Methods to Safely Scale Modern Neural Networks



Jishnu Mukhoti

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

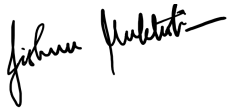
Michaelmas 2023

To Maa and Baba for everything.

Declaration

I hereby declare that this thesis (and the work presented in it) is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements of the degree of Doctor of Philosophy, and that this thesis is entirely my own work based on publications in collaboration with the co-authors where due acknowledgement is made.

January 2024
Jishnu Mukhoti



Acknowledgements

It has been a long journey of learning. Maa, Baba, without your sweet presence in my life, none of it would have been possible. Everything that I am, and everything that I have done, I owe it to you.

I would like to express my deepest and sincerest gratitude to my supervisors, Philip Torr, Yarin Gal and Puneet Dokania. Phil, thank you for giving me this amazing opportunity to pursue a DPhil in your lab. Your unwavering energy, enthusiasm, outlook, and guidance never fail to amaze me and have taught me just how exciting the life of a researcher can be. Yarin, you have been the perfect blend of hands-on and hands-off supervision and I am fortunate to have had your smart and productive insights while working through several problems in my DPhil. Puneet, I can't be grateful enough for how patiently you have taught me all the different aspects of thinking through a research problem. I have thoroughly enjoyed our detailed paper-pen, marker-board discussions and I believe a lot of the outcomes of this thesis can be attributed to those moments.

I would like to thank my collaborators in Oxford, especially Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Andreas Kirsch, and Joost van Amersfoort in chronological order. I have learnt a lot about research and team-work from our discussions, pair coding sessions, general chats and I have you to thank for that. I am also grateful to researchers at Meta AI, Tsung-Yu Lin, Ser-Nam Lim, Ashish Shah, Bor-Chun Chen, Omid Poursaeed, and Rui Wang for hosting me on multiple research internships during my DPhil and for the opportunity to have a taste of problems at the cutting edge of industrial research.

Life in Oxford would not have been sustainable without my beautiful partner, Priyanka. Not only have you supported and encouraged me throughout this journey, your presence has made it constantly electrifying and exciting to be in Oxford. I am looking forward to the future of our journey together.

I am grateful to all the members of Torr Vision Group (TVG) and Oxford Applied and Theoretical Machine Learning Group (OATML). You are some of the smartest people I have met and I have constantly learnt by being in your presence and having discussions at various points in the lab. I am grateful to my dear friends, Megha and Priya, for being an unfailing source of comfort, joy and support throughout my DPhil years. Finally, I am grateful to FiveAI and Meta AI for funding my DPhil.

A special mention to Denis, my landlord and friend, for giving me an excellent place to stay during my final year of DPhil and for the constant supply of warm porridge in the mornings to keep me going.

Abstract

Deep learning has witnessed radical progress in the past few years with state-of-the-art neural networks obtaining excellent performance on several real-world applications. In addition, self-supervised pre-training at scale has led to the advent of generalist foundation models which are capable of solving a wide variety of downstream tasks. However, in spite of its tremendous progress, deep learning still suffers from well-known issues in safety and scalability of neural networks, issues which persist even today.

First, in the area of safety, a core technical problem with even the most performant neural networks lies in their failure to know what they don't know. Neural networks can make confident predictions even when they are incorrect with potentially catastrophic consequences in safety-critical downstream tasks like medical diagnosis and autonomous driving. Second, in the area of scalability, deep learning remains a data and compute-hungry field, with state-of-the-art models often requiring massive training datasets, expensive annotations and large compute resources. Thus, the cost involved in further increasing the abilities of such models can be prohibitive. To mitigate some of these issues, this thesis proposes practical methods to safely scale modern neural networks.

The first part of the thesis deals with the area of safety and focuses on developing simple methods to reliably quantify and evaluate uncertainty estimates in modern neural networks. In this part, we begin with a study of the problem of miscalibration in classification neural networks from the lens of overfitting on Negative Log Likelihood (NLL) and propose an alternative loss function, focal loss, to train calibrated neural networks from scratch. Next, we move to the general problem of uncertainty quantification in the open-world setting where inputs can lie outside the closed set of training classes. In this area, we propose a simple method to estimate two different types of uncertainty – epistemic and aleatoric uncertainty, in deterministic single-forward pass models. Our proposed method is competitive with previous state-of-the-art on several benchmarks while providing computational benefits over prior methods. Finally, we study the popular application of Out-of-Distribution (OoD) detection, commonly used to evaluate uncertainty methods and provide a critique on the conventional evaluation benchmarks used in OoD detection. We identify desirable properties of OoD samples and propose an alternative method to generate such OoD samples. We provide an

extensive analysis to show that such generated OoD samples can be used to form stronger yet reliable benchmarks for OoD detection.

Our work on uncertainty can be used to identify gaps in the knowledge of trained models, thereby helping us acquire data or annotations that the model needs further fine-tuning on. The second part of this thesis focuses on the challenges of fine-tuning and scaling the abilities of modern neural networks. In this part, we develop fine-tuning methods which can reduce dependence on specialized annotations and preserve the expensive pre-trained knowledge of foundation models. We first analyze emergent properties in pre-trained multi-modal foundation models like CLIP to propose a contrastive fine-tuning method which can transfer such models to dense prediction tasks like semantic segmentation. Our method removes the dependence of semantic segmentation on expensive pixel-level annotations while at the same time bringing the scale of CLIP's pre-training to dense prediction tasks. Next, we study end-to-end fine-tuning for multi-modal foundation models to analyze their effect on pre-trained knowledge. We find most popular fine-tuning methods to cause the model to forget pre-trained concepts and propose a simple alternative which can mitigate this issue to a large extent. Finally, we analyze and evaluate our proposed method to show its efficacy in both single step as well as continual fine-tuning scenarios for foundation models.

Contents

List of Figures **xiii**

1 Introduction **1**
1.1 Deep Learning: The Big Picture . . . 2
1.1.1 Challenges in Safety and Scalability . . . 5
1.2 Thesis Outline and Contributions . . . 8
1.3 Publications . . . 10

2 Preliminaries **12**
2.1 Uncertainty in Deep Learning . . . 14
2.1.1 Problem Setting: Multi-class classification . . . 14
2.1.2 Calibrating the Softmax Probability . . . 16
2.1.3 Uncertainty beyond the Closed World Setting . . . 19
2.2 Scaling Deep Neural Networks . . . 24
2.2.1 CLIP: The Canonical Multi-modal Foundation Model . . . 25
2.2.2 Scaling Foundation Models Further . . . 27

3 Calibrating Deep Neural Networks using Focal Loss **29**
3.1 Introduction . . . 31
3.2 Problem Formulation . . . 33
3.3 What Causes Miscalibration? . . . 35
3.4 Improving Calibration using Focal Loss . . . 36
3.5 Experiments . . . 42
3.6 Conclusion . . . 46

4 Deep Deterministic Uncertainty: A New Simple Baseline **48**
4.1 Introduction . . . 50
4.2 Background . . . 54
4.3 Deep Deterministic Uncertainty . . . 57
4.4 Experiments . . . 59

4.4.1 Active Learning . . . 60
4.4.2 OoD Detection . . . 62
4.4.3 Semantic Segmentation . . . 65
4.5 Additional Insights . . . 67
4.6 Conclusion . . . 71

5 Raising the Bar on the Evaluation of Out-of-Distribution Detection 73
5.1 Introduction . . . 75
5.2 Related Work . . . 79
5.3 Method . . . 81
5.4 Experiments and Discussion . . . 85
5.4.1 Implementation Details . . . 85
5.4.2 Sanity Check on Benchmarks . . . 86
5.4.3 Evaluating Near OoD Datasets . . . 87
5.4.4 Evaluating Shifted Datasets . . . 91
5.4.5 Evaluation of OoD detection in real life . . . 93
5.5 Conclusion . . . 94

6 Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning 95
6.1 Introduction . . . 97
6.2 Related Work . . . 100
6.3 Patch Level Alignment in CLIP . . . 101
6.4 Semantic Coherence in Vision Encoders . . . 103
6.5 Patch Aligned Contrastive Learning (PACL) . . . 105
6.6 Experiments and Discussion . . . 108
6.6.1 Zero-shot Semantic Segmentation . . . 108
6.6.2 Image Classification . . . 114
6.7 Conclusion . . . 115

7 The Crippling Effect of Fine-tuning on Foundation Models 116
7.1 Introduction . . . 118
7.2 A brief note on fine-tuning . . . 120
7.3 The crippling effect of end-to-end fine-tuning . . . 123
7.3.1 Quantifying Concept Forgetting . . . 123
7.3.2 Benchmarking concept forgetting . . . 125
7.4 Can preserving features help? . . . 128

7.5 A nudge towards continual fine-tuning . . . 132
7.6 Related Work and Discussion . . . 133
7.7 Conclusion . . . 136

8 Conclusion **140**
8.1 Summary of Contributions . . . 141
8.1.1 Chapter 3: Calibrating Deep Neural Networks using Focal Loss . . . 141
8.1.2 Chapter 4: Deep Deterministic Uncertainty: A New Simple Baseline 142
8.1.3 Chapter 5: Raising the Bar on the Evaluation of Out-of-Distribution Detection . . . 144
8.1.4 Chapter 6: Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning . . . 145
8.1.5 Chapter 7: The Crippling Effect of Fine-tuning on Foundation Models 146
8.2 Future Directions . . . 147
8.3 Concluding Remarks . . . 149

Appendices

A Appendix: Calibrating Deep Neural Networks using Focal Loss **152**
A.1 Reliability Plots . . . 153
A.2 Relation between Focal Loss and Entropy Regularised KL Divergence . . . 154
A.3 Focal Loss and Cross-Entropy on a Linear Model . . . 156
A.4 Proofs . . . 158
A.5 Dataset Description and Implementation Details . . . 161
A.6 Additional Results . . . 163
A.7 Bar plots . . . 167
A.8 Focal Loss is Confident and Calibrated . . . 168
A.9 Ordering of Feature Norms . . . 171
A.10 Early stopping . . . 172
A.11 Machine Translation: A Downstream Task . . . 173

B Appendix: Deep Deterministic Uncertainty: A New Simple Baseline **176**
B.1 Related Work . . . 177
B.1.1 Predictive Entropy and Confidence in Recent Works . . . 178
B.2 Ambiguous- and Dirty-MNIST . . . 178
B.3 Algorithm . . . 179
B.3.1 Increasing sensitivity . . . 179

B.3.2 Algorithm Implementation . . . 180
B.3.3 Computational Complexity . . . 181
B.4 Experimental Details . . . 182
B.4.1 Dirty-MNIST . . . 182
B.4.2 OoD Detection Training Setup . . . 182
B.4.3 Semantic Segmentation Training Setup . . . 183
B.4.4 Compute Resources . . . 183
B.5 Additional Results . . . 184
B.6 Ablations and Toy Experiments . . . 189
B.6.1 Motivational Example in Figure 4.1 . . . 190
B.6.2 Effects of a well-regularized feature space on feature collapse . . . 192
B.6.3 QUBIQ Challenge . . . 193
B.6.4 Additional Baselines . . . 193
B.6.5 Calibration Metrics . . . 194
B.6.6 Two Moons . . . 195
B.6.7 5-Ensemble Visualisation . . . 196
B.6.8 Feature-Space Density &Epistemic Uncertainty vs Softmax Entropy &Aleatoric Uncertainty . . . 197
B.6.9 Objective Mismatch Ablation . . . 199
B.7 Theoretical Results . . . 199
B.7.1 Softmax entropy “cannot” capture epistemic uncertainty because Deep Ensembles “can” . . . 199
B.7.2 Objective Mismatch . . . 200

C Appendix: Raising the Bar on the Evaluation of Out-of-Distribution Detection 206
C.1 Additional Training Details . . . 206
C.1.1 Classifier Training . . . 206
C.1.2 Training Pix-2-Pix GAN . . . 208
C.1.3 Training GAN . . . 208
C.2 Outlier Exposure . . . 209
C.3 Additional Results . . . 210
C.4 Qualitative Examples of Generated Samples . . . 212

D Appendix: Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning **220**
D.1 Additional Implementation Details 220
D.1.1 Vision Embedder Architecture 221
D.1.2 Training details for Vision Embedder 221
D.1.3 Training and Evaluation Datasets 223
D.2 Additional Results 225
D.2.1 Semantic Coherence in CLIP 225
D.2.2 Qualitative Segmentation Results 226
D.2.3 Zero-shot Image Classification 226

E Appendix: The Crippling Effect of Fine-tuning on Foundation Models **228**
E.1 Training details 229
E.2 Computing $\Phi_\theta(\mathbf{x})$ 230
E.3 Additional results 231
E.3.1 Crippling effect of fine-tuning 231
E.3.2 Analysing L2SP 231
E.3.3 Performance of LDIFS 232
E.3.4 Comparison with Adapters and Prompt Tuning 234
E.3.5 Comparison with Wise-FT 235
E.3.6 Visualizing Table 7.1 235
E.3.7 Generality of Concept Forgetting beyond CLIP ViT-B/32 236

References **241**

List of Figures

1.1 **General Deep Learning Setup:** High level schematic overview of a training and test setup involving a pre-training, fine-tuning pipeline for training followed by real-world deployment. The figure also highlights a) safety issues of this setup showing noisy or out-of-distribution inputs post-deployment as well as b) challenges in scaling models with the requirement for specialized fine-tuning for downstream tasks as well as continual fine-tuning as a solution to amplifying model capabilities. 4

3.1 Metrics related to calibration plotted whilst training a ResNet-50 network on CIFAR-10. 36

3.2 How metrics related to model calibration change whilst training several ResNet-50 networks on CIFAR-10, using either cross-entropy loss, or focal loss with γ set to 1, 2 or 3. 38

3.3 (a): $g(p, \gamma)$ vs. p and (b-d): histograms of the gradient norms of the last linear layer for both cross-entropy and focal loss. 40

3.4 Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE, computed for ResNet-50 (first 3 figures) and ResNet-110 (last 3 figures) on CIFAR-10. 43

3.5 ROC plots obtained from ResNet-110 and Wide-ResNet-26-10 architectures trained on CIFAR-10 (in-distribution) and tested on SVHN (OoD), both pre and post temperature scaling. 46

4.1 *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and Fashion-MNIST (OoD) (**a**) requires using* softmax entropy *(**b**) and* feature-space density (GMM) *(**c**) with a well-regularised feature space (*ResNet-18+SN *vs* LeNet *&*VGG-16 *without smoothness &sensitivity).* **(b):** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples (stacked histogram). However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **(c):** With a well-regularised feature space (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without such feature space (LeNet &VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap. 51

4.2 *Epistemic and aleatoric uncertainty of ResNet-18+SN models trained on increasingly large subsets of Dirty-MNIST.* The feature-space density increases while the softmax entropy stays roughly the same, consistent with epistemic and aleatoric uncertainty being reducible and irreducible with more data, respectively. See §B.6.8 for a discussion on this. 57

4.3 DDU Pseudo-Code 58

4.4 *Active Learning experiments.* Acquired training set size vs test accuracy. DDU performs on par with Deep Ensembles. 62

4.5 AUROC vs corruption intensity averaged over all corruption types in CIFAR-10-C for 4 architectures. More details in §4.4.2 and more ablations in §B.4 in the appendix. 63

4.6 *Visualisation of uncertainty baselines on four PASCAL VOC validation samples (**rows**).* **Columns**: **a** shows pixel-wise accuracy; **b, c** predictive entropy (PE) and mutual information (MI) obtained for MC Dropout (MCD); **d, e** for deep ensembles; **f** per-pixel softmax entropy, *the aleatoric estimate of DDU*; and **g** feature density, *the epistemic component of DDU*. For all but g: the brighter, the more uncertain, whereas DDU's density g captures certainty: hence, the brighter, the more certain. 68

4.7 $p(\text{accurate}|\text{certain})$, $p(\text{uncertain}|\text{inaccurate})$ and PAVPU evaluated on PASCAL VOC validation set. DDU outperforms all other baselines. 68

5.1 ***Shifted* and *Near OoD* samples obtained from ImageNet.** *Shifted samples* are visually different but semantically similar to iD. *Near OoD images* are perceptually similar to iD but are semantically dissimilar. 75

5.2 **Categories of OoD samples.** Samples obtained from a distribution shift retain their conditional distribution over classes, whereas samples obtained from an unseen category don't belong to any of the training classes. 77
5.3 **Schematic of the proposed method to generate Shifted and Near OoD samples.** G_{P2P} and D_{P2P} represent the Pix-2-Pix to generate shifted samples and G and D represent the GAN to generate Near OoD images. The dotted lines show paths for gradient propagation. 84
5.4 **Confusion matrices for ensemble predictions on versions of CIFAR-10.** The ensemble broadly predicts Shifted samples correctly unlike Near OoD samples. 87
5.5 **FID scores between CIFAR-10 and generated OoD sets.** FID of Shifted CIFAR-10 is higher than most corruption types in CIFAR-10-C, indicating perceptual dissimilarity from CIFAR-10, whereas FID of N CIFAR-10 is significantly lower than SVHN and even lower than CIFAR-100 indicating perceptual similarity to CIFAR-10 samples. 88
5.6 AUROC % for different models, DenseNet-121 (DN), ResNet-50 (RN50), ResNet-110 (RN110), VGG-16, Wide-ResNet-28-10 (WRN) and Inception-v3 (INC), ViT-B-16/32 (VB16/32) and ViT-L-16/32 (VL16/32) trained on CIFAR-10 (first row) and CIFAR-100 (second row) using SVHN, CIFAR-10/100 and Near OoD (N) CIFAR-10/100 as OoD datasets. 90
5.7 Reliability plots for ImageNet shifts 91
5.8 **Ordering of AUROC scores of OoD detection baselines on 4 ViT and 2 convolutional architectures** trained on ImageNet-1K using OoD sets: ImageNet-O, Texture, Species, OpenImage-O and our Near OoD ImageNet (N-Im). 91
5.9 (a) **average LPIPS between ImageNet val and corrupted/shifted samples**, (b) **Near OoD samples from 10 ImageNet classes.** We observe the Near OoD samples to contain patches from their respective original classes but in an odd order, making the images unrecognizable compared to the original class. 92

6.1 **High level overview of our model.** We train an alignment between the patch level embeddings from the image encoder and the CLS embedding from the text encoder. This alignment can then be used to perform open-vocabulary semantic segmentation in a zero-shot manner. 97
6.2 **Patch level alignment between the word "cat" and images of cats.** In the first row, we show the original images, in the second row, we show the patch level alignment in CLIP ViT-B/16 and in the third row, we show the alignment for our method. 102

6.3 **ROC curve indicating semantic coherence of CLIP and DINO vision encoders.** CLIP encoders outperform DINO. 103
6.4 **Qualitative results on semantic coherence between CLIP and DINO ViT-B/16. a)**: we show the original image of a dog class with the patch marker (yellow X near the centre). **b, c)**: we show CLIP vision encoder cosine similarity across all patches for the same and a different image of a dog. **d, e)**: we show the same for DINO. See more examples in Appendix D.2.1. 104
6.5 **Compatibility function $\phi(\mathbf{x}, \mathbf{y})$ for Patch Aligned Contrastive Learning (PACL).** The image encoder $\hat{f}_v$ and embedder $\hat{e}_v$ produce patch level representations for each image whereas the text encoder f_t and embedder e_t produce the CLS representation for a given text. We compute the cosine similarity between the CLS text embedding and the vision patch embeddings and use them as weights to take a weighted sum over vision patch tokens. We use the cosine similarity between the weighted sum and the CLS text token as our compatibility $\hat{\phi}(\mathbf{x}, \mathbf{y})$. 107
6.6 **Qualitative results on zero-shot semantic segmentation.** The first row denotes the original images, the second row shows the corresponding labels, the third row shows results obtained from a vanilla CLIP ViT-B/16, and the fourth row shows results of our method, PACL trained on a CLIP ViT-B/16 encoder. The first 3 images from the left are from Pascal VOC and the next 3 images are from ADE20K. 111
6.7 **Qualitative results comparing segmentation of different encoders using PACL.** We use 3 images from PASCAL VOC val set and show their segmentations for DINO ViT-B/16, CLIP ViT-L/14 and CLIP ViT-B/16. . . 113
6.8 **Zero-shot image classification performance of PACL + CLIP vs vanilla CLIP on 12 datasets.** PACL + CLIP is competitive with or outperforms CLIP on most datasets. 115

7.1 **Test set $\Delta_{\text{LP}}\%$ for models fine-tuned on EuroSAT using different fine-tuning methods.** While EuroSAT Δ_{LP} rises, Δ_{LP} on all other datasets is almost always negative throughout the fine-tuning with a sole exception of DTD when fine-tuned using LP-init-L2SP. See §7.3.2 for details. 126
7.2 **Test set $\Delta_{\text{LP}}\%$ evaluated on different datasets for models fine-tuned on EuroSAT via various fine-tuning methods.** The L2SP baselines (particularly LP-init-L2SP) have the lowest negative Δ_{LP} on datasets other than EuroSAT. 127

7.3 ℓ_2 **distance in the parameter space** $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ of the image encoder over the course of fine-tuning. Except L2SP, all other baselines diverge away from their pre-trained counterparts. 128
7.4 ℓ_2 **distance in feature space** $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ between image encoders computed over the course of fine-tuning on EuroSAT. 129
7.5 **Test set** $\Delta_{\text{LP}}\%$ **evaluated on different datasets for models fine-tuned on EuroSAT.** LDIFS (Our) baselines (both ZS-init and LP-init) provides the best results in terms of avoiding concept forgetting without affecting downstream performance on EuroSAT. 131
7.6 $\mathcal{A}_{\text{LP}}$ for sequence: SVHN $\rightarrow$ CIFAR10 $\rightarrow$ RESISC45 (left 3 plots) and ii) SVHN $\rightarrow$ CIFAR100 $\rightarrow$ RESISC45 (right 3 plots). Vertical red line indicates a switch in the fine-tuning tasks. 134
7.7 $\mathcal{A}_{\text{LP}}$ for models trained with LDIFS on EuroSAT using the full concatenated feature vector vs just the last layer (LL) feature vectors. Full feature vectors are crucial for LDIFS's performance. 136

A.1.1 The confidence values for training samples at different epochs during the NLL training of a ResNet-50 on CIFAR-10 (see §3.3). Top row: reliability plots using 25 confidence bins; bottom row: % of samples in each bin. As training progresses, the model gradually shifts all training samples to the highest confidence bin. Notably, it continues to do so even after achieving 100% training accuracy by the 300 epoch point. 154
A.2.1 Optimal $\hat{p}$ for various values of q. 155
A.3.1 (a): Norm of logits (b): Norm of weights. 156
A.3.2 (a): Confidence of mis-classifications (b): Decision boundary of linear classifier trained using cross entropy (c): Decision boundary of linear classifier trained using focal loss 157
A.7.1 Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for ResNet-50 (first 3 figures) and ResNet-110 (last 3 figures) on CIFAR-100. 168
A.7.2 Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for Wide-ResNet-26-10 (first 3 figures) and DenseNet-121 (last 3 figures) on CIFAR-100. 169
A.7.3 Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for Wide-ResNet-26-10 (first 3 figures) and DenseNet-121 (last 3 figures) on CIFAR-10. 169

A.7.4 Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for ResNet-50 on Tiny-ImageNet. 169

A.8.1 Qualitative results showing the performance of Cross Entropy, Brier loss, MMCE and Focal Loss (sample-dependent γ 5,3) for a ResNet-50 trained on CIFAR-10. The first row of images have been correctly classified by networks trained on all four loss functions and the second row of images have all been incorrectly classified. For each image, we present the actual label, the predicted label and the confidence of the prediction both before and after temperature scaling. 171

A.9.1 L_2 norm of features obtained from the last ResNet block (before the linear layer) of ResNet-50 averaged over entire training dataset of CIFAR-10 using a batch size of 128. 175

A.11.1 Changes in *test set* BLEU score and *validation set* ECE with temperature, for models trained using (a) cross-entropy with hard targets (CE) (b) cross-entropy with label smoothing (LS) ($\alpha = 0.1$), and (c) focal loss (FL) ($\gamma = 1$). 175

B.2.1 *Samples from Ambiguous-MNIST.* 180

B.4.1 L2 distances between the feature space means of different classes for a pair of distant pixels on the Pascal VOC 2012 val set: (left) Pixels $(10, 255)$ and $(500, 255)$, (middle) Pixels $(234, 349)$ and $(36, 22)$ and (right) Pixels $(300, 500)$ and $(400, 255)$. 183

B.4.2 AUROC vs corruption intensity for all corruption types in CIFAR-10-C with Wide-ResNet-28-10 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density. 187

B.4.3 AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-50 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density. 188

B.4.4 AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-110 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density. 189

B.4.5 AUROC vs corruption intensity for all corruption types in CIFAR-10-C with DenseNet-121 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density. 190

B.6.1 *Uncertainty on Two Moons dataset*. Blue indicates high uncertainty and yellow indicates low uncertainty. Both the softmax entropy of a single model as well as the predictive entropy (PE) of a deep ensemble are uncertain only along the decision boundary whereas the feature-space density of DDU is uncertain everywhere except on the data distribution (the ideal behaviour). However, the feature density of a normal fully connected network (FC-Net) without any inductive biases can't capture uncertainty properly. 195

B.6.2 *Softmax outputs &entropies for 5 softmax models along with the predictive entropy (PE) and mutual information (MI) for the resulting 5-Ensemble.* a and b show that the softmax entropy is only reliably high for ambiguous iD points ($\pm$3.5–4.5), whereas it can be low or high for OoD points (-2–2). The different colors are the different ensemble components. Similarly, c shows that the MI of the ensemble is only high for OoD, whereas the PE is high for both OoD and for regions of ambiguity. See §B.6.7. 197

B.6.3 *Comparison of epistemic and aleatoric uncertainty captured b(ResNet-18+SN) on increasingly large subsets of Dirty-MNIST (DMN) and CIFAR-10 (C10). Clearly, feature density captures epistemic uncertainty which reduces when the model is trained on increasingly large subsets of training data, whereas softmax entropy (SE) does not. For comparison, we also plot a deep-ensemble's epistemic uncertainty, through mutual information (MI) for the same settings. For more details, see Table B.6.5.* 198

B.6.4 *3-component GMM fitted to a synthetic dataset with 3 different classes (differently colored) with 4% label noise using different objectives*. **a:** The optimas for conditional log-likelihood $\mathbb{H}_\theta[Y \mid Z]$, joint log-likelihood $\mathbb{H}_\theta[Y, Z]$, and marginalised log-likehood $\mathbb{H}_\theta[Z]$ all differ. Hence, the best calibrated model ($\mathbb{H}_\theta[Y \mid Z]$) will not provide the best density estimate ($\mathbb{H}_\theta[Z]$), and vice-versa. **b:** A mixture model that optimizes $\mathbb{H}_\theta[Y, Z]$ (GDA) does not have calibrated decision boundaries for aleatoric uncertainty: the ambiguous sample (due to label noise) marked by the yellow star has no aleatoric uncertainty under the GDA model. 199

B.7.1 *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and Fashion-MNIST (OoD)* ***a*** *requires using* softmax entropy ***b*** *and* feature-space density (GMM) ***c*** *with appropriate inductive biases (*ResNet-18+SN *vs* LeNet *&*VGG-16 *without them). Enlarged version.* **b:** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples. However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **c:** With appropriate inductive biases (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without appropriate inductive biases (LeNet &VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap. **Note:** Unambiguous MNIST samples and Ambiguous-MNIST samples are shown as stacked histograms with the total fractions adding up to 1 for Dirty-MNIST. 205

C.3.1 MI of the ensemble for Near OoD (N) CIFAR-10 and real CIFAR-10 samples. We use this to find plausible thresholds of MI. 211

C.4.1 AUPRC % for different models, DenseNet-121 (DN), ResNet-50 (RN50), ResNet-110 (RN110), VGG-16, Wide-ResNet-28-10 (WRN) and Inception-v3 (INC), ViT-B-16/32 (VB16/32) and ViT-L-16/32 (VL16/32) trained on CIFAR-10 (first row) and CIFAR-100 (second row) using SVHN, CIFAR-10/100 and Near OoD (N) CIFAR-10/100 as OoD datasets and softmax entropy, confidence, Mahalanobis distance and deep ensemble baselines. . . 215

C.4.2 ECE% of models on CIFAR-10-C (top row) and ImageNet-C (bottom row) for different corruption types. 216

C.4.3 Additional qualitative samples for CIFAR-10 and CIFAR-100 datasets. Left column shows real samples, and the right column shows corresponding shifted/transformed samples. 217

C.4.4 Additional qualitative samples for ImageNet datasets. Top shows real samples, and bottom shows corresponding shifted/transformed samples. . . 218

C.4.5 Additional qualitative samples for CIFAR-10, CIFAR-100 and ImageNet datasets. Samples show Near OoD images. 219

D.1.1 **Vision embedder $\hat{e}_v$ architecture for PACL.** The image encoder $\hat{f}_v$ produces token/patch-wise representations $\hat{f}_v(\mathbf{x}) \in \mathbb{R}^{T,D_v}$ of an input image $\mathbf{x}$. The vision embedder $\hat{e}_v$ converts the patch-wise representations to the multi-modal shared dimensional space, $\hat{e}_v(\hat{f}_v(\mathbf{x})) \in \mathbb{R}^{T,D}$ where T is the number of tokens or patches. 223

D.2.1 **Additional qualitative results on semantic coherence between CLIP and DINO ViT-B/16. a)**: we show the original image of a class (bird in top row, aeroplane in middle row and sheep in bottom row) with the patch marker (yellow X near the centre). **b, c)**: we show CLIP vision encoder cosine similarity across all patches for the same and a different image of the same class. **d, e)**: we show the same for DINO. 226

D.2.2 **Additional qualitative results on zero-shot semantic segmentation on Pascal VOC.** The first row shows the original images, the second row shows the corresponding ground-truth labels and the third row shows the predictions from our best performing model, i.e., PACL trained using a pre-trained CLIP ViT-B/16 encoder on GCC-3M + GCC-12M + YFCC-15M. 227

D.2.3 **Additional qualitative results on zero-shot semantic segmentation on ADE-20K.** The first row shows the original images, the second row shows the corresponding ground-truth labels and the third row shows the predictions from our best performing model, i.e., PACL trained using a pre-trained CLIP ViT-B/16 encoder on GCC-3M + GCC-12M + YFCC-15M. 227

E.3.1 **Test set ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for different fine-tuning methods** on 9 image classification tasks: (a) Stanford Cars, (b) CIFAR-10, (c) CIFAR-100, (d) DTD, (e) EuroSAT, (f) GTSRB, (g) MNIST (h) RESISC45 and (i) SVHN. $\mathcal{A}_{\text{ZS}}$ generally rises over the course of fine-tuning. However, for the ZS-init-L2SP and LP-init-L2SP baselines, the gain in $\mathcal{A}_{\text{ZS}}$ is relatively lower. Furthermore, for LP-init baselines, the performance is consistently lower compared to other baselines 232

E.3.2 **ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for models fine-tuned on EuroSAT (first row), GTSRB (second row), and SVHN (third row) and evaluated on 8 different datasets different from their fine-tuning dataset.** Most fine-tuning methods show a drop in $\mathcal{A}_{\text{ZS}}$ performance over the course of fine-tuning indicating a reduction in the model's transferability. 233

E.3.3 **ℓ_2 distance in weight space** $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ between the image encoder fine-tuned to the current time-step $f_{\theta_{v(t)}}$ and the pre-trained image encoder $f_{\theta_{v(0)}}$ over the course of fine-tuning. 233

E.3.4 **ℓ_2 distance in feature space** $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$, between image encoders, $f_{\theta_{v(t)}}$ and $f_{\theta_{v(0)}}$ computed over different fine-tuning methods for models fine-tuned on EuroSAT. 234

E.3.5 **Visualizing test LP accuracy of fine-tuning baselines on different datasets.** For each dataset, we plot its test set LP accuracy on the x-axis and the average test set LP accuracy of the remaining datasets on the y-axis. 236

E.3.6 **Comparing test set ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for different fine-tuning methods with LDIFS** on 9 image classification tasks. While LP-init baselines generally underperform on $\mathcal{A}_{\text{ZS}}$, we find ZS-init-LDIFS to be competitive with the best baselines. 237
E.3.7 **Comparing ZS Accuracy $\mathcal{A}_{\text{ZS}}$ of different fine-tuning methods with LDIFS for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) and evaluated on 8 datasets different from their fine-tuning dataset.** LP-init-LDIFS almost consistently beats all other baselines including LP-init-L2SP in preserving the model's transferability. . 237
E.3.8 **Test set LP Accuracy $\mathcal{A}_{\text{LP}}$ for different fine-tuning methods** on 9 image classification tasks: (a) Stanford Cars, (b) CIFAR-10, (c) CIFAR-100, (d) DTD, (e) EuroSAT, (f) GTSRB, (g) MNIST (h) RESISC45 and (i) SVHN. $\mathcal{A}_{\text{LP}}$ generally rises over the course of fine-tuning. However, for the ZS-init-L2SP and LP-init-L2SP baselines, the gain in $\mathcal{A}_{\text{LP}}$ is relatively lower. Furthermore, for LP-init baselines, the performance is consistently lower compared to other baselines 237
E.3.9 **LP Accuracy $\mathcal{A}_{\text{LP}}$ for models fine-tuned on EuroSAT (first row), GTSRB (second row), and SVHN (third row) and evaluated on 8 different datasets different from their fine-tuning dataset.** Most fine-tuning methods show a drop in $\mathcal{A}_{\text{LP}}$ performance over the course of fine-tuning indicating a reduction in the model's transferability. 238
E.3.10 **Comparing test set LP Accuracy $\mathcal{A}_{\text{LP}}$ for different fine-tuning methods with LDIFS** on 9 image classification tasks. 238
E.3.11 **Comparing LP Accuracy $\mathcal{A}_{\text{LP}}$ of different fine-tuning methods with LDIFS for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) and evaluated on 8 datasets different from their fine-tuning dataset.** LP-init-LDIFS almost consistently beats all other baselines including LP-init-L2SP in preserving the model's transferability. . 238
E.3.12 **ℓ_2 distance in weight space** $||\theta - \theta_0||_2^2$ between pre-trained image encoder f_{θ_0} and fine-tuned image encoder f_θ over the course of fine-tuning. Apart from ZS-init-L2SP and LP-init-L2SP, all fine-tuning baselines diverge in weight space over the course of fine-tuning. 239
E.3.13 **ℓ_2 distance in feature space** $d(\theta, \theta_0, \mathcal{D})$, between fine-tuned and pre-trained image encoders, f_θ and f_{θ_0} computed over different fine-tuning methods for models fine-tuned on EuroSAT. 239

The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer.

— Alan Turing

1 Introduction

Contents

1.1 Deep Learning: The Big Picture 2
1.1.1 Challenges in Safety and Scalability 5
1.2 Thesis Outline and Contributions 8
1.3 Publications 10

This thesis follows the integrated format where academic papers which are either published or accepted in peer-reviewed conferences, workshops and journals have been reformatted as individual chapters. In Section 1.1 of this introduction, we provide a brief history of deep learning following which, we describe the general training and test setup commonly used in deep learning systems. In Section 1.1.1 we introduce some of the major challenges in the field particularly in the area of safety and scalability of modern neural networks. These challenges form the core problems for the content chapters in this thesis. Section 1.2 then provides a brief outline for the chapters of the thesis mentioning the contributions of each individual chapter. Finally, Section 1.3 lists the publications associated with the main chapters of this thesis.

1.1 Deep Learning: The Big Picture

A Brief History Much before the advent of AI systems, humans had already begun philosophising about the existence of artificially created intelligent beings with the ability to perceive and reason in the real world. Such seeds can be found as early as Greek mythology with legends like Talos [1], a bronze giant constructed to guard the island of Crete from invading ships. While such artificially intelligent entities have always been the subject of interest in philosophy and science fiction, the field of artificial intelligence was born in reality due to a combination of ideas in the late 1930s, 1940s and early 1950s. A lot of the groundbreaking work which laid the foundation for the field can be credited to Alan Turing. His work on the theory of computation [2] introduced the famous “Turing Machine”, which showed that any computation can be described digitally. His paper “On Computable Numbers” [3] influenced Pitts and McCulloch who, in 1943, analyzed “artificial neurons” for the first time [4], what in the modern day is known as a neural network. In 1950, Turing published his seminal paper, “Computing Machinery and Intelligence” [5] which introduced the concept of the “Turing test", arguably the most popular philosophical test, till date, for a thinking machine. While Turing himself used the term “Machine Intelligence", the phrase “Artificial Intelligence" was coined by John McCarthy in 1956 during the Dartmouth Workshop, which marked the formal introduction of AI as an academic field. Although neural networks were introduced by Pitts and McCullouch [4] and then further developed on by Frank Rosenblatt with his work on Multi-Layer Perceptrons [6], efforts on training the weights of more than a single layer were unsuccessful till the invention and eventual popularisation of the Backpropagation algorithm [7–9] in the 1980s. The backpropagation algorithm reinvigorated interest in training multi-layer perceptrons and neural networks in general. The popularity of neural networks with multiple hidden layers, or in other words “Deep Neural Networks" which could now be trained with backpropagation, led to the development of the area of Deep Learning. The term “Deep Learning” was first introduced in the machine learning community by Rina Dechter in 1986 [10] and later by Aizenberg et al. in their study on binary neurons [11].

With the availability of more training data [12], better compute facilities [13] and advanced neural network architectures [14–16], in the 2000s, deep learning has experienced a tremendous boom making it possible for deep neural networks to become the unequivocal state-of-the-art in nearly all machine learning applications. These applications encompass several areas including computer vision [17], natural language processing [18], speech recognition [19], reinforcement learning [20], drug discovery [21], recommendation systems [22] and many others. Adding to this, the most recent feather in the cap of deep learning is the development of foundation models [23], a powerful family of neural networks which are pre-trained on massive internet-scale datasets due to which they are applicable on a wide range of downstream tasks and applications. With their astonishing performance, foundation models are widely considered to be the strongest step yet towards Artificial General Intelligence (AGI) [24], a hypothetical intelligent agent capable of performing any general task at human level. This has also sparked a huge discourse and debate on the emergent capabilities of foundation models and their safety in the real world [25], with AI safety becoming a pressing issue for academia, industry and governments alike. In spite of its massive success, deep learning is however not without its fair share of challenges. In what follows, we first describe the general setup of deep learning including training and deployment of deep neural networks along with some of the well-known challenges which the field currently faces.

The General Deep Learning Setup Figure 1.1 provides a high-level overview of the general setup used for training and deploying deep neural networks. The training stage conventionally follows a sequence of two main steps: i) pre-training and ii) fine-tuning [26]. The pre-training stage trains a randomly initialized neural network on a pre-training dataset. The main purpose of this stage is to learn features which can make the model suitable for downstream applications. If the pre-training dataset comes with labels, the model can be trained in a fully supervised manner. For instance, pre-training on ImageNet [27] has long been the norm for several downstream tasks in computer vision. However, in order to learn good features, the pre-training set often requires a large number of samples. While it is relatively easy to gather unlabeled samples, labeling them can be laborious and expensive,

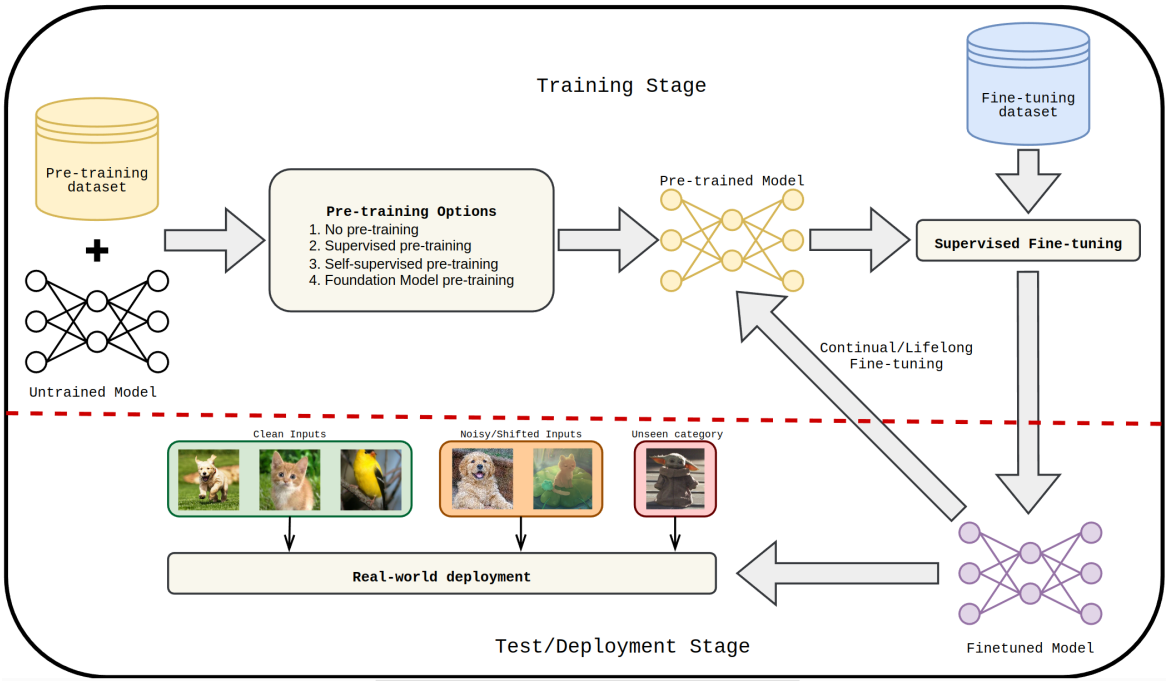


Figure 1.1: General Deep Learning Setup: High level schematic overview of a training and test setup involving a pre-training, fine-tuning pipeline for training followed by real-world deployment. The figure also highlights a) safety issues of this setup showing noisy or out-of-distribution inputs post-deployment as well as b) challenges in scaling models with the requirement for specialized fine-tuning for downstream tasks as well as continual fine-tuning as a solution to amplifying model capabilities.

thereby limiting the size of such training sets. This led to the exploration of self-supervised methods like contrastive training [28, 29] and masked autoregressive modeling [30–32] for pre-training. Self-supervised methods don't require labeled samples but define inherent tasks which leverage the structure of the data to produce meaningful training signals.They have thus been especially useful in cases where downstream data is limited and can inhibit the learning of good features. Finally, the creation of pre-training datasets from the internet containing several millions or billions of samples [33, 34], along with industry-scale compute resources has led to the pre-training of foundation models [23]. The main difference between self-supervised pre-training and foundation model pre-training lies in the scale of the pre-training dataset. Foundation models owing to their scale have shown emergent properties [35], capabilities which these models were not explicitly trained for. This makes them applicable

in a wide range of downstream tasks often with little to no further fine-tuning required.

The second training step involves fine-tuning the pre-trained model on a downstream task. For instance, some of the well-known downstream tasks in computer vision in increasing levels of prediction granularity are image classification and retrieval [36] which make predictions at the image level, object detection [37] which makes predictions using bounding boxes marking objects within an image and semantic segmentation [38] which requires models to predict and classify each pixel in an image. These tasks often require further fully supervised training on labeled downstream datasets using loss functions like Cross-Entropy, Mean Square Error (MSE) and Intersection-over-Union (IoU) [39]. Normally, the downstream fine-tuning datasets contain a lesser number of concepts compared to the pre-training sets. Once a model is fine-tuned on the downstream task, and achieves satisfactory performance on the respective test sets, it is ready for real-world deployment. It should be noted that in this overview we discuss a high-level paradigm of training and deployment of a machine learning model. There are can be many other specialized methods in machine learning which leverage parts of this general pipeline. A popular example for instance, is *in-context learning* [40], which involves providing demonstrations or examples of a task through input prompts at test time in an attempt to increase the flexibility of a model to solve a wide range of downstream tasks through the demonstrated examples. However, in-context learning is in turn dependent on a strong pre-training followed by instruction fine-tuning [41] in order to be effective.

In the next section, we discuss some of the challenges with this general setup particularly from the lens of safety of fine-tuned models and scaling such models further to expand their capabilities.

1.1.1 Challenges in Safety and Scalability

While deep learning has witnessed tremendous success in several real-world use-cases, it still suffers from its fair share of challenges. In this section, we highlight specific challenges relating to the safety and scalability of fine-tuned models which forms the focus for this thesis.

Challenges in safety The training and test setup in deep learning generally makes the i.i.d assumption [42], i.e., training and test samples are *independent and identically distributed*, which implies that they are drawn from the same probability distribution. This assumption however may be false in real-world deployment where a model may receive inputs which don't belong to the distribution of its training set. As shown in Figure 1.1, such inputs could be noisy or ambiguous or could completely lie outside the closed set of concepts on which the model has been fine-tuned. Even if the test inputs are clean samples, the fine-tuned model can still suffer from a generalization gap [43] and hence make mistakes on such inputs. In safety-critical applications like medical diagnosis [44, 45], autonomous driving [46] or financial prediction [47], incorrect predictions by a model can lead to potentially catastrophic consequences. Thus, in order to ensure the safe deployment of neural networks in such real-world applications, models need to provide a reliable measure of uncertainty in addition to their predictions. The uncertainty estimate can help users decide if the model's predictions can be trusted. While there exists research on uncertainty in neural networks [48, 49], state-of-the-art methods often impose additional computational overhead, either requiring multiple models to be stored in memory or multiple forward passes for each test input sample. This overhead inhibits their application on large-scale downstream tasks with real-time prediction requirements like semantic segmentation in autonomous driving. Thus, the first part of this thesis, specifically Chapter 3 and Chapter 4, focuses on developing reliable uncertainty quantification methods in deterministic single forward-pass neural networks which can handle incorrect predictions on clean inputs, noisy or ambiguous inputs, and inputs belonging to previously unseen classes. Chapter 5 then studies and critiques conventional evaluation benchmarks used for such uncertainty methods and proposes an alternative way to mitigate the shortcomings of current benchmarks.

Challenges in scalability In spite of all its potential, deep learning is also known to be a data hungry field, a problem which severely impedes the scaling of deep neural networks. One way in which the dependence on data can arise is in the form of requiring supervised labels in downstream tasks. In vision for instance, tasks like semantic segmentation [38] which

require localized recognition of real world concepts in an image are known to particularly suffer from this problem. Semantic segmentation requires pixel-level annotations to train models in a supervised manner. Since localized annotations in vision are expensive to obtain and generate, segmentation models are often limited in terms of the number of real-world concepts which the model can learn. Secondly, the data dependence in deep learning can also come in the form of the sheer number of samples required to train a powerful model. As an example, the major success in foundation models [23] can be hugely attributed to the availability of large pre-training datasets with size of the order of hundreds of millions or even billions of samples gathered from the internet, coupled with industry scale compute resources to train these models. This has led to pre-training costs running up to millions of dollars [50]. Even after such expensive pre-training, foundation models exhibit gaps in their knowledge such as knowledge cut-offs in LLMs [51] or poor performance in niche downstream areas [52]. They can also exhibit undesirable behaviours such as hallucination [53] in LLMs or bias in text-to-image models [54]. In order to then cover knowledge gaps or fix undesirable behaviours, these models need to undergo further fine-tuning. Since such fine-tuning is done on smaller datasets compared to the model's pre-training set, they can cause models to forget their pre-trained knowledge [55]. This is highly undesirable particularly due to the massively expensive nature of pre-training. A scalable solution to expand the abilities of foundation models would therefore require the continual fine-tuning of these models. The second part of the thesis, particularly Chapter 6 and Chapter 7, proposes methods to mitigate these scalability problems in deep learning. Chapter 6 leverages emergent properties of pre-trained foundation models to reduce the dependency on localized annotations in downstream vision tasks like semantic segmentation. Chapter 7 proposes a simple fine-tuning method which can preserve the pre-trained knowledge of foundation models during fine-tuning, thereby aiding in continual fine-tuning and the scalable knowledge expansion of such models.

1.2 Thesis Outline and Contributions

The first 3 content chapters (Chapter 3, Chapter 4 and Chapter 5) in the thesis deal with neural network safety and propose scalable methods of uncertainty quantification in deep learning and improvements in their evaluation benchmarks. The second 2 content chapters (Chapter 6 and Chapter 7) look at the above mentioned scalability problems in deep learning and propose fine-tuning methods for multi-modal foundation models which amplify their abilities to recognize real-world concepts. With this being an integrated thesis, Chapters 3-6 are published and Chapter 7 is accepted in leading peer-reviewed conferences or journals in machine learning and computer vision research. The chapters have been reformatted according to the format of the thesis and the supplementary materials for each paper have been added as Appendix chapters at the end of the thesis after Chapter 8. Each chapter is self-sufficient and has its individual related work sections specifically around the problem which the chapter focuses on. Overall, the chapters of the thesis are outlined as follows:

Chapter 2 first provides an overview of the progress and challenges in uncertainty quantification in deep learning. It then briefly discusses developments in scaling neural networks touching on multi-modal foundation models like CLIP. Finally, it discusses two open problems that challenge the further scaling of such foundation models.

Chapter 3 studies the problem of calibration in deep learning. It first expands on the Negative Log Likelihood (NLL) overfitting phenomenon observed in [56] which is known to cause miscalibration. To mitigate NLL overfitting, it proposes an alternative loss function, focal loss, which can calibrate models during training. It studies the properties of focal loss which leads to calibration and proposes a way to automatically select γ, the focal loss hyper-parameter, to optimally calibrate neural networks during training. The empirical results provide a significant improvement in calibration performance over other loss functions and can be further combined with post-hoc calibration methods like temperature scaling [56].

Chapter 4 proposes a simple method, Deep Deterministic Uncertainty (DDU), to quantify uncertainty in deterministic single forward-pass neural networks. It also discusses an

algorithm to use DDU to disentangle the two types of uncertainty: epistemic and aleatoric uncertainty. DDU simplifies previous efforts on deterministic uncertainty quantification like DUQ [57] and SNGP [58] by minimizing changes to model architecture and training setup. These simplifications allow DDU to be easily applied on large computer vision applications like semantic segmentation. Finally, the chapter also applies DDU on out-of-distribution detection and active learning showing competitive performance with state-of-the-art deep ensembles [49] while using a single deterministic model.

Chapter 5 studies out-of-distribution (OoD) detection, frequently used to evaluate uncertainty methods in deep learning. It critiques the conventional OoD evaluation benchmarks which use arbitrary pairs of in-distribution vs OoD datasets to measure if a model can separate the two. It proposes an alternative approach by first identifying desirable properties in OoD in terms of perceptual and semantic similarity to in-distribution samples and then formulating a generative model to generate such OoD samples from the in-distribution training set. Finally, it analyses the generated samples and provides a thorough empirical study across training datasets, model architectures and OoD detection methods to show that the generated OoD samples provide a stronger and more reliable evaluation benchmark for OoD detection compared to the current existing OoD datasets.

Chapter 6 tackles the label dependency problem in fine-grained visual recognition, particularly semantic segmentation, by identifying and leveraging an emergent property in CLIP's contrastive pre-training. The emergent property leads to a new compatibility function for contrastive loss, Patch Aligned Contrastive Learning (PACL), which can train a patch-level alignment between the image and text encoders in CLIP. The alignment helps in identifying localized regions in an image corresponding to a given text input and thus can be seamlessly applied to the task of open vocabulary semantic segmentation. The modified contrastive training only requires freely available image-text data and does not depend on any additional pixel-level annotations or pixel-level masks, thereby bringing the scale of CLIP's contrastive pre-training to downstream tasks like semantic segmentation.

Chapter 7 tackles the continual fine-tuning problem in multi-modal foundation models like CLIP. It first performs a comprehensive analysis of end-to-end fine-tuning methods in CLIP, across 10 downstream image classification tasks, and observes that all fine-tuning methods lead to forgetting of pre-trained concepts, an undesirable outcome. It studies why this is the case and proposes a simple fix in the form of a regularizer using l_2 Distance in Feature Space (LDIFS) between the model being fine-tuned and the pre-trained model. This regularizer mitigates the forgetting problem to a great extent without harming downstream task performance. Finally, the chapter applies LDIFS on a sequence of downstream tasks to show its beneficial effects in a continual fine-tuning setup for foundation models.

Chapter 8 concludes the thesis by summarizing the contributions and discussing future directions of research.

1.3 Publications

The contributions discussed for each chapter in the previous section are from the following publications:

1. **Chapter 3**: **Jishnu Mukhoti***, Viveka Kulharia*, Amartya Sanyal, Stuart Golodetz, Philip H.S. Torr, Puneet K. Dokania, "Calibrating Deep Neural Networks using Focal Loss". *Advances in Neural Information Processing Systems (NeurIPS),* 2020 (link).

2. **Chapter 4**: **Jishnu Mukhoti***, Andreas Kirsch*, Joost van Amersfoort, Philip H.S. Torr, Yarin Gal, "Deep Deterministic Uncertainty: A New Simple Baseline". *IEEE/CVF Computer Vision and Pattern Recognition Conference, CVPR,* 2023 (**Highlight**, link).

3. **Chapter 5**: **Jishnu Mukhoti**, Tsung-Yu Lin, Bor-Chun Chen, Ashish Shah, Philip H.S. Torr, Puneet K. Dokania*, Ser-Nam Lim*, "Raising the Bar on the Evaluation of Out-of-Distribution Detection". *International Conference on Computer Vision Workshop on Out-of-Distribution Generalization in Computer Vision (ICCV OOD-CV),* 2023 (**Oral**, link).

4. **Chapter 6**: **Jishnu Mukhoti**, Tsung-Yu Lin, Omid Poursaeed, Rui Wang, Ashish Shah, Philip H.S. Torr, Ser-Nam Lim, "Open Vocabulary Semantic Segmentation With Patch Aligned Contrastive Learning". *IEEE/CVF Computer Vision and Pattern Recognition Conference, CVPR,* 2023 (**Highlight**, link).

5. **Chapter 7**: **Jishnu Mukhoti**, Yarin Gal, Philip H.S. Torr, Puneet K. Dokania, "Fine-tuning can cripple your foundation model; preserving features may be the solution". *Transactions on Machine Learning Research*, (accepted with **Featured Certification**, link).

In the above list, "*" indicates equal contribution by authors. The following publication on the topic of uncertainty estimation in deep learning was also made during the course of this DPhil.

1. **Jishnu Mukhoti**, Puneet K. Dokania, Philip H.S. Torr, Yarin Gal, "On Batch-Normalization for Approximate Bayesian Inference". *3rd Symposium for Advances in Approximate Bayesian Inference (AABI),* 2021 (link).

2
Preliminaries

Contents

2.1 Uncertainty in Deep Learning 14
2.1.1 Problem Setting: Multi-class classification 14
2.1.2 Calibrating the Softmax Probability 16
2.1.3 Uncertainty beyond the Closed World Setting 19
2.2 Scaling Deep Neural Networks 24
2.2.1 CLIP: The Canonical Multi-modal Foundation Model 25
2.2.2 Scaling Foundation Models Further 27

Abstract

This chapter discusses preliminary concepts and problems that will help the reader understand the remainder of the thesis. In Section 2.1, we provide a brief overview of uncertainty quantification in deep learning. This provides the background for Chapter 3, Chapter 4 and Chapter 5 of the thesis. Next, in Section 2.2, we provide a discussion on developments in scaling the abilities of neural networks and introduce multi-modal foundation models, particularly the CLIP [52] model. We also discuss two problems with scaling such foundation models which provide background and motivation for our work in Chapter 6 and Chapter 7.

2.1 Uncertainty in Deep Learning

Deep neural networks have recently proven themselves to be extremely performant in a variety of applications. This has led to their widespread deployment in several real-life use cases, many of which are safety-critical. Classic examples of such use cases include autonomous driving [46] and medical diagnosis [44, 45]. In order to then ensure that neural networks are safe for deployment in such applications, they not only need to be accurate in their predictions but should also be able to identify cases where they may be incorrect. In other words, neural networks should be able to produce a reliable estimate of their uncertainty when making predictions. For instance, in the medical field, a possible misdiagnosis of a medical issue by a model should be referred for further investigation by a human expert.

While deep neural networks can be accurate in several domains, they are also known to be overconfident when making incorrect predictions [48, 56]. This makes reliable uncertainty quantification in neural networks an important and open research problem. In this section, we provide a brief overview of the main approaches that have been developed in this direction.

2.1.1 Problem Setting: Multi-class classification

One of the main problems in supervised machine learning is *classification* [59], the task of classifying a given input into one of a predefined set of classes or categories. Given a labeled training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ of N samples, where each sample (say sample i) has an input $\mathbf{x}_i$ and its corresponding label $y_i \in \{1, 2, ..., K\}$, which is one among a set of K classes, the task is to train a model which is able to correctly predict the class of test input $\mathbf{x}^*$ which may not belong to the dataset $\mathcal{D}$. We assume that the training and test samples are *independent and identically distributed (i.i.d.)* which means that the test samples are drawn from the same probability distribution which generated the training samples in $\mathcal{D}$. This implies that we assume test samples $\mathbf{x}^*$ to also belong to one of the set of K classes: $\{1, 2, ..., K\}$.

As the model, we use a neural network [15, 16, 60, 61] f_θ, parameterized by θ, to approximate the function mapping the inputs to the outputs in the training set. Given an input

$\mathbf{x}$, the neural network outputs a vector of *logits*, $f_\theta(\mathbf{x}) \in \mathbb{R}^K$, with an element or logit for each of the K classes. The predicted class is then obtained as $\hat{y} = \arg\max_c f_\theta(\mathbf{x})_c$, which is the class corresponding to the maximum logit value. While the predicted class can be obtained from the logit vector, the logits themselves don't have any numerical restrictions. In order to convert the logit vector into a categorical probability distribution over classes, the softmax operation [62] is commonly applied on the logit vector as shown below:

$$\hat{p}(y = c|\mathbf{x}, \theta) = \frac{e^{f_\theta(\mathbf{x})_c}}{\sum_{i=1}^{K} e^{f_\theta(\mathbf{x})_i}}. \tag{2.1}$$

The softmax squishes the logit vector into a categorical probability distribution over classes, such that $\sum_{c=1}^{K} \hat{p}(y = c|\mathbf{x}, \theta) = 1$. Naturally, the same $\arg\max$ operation can be applied on the softmax vector to obtain the model prediction on an input, with the predicted class being $\hat{y} = \arg\max_c \hat{p}(y = c|\mathbf{x}, \theta)$.

In order to then train the neural network f_θ on $\mathcal{D}$, for each training sample, we first convert the label $y \in \{1, 2, ..., K\}$ into a ground-truth categorical probability distribution $p(y|\mathbf{x})$ over K classes. Conventionally, this ground-truth distribution is a one-hot encoding, where $p(y = c_i|\mathbf{x}_i) = 1$ if $y = c_i$, i.e., the ground-truth correct class for input $\mathbf{x}_i$ is c_i and $p(y = k|\mathbf{x}_i) = 0 \; \forall k \neq c_i$. However, there can be other ways of defining the ground-truth distribution over classes as well. For instance, in label smoothing [63], a small probability mass is taken from the ground-truth class and uniformly distributed among other classes. The loss function is then the cross-entropy between two categorical distributions: the ground-truth distribution $p(y|\mathbf{x})$ and the softmax distribution predicted by the model $\hat{p}(y|\mathbf{x}, \theta)$, averaged over the entire training set $\mathcal{D}$:

$$\mathcal{L}_{\text{CE}}(\theta) = \mathbb{H}[p||\hat{p}] = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} p(y_i = c|\mathbf{x}_i) \log \hat{p}(y_i = c|\mathbf{x}_i, \theta). \tag{2.2}$$

In the one-hot case, the cross entropy loss reduces to:

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log \hat{p}(y_i = c_i|\mathbf{x}_i, \theta), \tag{2.3}$$

where c_i is the ground-truth correct class for input $\mathbf{x}_i$. With this loss function defined, we use the well-known backpropagation [9] algorithm to learn the parameters of the neural network. In the next section, we discuss if and when, the predicted softmax probability $\max_c \hat{p}(y = c|\mathbf{x}, \theta)$ can be used as a reliable measure of confidence from the neural network.

2.1.2 Calibrating the Softmax Probability

We have seen how the softmax operation allows us to predict a categorical probability distribution over classes. The predicted class has an associated probability estimate $\max_c \hat{p}(y = c|\mathbf{x}, \theta)$. It then may make intuitive sense to use this probability estimate as a measure of the model's confidence in its prediction, i.e., to have a high probability estimate when the model is confident and vice versa. However, the natural question then is *whether the softmax probabilities are indeed reliable measures of the model's confidence?* One way of tackling this question is to examine if the softmax probability scores are *calibrated*.

What is calibration? A probability score predicted for an event is calibrated if it represents the true likelihood of occurrence of the event. In the context of classification, given any test input $\mathbf{x}^*$ with true label y^*, the predicted softmax probability $\max_c \hat{p}(y = c|\mathbf{x}^*, \theta)$ is calibrated if it represents the true likelihood of the prediction to be accurate:

$$P\left[\hat{y} = y^* \middle| \max_c \hat{p}(y = c|\mathbf{x}^*, \theta)\right] = \max_c \hat{p}(y = c|\mathbf{x}^*, \theta), \tag{2.4}$$

where $\hat{y} = \arg\max_c \hat{p}(y = c|\mathbf{x}^*, \theta)$. As an example, if a model predicts a sample with 0.7 probability, this score is calibrated and represents the true likelihood of correct prediction if, among all samples predicted by the model with 0.7 probability, 70% are accurate predictions. The model is perfectly calibrated if this holds true for all prediction probabilities in the range $[0, 1]$. Since $[0, 1]$ is an uncountably infinite interval, it is impossible to check for perfect calibration for all possible values in this interval. We therefore use an approximation where we convert this interval into a set of finite bins and check for calibration in each bin.

How to measure calibration? The simplest way to convert the interval $[0, 1]$ into bins is to make them uniformly wide. If we make M such bins, then the i^{th} bin will represent

the interval $B_i = \left[\frac{i-1}{M}, \frac{i}{M}\right]$. Given a test dataset $\mathcal{D}_{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$, we first use the model to make predictions on each sample and place the sample in the bin which corresponds to its prediction probability. Since we want the predicted probabilities to match the true likelihood of the predictions being accurate, for each bin, we want the average confidence of samples within that bin to match the accuracy of prediction within that bin. For bin B_i, let A_i represent the accuracy of predictions in that bin and let C_i be its average prediction confidence. The calibration error is then an estimate of the absolute difference between the accuracy and confidence averaged over bins. For a perfectly calibrated model, we would like $A_i = C_i \; \forall i \in \{1, ..., M\}$. In [64], we get two very well known calibration error scores, the Expected Calibration Error (ECE) and the Maximum Calibration Error (MCE):

$$\begin{aligned} \text{ECE} &= \sum_{i=1}^{M} \frac{|B_i|}{L} |A_i - C_i| \\ \text{MCE} &= \max_{i \in 1,...,M} |A_i - C_i|. \end{aligned} \tag{2.5}$$

The ECE metric is extensively used in research literature as the go-to calibration error score. There are also improvements over these calibration error metrics [65] which include better ways of binning the interval $[0, 1]$ and using softmax probabilities for every class and not just the predicted class. Finally, a useful tool to visualize calibration is the reliability plot [66] which shows the accuracy for each confidence bin in a bar plot. While the calibration error scores are absolute positive values, they don't give an idea of whether the model is miscalibrated due to overconfidence or underconfidence. However, the reliability plots show if the bin accuracies are below the bin confidences, a case of overconfidence, or the other way round indicating underconfidence. A calibrated model will chart the $y = x$ line on a reliability plot indicating that the bin accuracies and bin confidences are perfectly equal.

Major approaches to calibrate neural networks As shown in [56], modern neural networks trained for the multi-class classification problem are often overconfident and miscalibrated. The main reason behind this is overfitting on the Negative Log Likelihood (NLL) or cross-entropy loss, where, even after obtaining perfect accuracy on the training set, the model can still minimize training set NLL to match the one-hot encoding distribution of

the ground-truth. This causes it to overfit on NLL during training, making it more confident on its predictions even when it is inaccurate on the test set. In order to fix this problem, there are two main families of approaches: a) *post-hoc calibration* methods which aim to first train the model as is and then calibrate the probabilities of the trained model and b) *regularization* methods which introduce additional regularizers to the loss or augment training samples to ensure that the neural networks remain calibrated during training.

Some of the popular post-hoc calibration methods include histogram binning [67], isotonic regression [68], Dirichlet calibration [69] and Platt Scaling [70]. One method particularly worth noting is the single-parameter extension of Platt Scaling called Temperature Scaling [56]. It uses a single parameter called temperature T which scales the logits of the model, with the scaled softmax probabilities being:

$$\hat{p}(y = c|\mathbf{x}, \theta) = \frac{e^{f_\theta(\mathbf{x})_c/T}}{\sum_{i=1}^{K} e^{f_\theta(\mathbf{x})_i/T}}. \tag{2.6}$$

The temperature T is tuned post-hoc by minimizing NLL on a held-out validation set. Although temperature scaling is a simple and effective method, some improvements have been proposed over classic temperature scaling. Two of these include Attended Temperature Scaling [71] which increases class-specific samples in the validation set and Adaptive Temperature Scaling [72] which proposes learning a temperature for each sample. Even so, temperature scaling still remains one of the most commonly used calibration methods in practice due to its simplicity, effectiveness and its ability to preserve model accuracy.

A second thread of work on calibration follows the idea of regularizing the loss function during training in order to calibrate the model from scratch. Some of the major approaches in this direction include entropy regularization [73] which maximizes entropy of the softmax distribution as a regularizer, Maximum Mean Calibration Error (MMCE) [74] and Differentiable ECE (DECE) [75] which propose differentiable proxies to calibration error as regularizers, label smoothing [63] which changes the one-hot ground truth by shifting some probability mass from the ground-truth class uniformly across other classes and mixup [76] which uses mixup augmentation specifically for calibration. ***Our work on calibration using focal loss***

[77] also belongs to this genre of approaches and we describe it in detail in Chapter 3. In the next section, we discuss the general problem of uncertainty quantification in the open world scenario where inputs may lie outside the set of pre-defined classes.

2.1.3 Uncertainty beyond the Closed World Setting

The concept of calibration inherently requires defining what we mean by correct or accurate predictions. What happens then when we encounter an input which does not belong to any of the classes in the training set? It is not possible to define what a correct prediction is in this case. The only sensible answer from the model for such an input is "*I don't know*". Thus, research directed towards ensuring that neural networks know what they don't know has led to the development of the field of uncertainty quantification in deep learning.

Types of uncertainty There are two types of uncertainty commonly investigated in deep learning [48, 78]. *Epistemic* uncertainty is the type which occurs due to a lack of information about model parameters which correctly explain the data. It is often associated with a dearth of training data and can therefore be reduced by introducing more training data. *Aleatoric* uncertainty is the type introduced due to inherent noise, ambiguity or missing features in the training data, for instance observation noise or label noise. Introducing more data without fixing the source of noise or ambiguity does not reduce aleatoric uncertainty. In real-life scenarios, it is often important to disentangle these two types of uncertainty as they can have different rectifying actions associated with them. In classification for instance, a high epistemic uncertainty might indicate that the input belongs to a class which the model hasn't seen before and hence the action would be to train the model on samples belonging to the new class. However, a high aleatoric uncertainty might mean that the input is noisy or inherently ambiguous and hence, can either be denoised before being presented to the model again or can simply not be classified.

How to evaluate uncertainty estimates? Unlike ground-truth labels which can be used to evaluate the accuracy of a model, there are no ground-truth uncertainty estimates. One way to then evaluate a model's uncertainty could be to tie the uncertainty estimate with the

model's accuracy. Naturally, a model should be uncertain when inaccurate and vice versa. This rationale leads to metrics similar to calibration error scores as described in Section 2.1.2. However, when we can't define a sensible measure of accuracy, the only way to evaluate uncertainty is to measure performance on uncertainty dependent downstream applications. One such application is *active learning* [79, 80] which can use uncertainty as the acquisition function to select samples from the pool set for labeling. Another application is *selective classification* or classification with the reject option [81], where a classifier has the option to reject predictions based on low confidence or high uncertainty samples. Yet another application of uncertainty can be found in *reinforcement learning* [82] where uncertainty estimates are used in imperfect information games and exploration of reinforcement learning agents. However, the most popular and commonly used downstream application to evaluate uncertainty is *Out-of-Distribution (OoD) Detection* [83]. OoD detection breaks the i.i.d assumption at test time, i.e., inputs seen at test time need not necessarily be sampled from the same distribution as those seen during training. Given that the training distribution is a joint distribution of inputs and labels $p_{\text{train}}(\mathbf{X}, \mathbf{Y})$, there are multiple ways by which one can generate an OoD sample from the test distribution $p_{\text{test}}(\mathbf{X}, \mathbf{Y})$. From the chain rule of probability, we have $p_{\text{train}}(\mathbf{X}, \mathbf{Y}) = p_{\text{train}}(\mathbf{Y}|\mathbf{X})p_{\text{train}}(\mathbf{X})$ and we can expand $p_{\text{test}}(\mathbf{X}, \mathbf{Y})$ in a similar manner. The general convention breaking i.i.d in the OoD test set is to have $p_{\text{train}}(\mathbf{X}) \neq p_{\text{test}}(\mathbf{X})$, i.e., the distribution of inputs changes at test time. This gives us two subcases: one where we keep $p_{\text{train}}(\mathbf{Y}|\mathbf{X}) = p_{\text{test}}(\mathbf{Y}|\mathbf{X})$, a case popularly known as covariate shift [84], and the other where we have $p_{\text{train}}(\mathbf{Y}|\mathbf{X}) \neq p_{\text{test}}(\mathbf{Y}|\mathbf{X})$ indicating that we encounter new classes at test time. These two subcases are important to note as we have different metrics for evaluating the two cases.

In case of covariate shift, the inputs still belong to one of the training classes. Therefore, ideally we want the model to make accurate predictions and if incorrect, we want the model to predict calibrated probability scores. Thus, we can use metrics like accuracy and ECE to evaluate models in this case. Inputs with covariate shift are generally obtained by introducing natural or synthetic transformations to clean inputs. Some examples of such datasets obtained

using ImageNet [12] as a reference are ImageNet-C [85] which introduces different types of synthetic noise like brightness, Gaussian noise and Gaussian blur, ImageNet-R [86] which has different renditions of ImageNet classes, and ImageNet-Sketch [87] which introduces images from ImageNet classes in the form of hand-drawn sketches.

If on the other hand, we have inputs which belong to a class outside the training set, it is not possible to define a sensible measure of accuracy. In this case, we estimate the model's predicted uncertainty on such OoD inputs to identify if they belong to the training set classes or not. To evaluate models, we use proxy OoD datasets which have classes outside the training set and measure the model's uncertainty as a binary classifier to identify if a given input is in-distribution, i.e., belonging to a training set class, or OoD. Since this is modeled as a binary classification problem, we can use metrics like Area Under Receiver Operating Characteristic (AUROC) and Area Under Precision Recall Curve (AUPRC) [88] to evaluate the model's OoD detection performance. The proxy OoD datasets are often obtained from conventional benchmarks used in literature to provide a common framework for comparing baselines. Some of the popularly used OoD detection benchmarks for image classification include CIFAR-10 [89] vs SVHN [90], CIFAR-10 vs CIFAR-100 [89], ImageNet vs ImageNet-O [91], ImageNet vs OpenImage-O [92] etc. While effort has been put into creating larger and better OoD datasets [93], many popular benchmarks still use OoD sets which are arbitrarily chosen while lacking a proper understanding of what forms a good OoD dataset. Additionally, many benchmarks are on the verge of saturation with state-of-the-art OoD detection methods [94] obtaining near 100% AUROC on them. ***In Chapter 5, we study this problem particularly to pin-point desirable properties of OoD samples and propose a method of generating such samples to evaluate and benchmark OoD detection without using any proxy OoD datasets.*** Finally, it is worth mentioning that the areas of Open-Set Recognition [95–97] and Anomaly Detection [98–100] are also heavily related to OoD Detection with methods proposed in one genre of works often being applicable and performant in others.

Major approaches to estimate uncertainty Having discussed how to evaluate uncertainty, here we discuss some of the major threads of work which have been developed

to model reliable uncertainty estimates in deep neural networks. A principled way of approaching the uncertainty problem lies in following the Bayesian formalism [101, 102]. This approach initially assumes a prior distribution $p(\theta)$ over model parameters θ which represents the initial belief on model parameters. Once the model is exposed to the training data $\mathcal{D}$, this belief is updated following Bayes' rule to obtain the posterior distribution over the model parameters $p(\theta|\mathcal{D})$ as follows:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \tag{2.7}$$

where $p(\mathcal{D}|\theta)$ is known as the likelihood and $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$ is known as the evidence or marginal likelihood. At test time, given input $\mathbf{x}^*$, the prediction is made by marginalizing over the model's posterior:

$$p(y|\mathbf{x}^*, \mathcal{D}) = \int p(y|\mathbf{x}^*, \theta)p(\theta|\mathcal{D})d\theta. \tag{2.8}$$

The problem with exact Bayesian inference lies in the computational intractability of marginalization. This has led to the development of approximate Bayesian inference methods like Markov Chain Monte Carlo (MCMC) [102–104], Variational Inference [105, 106] and Laplace approximations [101, 107, 108]. Of these, the most commonly used approximation is *variational inference* [105, 106], which learns an approximate posterior distribution $q_\phi(\theta)$ parameterized by ϕ, called the variational distribution and minimizes the KL divergence $\mathrm{KL}[q_\phi(\theta)||p(\theta|\mathcal{D})]$ between the variational distribution and the true posterior $p(\theta|\mathcal{D})$ using the Evidence Lower Bound (ELBO) loss function. Many popular methods which have found practical use in deep learning applications have been developed in this direction, some of which include MC Dropout [109], Concrete Dropout [110] and Bayes by Backprop [111]. Inference at test time in such methods involves multiple forward passes per input sample, with some source of stochasticity in the network leading to variance in the outputs. The mean prediction across the outputs becomes the prediction and a measure of variance across the outputs is used as the uncertainty estimate.

Another work on uncertainty which has proven to be particularly performant in practice is Deep Ensembles [49]. In this method, we train an ensemble of neural networks and use the average softmax distribution across all trained ensemble components as the test time prediction. Deep ensembles have also been interpreted as a form of Bayesian Model Averaging [112] with each trained network in the ensemble being a sample from an approximated posterior. Due to its effectiveness, many improvements over deep ensembles have been proposed in the literature including BatchEnsemble [113], MarkEnsembles [114] and Packed-Ensemble [115]. The deep ensemble genre of works is generally known to form the state-of-the-art in uncertainty quantification in deep learning.

Finally, while both approximate Bayesian inference and Deep Ensembles are effective methods of modeling uncertainty, they are both computationally expensive. They either require multiple models to be stored in memory or require multiple forward passes to be made for each sample at test time. This often limits their applicability in downstream applications which require real-time estimates. For instance, in a self-driving car, we often require uncertainty estimates in real-time for large vision models like semantic segmentation. Computationally expensive uncertainty methods can therefore fail in such applications. To tackle this, a thread of research has developed methods to estimate uncertainty in deterministic single forward-pass neural networks. Two popular approaches in this category of works include Deterministic Uncertainty Quantification (DUQ) [57], Spectral Normalized Gaussian Process (SNGP) [58]. Both these methods propose: a) a distance preserving feature space, which maintains the distance between input samples without collapsing them into the same region of the feature space, and b) a distance aware output layer, either in the form of a Radial Basis Function network or a Gaussian Process. The entire network is then trained end-to-end in both cases. ***In Chapter 4, we propose Deep Deterministic Uncertainty (DDU) [116], a deterministic uncertainty method which provides further simplifications on DUQ and SNGP by fitting a Gaussian Discriminant Analysis (GDA) model post-hoc on the feature space of a neural network.*** Our approach leads to a model which can both quantify and disentangle epistemic and aleatoric uncertainty with a single forward-pass, is competitive

with deep ensembles in performance, and requires minimal changes to model architecture and training setup. Furthermore, the post-hoc nature of DDU's GDA model allows us to easily apply DDU to tasks like semantic segmentation [117]. With this, we conclude our preliminary discussion on uncertainty in deep learning.

2.2 Scaling Deep Neural Networks

In the previous section, we provided a brief overview of uncertainty methods in deep learning. While one of the main purposes of uncertainty is to make models safe for deployment in real-world applications, a second major motivation is to scale the abilities of neural networks. If a model can identify previously unseen concepts, it can ask for further training on those concepts. This lends uncertainty as an important component in applications like active learning [80, 118] and continual learning [119, 120] where the model's uncertainty can be used to identify unknown concepts as well as guide its continual training on such concepts. However, active learning and continual learning are not the only areas which are motivated to scale the ability of a model to recognize more real-world concepts. The is in fact, one of the primary challenges of deep learning research and has been approached from multiple major angles:

- First, *dataset size* in supervised learning has grown massively. As an example, image classification dataset size has increased from MNIST [14] and CIFAR-10 [89] with 10 classes to ImageNet-21K [12] with $\approx$ 21K classes and JFT-300M [121] with 18291 classes.
- Second, neural network *model size* in terms of number of trainable parameters has grown [122] with new seminal architectures like VGG [61], ResNet [15] and most recently, Transformers [16, 123]. At the same time, regularization methods like Dropout [124], Batch Normalization [125] and Layer Normalization [126] have helped reduce the training time by accelerating the convergence of such large models.
- Third, improvements in *self-supervised pre-training* like contrastive learning [28, 29] and masked image/language modeling [31, 32] has led to the generation of the pre-training,

fine-tuning recipe for neural networks. The pre-trained models provide good features for subsequent downstream fine-tuning which leads to gains in terms of improved downstream performance and reduced requirement for task-specific fine-tuning samples.

- Finally, the combination of all these improvements along with the availability of massive pre-training datasets scraped from the internet [33, 34] and large compute facilities [13] has led to the advent of *foundation models* [23].

Foundation models are a family of powerful pre-trained models which can be applied to a wide variety of downstream tasks. What sets them apart from previous state-of-the-art neural networks is the sheer range of downstream tasks on which they have proven to be performant, often requiring very little further training on the tasks themselves. Foundation models are available in modalities like language with Large Language Models like GPT [51], Llama [127] and PaLM [128], vision with models like DINOv2 [129] and SAM [130] as well as multi-modal models like CLIP [52], CoCa [131], DALL-E [132] and Stable Diffusion [133] with wide-spread applications in content understanding, prediction as well as content generation. Collectively, they represent the current state-of-the-art in scaling the abilities of neural networks. However, even with their tremendous capabilities, foundation models are not without shortcomings and can fail in certain scenarios and tasks, thereby leaving room for further improvement. With our motivation to scale modern neural networks, in the second part of the thesis, we study some of these shortcomings of foundation models like CLIP [52] and develop methods to circumvent these failure modes. In what follows, we first provide a brief overview of CLIP following which we describe some of the major problems of multi-modal foundation models which can impede their usage in downstream tasks. In the second part of the thesis, specifically Chapter 6 and Chapter 7, we propose methods that can tackle these problems, thereby aiding the further scaling of foundation models.

2.2.1 CLIP: The Canonical Multi-modal Foundation Model

Contrastive Language Image Pre-training (CLIP) [52] is one of the most popular multi-modal foundation models used in literature. It has two main components: a vision encoder f_{θ_v}

with parameters θ_v and a text encoder f_{θ_t} with parameters θ_t, with the full set of training parameters being $\theta = \{\theta_v, \theta_t\}$. CLIP is pre-trained in a self-supervised manner using a corpus of 400 million image-text pairs scraped from the internet. The pre-training is done using contrastive loss [134] to align the image and text representations in the model's feature space. Let $\mathbf{x}$ represent an input image and $\mathbf{y}$ represent the corresponding input text. Then $f_{\theta_v}(\mathbf{x})$ and $f_{\theta_t}(\mathbf{y})$ are the feature space representations of the image, text pair and $\Phi(\mathbf{x}, \mathbf{y}) = \frac{f_{\theta_v}(\mathbf{x})}{|f_{\theta_v}(\mathbf{x})|} \cdot \frac{f_{\theta_t}(\mathbf{y})}{|f_{\theta_t}(\mathbf{y})|}$, is their cosine similarity, which forms the compatibility function for contrastive loss. Given a batch of inputs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{B}$, the contrastive loss tries to align the representations of the correct image, text pairs in nearby locations of the feature space, maximizing their cosine similarity, and aligns the incorrect pairs in distant locations, minimizing their cosine similarity:

$$\begin{aligned} \mathcal{L}_{\mathbf{x}} &= \frac{1}{B} \sum_{i=1}^{B} \frac{e^{\Phi(\mathbf{x}_i, \mathbf{y}_i)}}{\sum_{j=1}^{B} e^{\Phi(\mathbf{x}_i, \mathbf{y}_j)}} \\ \mathcal{L}_{\mathbf{y}} &= \frac{1}{B} \sum_{i=1}^{B} \frac{e^{\Phi(\mathbf{x}_i, \mathbf{y}_i)}}{\sum_{j=1}^{B} e^{\Phi(\mathbf{x}_j, \mathbf{y}_i)}} \\ \mathcal{L}_{\text{InfoNCE}} &= \frac{1}{2}(\mathcal{L}_{\mathbf{x}} + \mathcal{L}_{\mathbf{y}}). \end{aligned} \tag{2.9}$$

Once the model is pre-trained, there are several ways in which CLIP's encoders can be used to make predictions. The simplest and most popular is *zero-shot* prediction which does not require any further downstream fine-tuning. Given a test image $\mathbf{x}^*$ and a set of K natural language texts $\{\mathbf{y}_1, ..., \mathbf{y}_K\}$ describing K classes, the cosine similarities $[\Phi(\mathbf{x}^*, \mathbf{y}_1), \Phi(\mathbf{x}^*, \mathbf{y}_2), ..., \Phi(\mathbf{x}^*, \mathbf{y}_K)]$ can be used as classification logits for predicting on $\mathbf{x}^*$. Since the classes are provided as free form texts, this method of predicting is popularly known as *open vocabulary prediction*. Although this is the simplest method of predicting using a pre-trained CLIP model, there are several other methods including linear probe [52], prompt tuning [135, 136], adapters [137, 138] and full end-to-end fine-tuning [139, 140] which require varying amounts of exposure to downstream training samples. Due to its popularity, we use CLIP as the model of choice in our study on scaling the abilities of modern neural networks.

2.2.2 Scaling Foundation Models Further

In spite of their tremendous capabilities, foundation models can still encounter problems with certain types of downstream tasks which can bottleneck their usage. First, most pre-training methods, including contrastive learning, use weak-supervision signals at the image level. Image-text models for instance use datasets where each sample has a text description attached to a training image. Thus, at test time, image level prediction tasks like classification or retrieval, as shown in Section 2.2.1, can be easily solvable. However, the same cannot be said for tasks which require localized predictions like object detection [37] or semantic segmentation [38]. These tasks need additional supervision signals like localized bounding box or pixel-level annotations for training. Such annotations are difficult and expensive to obtain, which in turn limits the number of concepts that detection or segmentation models can learn. As an example, the largest classification dataset (i.e., ImageNet-21K) has about 21K classes, whereas the largest openly available semantic segmentation dataset, LVIS [141] has only 1203, merely about 6% of the total number of classes in ImageNet-21K. There does exist research on developing foundation models specifically for detection [142, 143] and segmentation [130, 144–146] tasks. However, they still require expensive annotations for training, or make significant changes to model architecture which needs the models to be trained end-to-end from scratch. ***In Chapter 6, we therefore look into emergent properties in contrastive pre-trained foundation models like CLIP, which can be used to transfer them to localized prediction tasks like semantic segmentation without requiring any expensive annotations or re-training the model end-to-end from scratch.*** Our work therefore brings the scale of pre-trained foundation models to downstream tasks involving localized prediction in vision.

A second well-known problem with the pre-training process for foundation models is that they are a very expensive affair often requiring millions of dollars, industry-scale compute resources and several months of human effort. Even so, foundation models often exhibit gaps in their knowledge. For instance, pre-trained CLIP is known to underperform

in the zero-shot setting on satellite image classification like EuroSAT [147] or lymph node tumor detection like PatchCamelyon [148]. These domains are hypothesized to be under-represented in CLIP's pre-training dataset. Additionally, LLMs like Chat-GPT [51] have knowledge-cutoff dates, and therefore may not be aware of the latest events. While methods like Retrieval Augmented Generation (RAG) [149] exist to circumvent the knowledge-cutoff problem in LLMs, such methods also impose additional overheads like the need to constantly maintain and update a vector database and searching through the database for every input query. To thus fill these knowledge gaps, pre-trained foundation models may require further downstream fine-tuning to raise their performance to state-of-the-art standards. A lot of research has explored fine-tuning methods specifically for foundation models [135, 137, 140, 150–152]. However, the main goal of these methods is often to maximize performance on the downstream task at hand or to make the fine-tuning process more efficient. With the immense cost involved in pre-training a foundation model, an important use-case of such fine-tuning methods should be to also preserve the pre-trained knowledge of foundation models. However, this aspect of fine-tuning has not been well-studied in literature. Thus, ***in Chapter 7, we perform an extensive analysis of the latest fine-tuning methods for multi-modal foundation models like CLIP and find that all such methods lead to the model forgetting its pre-trained knowledge on real-world concepts. We research why this is the case and propose a simple solution to mitigate this forgetting issue.*** While there are few works [153] which explore classic continual learning for foundation models, to the best of our knowledge, our work is the first to study the forgetting problem specifically from the lens of downstream fine-tuning methods. This brings us to the end of our preliminary overview on problems related to scaling foundation models.

3

Calibrating Deep Neural Networks using Focal Loss

Contents

3.1 Introduction 31
3.2 Problem Formulation 33
3.3 What Causes Miscalibration? 35
3.4 Improving Calibration using Focal Loss 36
3.5 Experiments 42
3.6 Conclusion 46

Abstract

Miscalibration – a mismatch between a model's confidence and its correctness – of Deep Neural Networks (DNNs) makes their predictions hard to rely on. Ideally, we want networks to be accurate, calibrated and confident. We show that, as opposed to the standard cross-entropy loss, focal loss [154] allows us to learn models that are already very well calibrated. When combined with temperature scaling, whilst preserving accuracy, it yields state-of-the-art calibrated models. We provide a thorough analysis of the factors causing miscalibration, and use the insights we glean from this to justify the empirically excellent performance of focal loss. To facilitate the use of focal loss in practice, we also provide a principled approach to automatically select the hyperparameter involved in the loss function. We perform extensive experiments on a variety of computer vision and NLP datasets, and with a wide variety of network architectures, and show that our approach achieves state-of-the-art calibration without compromising on accuracy in almost all cases. Code is available at https://github.com/torrvision/focal_calibration.

3.1 Introduction

Deep neural networks have dominated computer vision and machine learning in recent years, and this has led to their widespread deployment in real-world systems [155–159]. However, many current multi-class classification networks in particular are poorly calibrated, in the sense that the probability values that they associate with the class labels they predict overestimate the likelihoods of those class labels being correct in the real world. This is a major problem, since if networks are routinely overconfident, then downstream components cannot trust their predictions. The underlying cause is hypothesised to be that these networks' high capacity leaves them vulnerable to overfitting on the negative log-likelihood (NLL) loss they conventionally use during training [56].

Given the importance of this problem, numerous suggestions for how to address it have been proposed. Much work has been inspired by approaches that were not originally formulated in a deep learning context, such as Platt scaling [70], histogram binning [67], isotonic regression [68], and Bayesian binning and averaging [64, 160]. As deep learning has become more dominant, however, various works have begun to directly target the calibration of deep networks. For example, Guo et al. [56] have popularised a modern variant of Platt scaling known as *temperature scaling*, which works by dividing a network's logits by a scalar $T > 0$ (learnt on a validation subset) prior to performing softmax. Temperature scaling has the desirable property that it can improve the calibration of a network without in any way affecting its accuracy. However, whilst its simplicity and effectiveness have made it a popular network calibration method, it does have downsides. For example, whilst it scales the logits to reduce the network's confidence in incorrect predictions, this also slightly reduces the network's confidence in predictions that were correct [74]. Moreover, it is known that temperature scaling does not calibrate a model under data distribution shift [161].

By contrast, Kumar et. al [74] initially eschew temperature scaling in favour of minimising a differentiable proxy for calibration error at training time, called Maximum Mean Calibration Error (MMCE), although they do later also use temperature scaling as a post-processing step

to obtain better results than cross-entropy followed by temperature scaling [56]. Separately, [63] propose training models on cross-entropy loss with label smoothing instead of one-hot labels, and show that label smoothing has a very favourable effect on model calibration.

In this chapter, we propose a technique for improving network calibration that works by replacing the cross-entropy loss conventionally used when training classification networks with the focal loss proposed by [154]. We observe that unlike cross-entropy, which minimises the KL divergence between the predicted (softmax) distribution and the target distribution (one-hot encoding in classification tasks) over classes, focal loss minimises a regularised KL divergence between these two distributions, which ensures minimisation of the KL divergence whilst *increasing the entropy* of the predicted distribution, thereby preventing the model from becoming overconfident. Since focal loss, as shown in §3.4, is dependent on a hyperparameter, γ, that needs to be cross-validated, we also provide a method for choosing γ automatically for each sample, and show that it outperforms all the baseline models.

The intuition behind using focal loss is to direct the network's attention during training towards samples for which it is currently predicting a low probability for the correct class, since trying to reduce the NLL on samples for which it is already predicting a high probability for the correct class is liable to lead to NLL overfitting, and thereby miscalibration [56]. More formally, we show in §3.4 that focal loss can be seen as *implicitly* regularising the weights of the network during training by causing the gradient norms for confident samples to be lower than they would have been with cross-entropy, which we would expect to reduce overfitting and improve the network's calibration.

Overall, we make the following contributions:

1. In §3.3, we study the link that [56] observed between miscalibration and NLL overfitting in detail, and show that the overfitting is associated with the predicted distributions for misclassified test samples becoming peakier as the optimiser tries to increase the magnitude of the network's weights to reduce the training NLL.

2. In §3.4, we propose the use of focal loss for training better-calibrated networks, and provide both theoretical and empirical justifications for this approach. In addition, we provide a principled method for automatically choosing γ for each sample during training.
3. In §3.5, we show, via experiments on a variety of classification datasets and network architectures, that DNNs trained with focal loss are more calibrated than those trained with cross-entropy loss (both with and without label smoothing), MMCE or Brier loss [162]. Finally, we also make the interesting observation that whilst temperature scaling may not work for detecting out-of-distribution (OoD) samples, our approach can. We show that our approach is better at detecting out-of-distribution samples, taking CIFAR-10 as the in-distribution dataset, and SVHN and CIFAR-10-C as out-of-distribution datasets.

3.2 Problem Formulation

Let $D = \langle(\mathbf{x}_i, y_i)\rangle_{i=1}^{N}$ denote a dataset consisting of N samples from a joint distribution $\mathcal{D}(\mathcal{X}, \mathcal{Y})$, where for each sample i, $\mathbf{x}_i \in \mathcal{X}$ is the input and $y_i \in \mathcal{Y} = \{1, 2, ..., K\}$ is the ground-truth class label. Let $\hat{p}_{i,y} = f_\theta(y|\mathbf{x}_i)$ be the probability that a neural network f with model parameters θ predicts for a class y on a given input $\mathbf{x}_i$. The class that f predicts for $\mathbf{x}_i$ is computed as $\hat{y}_i = \text{argmax}_{y\in\mathcal{Y}}\ \hat{p}_{i,y}$, and the predicted confidence as $\hat{p}_i = \max_{y\in\mathcal{Y}}\ \hat{p}_{i,y}$. The network is said to be *perfectly calibrated* when, for each sample $(\mathbf{x}, y) \in D$, the confidence $\hat{p}$ is equal to the model accuracy $\mathbb{P}(\hat{y} = y|\hat{p})$, i.e. the probability that the predicted class is correct. For instance, of all the samples to which a perfectly calibrated neural network assigns a confidence of 0.8, 80% should be correctly predicted.

A popular metric used to measure model calibration is the *expected calibration error* (ECE) [64], defined as the expected absolute difference between the model's confidence and its accuracy, i.e. $\mathbb{E}_{\hat{p}}\Big[\,|\mathbb{P}(\hat{y} = y|\hat{p}) - \hat{p}|\,\Big]$. Since we only have finite samples, the ECE cannot in practice be computed using this definition. Instead, we divide the interval $[0, 1]$ into M equispaced bins, where the i^{th} bin is the interval $\left(\frac{i-1}{M}, \frac{i}{M}\right]$. Let B_i denote the set of samples with confidences belonging to the i^{th} bin. The accuracy A_i of this bin is computed

as $A_i = \frac{1}{|B_i|} \sum_{j \in B_i} \mathbb{1}(\hat{y}_j = y_j)$, where $\mathbb{1}$ is the indicator function, and $\hat{y}_j$ and y_j are the predicted and ground-truth labels for the j^{th} sample. Similarly, the confidence C_i of the i^{th} bin is computed as $C_i = \frac{1}{|B_i|} \sum_{j \in B_i} \hat{p}_j$, i.e. C_i is the average confidence of all samples in the bin. The ECE can be approximated as a weighted average of the absolute difference between the accuracy and confidence of each bin: $\text{ECE} = \sum_{i=1}^{M} \frac{|B_i|}{N} |A_i - C_i|$.

A similar metric, the *maximum calibration error* (MCE) [64], is defined as the maximum absolute difference between the accuracy and confidence of each bin: $\text{MCE} = \max_{i \in \{1,\dots,M\}} |A_i - C_i|$.

AdaECE: One disadvantage of ECE is the uniform bin width. For a trained model, most of the samples lie within the highest confidence bins, and hence these bins dominate the value of the ECE. We thus also consider another metric, AdaECE (Adaptive ECE), for which bin sizes are calculated so as to evenly distribute samples between bins (similar to the adaptive binning procedure in [163]): $\text{AdaECE} = \sum_{i=1}^{M} \frac{|B_i|}{N} |A_i - C_i|$ s.t. $\forall i, j \cdot |B_i| = |B_j|$.

Classwise-ECE: The ECE metric only considers the probability of the predicted class, without considering the other scores in the softmax distribution. A stronger definition of calibration would require the probabilities of all the classes in the softmax distribution to be calibrated [69, 164–166]. This can be achieved with a simple classwise extension of the ECE metric: $\text{ClasswiseECE} = \frac{1}{K} \sum_{i=1}^{M} \sum_{j=1}^{K} \frac{|B_{i,j}|}{N} |A_{i,j} - C_{i,j}|$, where K is the number of classes, B_{ij} denotes the set of samples from the j^{th} class in the i^{th} bin, $A_{ij} = \frac{1}{|B_{ij}|} \sum_{k \in B_{ij}} \mathbb{1}(j = y_k)$ and $C_{i,j} = \frac{1}{|B_{ij}|} \sum_{k \in B_{ij}} \hat{p}_{kj}$.

A common way of visualising calibration is to use a *reliability plot* [66], which plots the accuracies of the confidence bins as a bar chart (see Appendix Figure A.1.1). For a perfectly calibrated model, the accuracy for each bin matches the confidence, and hence all of the bars lie on the diagonal. By contrast, if most of the bars lie above the diagonal, the model is more accurate than it expects, and is under-confident, and if most of the bars lie below the diagonal, then it is over-confident.

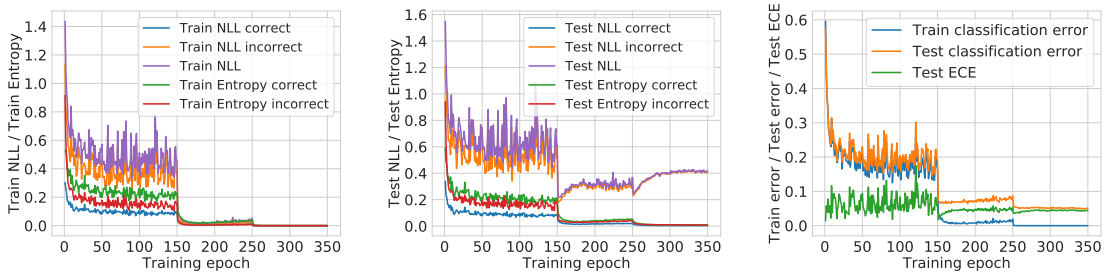
3.3 What Causes Miscalibration?

We now discuss why high-capacity neural networks, despite achieving low classification errors on well-known datasets, tend to be miscalibrated. A key empirical observation made by [56] was that poor calibration of such networks appears to be linked to overfitting on the negative log-likelihood (NLL) during training. In this section, we further inspect this observation to provide new insights.

For the analysis, we train a ResNet-50 network on CIFAR-10 with state-of-the-art performance settings [167]. We use Stochastic Gradient Descent (SGD) with a mini-batch of size 128, momentum of 0.9, and learning rate schedule of $\{0.1, 0.01, 0.001\}$ for the first 150, next 100, and last 100 epochs, respectively. We minimise cross-entropy loss (a.k.a. NLL) $\mathcal{L}_c$, which, in a standard classification context, is $-\log \hat{p}_{i,y_i}$, where $\hat{p}_{i,y_i}$ is the probability assigned by the network to the correct class y_i for the ith sample. Note that the NLL is minimised when for each training sample i, $\hat{p}_{i,y_i} = 1$, whereas the classification error is minimised when $\hat{p}_{i,y_i} > \hat{p}_{i,y}$ for all $y \neq y_i$. This indicates that even when the classification error is 0, the NLL can be positive, and the optimisation algorithm can still try to reduce it to 0 by further increasing the value of $\hat{p}_{i,y_i}$ for each sample (see Appendix A.1).

To study how miscalibration occurs during training, we plot the average NLL for the train and test sets at each training epoch in Figures 3.1(a) and 3.1(b). We also plot the average NLL and the entropy of the softmax distribution produced by the network for the correctly and incorrectly classified samples. In Figure 3.1(c), we plot the classification errors on the train and test sets, along with the test set ECE.

Curse of misclassified samples: Figures 3.1(a) and 3.1(b) show that although the average train NLL (for both correctly and incorrectly classified training samples) broadly decreases throughout training, after the 150^{th} epoch (where the learning rate drops by a factor of 10), there is a marked rise in the average test NLL, indicating that the network starts to overfit on average NLL. This increase in average test NLL is caused only by the incorrectly classified samples, as the average NLL for the correctly classified samples continues to decrease even



(a) **(b)** **(c)**

Figure 3.1: Metrics related to calibration plotted whilst training a ResNet-50 network on CIFAR-10.

after the 150^{th} epoch. We also observe that after epoch 150, the test set ECE rises, indicating that the network is becoming miscalibrated. This corroborates the observation in [56] that miscalibration and NLL overfitting are linked.

Peak at the wrong place: We further observe that the entropies of the softmax distributions for both the correctly and incorrectly classified *test* samples decrease throughout training (in other words, the distributions get peakier). This observation, coupled with the one we made above, indicates that *for the wrongly classified test samples, the network gradually becomes more and more confident about its incorrect predictions*.

Weight magnification: The increase in confidence of the network's predictions can happen if the network increases the norm of its weights W to increase the magnitudes of the logits. In fact, cross-entropy loss is minimised when for each training sample i, $\hat{p}_{i,y_i} = 1$, which is possible only when $||W|| \to \infty$. Cross-entropy loss thus inherently induces this tendency of weight magnification in neural network optimisation. The promising performance of weight decay [56] (regulating the norm of weights) on the calibration of neural networks can perhaps be explained using this. This increase in the network's confidence during training is one of the key causes of miscalibration.

3.4 Improving Calibration using Focal Loss

As discussed in §3.3, overfitting on NLL, which is observed as the network grows more confident on all of its predictions irrespective of their correctness, is strongly related to poor

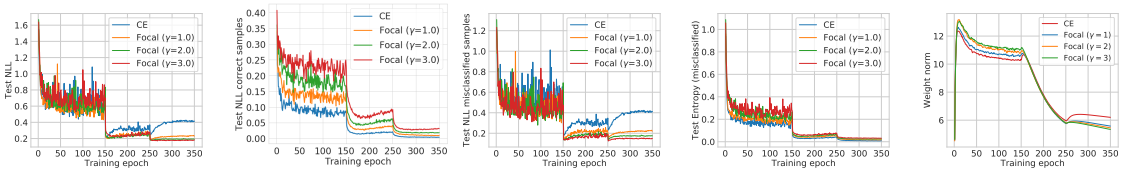
calibration. One cause of this is that the cross-entropy objective minimises the difference between the softmax distribution and the ground-truth one-hot encoding over an entire mini-batch, irrespective of how well a network classifies individual samples in the mini-batch. In this work, we study an alternative loss function, popularly known as *focal loss* [154], that tackles this by weighting loss components generated from individual samples in a mini-batch by how well the model classifies them. For classification tasks where the target distribution is a one-hot encoding, it is defined as $\mathcal{L}_f = -(1-\hat{p}_{i,y_i})^\gamma \log \hat{p}_{i,y_i}$, where γ is a user-defined hyperparameter[1].

Why might focal loss improve calibration? We know that cross-entropy forms an upper bound on the KL-divergence between the target distribution q and the predicted distribution $\hat{p}$, i.e. $\mathcal{L}_c \geq \mathrm{KL}(q||\hat{p})$, so minimising cross-entropy results in minimising $\mathrm{KL}(q||\hat{p})$. Interestingly, a general form of focal loss can be shown to be an upper bound on the regularised KL-divergence, where the regulariser is the negative entropy of the predicted distribution $\hat{p}$, and the regularisation parameter is γ, the hyperparameter of focal loss (a proof of this can be found in Appendix A.2):

$$\mathcal{L}_f \geq \mathrm{KL}(q||\hat{p}) - \gamma\mathbb{H}[\hat{p}]. \tag{3.1}$$

The most interesting property of this upper bound is that it shows that replacing cross-entropy with focal loss has the effect of adding a maximum-entropy regulariser [73] to the implicit minimisation that was previously being performed. In other words, trying to minimise focal loss minimises the KL divergence between $\hat{p}$ and q, whilst simultaneously increasing the entropy of the predicted distribution $\hat{p}$. Note, in the case of ground truth with one-hot encoding, only the component of the entropy of $\hat{p}$ corresponding to the ground-truth index, $\gamma(-\hat{p}_{i,y_i} \log \hat{p}_{i,y_i})$, will be maximised (refer Appendix A.2). Encouraging the predicted distribution to have higher entropy can help avoid the overconfident predictions produced by DNNs (see the 'Peak at the wrong place' paragraph of §3.3), and thereby improve calibration.

[1]We note in passing that unlike cross-entropy loss, focal loss in its general form is not a proper loss function, as minimising it does not always lead to the predicted distribution $\hat{p}$ being equal to the target distribution q (see Appendix A.2 for the relevant definition and a longer discussion). However, when q is a one-hot encoding (as in our case, and for most classification tasks), minimising focal loss does lead to $\hat{p}$ being equal to q.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 3.2: How metrics related to model calibration change whilst training several ResNet-50 networks on CIFAR-10, using either cross-entropy loss, or focal loss with γ set to 1, 2 or 3.

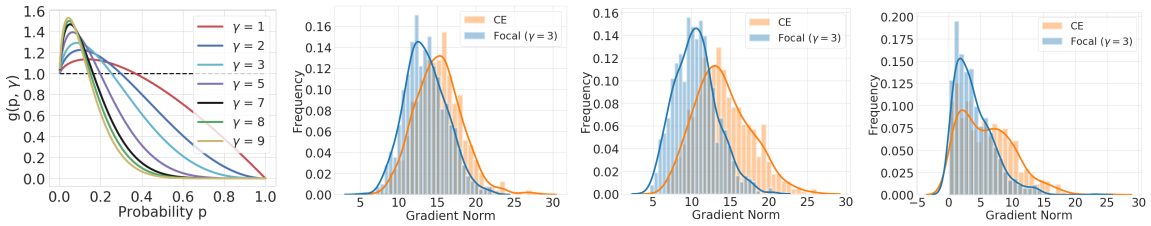
Empirical observations: To analyse the behaviour of neural networks trained on focal loss, we use the same framework as mentioned above, and train four ResNet-50 networks on CIFAR-10, one using cross-entropy loss, and three using focal loss with $\gamma = 1, 2$ and 3. Figure 3.2(a) shows that the test NLL for the cross-entropy model significantly increases towards the end of training (before saturating), whereas the NLLs for the focal loss models remain low. To better understand this, we analyse the behaviour of these models for correctly and incorrectly classified samples. Figure 3.2(b) shows that even though the NLLs for the correctly classified samples broadly-speaking decrease over the course of training for all the models, the NLLs for the focal loss models remain consistently higher than that for the cross-entropy model throughout training, implying that the focal loss models are relatively less confident than the cross-entropy model for samples that they predict correctly. This is important, as we have already discussed that it is overconfidence that normally makes deep neural networks miscalibrated. Figure 3.2(c) shows that in contrast to the cross-entropy model, for which the NLL for misclassified test samples increases significantly after epoch 150, the rise in this value for the focal loss models is much less severe. Additionally, in Figure 3.2(d), we notice that the entropy of the softmax distribution for misclassified test samples is consistently (if marginally) higher for focal loss than for cross-entropy (consistent with Equation 3.1).

Note that from Figure 3.2(a), one may think that applying early stopping when training a model on cross-entropy can provide better calibration scores. However, there is no ideal way of doing early stopping that provides the best calibration error and the best test set

accuracy. For fair comparison, we chose 3 intermediate models for each loss function with the best val set ECE, NLL and accuracy, and observed that: a) for every stopping criterion, focal loss outperforms cross-entropy in both test set accuracy and ECE, b) when using val set ECE as a stopping criterion, the intermediate model for cross-entropy indeed improves its test set ECE, but at the cost of a significantly higher test error. Please refer to Appendix A.10 for more details.

As per §3.3, an increase in the test NLL and a decrease in the test entropy for misclassified samples, along with no corresponding increase in the test NLL for the correctly classified samples, can be interpreted as the network starting to predict softmax distributions for the misclassified samples that are ever more peaky in the wrong place. Notably, our results in Figures 3.2(b), 3.2(c) and 3.2(d) clearly show that this effect is significantly reduced when training with focal loss rather than cross-entropy, leading to a better-calibrated network whose predictions are less peaky in the wrong place.

Theoretical justification: As mentioned previously, once a model trained using cross-entropy reaches high training accuracy, the optimiser may try to further reduce the training NLL by increasing the confidences for the correctly classified samples. It may achieve this by magnifying the network weights to increase the magnitudes of the logits. To verify this hypothesis, we plot the L_2 norm of the weights of the last linear layer for all four networks as a function of the training epoch (see Figure 3.2(e)). Notably, although the norms of the weights for the models trained on focal loss are initially higher than that for the cross-entropy model, *a complete reversal* in the ordering of the weight norms occurs between epochs 150 and 250. In other words, as the networks start to become miscalibrated, the weight norm for the cross-entropy model also starts to become greater than those for the focal loss models. In practice, this is because focal loss, by design, starts to act as a regulariser on the network's weights once the model has gained a certain amount of confidence in its predictions. This behaviour of focal loss can be observed even on a much simpler setup like a linear model (see Appendix A.3). To better understand this, we start by considering the following proposition (proof in Appendix A.4):



(a) **(b)** Epoch 10 **(c)** Epoch 100 **(d)** Epoch 200

Figure 3.3: (a): $g(p, \gamma)$ vs. p and (b-d): histograms of the gradient norms of the last linear layer for both cross-entropy and focal loss.

Proposition 1. *For focal loss* $\mathcal{L}_f$ *and cross-entropy* $\mathcal{L}_c$*, the gradients* $\frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} g(\hat{p}_{i,y_i}, \gamma)$*, where* $g(p, \gamma) = (1-p)^\gamma - \gamma p (1-p)^{\gamma-1} \log(p)$*,* $\gamma \in \mathbb{R}^+$ *is the focal loss hyperparameter, and* $\mathbf{w}$ *denotes the parameters of the last linear layer. Thus* $\left\| \frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} \right\| \leq \left\| \frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} \right\|$ *if* $g(\hat{p}_{i,y_i}, \gamma) \in [0, 1]$.

Proposition 1 shows the relationship between the norms of the gradients of the last linear layer for focal loss and cross-entropy loss, for the same network architecture. Note that this relation depends on a function $g(p, \gamma)$, which we plot in Figure 3.3(a) to understand its behaviour. It is clear that for every γ, there exists a (different) threshold p_0 such that for all $p \in [0, p_0]$, $g(p, \gamma) \geq 1$, and for all $p \in (p_0, 1]$, $g(p, \gamma) < 1$. (For example, for $\gamma = 1$, $p_0 \approx 0.4$.) We use this insight to further explain why focal loss provides implicit weight regularisation.

Implicit weight regularisation: For a network trained using focal loss with a fixed γ, during the initial stages of the training, when $\hat{p}_{i,y_i} \in (0, p_0)$, $g(\hat{p}_{i,y_i}, \gamma) > 1$. This implies that the confidences of the focal loss model's predictions will initially increase faster than they would for cross-entropy. However, as soon as $\hat{p}_{i,y_i}$ crosses the threshold p_0, $g(\hat{p}_{i,y_i}, \gamma)$ falls below 1 and reduces the size of the gradient updates made to the network weights, thereby having a regularising effect on the weights. This is why, in Figure 3.2(e), we find that the weight norms of the models trained with focal loss are initially higher than that for the model trained using cross-entropy. However, as training progresses, we find that the ordering of the weight norms reverses, as focal loss starts regularising the network weights. Moreover, we can draw similar insights from Figures 3.3(b), 3.3(c) and 3.3(d), in which we plot histograms of the gradient norms of the last linear layer (over all samples in the training set) at epochs 10, 100 and 200, respectively. At epoch 10, the gradient norms for

cross-entropy and focal loss are similar, but as training progresses, those for cross-entropy decrease less rapidly than those for focal loss, indicating that the gradient norms for focal loss are consistently lower than those for cross-entropy throughout training.

Finally, observe in Figure 3.3(a) that for higher γ values, the fall in $g(p, \gamma)$ is steeper. We would thus expect a greater weight regularisation effect for models that use higher values of γ. This explains why, of the three models that we trained using focal loss, the one with $\gamma = 3$ outperforms (in terms of calibration) the one with $\gamma = 2$, which in turn outperforms the model with $\gamma = 1$. Based on this observation, one might think that, in general, a higher value of gamma would lead to a more calibrated model. However, this is not the case, as we notice from Figure 3.3(a) that for $\gamma \geq 7$, $g(p, \gamma)$ reduces to nearly 0 for a relatively low value of p (around 0.5). As a result, using values of γ that are too high will cause the gradients to die (i.e. reduce to nearly 0) early, at a point at which the network's predictions remain ambiguous, thereby causing the training process to fail.

How to choose γ: As discussed, focal loss provides implicit entropy and weight regularisation, which heavily depend on the value of γ. Finding an appropriate γ is normally done using cross-validation. Also, traditionally, γ is fixed for all samples in the dataset. However, as shown, the regularisation effect for a sample i depends on $\hat{p}_{i,y_i}$, i.e. the predicted probability for the ground truth label for the sample. It thus makes sense to choose γ for each sample based on the value of $\hat{p}_{i,y_i}$. To this end, we provide Proposition 2 (proof in Appendix A.4), which we use to find a solution to this problem:

Proposition 2. *Given a p_0, for $1 \geq p \geq p_0 > 0$, $g(p, \gamma) \leq 1$ for all $\gamma \geq \gamma^* = \frac{a}{b} + \frac{1}{\log a} W_{-1}\Big(- \frac{a^{(1-a/b)}}{b} \log a \Big)$, where $a = 1 - p_0$, $b = p_0 \log p_0$, and W_{-1} is the Lambert-W function [168]. Moreover, for $p \geq p_0 > 0$ and $\gamma \geq \gamma^*$, the equality $g(p, \gamma) = 1$ holds only for $p = p_0$ and $\gamma = \gamma^*$.*

It is worth noting that there exist multiple values of γ where $g(p, \gamma) \leq 1$ for all $p \geq p_0$. For a given p_0, Proposition 2 allows us to compute γ s.t. (i) $g(p_0, \gamma) = 1$; (ii) $g(p, \gamma) \geq 1$ for $p \in [0, p_0)$; and (iii) $g(p, \gamma) < 1$ for $p \in (p_0, 1]$. This allows us to control the magnitude

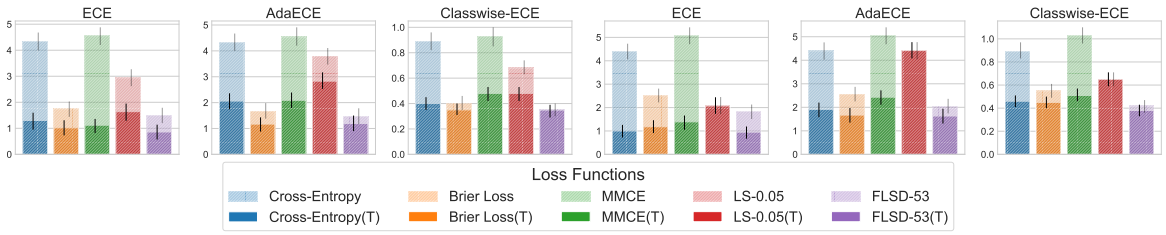
of the gradients for a particular sample i based on the current value of $\hat{p}_{i,y_i}$, and gives us a way of obtaining an informed value of γ for each sample. For instance, a reasonable policy might be to choose γ s.t. $g(\hat{p}_{i,y_i}, \gamma) > 1$ if $\hat{p}_{i,y_i}$ is small (say less than 0.25), and $g(\hat{p}_{i,y_i}, \gamma) < 1$ otherwise. Such a policy will have the effect of making the weight updates larger for samples having a low predicted probability for the correct class and smaller for samples with a relatively higher predicted probability for the correct class.

Following the aforementioned arguments, we choose a threshold p_0 of 0.25, and use Proposition 2 to obtain a γ policy such that $g(p, \gamma)$ is observably greater than 1 for $p \in [0, 0.25)$ and $g(p, \gamma) < 1$ for $p \in (0.25, 1]$. In particular, we use the following schedule: if $\hat{p}_{i,y_i} \in [0, 0.2)$, then $\gamma = 5$, otherwise $\gamma = 3$ (note that $g(0.2, 5) \approx 1$ and $g(0.25, 3) \approx 1$: see Figure 3.3(a)). We find this γ policy to perform consistently well across multiple classification datasets and network architectures. Having said that, one can calculate multiple such schedules for γ following Proposition 2, using the intuition of having a relatively high γ for low values of $\hat{p}_{i,y_i}$ and a relatively low γ for high values of $\hat{p}_{i,y_i}$.

3.5 Experiments

We conduct image and document classification experiments to test the performance of focal loss. For the former, we use CIFAR-10/100 [89] and Tiny-ImageNet [12] , and train ResNet-50, ResNet-110 [15], Wide-ResNet-26-10 [169] and DenseNet-121 [170] models, and for the latter, we use 20 Newsgroups [171] and Stanford Sentiment Treebank (SST) [172] datasets and train Global Pooling CNN [173] and Tree-LSTM [174] models. Further details on the datasets and training can be found in Appendix A.5.

Baselines Along with cross-entropy loss, we compare our method against the following baselines: a) *MMCE* (Maximum Mean Calibration Error) [74], a continuous and differentiable proxy for calibration error that is normally used as a regulariser alongside cross-entropy, b) *Brier loss* [162], the squared error between the predicted softmax vector and the one-hot ground truth encoding (Brier loss is an important baseline as it can be decomposed into



(a) ResNet-50 on CIFAR-10 (b) ResNet-110 on CIFAR-10

Figure 3.4: Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE, computed for ResNet-50 (first 3 figures) and ResNet-110 (last 3 figures) on CIFAR-10.

calibration and refinement [175]), and c) *Label smoothing* [63] (LS): given a one-hot ground-truth distribution $\mathbf{q}$ and a smoothing factor α (hyperparameter), the smoothed vector $\mathbf{s}$ is obtained as $\mathbf{s}_i = (1-\alpha)\mathbf{q}_i + \alpha(1-\mathbf{q}_i)/(K-1)$, where $\mathbf{s}_i$ and $\mathbf{q}_i$ denote the i^{th} elements of $\mathbf{s}$ and $\mathbf{q}$ respectively, and K is the number of classes. Instead of $\mathbf{q}$, $\mathbf{s}$ is treated as the ground truth. We train models using $\alpha = 0.05$ and $\alpha = 0.1$, but find $\alpha = 0.05$ to perform better. We thus report the results obtained from LS-0.05 with $\alpha = 0.05$.

Focal Loss: As mentioned in §3.4, our proposed approach is the sample-dependent schedule FLSD-53 ($\gamma = 5$ for $\hat{p}_y \in [0, 0.2)$, and $\gamma = 3$ for $\hat{p}_y \in [0.2, 1]$), which we find to perform well across most classification datasets and network architectures. In addition, we also train other focal loss baselines, including ones with γ fixed to $1, 2$ and 3, and also ones that have a training epoch-dependent schedule for γ. Among the focal loss models trained with a fixed γ, using validation set we find $\gamma = 3$ (FL-3) to perform the best. Details of all these approaches can be found in Appendix A.6.

Temperature Scaling: In order to compute the optimal temperature, we use two different methods: (a) learning the temperature by minimising val set NLL, and (b) performing grid search over temperatures between 0 and 10, with a step of 0.1, and finding the one that minimises val set ECE. We find the second approach to produce *stronger baselines* and report results obtained using this approach.

Performance Gains: We report ECE% (computed using 15 bins) along with optimal temperatures in Table 3.1, and test set error in Table 3.2. We report the other calibration

Dataset	Model	Cross-Entropy		Brier Loss		MMCE		LS-0.05		FL-3 (Ours)		FLSD-53 (Ours)	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	17.52	3.42(2.1)	6.52	3.64(1.1)	15.32	2.38(1.8)	7.81	4.01(1.1)	5.13	**1.97(1.1)**	**4.5**	2.0(1.1)
	ResNet-110	19.05	4.43(2.3)	**7.88**	4.65(1.2)	19.14	**3.86(2.3)**	11.02	5.89(1.1)	8.64	3.95(1.2)	8.56	4.12(1.2)
	Wide-ResNet-26-10	15.33	2.88(2.2)	4.31	2.7(1.1)	13.17	4.37(1.9)	4.84	4.84(1)	**2.13**	2.13(1)	3.03	**1.64(1.1)**
	DenseNet-121	20.98	4.27(2.3)	5.17	2.29(1.1)	19.13	3.06(2.1)	12.89	7.52(1.2)	4.15	**1.25(1.1)**	**3.73**	1.31(1.1)
CIFAR-10	ResNet-50	4.35	1.35(2.5)	1.82	1.08(1.1)	4.56	1.19(2.6)	2.96	1.67(0.9)	**1.48**	1.42(1.1)	1.55	**0.95(1.1)**
	ResNet-110	4.41	1.09(2.8)	2.56	1.25(1.2)	5.08	1.42(2.8)	2.09	2.09(1)	**1.55**	**1.02(1.1)**	1.87	1.07(1.1)
	Wide-ResNet-26-10	3.23	0.92(2.2)	**1.25**	1.25(1)	3.29	0.86(2.2)	4.26	1.84(0.8)	1.69	0.97(0.9)	1.56	**0.84(0.9)**
	DenseNet-121	4.52	1.31(2.4)	1.53	1.53(1)	5.1	1.61(2.5)	1.88	1.82(0.9)	1.32	1.26(0.9)	**1.22**	**1.22(1)**
Tiny-ImageNet	ResNet-50	15.32	5.48(1.4)	4.44	4.13(0.9)	13.01	5.55(1.3)	15.23	6.51(0.7)	1.87	1.87(1)	**1.76**	**1.76(1)**
20 Newsgroups	Global Pooling CNN	17.92	2.39(3.4)	13.58	3.22(2.3)	15.48	6.78(2.2)	**4.79**	2.54(1.1)	8.67	3.51(1.5)	6.92	**2.19(1.5)**
SST Binary	Tree-LSTM	7.37	2.62(1.8)	9.01	2.79(2.5)	5.03	4.02(1.5)	**4.84**	4.11(1.2)	16.05	1.78(0.5)	9.19	**1.83(0.7)**

Table 3.1: ECE $(\%)$ computed for different approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets. $T \approx 1$ indicates innately calibrated model.

Dataset	Model	Cross-Entropy	Brier Loss	MMCE	LS-0.05	FL-3 (Ours)	FLSD-53 (Ours)
CIFAR-100	ResNet-50	23.3	23.39	23.2	23.43	22.75	23.22
	ResNet-110	22.73	25.1	23.07	23.43	22.92	22.51
	Wide-ResNet-26-10	20.7	20.59	20.73	21.19	19.69	20.11
	DenseNet-121	24.52	23.75	24.0	24.05	23.25	22.67
CIFAR-10	ResNet-50	4.95	5.0	4.99	5.29	5.25	4.98
	ResNet-110	4.89	5.48	5.4	5.52	5.08	5.42
	Wide-ResNet-26-10	3.86	4.08	3.91	4.2	4.13	4.01
	DenseNet-121	5.0	5.11	5.41	5.09	5.33	5.46
Tiny-ImageNet	ResNet-50	49.81	53.2	51.31	47.12	49.69	49.06
20 Newsgroups	Global Pooling CNN	26.68	27.06	27.23	26.03	29.26	27.98
SST Binary	Tree-LSTM	12.85	12.85	11.86	13.23	12.19	12.8

Table 3.2: Test set error $(\%)$ computed for different approaches.

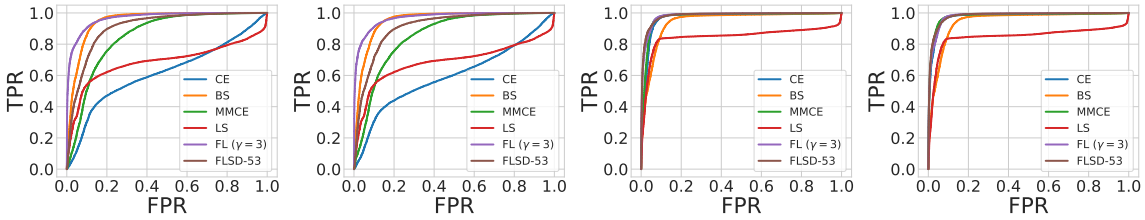
Dataset	Model	Cross-Entropy		Brier Loss		MMCE		LS-0.05		FL-3 (Ours)		FLSD-53 (Ours)	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10/SVHN	ResNet-110	61.71	59.66	94.80	95.13	85.31	85.39	68.68	68.68	**96.74**	**96.92**	90.83	90.97
	Wide-ResNet-26-10	96.82	97.62	94.51	94.51	97.35	97.95	84.63	84.66	98.19	98.05	**98.29**	**98.20**
CIFAR-10/CIFAR-10-C	ResNet-110	77.53	75.16	84.09	83.86	71.96	70.02	72.17	72.18	82.27	82.18	**85.05**	**84.70**
	Wide-ResNet-26-10	81.06	80.68	85.03	85.03	82.17	81.72	71.10	71.16	82.17	81.86	**87.05**	**87.30**

Table 3.3: AUROC $(\%)$ computed for models trained on CIFAR-10 (in-distribution), and using SVHN and CIFAR-10-C (Gaussian Noise corruption with severity level 5) respectively as the OoD datasets.

scores (AdaECE, Classwise-ECE, MCE and NLL) in Appendix A.6. Firstly, for all dataset-network pairs, we obtain very competitive classification accuracies (shown in Table 3.2). Secondly, *it is clear from Table 3.1, and Tables A.6.1 and A.6.2 in the appendix, that focal loss with sample-dependent γ and with $\gamma = 3$ outperform all the baselines: cross-entropy, label smoothing, Brier loss and MMCE.* They broadly produce the lowest calibration errors

both before and after temperature scaling. This observation is particularly encouraging, as it also indicates that a principled method of obtaining values of γ for focal loss can produce a very calibrated model, with no need to use validation set for tuning γ. As shown in Figure 3.4, we also compute 90% confidence intervals for ECE, AdaECE and Classwise-ECE using 1000 bootstrap samples following [166], and using ResNet-50/110 trained on CIFAR-10 (see Appendix A.7 for more results). Note that FLSD-53 produces the lowest calibration errors in general, and the difference in the metric values between FLSD-53 and other approaches (except Brier loss) is mostly statistically significant (i.e., confidence intervals don't overlap), especially before temperature scaling. In addition to the lower calibration errors, there are other advantages of focal loss as well, which we explore next.

More advantages of focal loss: *Behaviour on Out-of-Distribution (OoD) data:* A perfectly calibrated model should have low confidence whenever it misclassifies, including when it encounters data which is OoD [76]. Although temperature scaling calibrates a model under the i.i.d. assumption, it is known to fail under distributional shift [161]. Since focal loss has implicit regularisation effects on the network (see §3.4), we investigate if it helps to learn representations that are more robust to OoD data. To do this, we use ResNet-110 and Wide-ResNet-26-10 trained on CIFAR-10 and consider the SVHN [90] test set and CIFAR-10-C [85] with Gaussian noise corruption at severity 5 as OoD data. We use the entropy of the softmax distribution as the measure of confidence or uncertainty, and report the corresponding AUROC scores both before and after temperature scaling in Table 3.3. For both SVHN and CIFAR-10-C (using Gaussian noise), models trained on focal loss clearly obtain the highest AUROC scores. *Note that Focal loss even without temperature scaling performs better than other methods with temperature scaling.* We also present the ROC plots pre and post temperature scaling for models trained on CIFAR-10 and tested on SVHN in Figure 3.5. Thus, it is quite encouraging to note that models trained on focal loss are not only better calibrated under the i.i.d. assumption, but also seem to perform better than other competitive loss functions when we try shifting the distribution from CIFAR-10 to SVHN or CIFAR-10-C (pre and post temperature scaling).



(a) ResNet-110 (pre-T) **(b)** ResNet-110 (post-T) **(c)** Wide-ResNet (pre-T) **(d)** Wide-ResNet (post-T)

Figure 3.5: ROC plots obtained from ResNet-110 and Wide-ResNet-26-10 architectures trained on CIFAR-10 (in-distribution) and tested on SVHN (OoD), both pre and post temperature scaling.

Confident and Calibrated Models: It is worth noting that focal loss with sample-dependent γ has optimal temperatures that are very close to 1, mostly lying between 0.9 and 1.1 (see Table 3.1). This property is shown by the Brier loss and label smoothing models as well, albeit with worse calibration errors. By contrast, the temperatures for cross-entropy and MMCE models are significantly higher, with values lying between 2.0 and 2.8. An optimal temperature close to 1 indicates that the model is innately calibrated, and cannot be made significantly more calibrated by temperature scaling. In fact, a temperature much greater than 1 can make a model underconfident in general, as it is applied irrespective of the correctness of model outputs. We observe this empirically for ResNet-50 and ResNet-110 trained on CIFAR-10. Although models trained with cross-entropy have much higher confidence before temperature scaling than those trained with focal loss, after temperature scaling, focal loss models are significantly more confident in their predictions. We provide quantitative and qualitative empirical results to support this claim in Appendix A.8.

3.6 Conclusion

In this chapter, we have studied the properties of focal loss, an alternative loss function that can yield classification networks that are more naturally calibrated than those trained using the conventional cross-entropy loss, while maintaining accuracy. In particular, we show in §3.4 that focal loss implicitly maximises entropy while minimising the KL divergence between the predicted and the target distributions. We also show that, because of its

design, it naturally regularises the weights of a network during training, reducing NLL overfitting and thereby improving calibration. Furthermore, we empirically observe that models trained using focal loss are not only better calibrated under i.i.d. assumptions, but can also be better at detecting OoD samples which we show by taking CIFAR-10 as the in-distribution dataset and SVHN and CIFAR-10-C as out-of-distribution datasets, something which temperature scaling fails to achieve.

4

Deep Deterministic Uncertainty: A New Simple Baseline

Contents

4.1 Introduction 50
4.2 Background 54
4.3 Deep Deterministic Uncertainty 57
4.4 Experiments 59
4.4.1 Active Learning 60
4.4.2 OoD Detection 62
4.4.3 Semantic Segmentation 65
4.5 Additional Insights 67
4.6 Conclusion 71

Abstract

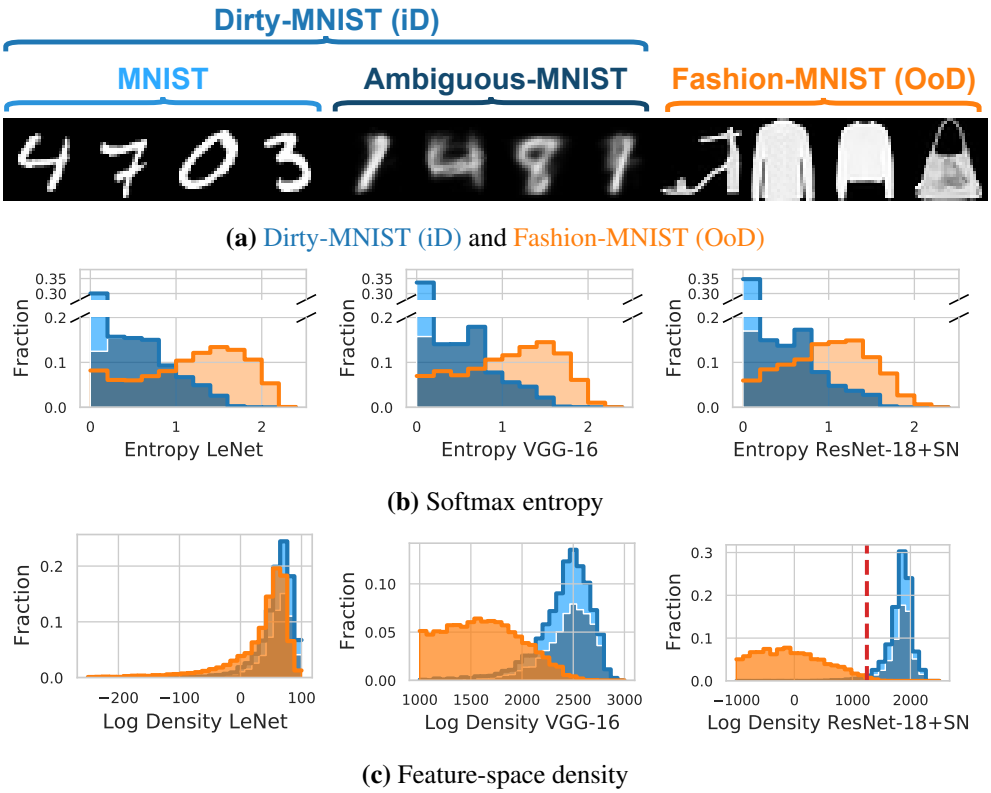
Reliable uncertainty from deterministic single-forward pass models is sought after because conventional methods of uncertainty quantification are computationally expensive.We take two complex single-forward-pass uncertainty approaches, DUQ and SNGP, and examine whether they mainly rely on a well-regularised feature space. Crucially, without using their more complex methods for estimating uncertainty, we find that a single softmax neural net with such a regularised feature-space, achieved via residual connections and spectral normalisation, outperforms DUQ and SNGP's epistemic uncertainty predictions using simple Gaussian Discriminant Analysis post-training as a separate feature-space density estimator—without fine-tuning on OoD data, feature ensembling, or input pre-processing. Our conceptually simple Deep Deterministic Uncertainty (DDU) baseline can also be used to disentangle aleatoric and epistemic uncertainty and performs as well as Deep Ensembles, the state-of-the art for uncertainty prediction, on several OoD benchmarks (CIFAR-10/100 vs SVHN/Tiny-ImageNet, ImageNet vs ImageNet-O), active learning settings across different model architectures, as well as in large scale vision tasks like semantic segmentation, while being computationally cheaper.

4.1 Introduction

Two types of uncertainty are often of interest in ML: *epistemic uncertainty*, which is inherent to the model, caused by a lack of training data or model misspecification, and hence reducible with more data, and *aleatoric uncertainty*, caused by inherent noise, ambiguity or missing features in the data, and hence often irreducible with more data [176]. Disentangling these two is critical for applications such as active learning [79] or detection of out-of-distribution (OoD) samples [177]: in active learning, we wish to avoid inputs with high aleatoric but low epistemic uncertainty, and in OoD detection, we wish to avoid mistaking ambiguous in-distribution (iD) examples as OoD. Disentangling uncertainties is particularly challenging for noisy and ambiguous datasets found in safety-critical applications like autonomous driving [46] and medical diagnosis [44, 45].

Related Work: Most well-known methods of uncertainty quantification in deep learning [49, 109, 111, 113, 178] require multiple forward passes at test time. Amongst these, Deep Ensembles have generally performed best in uncertainty prediction [161], but their significant memory and compute burden at training and test time hinders their adoption in real-life and mobile applications. Consequently, there has been an increased interest in uncertainty quantification using deterministic single forward-pass neural networks which have a smaller footprint and lower latency. Among these approaches, [179] uses Mahalanobis distances to quantify uncertainty by fitting a class-wise Gaussian distribution (with shared covariance matrices) on the feature space of a pre-trained ResNet encoder. They do not consider the structure of the underlying feature-space however, which might explain why their competitive results require input perturbations, the ensembling of OoD metrics over multiple layers, and fine-tuning on OoD hold-out data.

DUQ & SNGP: Two popular works in single forward-pass uncertainty, DUQ [57] and SNGP [58], propose distance-aware output layers, in the form of RBFs (radial basis functions) or GPs (Gaussian processes), and introduce additional inductive biases in the feature extractor using a Jacobian penalty [180] or spectral normalisation [181], respectively, which encourage



(a) Dirty-MNIST (iD) and Fashion-MNIST (OoD)

(b) Softmax entropy

(c) Feature-space density

Figure 4.1: *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and Fashion-MNIST (OoD) **(a)** requires using* softmax entropy *(**b**) and* feature-space density (GMM) *(**c**) with a well-regularised feature space (*ResNet-18+SN *vs* LeNet & VGG-16 *without smoothness & sensitivity).* **(b):** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples (stacked histogram). However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **(c):** With a well-regularised feature space (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without such feature space (LeNet & VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap.

smoothness and sensitivity in the latent space. These methods perform well and are almost competitive with Deep Ensembles on OoD benchmarks. However, they require training to be changed substantially, and introduce additional hyper-parameters due to the specialised output layers used at training. Furthermore, it is not obvious how DUQ and SNGP can be used to disentangle aleatoric and epistemic uncertainty. Particularly, in DUQ, the feature representation of an ambiguous data point, high on aleatoric uncertainty, will be in between two centroids, but due to the exponentiated distance of the RBF kernel (see eq (1) in [57]),

which is used as the distance measure in DUQ, it may seem far from both and thus may have uncertainty similar to epistemically uncertain data points which are far from all centroids. In SNGP, the predictive variance is computed using a mean-field approximation of the softmax likelihood, which cannot be disentangled. The variance can also be computed using MC samples of the softmax likelihood which, in theory, can allow disentangling uncertainties (see Equation (4.1)), but requires modelling the covariance between the classes, which is not the case in SNGP. We provide a more extensive review of related work in §B.1.

Contributions: Firstly, we investigate the question whether complex methods to estimate uncertainty like in DUQ and SNGP are necessary beyond feature-space regularisation that encourages bi-Lipschitzness. When we use spectral normalisation like SNGP does, the short answer is an empirical no. Indeed, with a well-regularised feature space using spectral normalisation, we find that simply performing GDA (Gaussian Discriminant Analysis) *after training* as feature-space density estimator can reliably capture epistemic uncertainty. However, unlike [179], which does not place any constraints on the feature space, we do not require training on "OoD" hold-out data, feature ensembling, and input pre-processing to obtain good performance (see Table 4.1). This results in a conceptually simpler method. Moreover, we find that using a separate covariance matrix for each class improves OoD detection performance as compared to a shared covariance matrix like in [179]. **Secondly**, we investigate how to disentangle aleatoric and epistemic uncertainty which is not directly addressed in prior works: DUQ and SNGP. As we only perform GDA after training, the original softmax layer is trained using cross-entropy as a proper scoring rule [182] and can be temperature-scaled to provide good in-distribution calibration and aleatoric uncertainty. Thus, the combination of using GDA for epistemic uncertainty and the softmax predictive distribution for aleatoric uncertainty after training with feature-space regularisation, e.g. residual connections with spectral normalisation, provides a simple baseline which we call *Deep Deterministic Uncertainty (DDU)*. DDU outperforms regular softmax neural networks, as illustrated in Figure 4.1. Furthermore, DDU is competitive with Deep Ensembles [49] and outperforms SNGP and DUQ [57, 58], with no changes to the model architecture beyond

spectral normalisation, in several OoD benchmarks and active learning settings. **Finally**, using DeepLab-v3+ [183] on Pascal VOC 2012 [184], we show that DDU improves upon two classic uncertainty methods: MC Dropout [109] and Deep Ensembles, popularly used on the task of semantic segmentation, while being significantly faster to compute.

Additional Insights: Beyond the above contributions, we also provide additional insights on potential pitfalls for practitioners, which informed the design of DDU. **Firstly**, predictive entropy confounds aleatoric and epistemic uncertainty (Figure 4.1b). This can be an issue in active learning in particular. Yet, this issue is often not visible for standard benchmark datasets without aleatoric noise. To examine this failure in more detail, we introduce a new dataset, Dirty-MNIST, which showcases the issue more clearly than artificially curated datasets like MNIST or CIFAR-10. *Dirty-MNIST* is a modified version of MNIST [14] with additional ambiguous digits (Ambiguous-MNIST) having multiple plausible labels and thus higher aleatoric uncertainty (Figure 4.1a). **Secondly**, we observe that the softmax entropy of a deterministic model trained with maximum likelihood, while being high for ambiguous points (i.e., with high aleatoric uncertainty), might not be consistent for points with high epistemic uncertainty, i.e. the softmax entropy for an OoD sample might be low, high or anything in between for different models trained on the same data (Figure 4.1b). **Thirdly**, we note that feature-space regularisation [58] is crucial for the estimation of epistemic uncertainty[1]. Without such regularisation feature-space density alone might not separate iD from OoD data, possibly explaining the limited empirical success of previous approaches which attempt to use feature-space density [186]. This can be seen in Figure 4.1c where the feature-space density of a VGG-16 or LeNet model are not able to differentiate iD Dirty-MNIST from OoD Fashion-MNIST while a ResNet-18 with spectral normalisation can do so better.

Scope: Our focus is on obtaining a well-regularised feature space using spectral normalisation in model architectures with residual connections, following [58]. Note that unsupervised methods using contrastive learning [187] might also obtain such a feature space by training on

[1] [185] argue for softmax confidence and entropy in their paper, yet feature-space density performs better in their experiments, too.

very large datasets, but training on them can be very expensive [121]. We only use GDA for estimating the feature-space density as it is straight-forward to implement and does not require performing expectation maximization or variational inference like other density estimators. normalising flows [188] or other more complex density estimators might provide even better density estimates, of course. Yet despite its simplicity, GDA is already sufficient to outperform other more complex approaches and obtain good results. As the amount of training data available grows and feature extractors improve, the quality of feature representations might improve as well. The underlying motivation of this chapter is that simple approaches will remain more applicable than more complex ones as our empirical results suggest.

4.2 Background

In this section, we review relevant concepts for quantifying uncertainty.

Epistemic Uncertainty at point x is a quantity which is high for a previously unseen x, and decreases when x is added to the training set and the model is updated [78]. This conforms with using mutual information in Bayesian models and deep ensembles [80] and feature-space density in deterministic models as surrogates for epistemic uncertainty [186] as we examine below (see Figure 4.2a and §B.6.8).

Aleatoric Uncertainty at point x is a quantity which is high for ambiguous or noisy samples [78]. It does not decrease with more data (see Figure 4.2b). Note that aleatoric uncertainty is only meaningful in-distribution, as, by definition, it quantifies the level of ambiguity between the different classes which might be observed[2].

Bayesian Models [101, 102] provide a principled way of measuring uncertainty. Starting with a prior distribution $p(\omega)$ over model parameters ω, they infer a posterior $p(\omega|\mathcal{D})$, given the training data $\mathcal{D}$. The predictive distribution $p(y|x, \mathcal{D})$ for a given input x is computed via marginalisation over the posterior: $p(y|x, \mathcal{D}) = \mathbb{E}_{\omega \sim p(\omega|\mathcal{D})}[p(y|x, \omega)]$. Its predictive entropy $\mathbb{H}[Y|x, \mathcal{D}]$ upper-bounds the epistemic uncertainty, where epistemic uncertainty

[2] If the probability of observing x under the data generating distribution is zero, $p(y|x) = \frac{p(x,y)}{p(x)}$, and hence, the entropy as a measure of aleatoric uncertainty, is not defined.

is quantified as the mutual information $\mathbb{I}[Y;\omega|x,\mathcal{D}]$ (*expected information gain*) between parameters ω and output y [48, 189]:

$$\underbrace{\mathbb{H}[Y|x,\mathcal{D}]}_{\text{predictive}} = \underbrace{\mathbb{I}[Y;\omega|x,\mathcal{D}]}_{\text{epistemic}} + \underbrace{\mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})}[\mathbb{H}[Y|x,\omega]]}_{\text{aleatoric (for iD } x)}. \tag{4.1}$$

Predictive uncertainty will be high whenever either epistemic uncertainty or aleatoric uncertainty is high. However, the intractability of exact Bayesian inference in deep learning has led to the development of approximate inference methods [109, 111, 190, 191]. In practice, however, these methods are either unable to scale to large datasets and model architectures, suffer from low uncertainty quality, or require expensive Monte-Carlo sampling.

Deep Ensembles [49] average the outputs of an ensemble of neural networks. Uncertainty is estimated as the entropy of the averaged softmax outputs. Despite the additional computational overhead at training and test time, Deep Ensembles along with recent extensions [113, 178, 189] form the state-of-the-art in uncertainty quantification in deep learning.

Deterministic Models produce a softmax distribution $p(y|x,\omega)$, and commonly either the *softmax confidence* $\max_c p(y = c|x,\omega)$ or the *softmax entropy* $\mathbb{H}[Y|x,\omega]$ are used as a measure of uncertainty [177]. Popular approaches to improve these metrics include pre-processing of inputs and post-hoc calibration methods [56, 192], alternative objective functions [193, 194], and exposure to outliers [98]. However, these methods are known to suffer from shortcomings like failing under distribution shift [161], requiring significant changes to the training setup, or assuming the availability of OoD samples during training.

Feature-Space Distances [57, 58, 179] and **Feature-Space Density** [186, 195] offer a different approach for estimating uncertainty in deterministic models. Following the definition of epistemic uncertainty above, it decreases when previously unseen samples are added to the training set. Feature-space distance and density methods realise this by estimating distance or density, respectively, to training data in the feature space (see again Figure 4.2a). A previously unseen point with high distance (low density), once added to the training data, will have low distance (high density). Hence, they can be used as a proxy

for epistemic uncertainty, under important assumptions about the feature space as detailed below. None of these methods, however, is competitive with Deep Ensembles, in uncertainty quantification, potentially for the reasons discussed next.

Feature Collapse [57] is why distance and density estimation in the feature space may fail to capture epistemic uncertainty: feature extractors might map the features of OoD inputs to iD regions in the feature space [196].

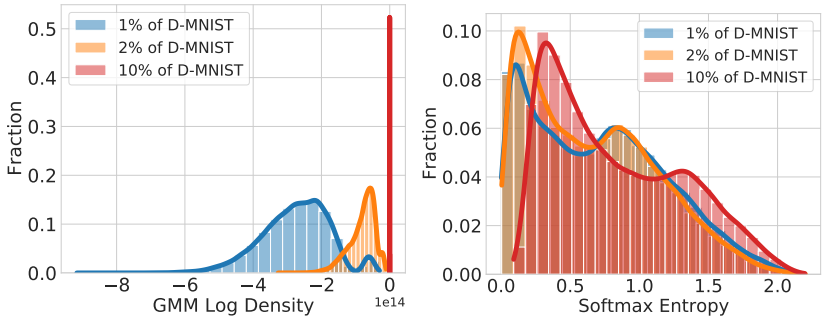
Smoothness & Sensitivity can be encouraged to prevent feature collapse by subjecting the feature extractor f_θ, with parameters θ to a *bi-Lipschitz constraint*:

$$K_L \; d_I(\mathrm{x}_1, \mathrm{x}_2) \leq d_F(f_\theta(\mathrm{x}_1), f_\theta(\mathrm{x}_2)) \leq K_U \; d_I(\mathrm{x}_1, \mathrm{x}_2), \tag{4.2}$$

for all inputs, x_1 and x_2, where d_I and d_F denote metrics for the input and feature space respectively, and K_L and K_U the lower and upper Lipschitz constants [58]. The lower bound ensures *sensitivity* to distances in the input space, and the upper bound ensures *smoothness* in the features, preventing them from becoming too sensitive to input variations, which, otherwise, can lead to poor generalisation and loss of robustness [57].

Lipschitz functions have an interesting property whereby if two functions f and g are Lipschitz with constants K_f and K_g, then their composition $f \circ g$ is Lipschitz with constant $K_f K_g$ [197]. As most activation functions in neural networks are 1-Lipschitz, this property helps design methods which can constrain Lipschitzness of the whole model by ensuring that learnable weight matrices in neural networks are individually Lipschitz constrained. Two popular methods which directly target encouraging bi-Lipschitzness in modern convolutional neural networks are: **i)** gradient penalty, by applying a two-sided penalty to the L2 norm of the Jacobian [180], and **ii)** spectral normalisation [181] in models with residual connections, like ResNets [15]. Spectral normalisation in particular, works by normalising each learnable weight matrix in the network $\mathbf{W}$ by its spectral norm $||\mathbf{W}||_2$, i.e., its largest singular value:

$$\hat{\mathbf{W}} = \frac{\mathbf{W}}{||\mathbf{W}||_2}. \tag{4.3}$$



(a) Epistemic **(b)** Aleatoric

Figure 4.2: *Epistemic and aleatoric uncertainty of ResNet-18+SN models trained on increasingly large subsets of Dirty-MNIST.* The feature-space density increases while the softmax entropy stays roughly the same, consistent with epistemic and aleatoric uncertainty being reducible and irreducible with more data, respectively. See §B.6.8 for a discussion on this.

where the spectral norm is computed using the power iteration method. Since the neural network is a composition of several functions, ensuring that the learnable weights have an upper bound Lipschitz constant of 1 can be used to in turn upper bound the Lipschitz constant of the whole model, thereby encouraging *smoothness* in the feature space. On the other hand, the presence of skip connections in residual architectures like ResNet ensures that residual blocks have a lower bound Lipschitz constant of 1 [198]. The authors in [199] provide an in-depth analysis which supports that spectral normalisation on residual networks can lead to bi-Lipschitzness. Compared to the Jacobian gradient penalty [57], spectral normalisation has also been shown to be significantly faster with comparatively more stable training dynamics.

4.3 Deep Deterministic Uncertainty

We discuss the primary components of DDU in this section.

Ensuring Sensitivity & Smoothness: We ensure sensitivity and smoothness using spectral normalisation in models with residual connections. In addition, we make minor changes to the standard residual block to further encourage sensitivity without sacrificing accuracy (see details in §B.3.1).

Disentangling Epistemic & Aleatoric Uncertainty: To quantify epistemic uncertainty, we fit a feature-space density estimator after training. We use GDA, a GMM with a single

```
# instantiate models
model = create_sensitive_smooth_model()
gda = create_gda()

# train
training_samples, training_labels = load_training_set()
model.fit(training_samples, training_labels)

training_features = model.features(training_samples)
gda.fit(training_features, training_labels)

# test
test_features = model.features(test_sample)

epistemic_uncertainty = -gda.log_density(test_features)

is_ood = epistemic_uncertainty <= ood_threshold
if not is_ood:
  predictions = model.softmax_layer(test_features)
  aleatoric_uncertainty = entropy(predictions)
  return 0, aleatoric_uncertainty

return epistemic_uncertainty, np.log(num_classes)
```

Figure 4.3: DDU Pseudo-Code

Gaussian component per class, and fit each class component by computing the empirical mean and covariance, per class, of the feature vectors $z = f_\theta(x)$, which are the outputs of the last convolutional layer of the model computed on the training samples x. The conditional density $\mathrm{q}(z|y = c)$ for class c is modeled using the Gaussian $\mathrm{q}(z|y = c) = \mathcal{N}(z|\mu_c, \mathbf{\Sigma}_c)$ where:

$$\begin{aligned} \mu_c &= \frac{1}{N_c} \sum_{i:y_i=c} z_i \\ \mathbf{\Sigma}_c &= \frac{1}{N_c - 1} \sum_{i:y_i=c} (z_i - \mu_c)(z_i - \mu_c)^{\mathrm{T}}. \end{aligned} \tag{4.4}$$

The feature density $\mathrm{q}(z)$ can then be computed as $\mathrm{q}(z) = \sum_c \mathrm{q}(z|y = c)\,\mathrm{q}(y = c)$. In this equation, $\mathrm{q}(y = c) = \frac{N_c}{N}$ where N_c is the number of training samples belonging to class c and N is the total number of training samples. *Note that we do not require OoD data to fit these and unlike [179] we use a separate covariance matrix for each class.* Fitting a GDA on the feature space, thus requires no further training and only requires a single forward pass through the training set.

Evaluation: At test time, we estimate the epistemic uncertainty by evaluating the marginal likelihood of the feature representation under our density $\mathrm{q}(z)$. To quantify aleatoric uncertainty for in-distribution samples, we use the entropy $\mathbb{H}[Y|x, \theta]$ of the softmax

distribution $\mathrm{p}(y \mid x, \theta)$. The softmax distribution thus obtained can be further calibrated using temperature scaling [56]. Note that in contrast with Chapter 3, we are not concerned about producing the most calibrated softmax distribution in this chapter. We are solving the broader question around general uncertainty quantification and its disentanglement into epistemic and aleatoric uncertainty. Combining the two approaches, i.e., applying DDU on a network trained on focal loss is surely an appealing option for someone who is interested in reaping the benefits of focal loss calibration in addition to the epistemic uncertainty estimates of DDU.

Thus, in a nutshell, the DDU algorithm at test time, operates as follows. For a given input, a high feature-space density indicates low epistemic uncertainty (iD), at which point, we can trust the aleatoric estimate from the softmax entropy. The sample can then be either unambiguous (low softmax entropy) or ambiguous (high softmax entropy). Conversely, a low feature density indicates high epistemic uncertainty (OoD), and we cannot trust softmax predictions. A simple Python pseudo-code using a scikit-learn-like API [200] is shown in Figure 4.3. A more detailed algorithm and the corresponding computational complexity can be found in §B.3.

Sanity Check: To verify our claims on DDU's ability to quantify epistemic and aleatoric uncertainty, we train a ResNet-18 model with spectral normalisation (ResNet-18+SN) on increasingly large subsets of Dirty-MNIST (1%, 2% and 10% particularly) and plot the feature-space density as well as the softmax entropy for each of these models in Figure 4.2. With increasing training set size, feature-space density on the test set increases, following the definition of epistemic uncertainty, whereas softmax entropy remains similar, indicative of aleatoric uncertainty.

4.4 Experiments

We evaluate DDU's quality of epistemic uncertainty estimation in active learning [201] using MNIST, CIFAR-10 and an ambiguous version of MNIST (Dirty-MNIST). We also test

DDU on several OoD detection settings including CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C, CIFAR-100 vs SVHN/Tiny-ImageNet and ImageNet vs ImageNet-O dataset pairings, where we outperform other deterministic single-forward-pass methods and perform on par with deep ensembles. Finally, we also evaluate DDU on the task of semantic segmentation on Pascal VOC, comparing with a deterministic model, MC Dropout (MCDO) [109] and deep ensembles. In the appendix, we also examine DDU's performance on the real-world QUBIQ challenge in §B.6.3 and on the well-known Two Moons toy dataset in §B.6.6. We elaborate on how DDU can disentangle epistemic and aleatoric uncertainty, the setting depicted in Figure 4.1, in §B.6.1, and the effect of feature-space regularisation in §B.6.2.

4.4.1 Active Learning

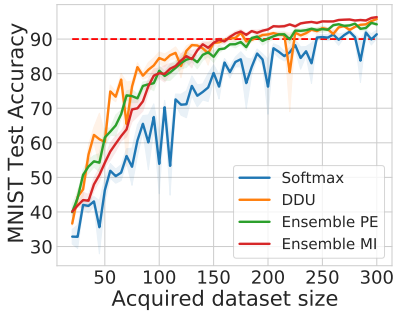
We first demonstrate the quality of our uncertainty disentanglement in active learning (AL) [201]. AL aims to train models in a data-efficient manner. Additional training samples are iteratively acquired from a large pool of unlabelled data and labelled with the help of an expert. After each acquisition step, the model is retrained on the newly expanded training set. This is repeated until the model achieves a desirable accuracy—or when a maximum number of samples have been acquired.

Data-efficient acquisition relies on acquiring labels for the most informative samples. This can be achieved by selecting points with high epistemic uncertainty [79]. Conversely, repeated acquisition of points with high aleatoric uncertainty is not informative for the model and such acquisitions lead to data inefficiency. As mentioned before, points with high aleatoric uncertainty often suffer from noise or ambiguity which can make it inherently difficult for the expert to correctly label them. The presence of label noise in the acquired samples then prevents the model from generalising well on the test set, thereby also increasing the amount of labeled training data the model needs to acquire. AL, therefore, makes an excellent application for evaluating epistemic uncertainty and the ability of models to separate different sources of uncertainty. We evaluate DDU on three different setups: **i)** with clean MNIST samples in the pool set, **ii)** with clean CIFAR-10 samples in the pool set, and **ii)**

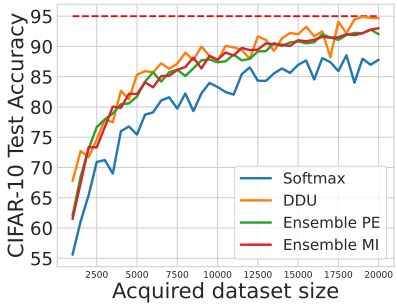
with Dirty-MNIST, having a 1:60 ratio of MNIST to Ambiguous-MNIST samples, in the pool set. In the first two setups, we compare 3 baselines: **i)** a ResNet-18 with softmax entropy as the acquisition function, **ii)** DDU trained using a ResNet-18 with feature density as acquisition function, and **iii)** a Deep Ensemble of 3 ResNet-18s with the predictive entropy (PE) and mutual information (MI) of the ensemble as the acquisition functions. In the last setup, in addition to the above 3 approaches, we also use **iv)** feature density of a VGG-16 instead of ResNet-18+SN as an ablation to see if feature density of a model without inductive biases performs well, **v)** SNGP and **vi)** DUQ as additional baselines. For MNIST and Dirty-MNIST, we start with an initial training-set size of 20 randomly chosen MNIST points, and in each iteration, acquire the 5 samples with highest reported epistemic uncertainty. For each step, we train the models using Adam [202] for 100 epochs and choose the one with the best validation set accuracy. We stop the process when the training set size reaches 300. For CIFAR-10, we start with 1000 samples and go up to 20000 samples with an acquisition size of 500 samples in each step.

MNIST & CIFAR-10 In Figure 4.4a and Figure 4.4b, for regular curated MNIST and CIFAR-10 in the pool set, DDU clearly outperforms the deterministic softmax baseline and is competitive with Deep Ensembles. For MNIST, the softmax baseline reaches 90% test-set accuracy at a training-set size of 245. DDU reaches 90% accuracy at a training-set size of 160, whereas Deep Ensemble reaches the same at 185 and 155 training samples with PE and MI as the acquisition functions respectively. Note that DDU is three times faster than a Deep Ensemble, which needs to train three models independently after every acquisition.

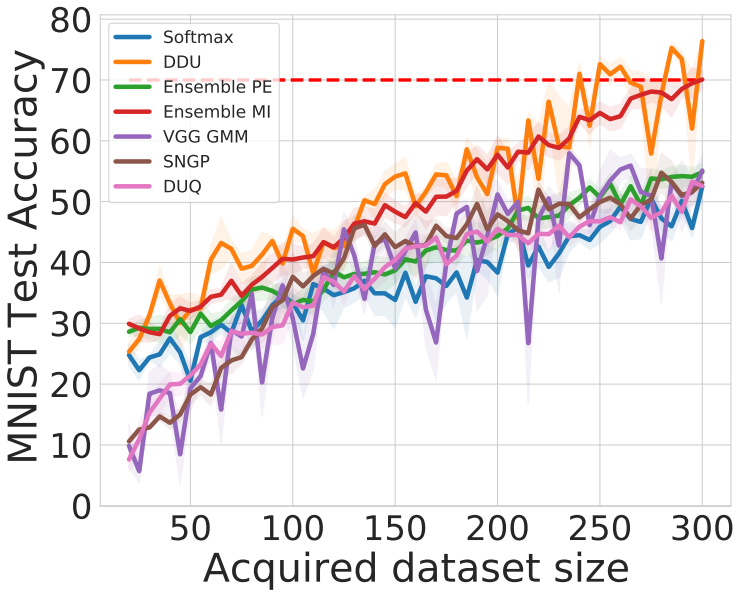
Dirty-MNIST. Real-life datasets often contain observation noise and ambiguous samples. What happens when the pool set contains a lot of such noisy samples having high aleatoric uncertainty? In such cases, it becomes important for models to identify unseen and informative samples with high epistemic uncertainty and not with high aleatoric uncertainty. To study this, we construct a pool set with samples from Dirty-MNIST (see §B.2). We significantly increase the proportion of ambiguous samples by using a 1:60 split of MNIST to Ambiguous-MNIST (a total of 1K MNIST and 60K Ambiguous-MNIST samples).



(a) MNIST



(b) CIFAR-10



(c) Dirty-MNIST

Figure 4.4: *Active Learning experiments.* Acquired training set size vs test accuracy. DDU performs on par with Deep Ensembles.

In Figure 4.4c, for Dirty-MNIST in the pool set, the difference in the performance of DDU and the deterministic softmax model is stark. While DDU achieves a test set accuracy of 70% at a training set size of 240 samples, the accuracy of the softmax baseline peaks at a mere 50%. In addition, all baselines, including SNGP, DUQ and the feature density of a VGG-16, which fail to solely capture epistemic uncertainty, are significantly outperformed by DDU and the MI baseline of the deep ensemble. However, note that DDU also performs better than Deep Ensembles with the PE acquisition function. The difference gets larger as the training set size grows: DDU's feature density and Deep Ensemble's MI solely capture epistemic uncertainty and hence, do not get confounded by iD ambiguous samples with high aleatoric uncertainty.

4.4.2 OoD Detection

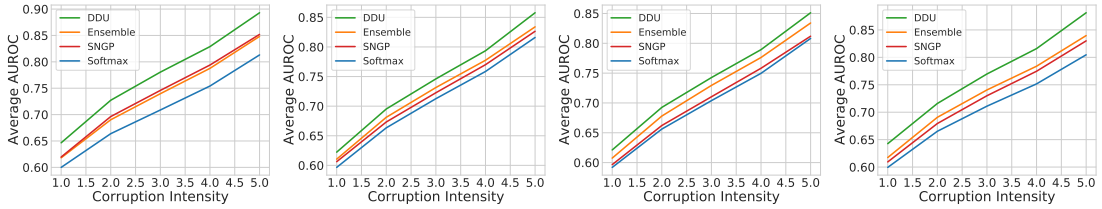
OoD detection is an application of epistemic uncertainty quantification: if we do not train on OoD data, we expect OoD data points to have higher epistemic uncertainty than iD data.

Train Dataset	Method	Penalty	Aleatoric Uncertainty	Epistemic Uncertainty	Accuracy (↑)	ECE (↓)	AUROC SVHN (↑)	AUROC CIFAR-100 (↑)	AUROC Tiny-ImageNet (↑)
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	95.98 ± 0.02	$\mathbf{0.85 \pm 0.02}$	94.44 ± 0.43	89.39 ± 0.06	88.42 ± 0.05
	Energy-based [195]	-		Softmax Density			94.56 ± 0.51	88.89 ± 0.07	88.11 ± 0.06
	DUQ [57]	JP	Kernel Distance	Kernel Distance	94.6 ± 0.16	1.55 ± 0.08	93.71 ± 0.61	85.92 ± 0.35	86.83 ± 0.12
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	$\mathbf{96.04 \pm 0.09}$	1.8 ± 0.1	94.0 ± 1.3	91.13 ± 0.15	89.97 ± 0.19
	DDU (ours)	**SN**	**Softmax Entropy**	**GDA Density**	95.97 ± 0.03	$\mathbf{0.85 \pm 0.04}$	$\mathbf{97.86 \pm 0.19}$	$\mathbf{91.34 \pm 0.04}$	$\mathbf{91.07 \pm 0.05}$
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	$\mathbf{96.59 \pm 0.02}$	$\mathbf{0.76 \pm 0.03}$	97.73 ± 0.31	$\mathbf{92.13 \pm 0.02}$	90.06 ± 0.03
				Mutual Information			97.18 ± 0.19	91.33 ± 0.03	90.90 ± 0.03
					Accuracy (↑)	ECE (↓)	AUROC SVHN (↑)		AUROC Tiny-ImageNet (↑)
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	80.26 ± 0.06	4.62 ± 0.06	77.42 ± 0.57		81.53 ± 0.05
	Energy-based [195]	-		Softmax Density			78 ± 0.63		81.33 ± 0.06
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	80.00 ± 0.11	4.33 ± 0.01	85.71 ± 0.81		78.85 ± 0.43
	DDU (ours)	**SN**	**Softmax Entropy**	**GDA Density**	$\mathbf{80.98 \pm 0.06}$	$\mathbf{4.10 \pm 0.08}$	$\mathbf{87.53 \pm 0.62}$		$\mathbf{83.13 \pm 0.06}$
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	$\mathbf{82.79 \pm 0.10}$	$\mathbf{3.32 \pm 0.09}$	79.54 ± 0.91		82.95 ± 0.09
				Mutual Information			77.00 ± 1.54		82.82 ± 0.04

Table 4.1: *OoD detection performance of different baselines using a Wide-ResNet-28-10 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs. SN:* Spectral Normalisation, *JP:* Jacobian Penalty. The best deterministic single-forward pass method and the best method overall are in bold for each metric.

Model	Accuracy (↑)		ECE (↓)		AUROC (↑)				
	Deterministic	3-Ensemble	Deterministic	3-Ensemble	Softmax Entropy	Energy-based Model	DDU	3-Ensemble PE	3-Ensemble MI
ResNet-50	74.8 ± 0.05	76.01	2.08 ± 0.11	2.07	51.42 ± 0.61	55.76 ± 0.81	$\mathbf{71.29 \pm 0.08}$	60.3	62.43
Wide-ResNet-50-2	76.75 ± 0.11	77.58	1.18 ± 0.07	1.22	52.71 ± 0.23	57.13 ± 0.4	$\mathbf{73.12 \pm 0.19}$	60.45	64.81
VGG-16	72.48 ± 0.02	73.54	2.62 ± 0.11	2.59	50.67 ± 0.22	52.04 ± 0.23	54.32 ± 0.14	58.74	**60.56**

Table 4.2: *OoD detection performance of different baselines using ResNet-50, Wide-ResNet-50-2 and VGG-16 architectures on ImageNet vs ImageNet-O [91].* Best AUROC scores are marked in bold.



(a) WRN-28-10 **(b)** RN-50 **(c)** RN-110 **(d)** DN-121

Figure 4.5: AUROC vs corruption intensity averaged over all corruption types in CIFAR-10-C for 4 architectures. More details in §4.4.2 and more ablations in §B.4 in the appendix.

We evaluate CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C, CIFAR-100 vs SVHN/Tiny-ImageNet and ImageNet vs ImageNet-O as iD vs OoD dataset pairs for this experiment [12, 89, 90, 203]. We also evaluate DDU on different architectures: Wide-ResNet-28-10, Wide-ResNet-50-2, ResNet-50, ResNet-110 and DenseNet-121 [15, 169, 170]. The training setup is described in §B.4.2. In addition to using softmax entropy of a deterministic model (*Softmax*) for both aleatoric and epistemic uncertainty, we also compare with the following **baselines** that do not require training or fine-tuning on OoD data:

- *Energy-based model* [195]: We use the softmax entropy as aleatoric uncertainty and

the unnormalised softmax density (the logsumexp of the logits) as epistemic uncertainty *without* regularisation to avoid feature collapse.

- *DUQ [57] & SNGP [58]*: We compare with the state-of-the-art deterministic methods for uncertainty quantification including DUQ and SNGP. For SNGP, we use the exact predictive covariance computation and we use the entropy of the average of the MC softmax samples as uncertainty. For DUQ, we use the closest kernel distance. Note that for CIFAR-100, DUQ's one-vs-all objective did not converge during training and hence, we do not include the DUQ baseline for CIFAR-100.

- *5-Ensemble*: We use an ensemble of 5 networks and compute the predictive entropy of the ensemble as both epistemic and aleatoric uncertainty and mutual information as epistemic uncertainty.

Results: Table 4.1 presents the AUROC for Wide-ResNet-28-10 models on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet along with their respective test set accuracy and ECE post temp-scaling (additional calibration scores in §B.6.5 and comparison with more baselines in §B.6.4). The equivalent results for other architectures: ResNet-50/110 and DenseNet-121 can be found in Table B.4.1, Table B.4.2 and Table B.4.3. Note that for DDU, post-hoc calibration with temperature scaling [56], is simple as it does not affect the GMM density. We also plot the AUROC averaged over corruption types vs corruption intensity for CIFAR-10 vs CIFAR-10-C in Figure 4.5, with AUROC plots per corruption type in Figure B.4.2, Figure B.4.3, Figure B.4.4 and Figure B.4.5 of the appendix. Finally, in Table 4.2, we present AUROC for models trained on ImageNet.

For OoD detection, *DDU outperforms all other deterministic single-forward-pass methods, DUQ, SNGP and the energy-based model approach from [195], on CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet, CIFAR-10 vs CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet, often performs on par with state-of-the-art Deep Ensembles—and even performing better in a few cases*. This holds true for all the architectures we experimented on. Similar observations can be made on ImageNet vs ImageNet-O as well. Importantly, the great

performance in OoD detection comes without compromising on the single-model test set accuracy in comparison to other deterministic methods.

Ablations: Additional ablations for the CIFAR-10/100 experiments are detailed in §B.5 with results in Table B.4.4 and Table B.4.5. These tables along with observations in Table 4.2, show that *the feature density of a VGG-16 (i.e. without residual connections and spectral normalisation) is unable to beat a VGG-16 ensemble, whereas a Wide-ResNet-28-10 with spectral normalisation outperforms its corresponding ensemble in almost all the cases*. This result further validates the importance of having a regularised feature space on the model to obtain smoothness and sensitivity. Between spectral normalisation (smoothness) and residual connections (sensitivity), we note that even without spectral normalisation, a Wide-ResNet-28 is able to perform competitively with a Wide-ResNet-28 with spectral normalisation. This indicates that residual connections can be a larger contributing factor towards good performance compared to spectral normalisation, as residual connections make the model sensitive to changes in the input space. Finally, we also provide an ablation using LDA [179], which uses a shared covariance matrix over all classes, instead of GDA with covariance matrices per class. The resulting AUROC for Wide-ResNet-28-10 trained on CIFAR-10/100 and for Wide-ResNet-50-2 and ResNet-50 trained on ImageNet in Table B.4.6 in §B.5. LDA only outperforms GDA when using SVHN as an OoD dataset. In all other cases, GDA obtains significantly higher AUROC, thereby indicating the advantage of modeling density using individual covariance matrices per class.

4.4.3 Semantic Segmentation

In this section, we apply DDU to the task of semantic segmentation on Pascal VOC 2012 [184], comparing with a vanilla softmax model, MC Dropout and deep ensembles. Semantic segmentation [38], classifies every pixel of a given to one of a fixed set of classes. Since different classes can have different levels of representation in a segmentation dataset, it forms a classic example of a problem with class imbalance, thereby requiring reliable estimates of epistemic uncertainty. Furthermore, due to the computationally heavy nature of semantic

segmentation, classic uncertainty quantification approaches like MC Dropout and deep ensemblesare rendered infeasible in real-world applications.

Pixel-Independent Class-Wise Means and Covariances: As each pixel has a corresponding prediction in semantic segmentation, it is natural to ask if the Gaussian means and covariance matrices need to be computed per pixel. To examine this, in Figure B.4.1 of §B.4.3, we plot the L2 distances between feature space means of all pairs of classes obtained from a DeepLab-v3+ [183] model with a ResNet-101 backbone for two "distant" pixels. We observe that pixels of the same class are much closer in the feature space than pixels of different classes, irrespective of their location in the image. In spirit of a new simple baseline, we thus compute the Gaussian means and covariances per class, taking each pixel as a separate data point.

Architecture, Training and Evaluation Metrics: As mentioned above, we evaluate DDU on Pascal VOC 2012 and compare to a vanilla softmax model, MC Dropout with 5 forward passes at test time, and a deep ensembles with 3 members. We use DeepLab-v3+ with a ResNet-101 backbone as the model architecture. Further training details are in §B.4.3. Finally, to evaluate the uncertainty estimates, we use metrics proposed in [204]: $p(\text{accurate}|\text{certain})$, $p(\text{uncertainty}|\text{inaccurate})$ and PAVPU. $p(\text{accurate}|\text{certain})$ computes the probability of the model being accurate given that it is confident. Similarly, $p(\text{uncertainty}|\text{inaccurate})$ measures probability of the model being uncertain given that it is inaccurate and $PAVPU$ computes the probability of the model being confident on accurate predictions and uncertain on inaccurate ones. Ideally, high values for these metrics indicate better uncertainty estimates in segmentation. Furthermore, note that these metrics can be computed at different thresholds of uncertainty (defining if a model is certain or not).

Results and Discussion: In Figure 4.7, we present the above 3 metrics for all segmentation baselines evaluated on the Pascal VOC validation set. We also report the val set accuracy and runtime of a single forward pass in Table 4.3. Finally, we visualise uncertainty estimates from each baseline in Figure 4.6. Firstly, from Table 4.3, it is clear that DDU has the runtime of a deterministic model which is significantly faster than both MC Dropout and

Baseline	Softmax	MC Dropout	Deep Ensemble	**DDU**
mIoU	78.53	78.61	78.47	78.53
Runtime (ms)	275.48 ± 1.91	1576.75 ± 1.56	875.87 ± 0.79	$\mathbf{263.83 \pm 2.79}$

Table 4.3: *Pascal VOC val set mIoU and runtime in milliseconds averaged over 10 forward passes.* For MC Dropout, we perform 5 stochastic forward passes.

deep ensembles. Also note that DDU's mIoU is the same as that of the vanilla softmax model. Secondly, from Figure 4.7, we see that DDU consistently performs better on all 3 evaluation metrics compared to the other baselines. Finally, Figure 4.6 qualitatively validates that DDU's feature-space density captures epistemic uncertainty while the softmax entropy captures aleatoric uncertainty. For DDU, for the first two samples (first two rows, Figure 4.6g), the epistemic uncertainty is not high and only aleatoric uncertainty is captured along edges of objects. However, for the last sample (row 4, Figure 4.6g), the epistemic uncertainty is high for a relatively large patch on the image which is inaccurately predicted by the model as well. Note that only DDU's feature density is significantly lower for that entire region, whereas softmax entropy does not capture high uncertainty there and is only high along the edges. These observations are in line with [78]: aleatoric uncertainty is high on edges of objects as they correspond to regions of high ambiguity and noise; on the other hand, epistemic uncertainty is high for regions of the image which are previously unseen.

4.5 Additional Insights

We conclude the chapter with a discussion on potential pitfalls of predictive entropy in general and softmax entropy of deterministic models in particular. Proofs for the statements in this section are provided in §B.7.

Potential Pitfalls of *Predictive Entropy*: We start with a classic deep ensemble as defined in [49] and §4.2 where each component of the deep ensemble is a deterministic neural network producing a softmax distribution. Such ensembling can be interpreted as a form of Bayesian Model Averaging [112, 205], with each ensemble member approximating a sample from a posterior. Hence, eq. (4.1) can be applied to ensembles to disentangle

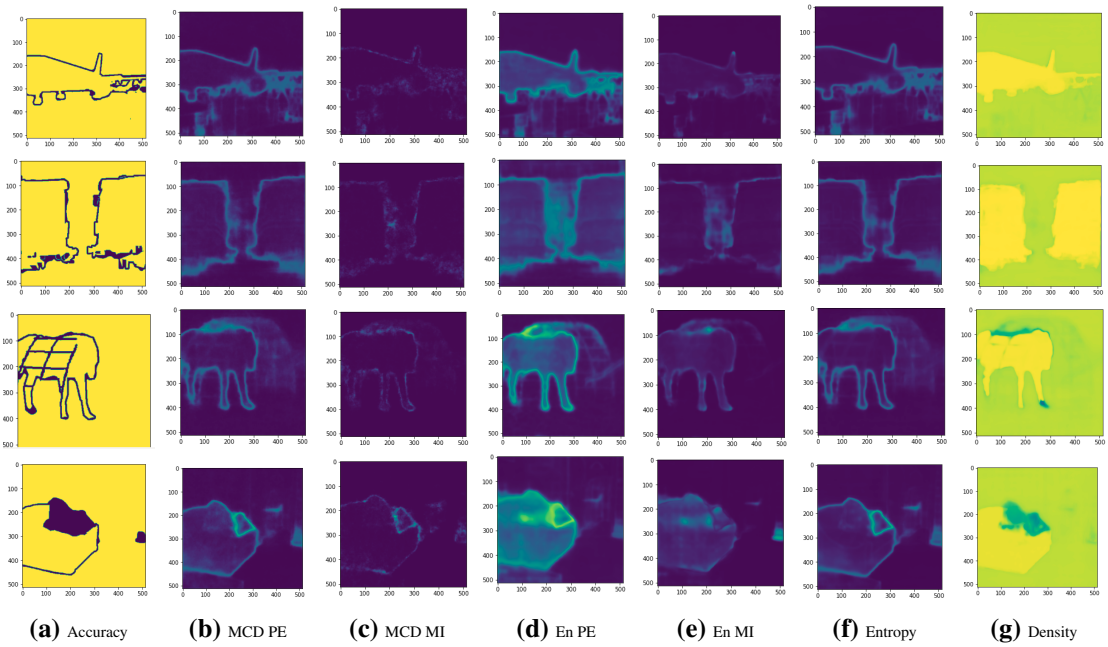


Figure 4.6: *Visualisation of uncertainty baselines on four PASCAL VOC validation samples (**rows**).* **Columns**: **a** shows pixel-wise accuracy; **b, c** predictive entropy (PE) and mutual information (MI) obtained for MC Dropout (MCD); **d, e** for deep ensembles; **f** per-pixel softmax entropy, *the aleatoric estimate of DDU*; and **g** feature density, *the epistemic component of DDU*. For all but g: the brighter, the more uncertain, whereas DDU's density g captures certainty: hence, the brighter, the more certain.

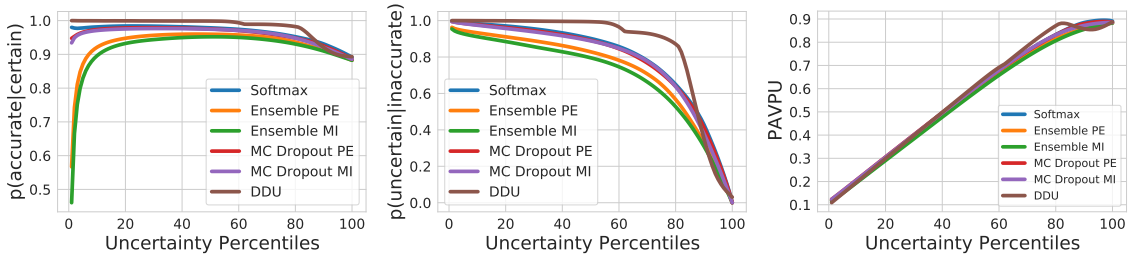


Figure 4.7: $p(\text{accurate}|\text{certain})$, $p(\text{uncertain}|\text{inaccurate})$ and PAVPU evaluated on PASCAL VOC validation set. DDU outperforms all other baselines.

epistemic and aleatoric uncertainty. In eq. (4.1), predictive entropy $\mathbb{H}[Y|x, \mathcal{D}]$ is the entropy of the average softmax distribution produced by all the components of the ensemble, and mutual information $\mathbb{I}[Y; w|x, \mathcal{D}]$ is the difference between the predictive entropy and the average of the individual softmax entropies $\mathbb{E}_{p(w|\mathcal{D})}[\mathbb{H}[Y|x, w]]$. As mutual information provides an estimate of epistemic uncertainty specifically, *predictive entropy can confound both epistemic and aleatoric uncertainty*.

In practice, both mutual information $\mathbb{I}[Y; \omega \mid x, \mathcal{D}]$ and predictive entropy $\mathbb{H}[Y \mid x, \mathcal{D}]$ could be used to detect OoD samples. However, previous empirical findings show predictive entropy outperforming mutual information [206]. Indeed, much of the recent literature only focuses on predictive entropy for OoD detection (see §B.1.1). We explain these findings using the following observation:

Observation 4.5.1. If we *know* that *either* aleatoric *or* epistemic uncertainty is *low* for a sample, predictive entropy is a good measure of the uncertainty type which is high.

Thus, predictive entropy, as an upper-bound of mutual information, can separate iD and OoD data better when datasets are curated and have low aleatoric uncertainty. However, as seen in eq. (4.1), predictive entropy can be high for both iD ambiguous samples (high aleatoric) as well as OoD samples (high epistemic) (see Figure B.6.2) and might *not* be an effective measure for OoD detection when used with datasets that are not curated with ambiguous samples, like Dirty-MNIST, as seen in our active learning results.

Potential Pitfalls of *Softmax Entropy*: The softmax entropy for deterministic models trained with maximum likelihood can be *inconsistent*. In fact, if we assume that mutual information of a deep ensemble is an effective and reliable measure of epistemic uncertainty, softmax entropy should not be one, as we describe in our observation below.

Observation 4.5.2. Let x_1 and x_2 be points such that x_1 has **higher** mutual information than x_2 under the ensemble: $\mathbb{I}[Y_1; \omega \,|\, x_1, \mathcal{D}] > \mathbb{I}[Y_2; \omega \,|\, x_2, \mathcal{D}] + \delta$, $\delta \geq 0$. Further assume both have similar predictive entropy $|\, \mathbb{H}[Y_1 \,|\, x_1, \mathcal{D}] - \mathbb{H}[Y_2 \,|\, x_2, \mathcal{D}]| \leq \epsilon$, $\epsilon \geq 0$. Then, there exist sets of ensemble members Ω with $\mathrm{p}(\Omega \,|\, \mathcal{D}) > 0$, such that for all softmax models $\omega \in \Omega$ the softmax entropy of x_1 is **lower** than the softmax entropy of x_2: $\mathbb{H}[Y_1 \,|\, x_1, \omega] < \mathbb{H}[Y_2 \,|\, x_2, \omega] - (\delta - \epsilon)$.

In simpler terms, if a sample x_1 is assigned higher epistemic uncertainty (quantified as mutual information) by an ensemble than another sample x_2, it will necessarily be assigned lower softmax entropy by at least one of the ensemble's members. We can prove this through simple contradiction as well. Note that mutual information $\mathbb{I}[Y; w|x, \mathcal{D}]$ captures a measure of disagreement between the ensemble components. If all the ensemble components confidently predict (i.e., make low/zero entropy predictions on) *different* labels,

$\mathbb{I}[Y; w|x, \mathcal{D}]$ is maximized. On the other hand, if all the ensemble components either confidently predict the same label or are not confident (high entropy/uniform predictions by all ensemble components) on any label, $\mathbb{I}[Y; w|x, \mathcal{D}]$ is minimized (close to 0). Now let's say $\mathbb{I}[Y_1; w|x_1, \mathcal{D}] > \mathbb{I}[Y_2; w|x_2, \mathcal{D}]$. Also, let's say all the ensemble components produced a higher entropy (low confidence) prediction on x_1 compared to x_2. However, if that is the case, $\mathbb{I}[Y_1; w|x_1, \mathcal{D}] < \mathbb{I}[Y_2; w|x_2, \mathcal{D}]$ as higher entropy predictions from all ensemble components for x_1 would indicate a higher level of agreement and a lower mutual information for x_1, thereby leading to a contradiction. Thus, there must be at least one ensemble component which produced a lower softmax entropy for x_1 than x_2.

As a result, a priori, we cannot know whether a softmax model preserves the order or not, and *the empirical observation that the mutual information of an ensemble can quantify epistemic uncertainty well implies that the softmax entropy of a deterministic model might not*. We see this in Figure 4.1b and Figure B.6.2 where softmax entropy for OoD samples can be high, low or anywhere in between. While *not all* model architectures or versions of a deep ensemble might behave like this, for cases when the mutual information of a Deep Ensemble works well empirically, Observation 4.5.2 holds.

Objective Mismatch: The predictive probability induced by a feature-density estimator will generally not be well-calibrated as there is an objective mismatch. This was overlooked in previous research on uncertainty quantification for deterministic models [15, 57, 58, 179, 186]. Specifically, a mixture model $\mathrm{q}(y, z) = \sum_y \mathrm{q}(z \mid y)\, \mathrm{q}(y)$, using one component per class, may not be optimal for both feature-space density and predictive distribution estimation as there is an *objective mismatch* [207, Ex. 4.20, p. 145]. We summarize this intuition in the following observation:

Observation 4.5.3. For an input x, let $z = f_\theta(x)$ denote its feature representation in a feature extractor f_θ with parameters θ. Then the following hold:

1. A discriminative classifier $\mathrm{p}(y \mid z)$, e.g. a softmax layer, is well-calibrated in its predictions when it maximises the conditional log-likelihood $\log \mathrm{p}(y \mid z)$;

2. A feature-space density estimator $\mathrm{q}(z)$ is optimal when it maximises the marginalised log-likelihood $\log \mathrm{q}(z)$;

3. A mixture model $\mathrm{q}(y, z) = \sum_y \mathrm{q}(z \mid y)\, \mathrm{q}(y)$ might not maximise both objectives, conditional log-likelihood and marginalised log-likelihood, at the same time. In the specific instance that a GMM with one component per class does maximise both, the resulting model must be a GDA (but the opposite does not hold).

Hence, importantly, DDU uses *both* a discriminative classifier (softmax layer) to capture aleatoric uncertainty for iD samples and a separate feature-density estimator to capture epistemic uncertainty even on a model trained using conditional log-likelihood, i.e. the usual cross-entropy objective. Figure B.6.4 provides additional intuitions on this point.

4.6 Conclusion

Deep Deterministic Uncertainty (DDU) can outperform state-of-the-art deterministic single-pass uncertainty methods in active learning and OoD detection by fitting a GDA for feature-space density estimation after training a model with residual connections and spectral normalisation [58, 179] while performing as well as deep ensembles in several settings. Hence, DDU provides a very simple method to obtain good epistemic and aleatoric uncertainty estimates and might be taken into consideration as an alternative to deep ensembles without requiring the complexities or computational cost of the current state-of-the-art. Reliable uncertainty quantification is an important requirement to make deep neural nets safe for deployment. Thus, we hope our work will contribute to increasing safety, reliability and trust in AI.

Limitations & future work: First, while DDU provides a simple and effective solution to the problem of quantifying and disentangling uncertainty using a deterministic neural network, there can be room for improvement both on the front of better methods for regularising the feature space as well as better density models. Second, DDU does not produce a predictive joint uncertainty estimate of the likes of predictive entropy in deep ensembles [49], eq. (4.1) or joint model and data uncertainty in HetSNGP [208]. This is due to the fact that feature density

and softmax entropy in DDU are in different units of measurement and it is not obvious how these two measures can be easily combined into a single joint uncertainty estimate.

5

Raising the Bar on the Evaluation of Out-of-Distribution Detection

Contents

5.1 Introduction 75
5.2 Related Work 79
5.3 Method 81
5.4 Experiments and Discussion 85
5.4.1 Implementation Details 85
5.4.2 Sanity Check on Benchmarks 86
5.4.3 Evaluating Near OoD Datasets 87
5.4.4 Evaluating Shifted Datasets 91
5.4.5 Evaluation of OoD detection in real life 93
5.5 Conclusion 94

Abstract

In image classification, a lot of development has happened in detecting out-of-distribution (OoD) data. However, most OoD detection methods are evaluated on a standard set of datasets, arbitrarily different from training data. There is no clear definition of what forms a "good" OoD dataset. Furthermore, the state-of-the-art OoD detection methods already achieve near perfect results on these standard benchmarks. In this chapter, we define 2 categories of OoD data using the subtly different concepts of perceptual/visual and semantic similarity to in-distribution (iD) data. We define *Near OoD* samples as perceptually similar but semantically different from iD samples, and *Shifted* samples as points which are visually different but semantically akin to iD data. We then propose a GAN based framework for generating OoD samples from each of these 2 categories, given an iD dataset. Through extensive experiments on MNIST, CIFAR-10/100 and ImageNet, we show that **a)** state-of-the-art OoD detection methods which perform exceedingly well on conventional benchmarks are significantly less robust to our proposed benchmark. Moreover, we observe that **b)** models performing well on our setup also perform well on conventional real-world OoD detection benchmarks and vice versa, thereby indicating that one might not even need a separate OoD set, to reliably evaluate performance in OoD detection.



(a) iD **(b)** Shifted **(c)** Near OoD

Figure 5.1: *Shifted* and *Near OoD* samples obtained from ImageNet. *Shifted samples* are visually different but semantically similar to iD. *Near OoD images* are perceptually similar to iD but are semantically dissimilar.

5.1 Introduction

With the wide-spread deployment of deep learning models in real-life applications like autonomous driving [209] and medical diagnosis [210], it is imperative to ensure that in addition to being accurate, such models are also able to reliably quantify their uncertainty and identify inputs which they "don't know". One of the major applications of such uncertainty quantification methods is the detection of inputs sampled from a distribution different from the model's training distribution (i.e, Out-of-Distribution or OoD inputs). A lot of work has been done in this direction from the perspective of uncertainty quantification [49, 57, 58, 211], OoD Detection [94, 177, 179, 187, 192, 212], open-set recognition [96, 213] and the like.

Since any point outside the training distribution can be considered OoD, the set of potential OoD inputs is infinite. This makes evaluating OoD detection a particularly challenging problem. The general evaluation practice involves using a proxy OoD dataset which is different from the training distribution (or in-distribution (iD) samples) to simulate an out-of-distribution scenario. The OoD detection algorithm is then evaluated on how well it can separate the iD samples from the OoD points. For the purposes of evaluation and benchmarking, it is natural to ask which proxy OoD dataset is best suited for measuring model performance. To answer this, we need to consider the different types of OoD inputs

that can arise in a real-world scenario.

In image classification, we model the conditional categorical distribution $p(y|\mathbf{x})$ over classes, given an input image $\mathbf{x}$. Under the i.i.d assumption, both the training and test images are assumed to be sampled from the same continuous distribution in image space, i.e., $p_{\text{train}}(\mathbf{x}) = p_{\text{test}}(\mathbf{x})$. In case of OoD samples, this assumption is broken, i.e., $p_{\text{train}}(\mathbf{x}) \neq p_{\text{ood}}(\mathbf{x})$. Based on the conditional distribution $p(y|\mathbf{x})$, we can then define two kinds of OoD.

Distribution Shift: Although the distribution in image space is different, the conditional distribution over class labels remains the same, i.e., $p_{\text{train}}(y|\mathbf{x}) = p_{\text{ood}}(y|\mathbf{x})$, and $p_{\text{train}}(\mathbf{x}) \neq p_{\text{ood}}(\mathbf{x})$. Such samples are generally derived from the training set by applying transformations like corruptions [203] and semantic shifts [214, 215], where the transformed images have the same labels as the originals from training. For example, ImageNet-C/P [203] contain synthetic corruptions/perturbations applied to ImageNet [12]. Such datasets provide a controlled environment to study models under specific synthetic and real-world distribution shifts.

Unseen Categories: The second category of OoD comprise images of classes which the model has not been trained on, i.e., $p_{\text{train}}(y|\mathbf{x}) \neq p_{\text{ood}}(y|\mathbf{x})$, and $p_{\text{train}}(\mathbf{x}) \neq p_{\text{ood}}(\mathbf{x})$. For a given training set, any dataset having a disjoint set of labels qualifies as OoD with unseen categories. How do we then decide which OoD dataset is good for evaluation? The convention is to use a well-known set of (iD vs OoD) dataset pairs like MNIST [14] vs Fashion-MNIST [216], CIFAR-10 [89] vs SVHN [90] etc. However, **firstly,** the choice of these dataset pairs is relatively arbitrary and there is no guarantee that performance on these benchmarks will generalise to the real-world. **Secondly**, in recent literature [94, 187], the terms “Near OoD" and “Far OoD" have been used to indicate the difficulty of an OoD detection task with Near OoD datasets (CIFAR-10 vs CIFAR-100) being more difficult than Far OoD (CIFAR-10 vs SVHN). With no model-agnostic metric quantifying the “nearness" of an OoD dataset, these terms are also not well-defined. **Finally,** one of the current state-of-the-art OoD detection baselines, Vision Transformer [94], obtains around 96% AUROC on CIFAR-100 vs CIFAR-10 and over 99.5% AUROC on CIFAR-10 vs SVHN. Hence, the most popular OoD detection benchmarks are not necessarily the most effective. They can

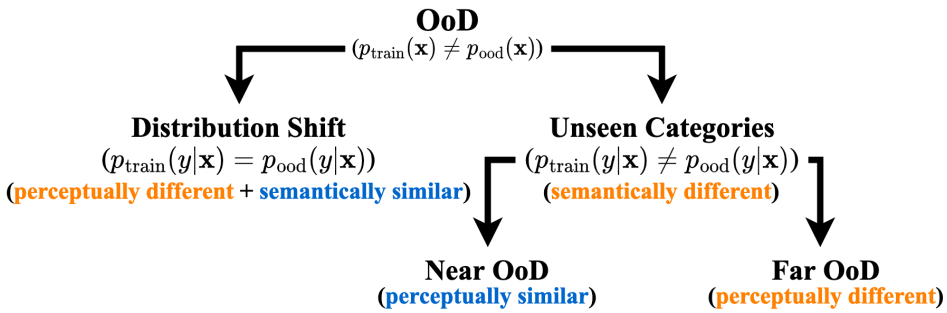


Figure 5.2: Categories of OoD samples. Samples obtained from a distribution shift retain their conditional distribution over classes, whereas samples obtained from an unseen category don't belong to any of the training classes.

be saturated and might give us the impression that state-of-the-art baselines are robust to OoD. The near perfect AUROC scores also indicate that these benchmarks might be rendered redundant in future for evaluating the performance of even better OoD detection methods which outperform the Vision Transformer [94].

In this work, we thus aim to take a step towards improving the conventional evaluation process for OoD detection in image classification. We start with the question: *What are the desirable properties of an OoD sample?* To answer this, we study the types of OoD mentioned above through the lens of *perceptual/visual similarity* and *semantic similarity* [217, 218] between images. Perceptual similarity between two images denotes how visually similar they are and semantic similarity captures the similarity of concepts that they represent. With this in mind, we define:

1. ***Shifted sets*** *as perceptually dissimilar but semantically similar to the training distribution.*
2. ***Near OoD sets*** *as perceptually similar but semantically dissimilar to the training distribution.*
3. ***Far OoD sets*** *as both perceptually and semantically dissimilar to the training distribution.*

Clearly, images which are both perceptually and semantically similar to the training distribution would be iD. We show the hierarchy of OoD samples in Figure 5.2. Next, knowing the desirable properties of OoD samples, we ask: *Can we generate OoD samples*

given the training set of a task? We tackle this by designing regularizers in the loss function of a generative model, which capture our understanding of perceptual and semantic similarity. Given the training set, it is difficult to define a single distance measure in the image space which can capture both perceptual and semantic similarity. Hence, we propose using a sampling based generative model, a Generative Adversarial Network (GAN) [219] and design regularisers for the GAN objective using the definitions above to generate OoD samples.

In this work, we particularly focus on generating Shifted and Near OoD sets. More formally, for a training set $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^N$, where $y_i \in \mathcal{S}$, $\forall i$, in order to generate *shifted samples*, we learn a transformation $t_{\text{shift}} : \mathbf{x} \rightarrow \hat{\mathbf{x}}$ in the image space, $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^{H,W,C}$, such that $\mathbf{x}$ and $\hat{\mathbf{x}}$ are perceptually different and semantically similar, i.e., have the same label: $\arg\max_c p(y_c|\mathbf{x}) = \arg\max_c p(y_c|\hat{\mathbf{x}})$. This is an Image-to-Image translation problem and hence, we use a Pix-2-Pix [220] model to learn a distribution shift. In case of *Near OoD*, we want to learn a distribution in the close perceptual vicinity of $\mathcal{D}$. This can be seen as a transformation $t_{\text{near ood}} : \mathbf{z} \rightarrow \hat{\mathbf{x}}$, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ where the generated image $\hat{\mathbf{x}}$ is perceptually similar to the training distribution but does not belong to any of the iD classes: $\arg\max_c p(y_c|\hat{\mathbf{x}}) \notin \mathcal{S}$. This is an image generation problem and hence, we use a GAN for generating Near OoD samples. *The proposed method therefore helps us generate an OoD set, given a task's training set without using any external OoD samples*. In Figure 5.1, we show examples from ImageNet of shifted as well as Near OoD samples.

Finally, with the generated OoD samples, it is natural to ask the question: *How do generated OoD samples compare with real-world OoD datasets for evaluating OoD detection?* Through extensive experiments using several popular OoD detection baselines comparing our generated samples with conventional benchmarks, we make the following observations. **Firstly,** the performance of state-of-the-art OoD detection baselines (Deep Ensembles [49] and Vision Transformers [94]), established to be relatively robust on standard benchmarks, is consistently worse under our proposed evaluation setup. This is true across datasets of all sizes: ImageNet, CIFAR-10/100 and MNIST, thereby showing that *there's still plenty of room for improvement in OoD detection research*. **Secondly,** we observe a consistent

trend where models which perform better on our benchmarks also perform well on standard real-world benchmarks and vice versa. Assuming that standard benchmarks are indicative of real-world OoD detection performance, an assumption made by most existing works on OoD detection, the fact then that our benchmarks have been created without the use of any external OoD dataset then indicates that *one might not need an OoD dataset to measure OoD detection performance*. In summary, the contributions of this chapter are as follows:

1. We identify desirable properties of OoD samples from the lens of perceptual and semantic similarity to iD samples. We define existing concepts, such as distribution shift, Near and Far OoD through through this lens.
2. We design loss function regularizers for GANs, which capture perceptual and semantic similarity, with the goal to generate Shifted and Near OoD samples given a training set.
3. We conduct thorough experiments across model architectures, OoD detection baselines and OoD detection benchmarks on 3 datasets: ImageNet, CIFAR-10/100 and MNIST, and show that our generated samples can provide a significantly tougher benchmark while also being reliable for assessing OoD detection performance.
4. To the best of our knowledge, our method is a novel way to generate benchmarks for the evaluation of any OoD detection method without using any external OoD samples. This can be particularly useful for niche use cases where proxy OoD datasets do not exist to evaluate performance.

5.2 Related Work

State-of-the-art on OoD Detection: The problem of OoD detection has been tackled from different angles like uncertainty quantification [49] and open set recognition [213]. As mentioned in §5.1, a popular approach is uncertainty quantification, which naturally serves as a solution for OoD detection as OoD inputs should intuitively be assigned higher uncertainty. A well-known thread of work uses the softmax distribution from a neural network to capture uncertainty. This starts from [177] where the authors simply use the softmax probability and

continues on to several methods including augmentations [192, 193, 221], calibration [56, 77], modified activation functions [222], logit normalization [223] and energy based models [195]. While these methods have a low computational overhead, the softmax distribution often fails to capture epistemic uncertainty [78] and is overconfident on incorrect predictions for OoD inputs [48]. A more principled approach uses the Bayesian formalism [102] and is applied to deep neural nets using approximate Bayesian inference methods [109, 111, 224]. Yet another popular uncertainty quantification method, often interpreted as a form of Bayesian Model Averaging [112], is deep ensembles [49] which uses an ensemble of neural networks and averages the softmax distributions to compute uncertainty. Deep ensembles and its modifications [113, 114] are often widely accepted as one of the the state-of-the-art methods for uncertainty quantification. Both approximate Bayesian inference and deep ensembles however, require either multiple forward passes at test time or multiple models to make predictions, thereby being computationally expensive. Attempting to achieve ensemble level performance from a single deterministic model, DUQ [57], SNGP [58, 225] and DDU [116], develop distance-aware deterministic models which can quantify uncertainty. In addition to the above, there are a set of works specifically targeted for OoD detection including using gram matrices [226], feature space singularity [227] and virtual logit matching [92]. Finally, [94] show that pre-trained Vision Transformers, when fine-tuned on a downstream dataset, achieve state-of-the-art AUROC scores on conventional OoD benchmarks. [83] contains a comprehensive set of OoD detection methods.

OoD evaluation procedure: OoD samples are generally one of two types: i) *distribution shifted* samples and ii) samples which belong to an *unseen category* which the model hasn't been trained on. For evaluation, the general practice is to use separate OoD datasets for testing. For shifted samples, some of the well-known datasets include ImageNet-C (corrupted) and ImageNet-P (perturbed) [203] which use synthetic corruptions and perturbations as well as stylised versions of ImageNet like ImageNet-R [86], ImageNet-Sketch [87] etc. There are also datasets containing specific real-world shifts like WILDS [215], Backgrounds [214], colored MNIST [228] etc. As shifted datasets retain the label information of the original

dataset, models are evaluated on their calibration error [161], which compares the model confidence with its accuracy on the provided test set. On the other hand, to test a model's performance on unseen categories, the convention is to use pairs of (iD vs OoD) datasets and measure how well a model is able to tell apart OoD from iD data using scores like AUROC. Most works in the current literature use MNIST [14] vs Fashion-MNIST [216], CIFAR-10/100 vs SVHN & CIFAR-100/10 and ImageNet [12] vs ImageNet-O [91] as the standard benchmarks for unseen distributions. Recently, [93] released the Species dataset as an OoD dataset for ImageNet-21K. However, the choice of these datasets is relatively arbitrary and a lot of the current OoD benchmarks including CIFAR-10 vs SVHN/CIFAR-100 and MNIST vs Fashion-MNIST are saturated [94]. In this work, we thus show that generating OoD samples given a training set can produce significantly more challenging benchmarks for even some of the state-of-the-art OoD detection methods.

Generative models for OoD: Generative models have been previously used to generate samples on the boundary of image classes [193, 229–232]. These include GAN based models [193, 229, 230], latent space sampling [231] and even text-to-image generators [232]. GANs in particular have often been employed to train their discriminators for anomaly detection [99, 100]. The purpose of these works however is to use generated samples during training to improve the performance on OoD detection [95, 97] or the general robustness of discriminative models to distribution shift. As mentioned before, with the lack of a clear definition of distance in image space, it is difficult to encode different types of OoD in a GAN and this is where one of our primary contributions lies. Secondly, our motivation is also orthogonal to these works. We use a GAN to improve the evaluation of OoD detection rather than improve the OoD detection methods themselves.

5.3 Method

In this section, we formalise our approach to generate shifted and near OoD samples given a training set. First, we encode perceptual and semantic similarity as quantifiable loss functions

for generative models. Then we discuss the GAN architectures and objective functions to generate shifted and Near OoD samples respectively.

Perceptual similarity as a loss function As mentioned in §5.1, perceptual similarity between images represents how visually similar they are. Since we have target images from the training set, we can use a Full-Reference Image Quality Assessment (FR-IQA) [233] metric to encode perceptual loss. There exist several FR-IQA metrics in the literature like SSIM [234], FSIM [235] and LPIPS [236] but we use the *Learned Perceptual Image Patch Similarity* (LPIPS) [236] as it is known to correlate with human judgement well. Let f_θ represent a pre-trained convolutional network. Given two images, $\mathbf{x}_1$ and $\mathbf{x}_2$, the LPIPS computes the cosine distance between feature space activations of $\mathbf{x}_1$ and $\mathbf{x}_2$ across different layers of the network f_θ as shown below:

$$\mathcal{L}_{\text{LPIPS}}(\mathbf{x}_1, \mathbf{x}_2) = \sum_l \left(\frac{1}{H_l W_l} ||f_\theta^l(\mathbf{x}_1) - f_\theta^l(\mathbf{x}_2)||_2^2 \right) \tag{5.1}$$

where $f_\theta^l(\mathbf{x}_1)$ and $f_\theta^l(\mathbf{x}_2) \in \mathbb{R}^{H_l, W_l, C_l}$ are the feature space representations from inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ in layer l of the network. In this work, we use LPIPS with a VGG, although other architectures can be used as well. From here on out, we represent perceptual loss as $\mathcal{L}_{\text{LPIPS}}$. Minimizing $\mathcal{L}_{\text{LPIPS}}(\mathbf{x}_1, \mathbf{x}_2)$ encourages images $\mathbf{x}_1$ and $\mathbf{x}_2$ to be perceptually similar and vice versa.

Semantic similarity as a loss function In image classification, the semantic meaning of an image is encoded in its class label. To identify if an image is semantically similar to the training distribution, we need a model which understands the semantic meaning of the training distribution. However, a single classifier can make incorrect and confident predictions on inputs [56], especially when they are not from any of the training classes [161]. In this work, we take inspiration from Bayesian literature [102] and quantify semantic similarity, using the *mutual information* (MI) $\mathbb{I}[y, \theta|\mathcal{D}, \mathbf{x}]$, (also known as *information gain*) [48] between the posterior distribution over parameters of a Bayesian model $p(\theta|\mathcal{D})$ and its predicted distribution over classes $p(y|\mathbf{x}, \theta)$. Let $(\mathbf{x}_g, y_g)$ be an input and $\mathcal{S}$ be the set of training classes. If $y_g \in \mathcal{S}$, i.e., the sample belongs to one of the training set classes,

seeing the sample $(\mathbf{x}_g, y_g)$ won't cause much information gain about the posterior $p(\theta|\mathcal{D})$. On the other hand, $y_g \notin \mathcal{S}$ will cause a high information gain about the posterior and $\mathbb{I}[y, \theta|\mathcal{D}, \mathbf{x}]$ will be high. Thus, in order to generate semantically similar/dissimilar images, we intuitively want $\mathbb{I}[y, \theta|\mathcal{D}, \mathbf{x}]$ to be low/high.

Due to the computational intractability of Bayesian inference in deep learning, we use a pre-trained deep ensemble which can be seen as a way to perform Bayesian Model Averaging [112] with each model in the ensemble being a sample from the posterior. Following [48], we approximate MI as the difference between the entropy of the average softmax distribution of the ensemble and the average entropy of the softmax distributions of each individual network in the ensemble. Let $p(y|\mathbf{x}, \theta_t)$ represent the softmax distribution produced by the t^{th} network in an ensemble of T networks, on an input $\mathbf{x}$. The average softmax distribution for the ensemble is $p(y|\mathbf{x}, \theta) = \frac{1}{T}\sum_{t=1}^{T} p(y|\mathbf{x}, \theta_t)$. Then, the MI for the ensemble on input $\mathbf{x}$ can be approximated as:

$$\mathcal{L}_{\text{MI}}(\mathbf{x}) = \hat{\mathbb{I}}[y, \theta|\mathcal{D}, \mathbf{x}] \approx \mathbb{H}[p(y|\mathbf{x}, \theta)] - \frac{1}{T}\sum_{t=1}^{T} \mathbb{H}[p(y|\mathbf{x}, \theta_t)] \tag{5.2}$$

where $\mathbb{H}[.]$ represents the entropy of a distribution. We use $\mathcal{L}_{\text{MI}}$ of a pre-trained ensemble as the semantic loss to quantify semantic similarity of an image to the training distribution. Semantically similar images (eg., images belonging to training classes) should have low $\mathcal{L}_{\text{MI}}$ and vice versa.

Generative Model Having defined perceptual and semantic similarity as quantifiable loss functions, we discuss two different GAN architectures for the two different types of OoD data. For distribution shift, we intend to learn a transformation on iD data and hence, use a conditional GAN architecture, Pix-2-Pix [220] in particular. For Near OoD, we intend to learn a distribution instead and hence, use a standard GAN. We do not change the loss for the discriminator in any of the GANs and only regularise the loss for the generator to produce the desired OoD type.

Distribution Shift: For distribution shift, we want to maximize the perceptual loss $\mathcal{L}_{\text{LPIPS}}$ (to generate perceptually different images) and minimize the semantic loss $\mathcal{L}_{\text{MI}}$ (to preserve

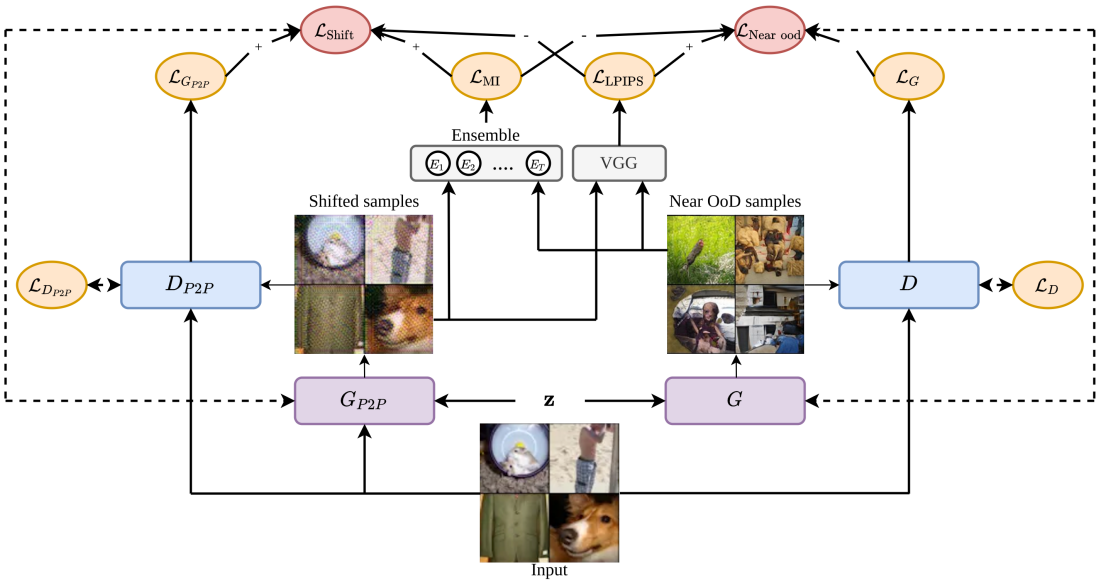


Figure 5.3: Schematic of the proposed method to generate Shifted and Near OoD samples. G_{P2P} and D_{P2P} represent the Pix-2-Pix to generate shifted samples and G and D represent the GAN to generate Near OoD images. The dotted lines show paths for gradient propagation.

the semantic meaning of generated images). Thus, the objective for the Pix-2-Pix generator is:

$$\begin{aligned}\mathcal{L}_{\text{Shift}} = & \underbrace{\mathbb{E}_{\mathbf{x},\mathbf{z}}[\log(1 - D_{P2P}(\mathbf{x}, G_{P2P}(\mathbf{x}, \mathbf{z})))]}_{\text{Pix-2-Pix Generator Loss}} \\ & - \underbrace{\lambda_p \mathbb{E}_{\mathbf{x},\mathbf{z}}[\mathcal{L}_{\text{LPIPS}}(\mathbf{x}, G_{P2P}(\mathbf{x}, \mathbf{z}))]}_{\text{maximize perceptual loss}} \\ & + \underbrace{\lambda_s \mathbb{E}_{\mathbf{x},\mathbf{z}}[\mathcal{L}_{\text{MI}}(G_{P2P}(\mathbf{x}, \mathbf{z}))]}_{\text{minimize semantic loss}},\end{aligned} \tag{5.3}$$

where G_{P2P} and D_{P2P} represent the generator and discriminator of the Pix-2-Pix GAN and λ_p and λ_s are the regularisation coefficients for the perceptual and semantic losses.

Near OoD: Similarly, for Near OoD, we want to minimize the perceptual loss $\mathcal{L}_{\text{LPIPS}}$ (to encourage perceptually similar images) and maximize the semantic loss $\mathcal{L}_{\text{MI}}$ (to generate semantically different images which don't belong to training set classes). We use a GAN

for Near OoD distributions with the generator objective as shown below:

$$\begin{aligned}\mathcal{L}_{\text{Near ood}} = &\underbrace{\mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]}_{\text{GAN Generator Loss}} \\ &+ \underbrace{\lambda_p \mathbb{E}_{\mathbf{x},\mathbf{z}}[\mathcal{L}_{\text{LPIPS}}(\mathbf{x}, G(\mathbf{z}))]}_{\text{minimize perceptual loss}} \\ &- \underbrace{\lambda_s \mathbb{E}_{\mathbf{z}}[\mathcal{L}_{\text{MI}}(G(\mathbf{z}))]}_{\text{maximize semantic loss}},\end{aligned} \tag{5.4}$$

where G and D denote the generator and discriminator of the GAN respectively. Figure 5.3 shows a schematic of our model along with the loss functions.

5.4 Experiments and Discussion

5.4.1 Implementation Details

Setting λ_p and λ_s and MI thresholding: In Equation (5.3) and Equation (5.4), we introduce the regularisation coefficients λ_p and λ_s. To set these, we generate samples using different combinations of λ_s and λ_p. We then measure the MI on the training ensemble for all generated samples and filter them such that MI is neither too high (avoid samples which are too dissimilar), nor too low (samples which coincide with iD). The lower bound is selected to be the lowest value which minimizes MI overlap between val and near OoD sets and the upper bound is chosen to be lower bound + 0.4 (see Figure C.3.1 in appendix). In particular, we use $[0.1, 0.5]$ for MNIST, $[0.2, 0.6]$ for CIFAR-10 (C10) and $[0.4, 0.8]$ for CIFAR-100 (C100) and ImageNet. We find these settings to empirically produce good OoD samples across datasets.

Ensemble for $\mathcal{L}_{\text{MI}}$: We implement the semantic loss $\mathcal{L}_{\text{MI}}$ on MNIST using an ensemble of 4 different networks: LeNet [14], AlexNet [59], VGG-11 [61] and ResNet-18 [15]. On C10/100, we use 6 networks: DenseNet-121 [170], ResNet-50/110, VGG-16, Wide-ResNet-28-10 and Inception-v3 [237] and on ImageNet, we use 3 networks: ResNet-18, MobileNet-v3-Large [238] and EfficientNet-B0 [239]. All the ensemble models are trained on their respective training sets.

Generating Near OoD Datasets We use a GAN to generate Near OoD samples from the training set using the $\mathcal{L}_{\text{Near ood}}$ loss. In particular, for MNIST, we use DCGAN for its simplicity and on C10/100 and ImageNet, we use BigGAN due to its superior performance in terms of FID scores. For all the GANs, we set λ_p to 1 and perform a grid search for λ_s over the interval $[0.5, 3]$ at steps of 0.25. As mentioned above, we use all the resulting trained GANs and filter samples out by thresholding on MI for the ensemble.

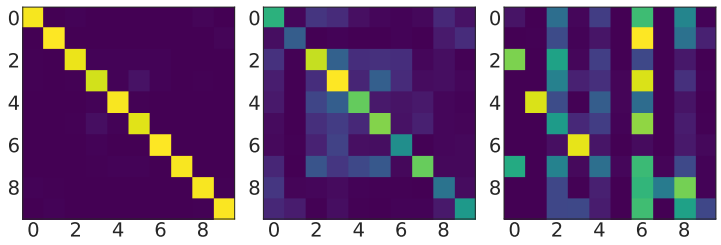
Generating Shifted Datasets We generate shifted samples using a Pix-2-Pix model trained on $\mathcal{L}_{\text{shift}}$ loss. After a grid search over different values, we found $\lambda_s = 1$ and $\lambda_p = 2$ to produce the best results on C10 and ImageNet. Further training details can be found in Appendix C.1. We show qualitative examples of both Near OoD and Shifted samples in Figure 6.1 and additional samples in Appendix C.4.

5.4.2 Sanity Check on Benchmarks

We perform a sanity check using CIFAR-10 to verify our claims on the generated samples: i.e., Near OoD samples are semantically dissimilar and perceptually similar while Shifted samples are semantically similar and perceptually dissimilar to the training set.

Semantic similarity to the training distribution To verify that Near OoD samples are semantically dissimilar and Shifted samples are semantically similar to iD, we compare the predictions of an ensemble of 6 models: DenseNet-121, ResNet-50/110, VGG-16, Wide-ResNet-28-10 and Inception-v3 on the CIFAR-10 test set with both near OoD and shifted versions of CIFAR-10. Note that the ensembles used for testing are different from the ones used in training. We present the corresponding confusion matrices in Figure 5.4. Clearly, in case of shifted samples, the label information is preserved as the ensemble's predicted classes broadly match the correct labels. Such is not the case, however, for near OoD samples where the predictions for each class are mostly incorrect, indicating that the dataset has not preserved the label information from the training distribution.

Perceptual similarity to the training distribution To measure perceptual similarity between images, we use the well-known FID score [240]. Again, note that the FID score



(a) iD **(b)** Shifted **(c)** Near OoD

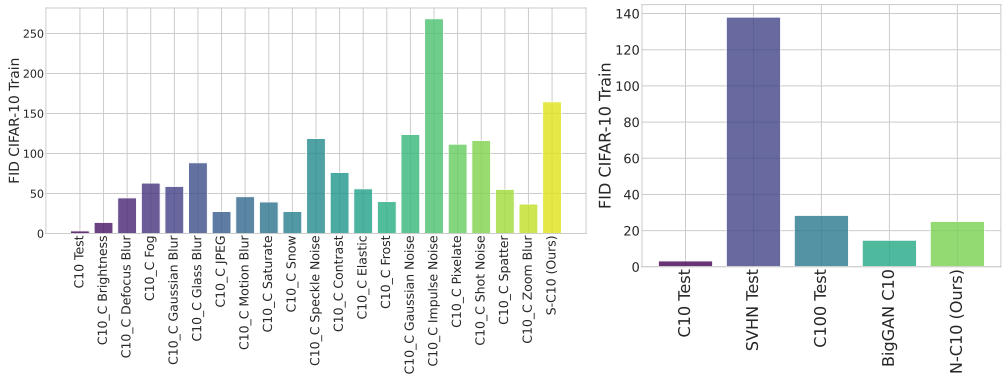
Figure 5.4: Confusion matrices for ensemble predictions on versions of CIFAR-10. The ensemble broadly predicts Shifted samples correctly unlike Near OoD samples.

is an independent metric which has not been used during our training. It is normally used to evaluate generative models on the perceptual quality of their outputs. We show the FID between the CIFAR-10 training set and Shifted (S) CIFAR-10 comparing with all the corruption types at intensity 5 of CIFAR-10-C [203]. Similarly, we compare the FID of Near OoD (N) CIFAR-10 with SVHN, CIFAR-100 and samples generated by a BigGAN trained on CIFAR-10. We present the results in Figure 5.5. It is evident that S CIFAR-10 has a very high FID score indicating perceptual dissimilarity from the training set. Note however, that it is not the highest among all the corruption types in CIFAR-10-C. On the other hand, N CIFAR-10 has a significantly lower FID than SVHN and slightly lower than CIFAR-100, providing evidence in favour of the fact that N CIFAR-10 is perceptually more similar to the training set as compared to CIFAR-100 or SVHN.

5.4.3 Evaluating Near OoD Datasets

Having sanity checked our claims on the perceptual and semantic similarity of generated samples, we evaluate Near OoD samples using two experiments. Firstly, we use generated Near OoD datasets obtained from MNIST, CIFAR-10/100 and ImageNet and compare them with conventional OoD benchmarks using multiple well-known OoD detection baselines. Secondly, in Appendix C.2 we also compare performance on models trained using outlier exposure [98] on our datasets with conventional ones.

OoD Detection: Architectures & Baselines For models trained on ImageNet, we use 6 different architectures: 2 convolutional models, ResNet-50 and Wide-ResNet-50-2, and 4



(a) Shifted **(b)** Near OoD

Figure 5.5: FID scores between CIFAR-10 and generated OoD sets. FID of Shifted CIFAR-10 is higher than most corruption types in CIFAR-10-C, indicating perceptual dissimilarity from CIFAR-10, whereas FID of N CIFAR-10 is significantly lower than SVHN and even lower than CIFAR-100 indicating perceptual similarity to CIFAR-10 samples.

vision transformers: ViT-B/16, ViT-B/32, ViT-L/16 and ViT-L/32. For each of the Vision transformers, we evaluate 5 well-known OoD detection baselines: softmax confidence, entropy [177], predictive entropy of a 5-ensemble [49], MaxLogit [93] and Energy [195] based OoD detection. Additionally, we also evaluate Mahalanobis distance [179] as a baseline for the 2 convolutional models. Note that the predictive entropy of a deep ensemble is the entropy of the average softmax distribution (first term in Equation (5.2)). The ensemble used to compute $\mathcal{L}_{\text{MI}}$ during training contains models with different architectures to encourage variability in predictions. During evaluation however, we follow standard procedure and use the same architecture for each model in the ensemble.

For CIFAR-10/100, we use 10 different model architectures: 6 convolutional, DenseNet-121, ResNet-50/110, VGG-16, Wide-ResNet-28-10, Inception-v3 and the 4 ViT based models mentioned above. On each of these architectures, we evaluate 3 OoD detection methods: softmax confidence, entropy and predictive entropy of a 5-ensemble.

Competitive Benchmarks For comparison, we use conventionally used OoD detection benchmarks. In particular, we compare with (MNIST vs Fashion-MNIST), (CIFAR-10 vs SVHN/CIFAR-100), (CIFAR-100 vs SVHN/CIFAR-10) and (ImageNet vs ImageNet-O [91]/ Texture [241]/ Species [93]/ OpenImage-O [92]). We present the test set accuracies and

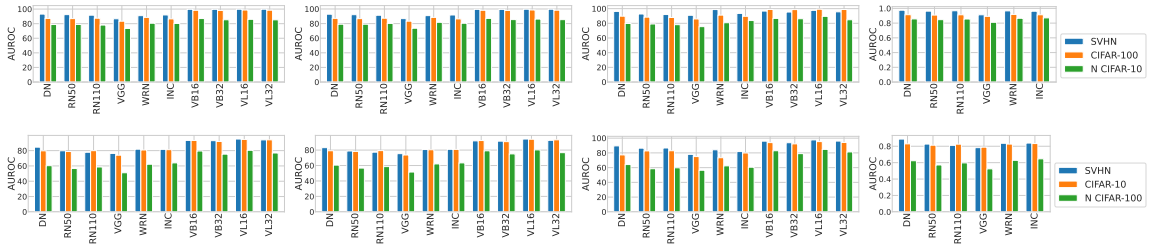
Architecture	Baseline	AUROC % ImageNet-O	Species	OpenImage-O	Texture	N-ImageNet (Ours)
ViT-B-16	SC	82.13	59.8	90.8	90.2	75.72
	SE	89.15	63.2	94.5	94.6	80.63
	Ensemble	95.1	75.6	98.2	98.3	88.12
	MaxLogit	90.3	68.1	95.4	95.6	82.1
	Energy	92.0	70.02	96.0	97.9	84.15
ViT-B-32	SC	75.96	53.4	85.4	87.8	74.2
	SE	78.94	56.5	87.3	89	76.98
	Ensemble	88.1	67.3	95.6	97	85.2
	MaxLogit	82.1	61.6	90.2	93.2	80.1
	Energy	85.6	63.5	93.0	94.0	83.21
ViT-L-16	SC	90.69	64.1	93.3	95.8	82.6
	SE	92.36	66.23	94.0	96.71	84.7
	Ensemble	95.8	77.3	98.62	99.2	88.6
	MaxLogit	94.2	72.1	96.8	97.8	86.8
	Energy	93.5	70.4	95.1	96.8	85.3
ViT-L-32	SC	86.91	56.6	88.6	90.3	80.06
	SE	89.51	60.2	91.0	94.2	81.65
	Ensemble	95.6	70.1	97.2	97.0	88.1
	MaxLogit	94.1	63.4	94.0	95.2	84.2
	Energy	92.8	62.1	93.2	94.7	83.0
RN50	SC	52.8	51.34	58.2	61.4	53.4
	SE	53.1	52.27	60.8	62.7	54.2
	Maha	50.1	50.3	55.4	54.3	51.1
	Ensemble	62.5	59.8	67.4	68.3	60.1
	MaxLogit	58.4	56.1	63.5	67.4	57.45
	Energy	57.6	55.3	63.2	66.2	56.8
WRN-50-2	SC	54.6	52.1	59.5	63.4	55.3
	SE	57.0	54.3	61.8	66.1	57.6
	Maha	53.1	50.6	57.2	57.6	52.8
	Ensemble	64.2	61.3	69.23	71.4	62.4
	MaxLogit	60.1	57.87	66.4	69.2	58.2
	Energy	58.5	57.1	65.3	67.8	57.8

Table 5.1: AUROC% of OoD detection on 4 transformer and 2 convolutional architectures trained on ImageNet using ImageNet-O, Species, OpenImage-O, Texture and Near OoD ImageNet (N-ImageNet) as OoD.

AUROC scores for MNIST in Table C.3.1 of the appendix, CIFAR-10/100 and ImageNet test accuracies in Table C.3.2 of the appendix, corresponding AUROC scores for CIFAR-10/100 shown as bar plots in Figure 5.6 and finally, AUROC scores for ImageNet models in Table 5.1. Related AUPRC scores are shown Appendix C.3.

Observations Our observations are as follows:

1. *Except for the ImageNet vs Species benchmark, AUROC & AUPRC for all model architectures across all training datasets and all baselines are lower for our Near OoD samples as compared to conventionally used OoD datasets*. This brings into question, the performance of baselines which are considered to be robust to OoD inputs, as evidenced



(a) Entropy **(b)** Confidence **(c)** Mahalanobis **(d)** Ensemble

Figure 5.6: AUROC % for different models, DenseNet-121 (DN), ResNet-50 (RN50), ResNet-110 (RN110), VGG-16, Wide-ResNet-28-10 (WRN) and Inception-v3 (INC), ViT-B-16/32 (VB16/32) and ViT-L-16/32 (VL16/32) trained on CIFAR-10 (first row) and CIFAR-100 (second row) using SVHN, CIFAR-10/100 and Near OoD (N) CIFAR-10/100 as OoD datasets.

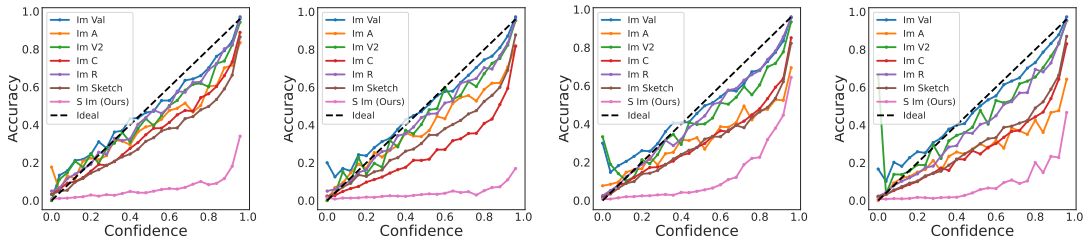
from their performance on standard benchmarks. It also provides evidence in favour of more challenging OoD detection benchmarks for evaluation which are far from saturation.

2. *The order of performance is preserved.* We consistently observe that if a model M_1 outperforms a model M_2 on our Near OoD benchmark, it broadly outperforms M_2 on all conventionally used real-world benchmarks and vice-versa. To see this, we present the normalized AUROC scores for ImageNet trained models (in Table 5.1) as a radar plot in Figure 5.8.

Discussion & Visualisation Firstly, as mentioned before, quite a few conventional OoD benchmarks (like CIFAR-10 vs SVHN) are nearing saturation and might become redundant in future for evaluating OoD detection methods. Our Near OoD datasets are much more effective at measuring performance as all the baselines broadly underperform in our setup. Secondly, even though the Near OoD samples might not look like real-world objects, the fact that the ordering of performance is consistently preserved between our benchmarks and conventional ones implies their potential to estimate real-world OoD detection performance. We visualise Near OoD samples from 10 ImageNet classes in Figure 5.9b and observe that such images contain patches from original classes in an odd order which makes the images unrecognizable, while still preserving close perceptual proximity to original classes. We find this to be true in general for other classes too. Thus, we posit that *it is not necessary for an OoD image to represent a real-world object as long as it captures certain desirable*

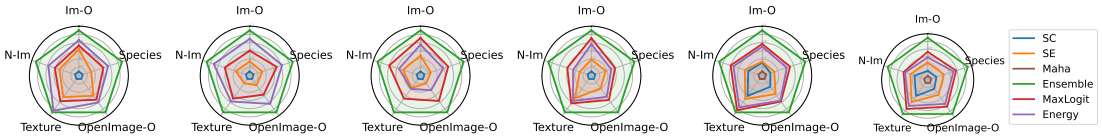
Model	Im-val	Im-A	Im-v2	Im-C		Im-R	Im-Sketch	S Im (Ours)
		ECE %		*Avg ECE %*	*Max ECE %*		*ECE %*	
ViT-B-16	3.62	14.16	7.43	11.18	18.61	5.38	15.44	**26.15**
ViT-B-32	3.70	23.13	8.15	11.32	19.21	7.69	17.64	**29.15**
ViT-L-16	2.35	12.67	7.30	9.42	13.44	4.79	14.97	**22.76**
ViT-L-32	2.51	13.20	7.62	11.03	15.76	4.85	15.12	**23.54**

Table 5.2: ECE % on standard ImageNet (Im) shifts compared to our Shifted ImageNet (S Im).



(a) VB16 **(b)** VB32 **(c)** VL16 **(d)** VL32

Figure 5.7: Reliability plots for ImageNet shifts



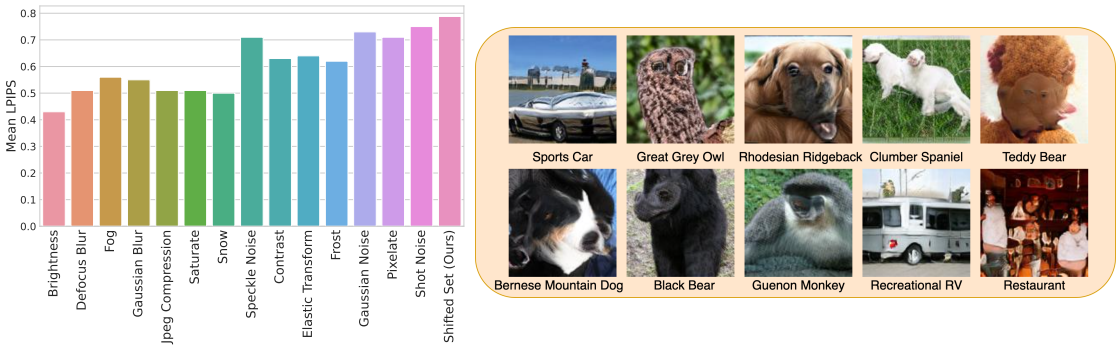
(a) VB16 **(b)** VB32 **(c)** VL16 **(d)** VL32 **(e)** RN50 **(f)** WRN

Figure 5.8: Ordering of AUROC scores of OoD detection baselines on 4 ViT and 2 convolutional architectures trained on ImageNet-1K using OoD sets: ImageNet-O, Texture, Species, OpenImage-O and our Near OoD ImageNet (N-Im).

properties. For near OoD, the desirable property is to be semantically dissimilar while lying in the close perceptual vicinity of the training distribution. Thus, our method provides a relatively easy alternative to gathering real-world OoD samples even for datasets where conventional benchmarks cannot be used.

5.4.4 Evaluating Shifted Datasets

In the second part of our experiments, we evaluate our method of learning distribution shifts by comparing the shifted datasets with other well-known synthetic and real-world shifts. For CIFAR-10, we compare with CIFAR-10-C [203] and for ImageNet, we compare with synthetic shifts: ImageNet-C, ImageNet-R (renditions) [86], ImageNet-Sketch [87] and real-world shifts: ImageNet-A [91] and ImageNet-V2 [242]. For CIFAR-10-C and



(a) **(b)**

Figure 5.9: (a) **average LPIPS between ImageNet val and corrupted/shifted samples**, (b) **Near OoD samples from 10 ImageNet classes.** We observe the Near OoD samples to contain patches from their respective original classes but in an odd order, making the images unrecognizable compared to the original class.

ImageNet-C, we use corrupted images at the highest intensity 5.

We report the Expected Calibration Error (ECE) in Table C.4.5 of the appendix for CIFAR-10 models and Table 5.2 for ImageNet. Detailed results for each corruption type can be found in Appendix C.3. For each model, the ECE for every competitive dataset is starkly lower than the ECE obtained on our Shifted datasets.

To better understand the effect of learned shifts, we compute the VGG LPIPS between each image in the ImageNet val set and the corresponding corrupted image obtained from using transformations in ImageNet-C (eg: brightness, Gaussian noise etc.). For the same validation image, we also compute the LPIPS with the transformed image using our Shift method. This is different from the FID analysis in Figure 5.5 as unlike FID, here, we are computing difference between individual pairs of images. Note that we ensure that the L2 norm of the difference between normal and corrupted images in the image space is same (set to 50) for all corruption types. We present the average LPIPS for each corruption type in Figure 5.9a. Although the input space L2 norm between image pairs remains the same for all corruption types, the high LPIPS for our shifted samples indicates that the corruptions learnt in our shifted dataset are again, not random transformations, rather they seem to be optimized to decrease feature space similarity (or increase feature space distance) between generated

images and their original counterparts, in a neural network. At the same time, minimizing an ensemble's MI in $\mathcal{L}_{\text{Shift}}$ encourages the Pix-2-Pix model to learn transformations which make the neural network confident on its predictions. Such a transformation should therefore lead to inaccurate but confident, and thereby miscalibrated predictions from pre-trained models. To empirically verify this, we present reliability plots for Shifted-ImageNet vs all other shifts in Figure 5.7. Clearly, for Shifted ImageNet, as expected, models consistently have lower accuracy as compared to confidence, i.e., they are overconfident and miscalibrated, thereby providing evidence in favour of our claim above.

5.4.5 Evaluation of OoD detection in real life

Having thoroughly evaluated our method, in order to use them in real-life applications, we need to verify if we can compare baselines sensibly using our benchmarks. Indeed that is the very purpose of a benchmark. We make the observation here that in all our experiments across datasets, the ordering of performance between baselines is broadly consistent between our benchmarks and conventional ones. *Models which perform well on our benchmarks also perform well on conventional benchmarks and vice versa.* We show this consistency for our ImageNet baselines in a set of radar plots in Figure 5.8 (using observations in Table 5.1), where we have normalized the AUROC scores of all baselines to lie between $[0.1, 1.1]$. Assuming that performance on conventional OoD detection benchmarks generalises to and is indicative of real-world OoD detection performance, we can then use the above observation as validation for our proposed benchmarks. Interestingly, our observations also indicate that we might not need an OoD dataset in the first place to evaluate a model's OoD detection performance. We could reliably estimate the OoD detection performance of any model just from the training set by following our proposed method. This has implications when working with new in-distribution training sets where OoD detection performance cannot be judged using conventional benchmarks. Even in such cases, our method can be used to provide a reasonable estimate of OoD detection performance.

5.5 Conclusion

Reliably evaluating the behaviour of models in unknown scenarios is an open problem and is extremely difficult because of infinitely many situations that could potentially fulfill our notion of “unknown" with respect to in-distribution. To our knowledge, our work is the first step in the direction where we do not advocate the naive approach of testing on arbitrarily chosen “other” datasets. Rather we propose to learn the distribution of samples satisfying constraints mimicking our notion of what is out-of-distribution. Through numerous experiments, we show that our generated samples provide for a more challenging and reliable benchmark for popularly used, state-of-the-art OoD detection baselines. In future, we want to explore how our approach can be extended to text-to-image generative models and if we can encode desirable out-of-distribution properties in natural language text. We thus hope that our work leads to an improvement in the standard procedure of evaluating OoD detection methods.

6

Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning

Contents

6.1 Introduction 97
6.2 Related Work 100
6.3 Patch Level Alignment in CLIP 101
6.4 Semantic Coherence in Vision Encoders 103
6.5 Patch Aligned Contrastive Learning (PACL) 105
6.6 Experiments and Discussion 108
6.6.1 Zero-shot Semantic Segmentation 108
6.6.2 Image Classification 114
6.7 Conclusion 115

Abstract

We introduce *Patch Aligned Contrastive Learning (PACL)*, a modified compatibility function for CLIP's contrastive loss, intending to train an alignment between the patch tokens of the vision encoder and the CLS token of the text encoder. With such an alignment, a model can identify regions of an image corresponding to a given text input, and therefore transfer seamlessly to the task of open vocabulary semantic segmentation without requiring any segmentation annotations during training. Using pre-trained CLIP encoders with PACL, we are able to set the state-of-the-art on the task of open vocabulary zero-shot segmentation on 4 different segmentation benchmarks: Pascal VOC, Pascal Context, COCO Stuff and ADE20K. Furthermore, we show that PACL is also applicable to image-level predictions and when used with a CLIP backbone, provides a general improvement in zero-shot classification accuracy compared to CLIP, across a suite of 12 image classification datasets.

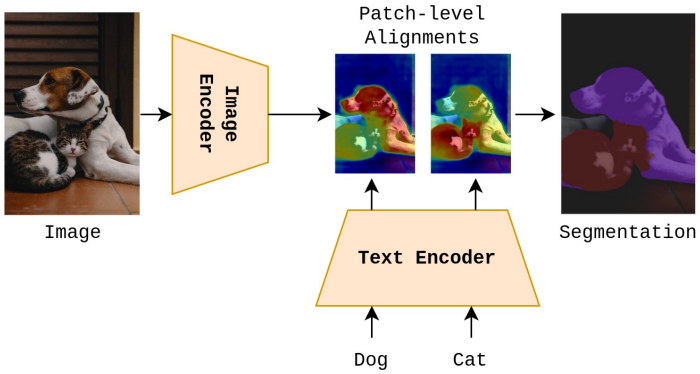


Figure 6.1: High level overview of our model. We train an alignment between the patch level embeddings from the image encoder and the CLS embedding from the text encoder. This alignment can then be used to perform open-vocabulary semantic segmentation in a zero-shot manner.

6.1 Introduction

Understanding the semantic content in visual scenes has been one of the most important problems studied in computer vision at various levels of granularity. Work on this problem has led to significant improvements along several threads including image level predictions like image classification [131, 243, 244], object level predictions like object detection [142, 245–249], as well as pixel level predictions like semantic segmentation [245, 246, 250, 251]. Although in image classification we require only a single label per image for prediction, for scene understanding at a higher level of granularity like segmentation, supervised training requires annotations at a pixel level. Such annotations require significant human effort and are often very expensive to obtain. This impedes training on a large scale with millions of images.

One way to tackle this problem could be to train models in an unsupervised manner without requiring any segmentation annotations. The best methods [252, 253] in this category exploit the similarity between internal representations of self-supervised image encoders [29]. This similarity is then used to identify and cluster similar regions of the image as segmentations. These models however are significantly outperformed by their fully supervised counterparts on most segmentation benchmarks.

Recent improvements in multi-modal foundation models has led to the possibility of training on very large scale datasets scraped off the internet [254]. These datasets contain

pairs of images and their corresponding natural language text descriptions. Models like CLIP [254], ALIGN [255], Florence [249] and CoCa [131] trained on such large internet scale datasets have been shown to transfer very well to several downstream tasks. Furthermore, having been trained on natural language textual descriptions, these models are often expected to recognize a wide variety of real-world visual concepts which can be expressed in natural language, a setting better known as *open vocabulary prediction*.

The natural question then is whether these multi-modal models can be used for pixel level predictions, i.e., semantic segmentation in the open vocabulary setting. Prior works on this topic [144–146, 256, 257] show that this is indeed possible. However, 3 of these works use either fully supervised segmentation annotations [257], class-agnostic segmentation masks [145] or a region proposal model trained using segmentation annotations [256], thereby being limited by the availability of expensive segmentation annotations/masks. To the best of our knowledge, only two models: ViL-Seg [257] and GroupViT [146] perform the task of open-vocabulary semantic segmentation while being trained solely on image-text data. Among these two, the better performer, GroupViT, defines a modified vision transformer (ViT) [123] architecture to naturally find semantic clusters within an image. Due to a different architecture, their model has to be trained end-to-end from scratch and cannot leverage pre-trained vision encoders.

In this work, we tackle the problem of open-vocabulary semantic segmentation without using any segmentation annotations or masks, with a model purely trained on image-text data. We start with the observation in [253] that self-supervised ViT models like DINO [29], have similar patch representations for semantically similar regions of an image. We find this observation to be true for CLIP's ViT based vision encoders as well. However, we also find that CLIP does not exhibit a patch level alignment between its vision and text encoders, primarily owing to the fact that its contrastive loss only aligns the CLS image and text tokens.

Inspired from previous work on contrastive learning for weakly supervised phrase grounding [258], we define a new compatibility function for contrastive loss to train an alignment between the patch tokens of the vision encoder and the CLS token of the text

encoder. In particular, we take the cosine similarity between the text CLS token and the vision patch tokens and use these similarities as weights to compute a weighted sum over vision tokens. The final compatibility function is then simply the cosine similarity between the weighted sum of the vision patch tokens thus obtained and the CLS text token. We find that models trained on our *Patch Aligned Contrastive Learning* loss indeed exhibit the desired patch level fine-grained alignment. Thus, at inference time, the compatibility function can be used to make image level predictions and the patch level alignment can be used for zero-shot transfer to semantic segmentation. A high level overview of our model is shown in Figure 6.1.

Note that unlike GroupViT, our PACL method is more flexible and general and can be used with any pre-trained ViT based encoders as well. We evaluate PACL with a pre-trained CLIP encoder on the task of zero-shot semantic segmentation using 4 different datasets: Pascal VOC [184], Pascal Context [259], COCO Stuff [260] and ADE20K [261]. On all 4 datasets, PACL consistently beats previous baselines [144–146, 257], even the ones which use segmentation annotations or segmentation masks for training. We also find that PACL trained on top of a CLIP backbone leads to a general improvement in zero-shot classification performance across a suite of 12 image classification datasets.

In a nutshell, our contributions are as follows. **Firstly,** we propose *Patch Aligned Contrastive Learning* (PACL), a modified compatibility function for contrastive loss in order to train an alignment between the patch representations of a ViT based vision encoder and the CLS text representation of a text encoder. We show that this alignment can be used to find regions within an image corresponding to a given text input and hence, can be used for zero-shot transfer to open-vocabulary semantic segmentation. **Secondly,** we show that PACL with a pre-trained CLIP encoder obtains state-of-the-art scores on zero-shot semantic segmentation across 4 different segmentation benchmarks: Pascal VOC, Pascal Context, COCO Stuff and ADE20K. **Finally,** PACL with a CLIP backbone also shows a general improvement in performance on zero-shot classification tasks across 12 different image classification datasets.

6.2 Related Work

In this section, we discuss some of the relevant works motivating our method.

Supervised semantic segmentation: Given an image, the task of semantic segmentation [38] involves classifying every pixel in the image to one of a fixed set of classes. Naturally, supervised datasets for semantic segmentation like Pascal VOC [184], ADE20K [261] and Cityscapes [262] contain images with class annotations for every pixel. A significant amount of work [183, 263–265] has been done to leverage these datasets and generate strong models for semantic segmentation. However, since annotating images at a pixel level is laborious and expensive, these datasets remain limited to a relatively small number of classes.

Unsupervised semantic segmentation: Identifying that the requirement of dense annotations is the problem, some works [252, 253, 266–269] have tried to leverage self-supervised techniques to train features which can be used for segmentation without requiring dense annotations. Notable among these works is STEGO [253] which uses the localized feature correspondences in self-supervised models like DINO [29] for the task of unsupervised segmentation. In our work, we study the existence of a similar feature correspondence in vision encoders of multi-modal models like CLIP [254] and use it to train a patch level alignment between image and text modalities. Note however, that it is still difficult for such unsupervised segmentation approaches to scale up to a large number of visual concepts.

Natural language supervision: Recently, the availability of datasets with millions of image-text pairs scraped from the internet has made it possible to train large-scale multi-modal fusion models. Such models [131, 249, 254, 255, 270, 271] are able to transfer well to several downstream tasks including vision-language pre-training (VLP) [272] tasks like image-text retrieval [273] and visual question answering [274], as well as vision specific tasks like zero-shot image classification [131, 254, 255] and object detection [142, 143]. Given the large-scale training of such multi-modal fusion models, it is natural to ask if these models can be leveraged to scale up the task of semantic segmentation and recognise a large number of visual concepts at a fine-grained level.

Natural language supervision for zero-shot segmentation: Some work has been done in this direction of using large-scale multi-modal models, like CLIP [254], for the task of semantic segmentation. For instance, LSeg [144] trains a segmentation model as its vision encoder and uses the frozen text encoder from CLIP to align pixel level embeddings with text. The resulting model is able to recognise conceptually similar labels which are not present within the training set. However, it trains the vision encoder in a fully supervised manner using segmentation annotations. OpenSeg [145] on the other hand is based on the ALIGN [255] model and trains using image-text data and class-agnostic segmentation annotations. ViL-Seg [257] trains using only image-text data with a vision based contrasting and a cross-modal contrasting objective along with an online clustering head to segment visual embeddings. Finally, GroupViT [146] proposes a modified ViT architecture which allows grouping semantically similar tokens into clusters useful for open vocabulary segmentation. To the best of our knowledge, ViL-Seg and Group-ViT are the only existing methods which solely use image-text data for training an open vocabulary semantic segmentation model. In our work, we propose a simple modification to the CLIP compatibility function for contrastive loss, which enables training an alignment between the patch tokens of a ViT based vision encoder and the CLS token of a text encoder. This alignment can then be seamlessly utilized for the task of semantic segmentation without using any segmentation annotations or class-agnostic segmentation masks during training.

6.3 Patch Level Alignment in CLIP

The contrastive training of CLIP ensures that the CLS tokens obtained from CLIP's transformer based vision and text encoders are aligned for similar image-text pairs. However, such an alignment between image and text at a patch level does not necessarily exist. To empirically study this, we use a semantic segmentation dataset, Pascal VOC [184], and classify each patch in the dataset to one of a fixed set of classes. The patch level vision tokens are classified using the same zero-shot classification [254] method normally used

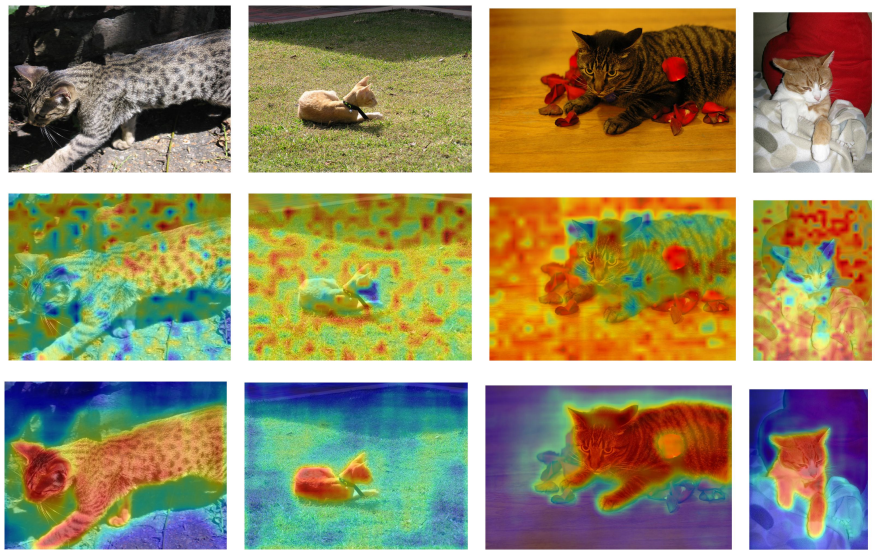


Figure 6.2: Patch level alignment between the word “cat" and images of cats. In the first row, we show the original images, in the second row, we show the patch level alignment in CLIP ViT-B/16 and in the third row, we show the alignment for our method.

on the CLS vision token. The classification accuracy, thus obtained, provides a measure of patch level alignment between the vision and text representations in the model, where a high classification accuracy indicates a high alignment and vice-versa.

More formally, let $\mathcal{D}_{\text{seg}} = (\mathbf{x}, \mathbf{y})_{i=1}^{N}$ be the semantic segmentation dataset where $\mathbf{x} \in \mathbb{R}^{C,H,W}$ and $\mathbf{y} \in \mathbb{R}^{H,W}$. We represent CLIP's vision and text encoders as $f_v : \mathbb{R}^{C,H,W} \rightarrow \mathbb{R}^{D_v}$ and $f_t : \mathbb{R}^{l} \rightarrow \mathbb{R}^{D_t}$ respectively. Similarly, let $e_v : \mathbb{R}^{D_v} \rightarrow \mathbb{R}^{D}$ and $e_t : \mathbb{R}^{D_t} \rightarrow \mathbb{R}^{D}$ be the linear embedders to project the vision and text encodings to the joint D dimensional space. Normally, for zero-shot classification, we measure the cosine similarity between the vision and text embeddings: $s(\mathbf{x}, c) = \frac{e_v(f_v(\mathbf{x}))}{|e_v(f_v(\mathbf{x}))|} \cdot \frac{e_t(f_t(c))}{|e_t(f_t(c))|}$ for each class name c and compute the predictive probability as: $p(c|\mathbf{x}) = \frac{e^{s(\mathbf{x},c)}}{\sum_{c'} e^{s(\mathbf{x},c')}}$. A simple modification to the vision encoder: $\hat{f}_v : \mathbb{R}^{C,H,W} \rightarrow \mathbb{R}^{T,D_v}$, where T is the number of tokens or patches, allows us to perform the same classification method on every patch.

In Table 6.1 second column (Pre-Alignment), we show the patch classification accuracy thus obtained for two CLIP models: ViT-B/16 and ViT-L/14. In Figure 6.2, first and second rows, we show qualitative samples of alignment on CLIP ViT-B/16, for 4 images of cats from

CLIP Vision Encoder	Patch Classification Accuracy *Pre-Alignment*	*Post-Alignment*
ViT-B-16	52.49	96.51
ViT-L/14	27.91	95.33

Table 6.1: Accuracy for patch level classification on Pascal VOC. For a pre-trained CLIP model, the accuracy is low indicating low patch level alignment between image and text. On applying our PACL alignment method, the accuracy significantly increases for both CLIP encoders indicating higher image text patch level alignment.

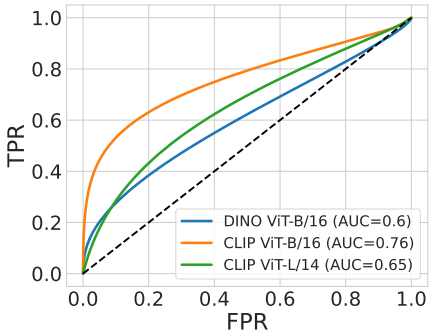


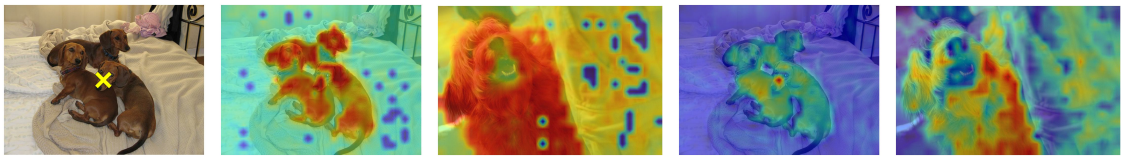
Figure 6.3: ROC curve indicating semantic coherence of CLIP and DINO vision encoders. CLIP encoders outperform DINO.

Pascal VOC. With a patch classification accuracy of 52.49% for ViT-B/16 and 27.91% for ViT-L/14, it is clear that the alignment we seek is very poor at the patch level. Surprisingly, note that for ViT-L/14, a model known to provide better image level prediction performance than ViT-B/16, the patch level alignment is significantly worse. Hence, *pre-trained CLIP models cannot be used for open vocabulary segmentation as the CLIP contrastive learning objective does not ensure patch level alignment between image and text modalities*.

6.4 Semantic Coherence in Vision Encoders

Due to the poor patch level alignment between pre-trained CLIP image and text encoders, our next question is whether we can train such an alignment in CLIP. This would however require the pre-trained vision encoder to be *sematically coherent*. In other words, semantically similar regions in an image should produce similar patch representations in the vision encoder. This property has been studied before in image self-supervised models like DINO [253]. We use a similar test to quantify semantic coherence of CLIP's vision encoders.

In particular, we collect all patch representations from the vision encoder for each image in Pascal VOC and store the corresponding target classes using the segmentation labels. Let $\hat{f}_v(\mathbf{x_1})_{i,j} \in \mathbb{R}^{D_v}$ and $\hat{f}_v(\mathbf{x_2})_{p,q} \in \mathbb{R}^{D_v}$ be the patch representations obtained at index (i, j) of image $\mathbf{x_1}$ and index (p, q) of image $\mathbf{x_2}$ respectively. We compute the cosine similarity



(a) (b) (c) (d) (e)

Figure 6.4: Qualitative results on semantic coherence between CLIP and DINO ViT-B/16. a): we show the original image of a dog class with the patch marker (yellow X near the centre). **b, c)**: we show CLIP vision encoder cosine similarity across all patches for the same and a different image of a dog. **d, e)**: we show the same for DINO. See more examples in Appendix D.2.1.

$\left(\frac{\hat{f}_v(\mathbf{x_1})_{i,j}}{|\hat{f}_v(\mathbf{x_1})_{i,j}|} \cdot \frac{\hat{f}_v(\mathbf{x_2})_{p,q}}{|\hat{f}_v(\mathbf{x_2})_{p,q}|}\right)$ between the patch representations and use this as a binary classifier to predict if the two patches have the same target label. Let the segmentation labels for the two patches be $l(\mathbf{x_1})_{i,j}$ and $l(\mathbf{x_2})_{p,q}$ respectively. Since we have labels for each pixel, we decide the label for each patch by majority-voting. The target value for binary classification is 1 if $l(\mathbf{x_1})_{i,j} = l(\mathbf{x_2})_{p,q}$, else 0. Note that performance on this binary classification task is indicative of semantic coherence, as a good classifier would require patch representations corresponding to same labels to have high cosine similarity and vice-versa.

We present the ROC curve and AUROC scores for CLIP and DINO in Figure 6.3. Surprisingly, we find that *CLIP's vision encoders outperform DINO on semantic coherence*[1]. This is encouraging as it indicates that we can indeed train a mapping between similar vision tokens and their corresponding text representations. We also present qualitative results in Figure 6.4 where we plot the patch level cosine similarity between a chosen patch (marked in yellow X in Figure 6.4a) and the remaining patches in the same image as well as a different image having the same class (dog). We do this for CLIP ViT-B/16 in Figure 6.4b and Figure 6.4c and for DINO ViT-B/16 in Figure 6.4d and Figure 6.4e. In both cases, CLIP's encoder seems to perform at par or better than DINO. Motivated by these observations, in the next section, we discuss a method to train a patch level alignment between the vision tokens and the CLS text token in CLIP using purely image-text data.

[1]CLIP's semantic coherence indicates that CLIP's vision encoders are good candidates for unsupervised segmentation approaches like STEGO [253], but further study of this feature is beyond the scope of this work.

6.5 Patch Aligned Contrastive Learning (PACL)

In the previous section, we showed that although CLIP lacks a patch level alignment between image and text representations, such an alignment can indeed be trained. However, note that this is a difficult problem as there is no ground-truth text data annotating each patch in an image-text dataset. Hence, training such an alignment can only be done in a weakly supervised fashion. Inspired from previous work on weakly supervised phrase grounding [258], in this section, we propose a modification on CLIP's contrastive loss, to learn an alignment between the vision patch tokens and the CLS text token.

A modified compatibility function for contrastive loss: Our method is simple in the sense that the only change we make to CLIP's training is in the compatibility function of its contrastive loss. Normally, for an image-text pair $(\mathbf{x}, \mathbf{y})$, CLIP computes the CLS vision and text embeddings as $e_v(f_v(\mathbf{x}))$ and $e_t(f_t(\mathbf{y}))$ respectively, where $f_v : \mathbb{R}^{C,H,W} \rightarrow \mathbb{R}^{D_v}$, $f_t : \mathbb{R}^{L} \rightarrow \mathbb{R}^{D_t}$ are the vision and text encoders and $e_v : \mathbb{R}^{D_v} \rightarrow \mathbb{R}^{D}$, $e_t : \mathbb{R}^{D_t} \rightarrow \mathbb{R}^{D}$ are the vision and text embedders to project the representations into the same dimensional space. The compatibility function $\phi(\mathbf{x}, \mathbf{y})$ is the cosine similarity between the vision and text CLS embeddings: $\phi(\mathbf{x}, \mathbf{y}) = \left(\frac{e_v(f_v(\mathbf{x}))}{|e_v(f_v(\mathbf{x}))|} \cdot \frac{e_t(f_t(\mathbf{y}))}{|e_t(f_t(\mathbf{y}))|}\right)$. Given this compatibility function, CLIP uses the InfoNCE [134] contrastive loss to learn vision and text representations which are aligned for similar image-text pairs:

$$\begin{aligned} \mathcal{L}_{\mathbf{x}} &= \frac{1}{k}\sum_{i=1}^{k}\left(\frac{e^{\phi(\mathbf{x_i},\mathbf{y_i})}}{\sum_{j=1}^{k} e^{\phi(\mathbf{x_i},\mathbf{y_j})}}\right) \\ \mathcal{L}_{\mathbf{y}} &= \frac{1}{k}\sum_{i=1}^{k}\left(\frac{e^{\phi(\mathbf{x_i},\mathbf{y_i})}}{\sum_{j=1}^{k} e^{\phi(\mathbf{x_j},\mathbf{y_i})}}\right) \end{aligned} \tag{6.1}$$

with the contrastive loss being $\mathcal{L}_{\text{InfoNCE}} = 1/2(\mathcal{L}_{\mathbf{x}} + \mathcal{L}_{\mathbf{y}})$.

Note that the above loss function produces an alignment between the CLS image and text tokens but as we observed in §6.3, it does not produce the desired alignment at patch level between vision and text encoders. In order to then train this alignment, we make the following changes to CLIP's loss. First, we use the patch tokens instead of the CLS token

from the vision encoder, $\hat{f}_v : \mathbb{R}^{C,H,W} \rightarrow \mathbb{R}^{T,D_v}$, where T is the number of tokens or patches. Next, we use a modified vision embedder $\hat{e}_v : \mathbb{R}^{T,D_v} \rightarrow \mathbb{R}^{T,D}$ to generate embeddings in the shared D-dimensional space for all patch tokens. We compute the patch level similarity

$$s(\mathbf{x}, \mathbf{y}) = \hat{e}_v(\hat{f}_v(\mathbf{x}))e_t(f_t(\mathbf{y})) \tag{6.2}$$

between all vision patch embeddings and the CLS text embedding, where $s(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^T$. We normalize the patch level similarity to the range $[0, 1]$ by applying a softmax function across tokens, $a(\mathbf{x}, \mathbf{y}) = \text{softmax}(s(\mathbf{x}, \mathbf{y}))$. Finally, we take a weighted sum across all vision patch embeddings where the weights of the tokens are obtained from the patch level similarities $a(\mathbf{x}, \mathbf{y})$ as:

$$\hat{v} = \hat{e}_v(\hat{f}_v(\mathbf{x}))^\intercal a(\mathbf{x}, \mathbf{y}) \tag{6.3}$$

where $\hat{v} \in \mathbb{R}^D$. The updated compatibility function $\hat{\phi}(\mathbf{x}, \mathbf{y})$ is then computed as the following dot product:

$$\hat{\phi}(\mathbf{x}, \mathbf{y}) = \left(\frac{\hat{v}}{|\hat{v}|} \cdot \frac{e_t(f_t(\mathbf{y}))}{|e_t(f_t(\mathbf{y}))|} \right). \tag{6.4}$$

We use this modified compatibility function with InfoNCE contrastive loss for training and we call this method *Patch Aligned Contrastive Learning*. Figure 6.5, shows a diagrammatic representation of the steps involved in computing the compatibility function for an image-text pair.

Grounded in Mutual Information: To understand how our compatibility function $\hat{\phi}(\mathbf{x}, \mathbf{y})$ works, we go back to the relation of the InfoNCE [134] loss with mutual information (MI). Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ be two multivariate random variables with a joint probability function $p(x, y)$. MI between x and y, computed as $\mathbb{I}[x, y] = \mathbb{E}_{(x,y)\sim p(x,y)} \log \left[\frac{p(x,y)}{p(x)p(y)} \right]$, captures the amount of information shared between x and y. However, MI is computationally intractable and hence requires approximations in order to be estimated. The InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}(\theta)$ defined using a compatibility function $\phi_\theta(x, y)$ with model parameters θ provides such an estimate and is a lower bound on MI as: $\mathbb{I}[x, y] \geq \log(k) - \mathcal{L}_{\text{InfoNCE}}(\theta)$, where k

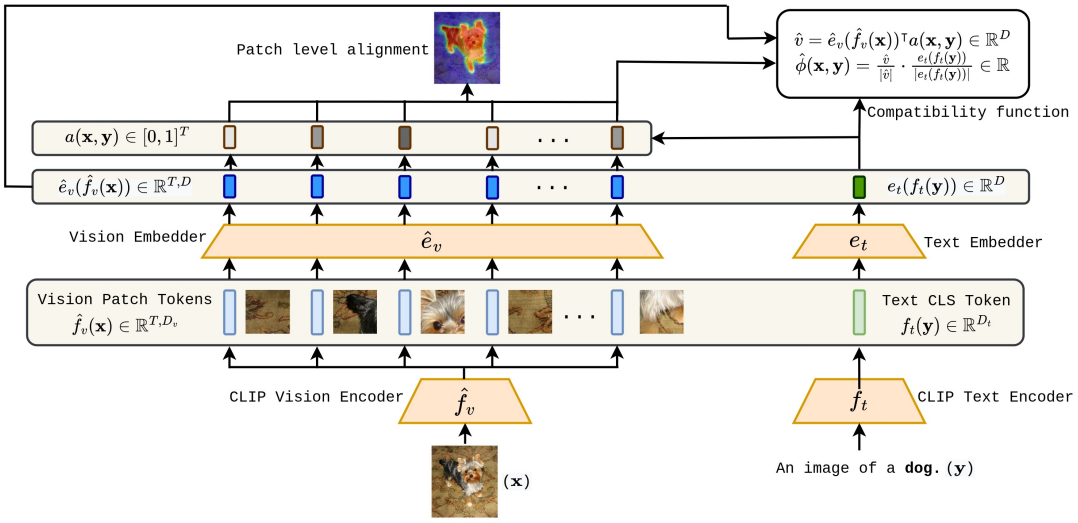


Figure 6.5: Compatibility function $\phi(\mathbf{x},\mathbf{y})$ for Patch Aligned Contrastive Learning (PACL). The image encoder $\hat{f}_v$ and embedder $\hat{e}_v$ produce patch level representations for each image whereas the text encoder f_t and embedder e_t produce the CLS representation for a given text. We compute the cosine similarity between the CLS text embedding and the vision patch embeddings and use them as weights to take a weighted sum over vision patch tokens. We use the cosine similarity between the weighted sum and the CLS text token as our compatibility $\hat{\phi}(\mathbf{x},\mathbf{y})$.

is the batch size in InfoNCE loss with one positive sample and $k-1$ negative samples per batch. Hence, minimizing $\mathcal{L}_{\text{InfoNCE}}$ maximises the lower bound estimate of MI.

In vanilla CLIP training, the random variables x and y are images $\mathbf{x}$ and texts $\mathbf{y}$ respectively, the compatibility function $\phi_\theta(x,y)$ is $\phi(\mathbf{x},\mathbf{y})$, i.e., the cosine similarity between CLS vision and text token embeddings, and the model parameters are $\theta = \{f_v, e_v, f_t, e_t\}$. Since, we modify the compatibility function $\hat{\phi}(\mathbf{x},\mathbf{y})$ using a weighted sum over vision tokens, to maximise MI $\mathbb{I}[x,y]$ between image and text, $\mathcal{L}_{\text{InfoNCE}}$ will have to attend to regions of the image which correspond to the text and assign such regions a higher value in $s(\mathbf{x},\mathbf{y})$. This indicates that $s(\mathbf{x},\mathbf{y})$ intuitively captures patch level alignment between image and text modalities. To empirically verify this, we conduct the same patch level classification task described in §6.3 where for each patch, we compute the similarity $s(\mathbf{x},\mathbf{y})$ for all classes and predict the class with the highest similarity. Results are in Table 6.1, third column (Post-Alignment) with qualitative results in Figure 6.2, third row. In both

cases, we observe a stark improvement in patch level alignment compared to vanilla CLIP using our compatibility function.

It is worth noting here that a similar contrastive learning approach has been used for the problem of weakly supervised phrase grounding in [258]. Their approach learns a mapping between ROI features from an object detector and word representations from a language model using an attention based weakly supervised contrastive learning. Although similar to our approach, they require the use of an object detector to provide ROI features, whereas we use CLIP's vision encoder patch tokens as region features, having shown (see §6.4) that such features indeed are semantically coherent. Furthermore, they also use a contextualised language model to generate negative samples for contrastive loss, whereas our method fits in seamlessly with the contrastive setting in CLIP. Finally, whereas they target weakly supervised phrase grounding, we aim to learn a multi-modal model which is zero-shot transferable to the task of open vocabulary semantic segmentation.

Inference: At inference time, we can compute both image level as well as dense predictions. For image level predictions, similar to CLIP, we simply use our compatibility function $\hat{\phi}(\mathbf{x}, \mathbf{y})$ to compute similarity between an image and text. For semantic segmentation, given an image $\mathbf{x}$ and a set of classnames $Y = \{\mathbf{y}_1, ..., \mathbf{y}_C\}$, we compute $s(\mathbf{x}, \mathbf{y_c}) \; \forall c \in \{1, ..., C\}$ as a mask for each class and then use a softmax across classes. In the next section, we provide a detailed set of experiments to show the performance of our approach at both zero-shot semantic segmentation as well as image classification tasks.

6.6 Experiments and Discussion

6.6.1 Zero-shot Semantic Segmentation

In the previous section, we described PACL, a multi-modal contrastive objective to train an alignment between vision patch embeddings and CLS text embeddings in CLIP. In this section, we evaluate the quality of this alignment through zero-shot transfer to semantic

segmentation. We present implementation and training details for PACL, evaluation settings for zero-shot segmentation, and finally, results and a discussion on the same.

Training a small vision embedder: In §6.4, we have shown that CLIP's pre-trained vision encoders $\hat{f}_v$ have a relatively strong semantic coherence. In order to leverage this coherence and the large scale pre-training of CLIP, we keep the image encoder $\hat{f}_v$, the text encoder f_t and the text embedder e_t frozen from a pre-trained CLIP model. We only train the vision embedder, i.e., $\theta = \{\hat{e}_v\}$. Note that the modification of the vision encoder from f_v (outputs the CLS vision token) to $\hat{f}_v$ (outputs the patch tokens) does not require any re-training. For $\hat{e}_v$, we use a residual block with two linear layers in the main branch and a single linear layer in the residual connection, There is a ReLU non-linearity between the two linear layers (see Appendix D.1.1). We find this simple architecture to work well for our applications.

Image-text datasets for training: We train our model purely on publicly available image-text datasets. In particular, we use Google Conceptual Captions (GCC) 3M [275], Google Conceptual Captions (GCC) 12M [276] and YFCC-15M, a subset of YFCC-100M [277] provided by CLIP [254], with a total number of approximately 30M training samples. Similar to GroupViT [146], in addition to the text descriptions in the datasets, we extract nouns from these descriptions, and randomly select one of 7 CLIP prompts (like "*itap of a ()*.", see Appendix D.1.2), to form sentences with these nouns. We add these sentences to the text descriptions as well. More details on the datasets can be found in Appendix D.1.3. Note that ***we do not use any segmentation annotations or class-agnostic segmentation masks during training***. Further training details are in Appendix D.1.2.

Stride trick at inference: Since CLIP ViT-B/16 and ViT-L/14 use either 16×16 or 14×14 patches, the number of tokens generated is much smaller than the number of pixels, which is a problem for fine-grained predictions in segmentation. One workaround is to upscale the image at inference time to a larger size. We however find instead that a change to the stride of the convolutional layer to extract image patches in ViT can provide better fine-grained patches at inference time. In particular, we use a stride of 4×4 at inference time, thereby generating a larger number of overlapping patch tokens, without requiring any change

in the model weights. For a given set of text inputs, we find the alignment between each patch and text embedding. The alignment scores are then interpolated to image dimensions. Finally, a softmax operation across text inputs provides a dense prediction in image dimensions.

Segmentation datasets for evaluation: Similar to recent works [145, 146] on zero-shot semantic segmentation, we use the following datasets for evaluation: **a)** *Pascal VOC* [184] (PV-20): 20 foreground classes with 1449 validation images, **b)** *Pascal Context* [259] (PC-59): 59 classes with 5k validation images, **c)** *COCO Stuff* [260] (CS-171): 171 "thing" or "stuff" classes and 5k validation images, **d)** *ADE20K* [261] (A-150): 150 classes with 2k validation images. Further details on these datasets can be found in Appendix D.1.3. For all datasets, we report the mean intersection over union (mIoU) [184], the most popular evaluation metric for semantic segmentation.

Comparative Baselines: We compare PACL with some of the most well-known recent methods on zero-shot semantic segmentation. In particular, we use *LSeg* [144], *ViL-Seg* [257], *GroupViT* [146] and *OpenSeg* [145] as baselines. In addition, we also compare with two relatively older approaches: *SPNet* [278] and *ZS3Net* [279]. Note that some of these methods work under relatively relaxed constraints. In particular, SPNet, ZS3Net and LSeg use full segmentation annotations during training and OpenSeg uses class-agnostic segmentation masks. Furthermore, unlike us, ViL-Seg, SPNet and ZS3Net evaluate on a small subset of "unseen" classes from Pascal VOC, Pascal Context and COCO Stuff. To our knowledge, GroupViT and ViL-Seg are the only two methods which solely use image-text data for training. We also add a baseline using vanilla CLIP by taking the alignment between the vision patch embeddings and the text CLS embedding from CLIP's pre-trained model.

Results & discussion: In Table 6.2, we report the mIoU for each baseline on the 4 segmentation datasets mentioned above. Note that the numbers shown for SPNet, ZS3Net and ViL-Seg are obtained from the ViL-Seg paper [257] and the numbers for all other baselines are obtained from their respective papers (cited in the table). In Figure 6.6, we show qualitative results of our method (i.e., PACL + CLIP) on PascalVOC and ADE20K images (more in Appendix D.2.2). With mIoU scores of 72.3, 50.1, 38.8 and 31.4 on Pascal VOC,

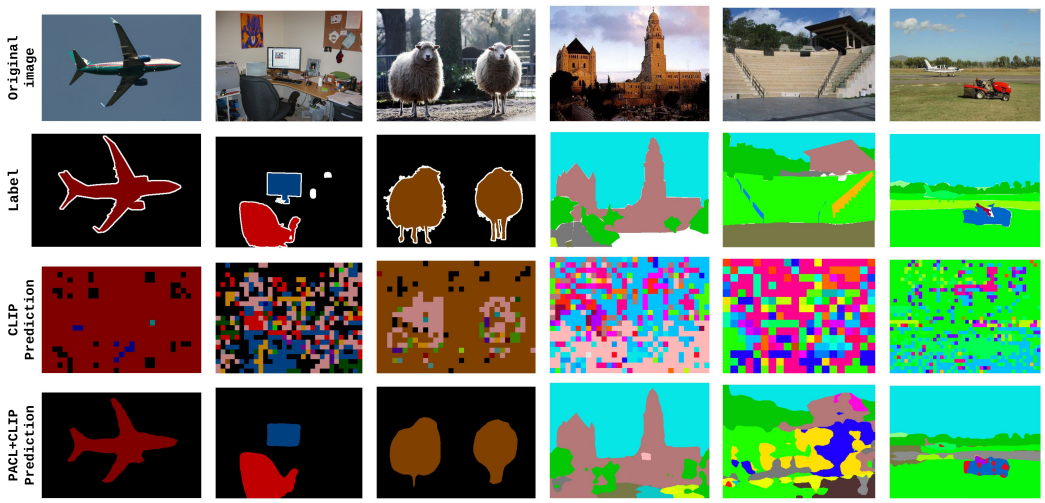


Figure 6.6: Qualitative results on zero-shot semantic segmentation. The first row denotes the original images, the second row shows the corresponding labels, the third row shows results obtained from a vanilla CLIP ViT-B/16, and the fourth row shows results of our method, PACL trained on a CLIP ViT-B/16 encoder. The first 3 images from the left are from Pascal VOC and the next 3 images are from ADE20K.

Method	Encoder (Pre-trained?)	External Training Set	Constraints		mIoU			
			Annotation	*Mask*	*PV-20* [184]	*PC-59* [259]	*CS-171* [260]	*A-150* [261]
SPNet [278]	ResNet-101 (✗)	✗	✓	✗	15.6	4.0	8.7	-
ZS3Net [279]	ResNet-101 (✗)	✗	✓	✗	17.7	7.7	9.6	-
LSeg [144]	ViT-L/16 (✗)	✗	✓	✗	52.3	-	-	-
OpenSeg [145]	EfficientNet-B7 (✗)	COCO [280] + Loc. Narr. [281]	✗	✓	72.2	48.2	-	28.6
ViL-Seg [257]	ViT-B/16 (✗)	GCC12M [276]	✗	✗	34.4	16.3	16.4	-
GroupViT [146]	ViT-S/16 (✗)	GCC12M [276] + YFCC15M [254, 277]	✗	✗	52.3	22.4	24.3	-
CLIP [254]	ViT-B/16 (✓)	WIT-400M [254]	✗	✗	8.4	2.3	2.6	1.3
CLIP + PACL (Ours)	ViT-B/16 (✓)	GCC3M [275] + GCC12M [276] + YFCC15M [254, 277]	✗	✗	**72.3**	**50.1**	**38.8**	**31.4**

Table 6.2: Results on zero-shot semantic segmentation on Pascal VOC (PV-20), Pascal Context (PC-59) and COCO Stuff (CS-171) and ADE20K (A-150) datasets. We provide the encoder architecture, external training dataset (if any) as well as if those methods use segmentation annotations or class-agnostic segmentation masks. Our method (CLIP + PACL) consistently outperforms all previous approaches.

Pascal Context, COCO Stuff and ADE20K respectively, it is clear that *PACL outperforms all other baselines consistently even though it works under a stricter set of assumptions*, i.e., it does not use any segmentation annotations and is evaluated on all classes of the segmentation datasets. This is further corroborated from our qualitative results in Figure 6.6. It is interesting to note from Figure 6.6 that vanilla CLIP mostly seems to identify the correct classes in its predictions, just not the locations of those classes within the image. This relates to the problem of a lack of alignment between the CLS text token and the vision patch tokens (see Figure 6.2) and this problem is solved through the introduction of the PACL objective. Since

PACL, as an approach, is not tied to any particular encoder, we next test its performance using different pre-trained encoders as well as different datasets on the zero-shot segmentation task.

Ablations on datasets and encoders: We perform an ablation by training PACL on a combination of different image-text training sets and different pre-trained vision encoders. For vision encoders we use CLIP ViT-B/16, CLIP ViT-L/14, DINO [29] ViT-B/16 as well as a Tiny-ViT 5M[2] model pre-trained on ImageNet-22K. For training sets, we use GCC3M for the Tiny-ViT encoder and ablate between GCC12M, (GCC12M + YFCC15M) and (GCC3M + GCC12M + YFCC15M) for all other encoders. We report the mIoU obtained on Pascal VOC from each of the (model, dataset) combinations in Table 6.3.

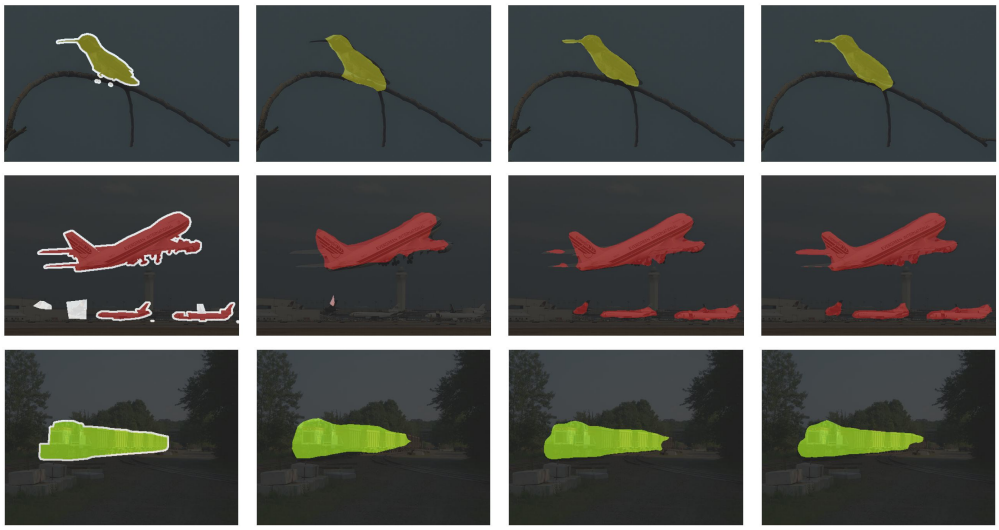
These results provide two surprising observations. Firstly, PACL seems to generate an alignment even between Tiny-ViT and DINO's pre-trained vision encoders and CLIP's text encoder, although these encoders have been trained independently. With an mIoU of 55.4, even the worst performing DINO baseline outperforms all competitive zero-shot segmentation baselines in Table 6.2 except OpenSeg. Secondly, PACL trained using CLIP's ViT-B/16 consistently outperforms ViT-L/14 even though ViT-L/14 is known to be a clear winner in terms of image level zero-shot tasks. In fact, there is a trend in performance where CLIP ViT-B/16 outperforms CLIP ViT-L/14 which outperforms DINO ViT-B/16. This is also noticeable in Figure 6.7 where CLIP encoders generate relatively better segmentation masks than DINO. This observation is strongly reminiscent of the one in §6.4 and Figure 6.3, where we note that semantic coherence is strongest in CLIP ViT-B/16 followed by CLIP ViT-L/14 and finally by DINO ViT-B/16. These empirical observations *suggest that PACL is a general contrastive learning method which can be used to train a patch level alignment and works independent of vision and text encoders as long as the vision encoders exhibit the property of semantic coherence*. Indeed, semantic coherence seems to be the most important factor behind the success of PACL.

ViT-B/16 vs ViT-L/14: To investigate the performance anomaly between ViT-B/16 and ViT-L/14, we train PACL with ViT-B/16 and ViT-L/14 encoders along with their respective

[2] github.com/microsoft/Cream/tree/main/TinyViT

Dataset	Vision Encoder	Text Encoder	mIoU PV-20
GCC3M	Tiny-ViT 5M	B/16	40.2
GCC12M	CLIP B/16	B/16	64.1
	CLIP L/14	L/14	62.7
	DINO B/16	B/16	55.4
GCC12M + YFCC15M	CLIP B/16	B/16	69.2
	CLIP L/14	L/14	68.4
	DINO B/16	B/16	62.6
GCC3M + GCC12M + YFCC15M	**CLIP B/16**	**B/16**	**72.3**
	CLIP L/14	L/14	71.7
	DINO B/16	B/16	64.8

Table 6.3: Ablation on zero-shot segmentation across encoder architectures and datasets on Pascal VOC (PV-20). In the Text Encoder column, B/16(L/14) indicates the pre-trained text encoder trained for CLIP ViT-B/16(L/14).



(a) GT **(b)** DINO B/16 **(c)** CLIP L/14 **(d)** CLIP B/16

Figure 6.7: Qualitative results comparing segmentation of different encoders using PACL. We use 3 images from PASCAL VOC val set and show their segmentations for DINO ViT-B/16, CLIP ViT-L/14 and CLIP ViT-B/16.

text encoders end-to-end from scratch on GCC12M + YFCC15M. We train both models for a total of 30 epochs. Additionally, we also perform end-to-end fine-tuning using CLIP's pre-trained encoders for 10 epochs on the same dataset (GCC12M + YFCC15M). We report the zero-shot mIoU on Pascal VOC in Table 6.4. Clearly, when the features of the vision encoder are modified through training, ViT-L/14 outperforms ViT-B/16. Furthermore, comparing with the results from Table 6.3, we see that end-to-end fine-tuning provides slight improvements over training on top of a frozen CLIP encoder. This leads us to conclude that CLIP's pre-

CLIP Encoder	mIoU PV-20 *Scratch*	mIoU PV-20 *Fine-tuned*
ViT-B/16	64.5	69.8
ViT-L/14	68.7	70.1

Table 6.4: Ablation on CLIP ViT-B/16 vs ViT-L/14 trained end-to-end from scratch or fine-tuned using PACL loss on GCC12M + YFCC15M. ViT-L/14 outperforms ViT-B/16.

	Linear embedder	Non-linear embedder
mIoU PV-20	23.3	57.6

Table 6.5: Ablation with linear vs non-linear embedder.

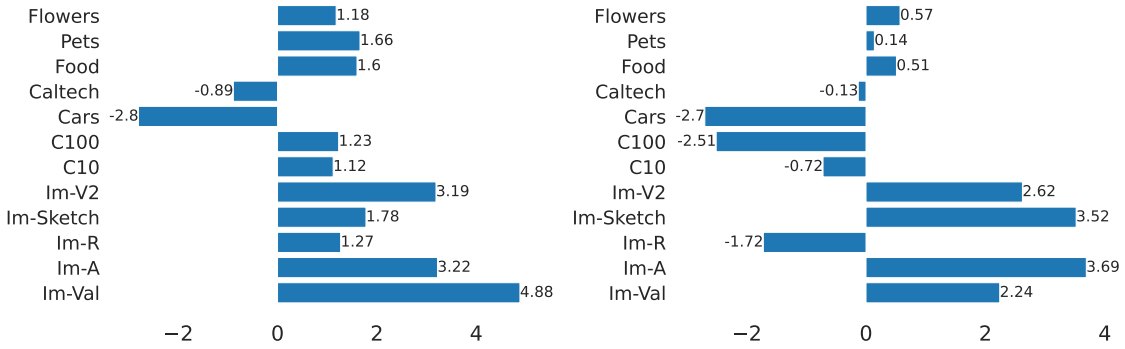
trained ViT-L/14 encoder underperforms due to CLIP's contrastive loss which uses only the CLS token and does not require semantic coherence to be explicitly incorporated into the patch tokens. When trained end-to-end on PACL, we find ViT-L/14 to be the superior model.

Linear vs non-linear vision embedder: Finally, to study the importance of a non-linear vision embedder, we train a linear and a non-linear (Figure D.1.1) embedder on top of CLIP ViT-B/16 using PACL loss. We train on GCC3M for 10 epochs and report the zero-shot mIoU on Pascal VOC in Table 6.5. Evidently, the performance of the linear embedder is inferior indicating that non-linearities in the embedder are important for PACL.

6.6.2 Image Classification

In §6.5, we mention that the modified compatibility function of PACL can be used to make image level predictions, similar to CLIP. In this section, we test our PACL models on zero-shot image classification.

Zero-shot image classification results: We apply PACL trained using CLIP ViT-B/16 and ViT-L/14 encoders on (GCC3M + GCC12M + YFCC15M) to zero-shot image classification on 12 different datasets including ImageNet [12], 4 datasets considered to be standard distribution shifts on ImageNet, ImageNet-A [91], ImageNet-R [86], ImageNet-Sketch [87] and ImageNet-V2 [242] as well as 7 other standard classification datasets, detailed in Appendix D.1.3. We report the difference in classification accuracy between PACL + CLIP and vanilla CLIP for all the datasets in Figure 6.8 (all classification accuracies



(a) ViT-B/16 **(b)** ViT-L/14

Figure 6.8: Zero-shot image classification performance of PACL + CLIP vs vanilla CLIP on 12 datasets. PACL + CLIP is competitive with or outperforms CLIP on most datasets.

in Appendix D.2.3). PACL + CLIP outperforms vanilla CLIP on 10 and 7 out of the 12 classification datasets for ViT-B/16 and ViT-L/14 encoders respectively. Also note that except on ImageNet-R for ViT-L/14, PACL consistently outperforms vanilla CLIP on ImageNet and its distribution shifts. This observation is encouraging as it provides evidence in favour of our approach being applicable for image level applications in addition to segmentation.

6.7 Conclusion

In this work, we explored *Patch Aligned Contrastive Learning* (PACL), a modified compatibility function for image-text contrastive loss which learns an alignment between patch tokens from a ViT vision encoder and the CLS token from a text encoder. We show that such an alignment allows a model to identify regions of an image corresponding to a given text input, thereby enabling a seamless zero-shot transfer to semantic segmentation, without requiring segmentation annotations or masks. On 4 different segmentation datasets, we beat previous approaches on zero-shot open vocabulary segmentation, including ones which use expensive segmentation annotations or masks. Finally, we show that PACL can also be used to make image level predictions and provides a general improvement in accuracy across 12 different image classification datasets.

7

The Crippling Effect of Fine-tuning on Foundation Models

Contents

7.1 Introduction 118
7.2 A brief note on fine-tuning 120
7.3 The crippling effect of end-to-end fine-tuning 123
7.3.1 Quantifying Concept Forgetting 123
7.3.2 Benchmarking concept forgetting 125
7.4 Can preserving features help? 128
7.5 A nudge towards continual fine-tuning 132
7.6 Related Work and Discussion 133
7.7 Conclusion 136

Abstract

Pre-trained foundation models, due to their enormous capacity and training using vast amounts of data can store knowledge about many real-world concepts. To further improve performance on downstream tasks, these models can be fine-tuned on task specific datasets. While various fine-tuning methods have been devised and have been shown to be highly effective, we observe that a fine-tuned model's ability to recognize concepts on tasks *different* from the downstream one is reduced significantly compared to its pre-trained counterpart. This is clearly undesirable as a huge amount of resources went into learning those very concepts in the first place. We call this undesirable phenomenon "concept forgetting" and via experiments show that most end-to-end fine-tuning approaches suffer heavily from this side effect. To this end, we also propose a simple fix to this problem by designing a method called LDIFS (short for ℓ_2 distance in feature space) that preserves the features of the original foundation model during fine-tuning. We show that LDIFS significantly reduces concept forgetting without having noticeable impact on the downstream task performance.

7.1 Introduction

Foundation models like CLIP [254], ALIGN [255] and CoCa [131] are trained using self-supervised methods on hundreds of millions or even billions of samples scraped from the internet. This massive, compute intensive pre-training makes such models a knowledge store on a vast number of real-world concepts, enabling them to easily transfer to a wide variety of downstream tasks and applications. Indeed, this ability to recognize real-world concepts and thereby transfer to downstream tasks is the primary advantage of such models and is the very reason behind their name [23].

While a foundation model can achieve impressive performance on a downstream task often without even requiring a single training sample from the task itself [254], in order to maximise performance, it conventionally requires some form of fine-tuning on the task at hand. There are multiple types of fine-tuning methods like linear probing [254], prompt-tuning [135, 136], adapters [137, 138], weight-space interpolation [150, 282] and full end-to-end fine-tuning [139, 140, 254, 283]. Among these types, end-to-end fine-tuning is well-known to produce the best downstream performance.

It is worth noting that the pre-training dataset of a foundation model, owing to its massive scale, contains information about several thousands of real-world concepts. Hence, it is highly likely that the downstream dataset for fine-tuning the model will only contain a significantly smaller number of concepts compared to its pre-training set. A natural question that arises then is: *How does end-to-end fine-tuning of a foundation model affect the vast knowledge it acquired through its pre-training?* In this work, this is precisely what we aim to answer.

Through a thorough study of popular end-to-end fine-tuning methods, we observe that for most of them, the fine-tuned model has significantly lost its ability to recognize real-world concepts outside the downstream task, a phenomenon which we call *concept forgetting*. This is highly undesirable as there are important use-cases where we would want the foundation model's vast knowledge on concepts to be preserved after fine-tuning.

One such use-case, for instance, is the requirement for *continual fine-tuning* to incorporate previously unknown information into a pre-trained model. Companies like OpenAI and Google have spent millions of dollars pre-training large foundation models from scratch. Even so, these powerful models do tend to have gaps in their knowledge. Specific examples are ChatGPT [51] not knowing about events after September 2021 or CLIP failing to classify satellite images or detect lymph node tumors [254]. Naturally, retraining the foundation model from scratch by combining the new data with prior training data, is too expensive to be a feasible option. Such scenarios require the model to be fine-tuned only on the new data while preserving its prior knowledge at the same time. This necessitates investigation into fine-tuning methods which prevent concept forgetting.

Although conventional end-to-end fine-tuning methods generally suffer from concept forgetting, we find that there can be a relatively simple fix to this problem. In particular, if a fine-tuning method ensures that the fine-tuned model is close in some sense to the original foundation model, it can significantly reduce concept forgetting. While the vicinity of the original model can be defined in terms of distance in the parameter space, as seen in the case of the L2SP regularizer [283], we find that it is much more effective to define it in terms of distance in the model's feature space which captures its input-output behaviour. This leads us to propose a new regularizer, **LDIFS (ℓ_2 distance in Feature Space)**, which minimizes the distance between the features of the original model and those obtained from the model being fine-tuned. Furthermore, we observe that simply preserving the last-layer features is not effective in reducing concept forgetting. Motivated from observations in [236], we preserve features extracted from different internal representations. We find that this relatively simple method of preserving the original model's features while fine-tuning on a downstream task, can significantly alleviate the problem of concept forgetting in the fine-tuned model, without affecting its performance on the downstream task. Thus, to summarize, our contributions in this work are as follows:

1. **Investigate concept forgetting.** We perform a thorough evaluation, benchmarking 6 existing end-to-end fine-tuning methods on 10 different downstream tasks and find that concept forgetting, as a phenomenon, exists in all of them.
2. **Analyze different end-to-end fine-tuning methods.** We find a consistent ordering in the ability of different fine-tuning methods to avoid concept forgetting. Particularly, we find the L2SP [283] regularizer to outperform other fine-tuning baselines. We analyze why this is the case.
3. **Propose a new regularizer for end-to-end fine-tuning.** Analyzing L2SP helps us propose a simple new regularizer (LDIFS) which minimizes feature space distance while fine-tuning.
4. **A nudge towards continual fine-tuning.** Finally, as a natural extension of the evaluation setting we evaluate on a sequence of multiple fine-tuning tasks and find LDIFS to be superior to other fine-tuning methods even in this setting.

7.2 A brief note on fine-tuning

Here we provide a description of CLIP [254] as our foundation model of choice and briefly discuss existing state-of-the-art methods used for fine-tuning CLIP.

CLIP, a brief overview Broadly speaking, a CLIP model has two components: **i)** the vision or image encoder[1] $f_{\theta_v} : \mathbb{R}^{C,H,W} \rightarrow \mathbb{R}^D$, and **ii)** the text encoder $f_{\theta_t} : \mathbb{R}^L \rightarrow \mathbb{R}^D$. CLIP is pre-trained on 400 million pairs of images and corresponding text descriptions scraped from the internet. For pre-training, it uses a contrastive loss to maximize cosine similarity between the correct (image, text) pairs and minimize the same for the incorrect ones. Due to its large-scale self-supervised pre-training, CLIP exhibits impressive performance on several downstream tasks, often without requiring a single training sample from the task itself. However, in order to maximise performance on a specific downstream task, the pre-trained

[1]Note, the vision encoder first extracts D_v dimensional image features and then projects them to a D dimensional space via a linear embedder $\mathbf{w}_v : \mathbb{R}^{D_v} \rightarrow \mathbb{R}^D$. Similarly for the text encoder.

CLIP model is conventionally fine-tuned further on the task itself. Below we provide a brief description of popular such fine-tuning methods relevant to our study.

Zero-shot (ZS) In image classification, given an input image $\mathbf{x}$ and a set of K class names $\{\mathbf{c}_i\}_{i=1}^K$ as natural language text, the D-dimensional encoding[2] for each class name $\psi(\mathbf{c}_i) = f_{\theta_t}(\mathbf{c}_i)$ and the image $\phi(\mathbf{x}) = f_{\theta_v}(\mathbf{x})$ are first obtained. The text encodings $\psi(\mathbf{c}_i)$ are then used as parameters of a K-class linear classifier, and the classification inference on $\mathbf{x}$ is performed as $\arg\max_i \psi(\mathbf{c}_i)^{\mathrm{T}}\phi(\mathbf{x})$. This is known as the zero-shot (ZS) prediction and CLIP's best model has been shown to have competitive ZS accuracies with a fully supervised ResNet-101 on ImageNet [254].

Linear Probe (LP) In this case, an additional linear layer $\mathbf{w} \in \mathbb{R}^{D\times K}$ is appended on top of the image encoder f_{θ_v} and the weights of this linear layer are trained by solving a standard logistic regression problem (e.g., scikit-learn's `LogisticRegression` module [284]) using the training set of the downstream task. The linear layer $\mathbf{w}$ is normally initialized using the text representations $\{\psi(\mathbf{c}_i)\}_{i=1}^K$, known as ZS initialization. Note that in the absence of any further downstream training, LP boils down to ZS, as the linear layer for LP is initialized using ZS weights.

End-to-end fine-tuning While a pre-trained CLIP encoder can obtain impressive ZS and LP accuracies on several tasks, in order to maximize performance on a specific downstream task, the general rule of thumb is to initialize a model from the weights of the pre-trained encoder and then fine-tune the model end-to-end on the downstream task. Here we list some of the most popular end-to-end fine-tuning methods which we study in this work. We provide a more detailed discussion on the different types of fine-tuning methods in §7.6.

1. **ZS-init-CE** [254]: This is the classic end-to-end fine-tuning method where, similar to the LP, a ZS initialized linear head $\mathbf{w} : \mathbb{R}^D \to \mathbb{R}^K$ is appended to the image encoder f_{θ_v}. However, differently from the LP, parameters of the entire model $\theta = \{\theta_v, \mathbf{w}\}$ (including the image encoder parameters) are fine-tuned using a cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$.

[2]The natural language class names are often augmented with a prompt like "an image of a {class name}."

2. **LP-init-CE (LP-FT)** [139]: This is similar to ZS-init-CE but instead of initializing the appended linear head via ZS, it is initialized by performing linear probing on the downstream task first. Once the linear head is initialized, the entire model is end-to-end fine-tuned using $\mathcal{L}_{\text{CE}}$.

3. **ZS-init-L2SP** [283]: In addition to the cross-entropy loss $\mathcal{L}_{\text{CE}}$, this method uses an additional regularizer to minimize the ℓ_2 distance between the pre-trained and fine-tuned *image encoder weights*, thereby tries to keep the fine-tuned model weights close to the pre-trained ones. Let the pre-trained image encoder weights be $\theta_{v(0)}$ and the encoder weights at time step t during fine-tuning be $\theta_{v(t)}$. Then, the fine-tuning loss in this case becomes

$$\mathcal{L}_{\text{L2SP}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{L2SP}} ||\theta_{v(t)} - \theta_{v(0)}||_2^2. \tag{7.1}$$

   Note that the ℓ_2 distance is only computed between the weights of the pre-trained and fine-tuned image encoders.

4. **LP-init-L2SP**: This is similar to ZS-init-L2SP but the linear head $\mathbf{w}$ is initialized by performing linear probing on the downstream dataset first. The loss for end-to-end fine-tuning then is the same as in Equation (7.1) [3].

5. **FLYP** [140]: The Fine-tune Like You Pre-train or FLYP baseline fine-tunes both the image and the text encoders of CLIP and uses contrastive loss $\mathcal{L}_{\text{cont}}$ instead of cross-entropy $\mathcal{L}_{\text{CE}}$ for fine-tuning on the downstream task. The parameters being fine-tuned here are $\theta = \{\theta_v, \theta_t\}$, i.e., both image and text encoders of CLIP.

6. **FLYP-CE** [140]: This is an ablation on FLYP where, instead of using contrastive loss, the fine-tuning is done using cross-entropy loss $\mathcal{L}_{\text{CE}}$, taking the cosine similarities between image and text embeddings as logits. Note that similar to FLYP, in this case as well, both image and text encoders are fine-tuned end-to-end.

[3] To the best of our knowledge, LP-init-L2SP, has not been evaluated or benchmarked prior to this work.

7.3 The crippling effect of end-to-end fine-tuning

The contrastive pre-training dataset of CLIP contains 400 million (image, text) pairs scraped from the internet [254]. Consequently, any downstream task that CLIP is fine-tuned on is highly likely to contain only a small fraction of concepts compared to what it has already been exposed to during pre-training. To investigate the impact of fine-tuning, here we perform a thorough study benchmarking 6 fine-tuning methods on 109 downstream classification tasks. We find that for most of these methods, while the fine-tuned model attains excellent improved performance on the downstream task itself, its general ability to recognize concepts outside the task is significantly reduced over the course of fine-tuning. We call this phenomenon ***concept forgetting*** and find this to be an undesirable effect of most fine-tuning methods. To explore this in more detail, in the remainder of the section, we first discuss how we quantify concept forgetting, then we propose our benchmarking setup for fine-tuning methods and finally present our observations.

7.3.1 Quantifying Concept Forgetting

During ZS and LP evaluation of a model (refer §7.2) the pre-trained image encoder weights θ_v remain frozen and unchanged irrespective of the downstream task at hand. Therefore, ZS and LP performance on a specific downstream task can be a good indicator of the pre-trained model's encoded knowledge about the task. This is the hypothesis we base our analysis on.

Furthermore, while ZS accuracy is based on how well the text encoder representations can separate task-specific categories in the image encoder's feature space, LP performance is indicative of whether image encoder representations are linearly separable. This distinction is important as fine-tuning the weights of just the image encoder θ_v can lead to a situation where its representations are no longer well-aligned with the text encoder. Even so, for a given task, if the image encoder representations are linearly separable, it shows that the model is still able to recognize concepts involved in the downstream task, thereby indicating the pre-trained model's encoded knowledge on the task.

Therefore, in order to quantify concept forgetting on a particular task represented through a dataset $\mathcal{D}$, we simply measure the difference in both ZS and LP accuracy between the pre-trained and fine-tuned image encoders on $\mathcal{D}$. To formalize this, let $f_{\theta_{v(0)}}$ and f_{θ_v} be the pre-trained and fine-tuned image encoders respectively, and let $\mathcal{D}_{\text{ft}}$ represent the dataset on which we fine-tune $\theta_{v(0)}$. Furthermore, let $\mathcal{A}_{\text{ZS}}(f_{\theta_v}, \mathcal{D})$ and $\mathcal{A}_{\text{LP}}(f_{\theta_v}, \mathcal{D})$ represent the ZS and LP accuracy of image encoder f_{θ_v} on dataset $\mathcal{D}$. Then, we can define concept forgetting on $\mathcal{D}$ as the change in LP (or ZS) accuracy on $\mathcal{D}$ between pre-trained and fine-tuned models: $\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}})$ (or in short, Δ_{LP}) as:

$$\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}}) = \mathcal{A}_{\text{LP}}(f_{\theta_v}, \mathcal{D}) - \mathcal{A}_{\text{LP}}(f_{\theta_{v(0)}}, \mathcal{D}) \tag{7.2}$$

We can define $\Delta_{\text{ZS}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}})$ in a similar manner. Clearly, as the image encoder θ_v is obtained after fine-tuning on $\mathcal{D}_{\text{ft}}$, we can expect $\Delta_{\text{LP}}(\mathcal{D}_{\text{ft}}, f_{\theta_v}, f_{\theta_{v(0)}})$ and $\Delta_{\text{ZS}}(\mathcal{D}_{\text{ft}}, f_{\theta_v}, f_{\theta_{v(0)}})$ to increase over the course of fine-tuning. However, the more interesting case is to capture concept forgetting of the model over datasets other than the fine-tuning dataset itself, i.e., $\Delta_{\text{ZS}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}})$ and $\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}})$ when $\mathcal{D} \neq \mathcal{D}_{\text{ft}}$. A negative value for these metrics indicates concept forgetting, a zero indicates knowledge accumulation and a positive value indicates knowledge gain or positive forward transfer on the task under inspection.

Catastrophic forgetting vs concept forgetting Catastrophic forgetting [55, 107, 285] is a well-known phenomenon which signifies how, when a model is trained on a new task, its performance on the previous task(s) drops catastrophically. For example, if a model trained on ImageNet is then trained on say CIFAR100, it significantly loses its performance on ImageNet. Though *concept forgetting* mentioned in this work is similar to catastrophic forgetting and we do not claim much conceptual novelty in terms of the underlying phenomenon, there is a subtle *difference in the setting in which we measure forgetting*, which makes us believe that there should be a distinction between the two.

In the pre-foundation model era, at each stage of training, it was known exactly on which task(s) or dataset(s) a model was trained (for example, ImageNet in the example above).

Therefore, it was possible to roughly quantify the degree of damage the model had once it was fine-tuned on a new task. However, in the case of foundation models, the pre-training dataset is either unavailable and even when available, it is often internet scale data which may be difficult to parse. Thus, it is not possible to quantify exactly what the model knows since it was pre-trained on several millions (or even billions) of examples. It is also not possible to quantify exactly what the fine-tuned model forgot and the catastrophe therein. Hence, it is necessary to devise poking and probing mechanisms similar to the ones we present in this work using ZS and LP, to quantify the effect (forgetting) of fine-tuning on a small downstream domain of concepts which are of interest (represented by the dataset $\mathcal{D}$ used to quantify Δ_{LP} or similar metrics).

7.3.2 Benchmarking concept forgetting

To quantify concept forgetting, here we use CLIP [254] ViT-B/32 pre-trained on the OpenAI dataset and released in the OpenCLIP repository [286] and measure its ZS and LP performance over fine-tuning on 10 different image classification downstream tasks with a high variability in their semantic concepts. These datasets, along with their respective train/test splits are:

1. *Stanford Cars* [287] containing 16185 images of 196 classes of cars with a train/test split of 8144 and 8041 images respectively,
2. *CIFAR-10/100 (C10/100)* [89] containing 60000 images of vehicles, flora and fauna, divided into 10/100 classes with the train/test split having 50000 and 10000 images respectively,
3. *DTD* [241] containing 3760 images of 47 classes of textures found in the wild with 1880 images each in the train and test sets,
4. *EuroSAT* [147] containing 25000 samples with 10 categories of satellite images of landscapes and 19600/5400 training/test images respectively,

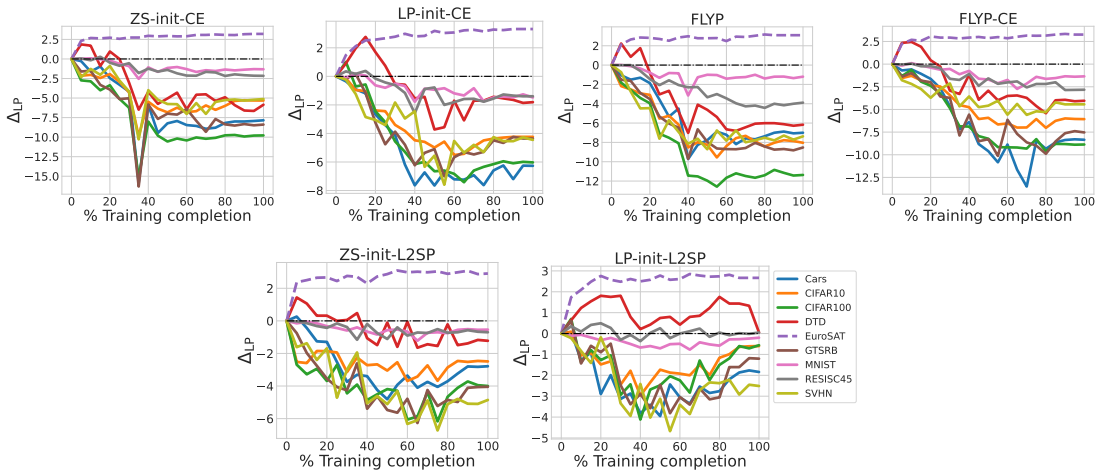
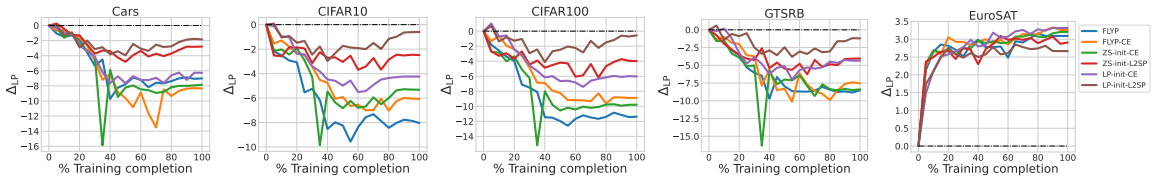


Figure 7.1: Test set $\Delta_{\text{LP}}\%$ for models fine-tuned on EuroSAT using different fine-tuning methods. While EuroSAT Δ_{LP} rises, Δ_{LP} on all other datasets is almost always negative throughout the fine-tuning with a sole exception of DTD when fine-tuned using LP-init-L2SP. See §7.3.2 for details.

5. *GTSRB* [288] containing 39270 images of 43 classes of German traffic signs with 26640 training images and 12630 test images,
6. *MNIST* [14] containing 60000 training images and 10000 test images of 10 handwritten digits from 0 to 9 in grayscale,
7. *RESISC45* (R45) [289] containing 25200 samples with 45 classes of various remote sensing image scenes with the train/test split having 18900 and 6300 images respectively,
8. *SVHN* [90] containing a total of 99289 colour images of street view house numbers, each image being categorized into one of 10 digits with 73257 training samples and 26032 test samples.
9. *ImageNet* [12] containing a total of 1.28 million training images and 50000 validation images of 1000 classes.

Finally, for this study, we use the 6 end-to-end fine-tuning methods discussed in §7.2. The training details can be found in Appendix E.1.

Fine-tuning causes concept forgetting. In Figure 7.1, we present the $\Delta_{\text{LP}}\%$ for models fine-tuned on EuroSAT using the 6 fine-tuning methods discussed in §7.2. Across all fine-tuning methods, we observe that *while performance on EuroSAT test set increases and*



(a) **(b)** **(c)** **(d)** **(e)**

Figure 7.2: Test set $\Delta_{\text{LP}}\%$ evaluated on different datasets for models fine-tuned on EuroSAT via various fine-tuning methods. The L2SP baselines (particularly LP-init-L2SP) have the lowest negative Δ_{LP} on datasets other than EuroSAT.

Δ_{LP} *is positive (also see Figure 7.2e), performance on all 8 other downstream datasets decreases and* Δ_{LP} *for all of them is negative.* This indicates, that *all 6 fine-tuning methods suffer from concept forgetting*. The only exception to this is when we evaluate on DTD and the model is fine-tuned using LP-init-L2SP. We observe Δ_{LP} to be mostly positive before it goes to zero at the end of fine-tuning. This is an interesting case as it indicates that LP-init-L2SP on EuroSAT might actually be helping increase knowledge about DTD before the fine-tuning becomes too specific to EuroSAT. This may be an example of positive forward transfer [290, 291] and exploring why such knowledge accumulation happens is an interesting avenue for further research.

Next, we compare between these fine-tuning methods by showing $\Delta_{\text{LP}}\%$ for these methods for different downstream datasets in Figure 7.2. Interestingly, *LP-init-L2SP consistently outperforms others in preserving the fine-tuned model's original performance across multiple downstream tasks*, as is observable from its low negative Δ_{LP} compared to other baselines. While it suffers an initial dip in performance in the early stages of fine-tuning, in the later stages, LP-init-L2SP regains the accuracy, often ending up with a near zero Δ_{LP}. Its impressive performance however, does seem to come at the cost of a relatively lower Δ_{LP} on the fine-tuning task, i.e., EuroSAT, itself. Our observations for other datasets are similar (see Appendix E.3). In what follows, we first investigate LP-init-L2SP further to gain insights on why it preserves concepts better than other baselines, and then use those insights to propose a new fine-tuning method that significantly outperforms all other baselines.

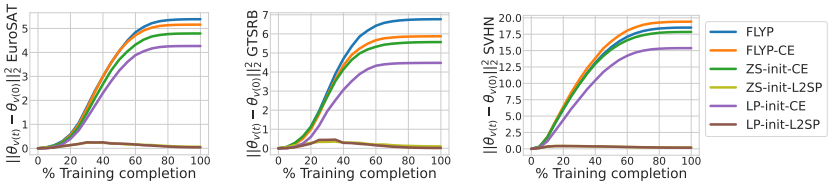


Figure 7.3: ℓ_2 distance in the parameter space $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ of the image encoder over the course of fine-tuning. Except L2SP, all other baselines diverge away from their pre-trained counterparts.

7.4 Can preserving features help?

The L2SP regularizer (Equation (7.1)) enforces the model $f_{\theta_{v(t)}}$ at time-step t of fine-tuning to be in the vicinity of the pre-trained model $f_{\theta_{v(0)}}$ by minimizing the ℓ_2 distance between the two in the parameter space. As evident from §7.3, this simple regularizer has a promising impact on the model's ability to avoid concept forgetting. To understand how correlated the parameter space ℓ_2 distance $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ is to concept forgetting, in Figure 7.3, we show how ℓ_2 distance changes as we fine-tune different datasets (EuroSAT, GTSRB and SVHN) using all the discussed fine-tuning methods. The observation is crystal clear. *Relatively speaking, all the fine-tuning methods except the two L2SP baselines (ZS-init-L2SP and LP-init-L2SP) cause the parameters of the fine-tuned model to move away from its pre-trained counterpart.* For the L2SP baselines, $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ first increases and then falls back down indicating that the model first diverges slightly and then gets closer again to the pre-trained model in the parameter space.

Though keeping the fine-tuned model in the vicinity of the pre-trained one shows a promising effect in preserving concepts, however, vicinity in the parameter space need not necessarily capture the input-output behaviour of the pre-trained model. In fact, it is trivial to construct two different sets of model parameters, far in ℓ_2 distance, outputting similar values in a specific domain. Additionally, regularizing in the parameter space might not keep the fine-tuned model in the *desired* vicinity that preserves the pre-trained knowledge and performs well on the downstream task at the same time. Therefore, we conjecture that vicinal distance in the feature space might be a better indicator of how effectively models can preserve concepts. Indeed, in the end, the internal representation space is where models

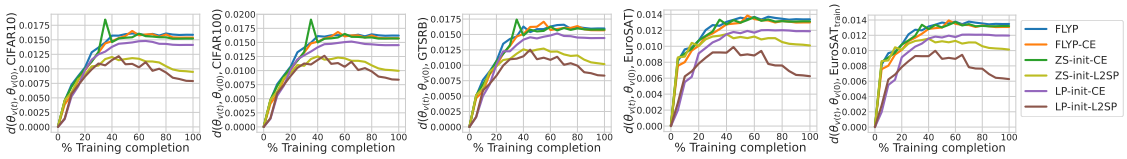


Figure 7.4: ℓ_2 distance in feature space $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ between image encoders computed over the course of fine-tuning on EuroSAT.

encode patterns. In Figure 7.4, we inspect ℓ_2 distance in feature space over fine-tuning using the following distance function:

$$d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} ||\Phi_{\theta_{v(t)}}(\mathbf{x}_i) - \Phi_{\theta_{v(0)}}(\mathbf{x}_i)||_2^2, \tag{7.3}$$

where, $\Phi_{\theta_{v(t)}}(\mathbf{x}_i)$ represents the features of the model with parameters $\theta_{v(t)}$ at time t for a given sample $\mathbf{x}_i$, and N is the number of samples in the dataset $\mathcal{D}$. The plots in Figure 7.4 are for a model fine-tuned on EuroSAT where we show $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ for the EuroSAT train and test sets, along with 3 other datasets: CIFAR-10, CIFAR-100 and GTSRB. Note that the feature vector $\Phi_{\theta_{v(t)}}(\mathbf{x}_i)$ is obtained by concatenating various internal representations (not just the last layer features) of the network architecture, similar to the perceptual features presented in [236]. The exact details as to how we compute the concatenated feature vector for a ViT-B/32 model is mentioned in Appendix E.2, and similar plots for other datasets is shown in Appendix E.3.

A strong correlation between concept forgetting and feature-space distance. Similar to our observations in Figure 7.3 (parameter space), we note that except L2SP, all other fine-tuning methods suffering significantly from concept forgetting cause the fine-tuned model to move away from the pre-trained model in terms of feature-space distance as well (Equation (7.3)). In case of ZS-init-L2SP and LP-init-L2SP, while initially diverging, the fine-tuned models recover their pre-trained behaviour to a certain extent in the later stages of fine-tuning. *Note that this observation is consistent for the training and test sets of the fine-tuning dataset, EuroSAT, as well as for all other datasets which were not used in fine-tuning, such as CIFAR-10/100.* The consistent behaviour across datasets is surprising

and indicates that other datasets need not necessarily be included in the fine-tuning set to achieve the desired outcome of minimizing concept forgetting.

Based on the results shown in Figure 7.2 and Figure 7.4 we conclude that the *fine-tuning methods which cause the model to significantly diverge away from the pre-trained model either in parameter space or in feature space suffer more from concept forgetting*. This observation, combined with our conjecture above leads to a natural extension of fine-tuning where we use $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ (distance in the feature space) as the regularizer. We call this regularizer **LDIFS: ℓ_2 distance in Feature Space**. The complete fine-tuning objective is:

$$\mathcal{L}_{\text{LDIFS}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{LDIFS}} \cdot d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D}_{\text{train}}) \tag{7.4}$$

where $\mathcal{D}_{\text{train}}$ is the training set and λ_{LDIFS} is the regularization coefficient. Note that similar to L2SP, we can initialize the linear head with both zero-shot weights or linear probe weights leading to two variants: ZS-init-LDIFS and LP-init-LDIFS.

LP-init-LDIFS minimizes concept forgetting with competitive fine-tuned performance. First, we evaluate LDIFS on the same setting as in §7.3 and present the $\Delta_{\text{LP}}\%$ over fine-tuning in Figure 7.5 (additional results presented in Appendix E.3). Furthermore, in Table 7.1, we report the Δ_{ZS} and Δ_{LP} of the fine-tuned models for 10 different image classification tasks. For each task, we report Δ_{ZS} and Δ_{LP} averaged over the remaining 9 tasks in the first row. For ImageNet, we leave out CIFAR-10/100 when calculating performance over other tasks since all of CIFAR-10/100's classes are within the set of ImageNet classes. In the second row, we show Δ_{ZS} and Δ_{LP} on the test set of the fine-tuning task itself. Full set of results are in Table E.3.3 and an alternative visualization of these results are in Figure E.3.5 of the appendix. Our observations are as follows.

1. ***Competitive downstream accuracy***: From Figure 7.5 and Table 7.1, we note that both ZS-init-LDIFS and LP-init-LDIFS are competitive with other fine-tuning baselines in terms of accuracy on the downstream fine-tuning task itself.

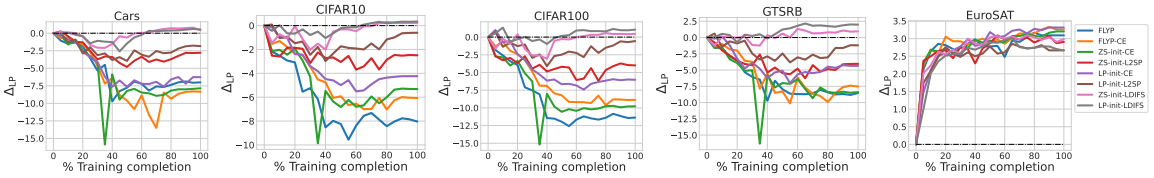


Figure 7.5: Test set Δ_{LP}% evaluated on different datasets for models fine-tuned on EuroSAT. LDIFS (Our) baselines (both ZS-init and LP-init) provides the best results in terms of avoiding concept forgetting without affecting downstream performance on EuroSAT.

Dataset / Fine-tune	Eval	FLYP $\Delta_{ZS}(\uparrow)$	FLYP $\Delta_{LP}(\uparrow)$	FLYP-CE $\Delta_{ZS}(\uparrow)$	FLYP-CE $\Delta_{LP}(\uparrow)$	ZS-init-CE $\Delta_{ZS}(\uparrow)$	ZS-init-CE $\Delta_{LP}(\uparrow)$	LP-init-CE (LP-FT) $\Delta_{ZS}(\uparrow)$	LP-init-CE (LP-FT) $\Delta_{LP}(\uparrow)$	ZS-init-L2SP $\Delta_{ZS}(\uparrow)$	ZS-init-L2SP $\Delta_{LP}(\uparrow)$	LP-init-L2SP $\Delta_{ZS}(\uparrow)$	LP-init-L2SP $\Delta_{LP}(\uparrow)$	ZS-init-LDIFS (Ours) $\Delta_{ZS}(\uparrow)$	ZS-init-LDIFS (Ours) $\Delta_{LP}(\uparrow)$	LP-init-LDIFS (Ours) $\Delta_{ZS}(\uparrow)$	LP-init-LDIFS (Ours) $\Delta_{LP}(\uparrow)$
Cars	Others	−4.40	−0.01	−5.14	−1.67	−4.78	−1.56	−3.73	−0.63	−1.5	−1.07	**0.25**	**0.47**	−0.58	−0.36	−0.17	−0.18
	Cars	9.2	**5.25**	5.73	3.95	18.24	2.67	−5.62	4.14	9.44	1.83	−3.31	3.06	**20.52**	4.71	−3.15	4.45
CIFAR10	Others	−18.79	−3.35	−12.85	−1.17	−13.95	−1.6	−10.41	−0.81	−0.82	1.04	0.99	1.16	0.06	1.03	**2.03**	**1.18**
	CIFAR10	7.87	2.79	7.8	2.65	**8.25**	**2.83**	7.5	2.79	8.05	2.8	6.38	2.74	7.72	2.46	6.39	2.33
CIFAR100	Others	−12.89	−1.16	−6.54	−0.5	−8.63	−0.96	−4.6	−0.11	−0.33	0.65	**0.33**	**1.03**	−1.05	0.82	−0.3	0.86
	CIFAR100	24.4	9.34	23.16	9.12	**24.63**	8.97	17.26	8.78	23.33	8.2	13.77	7.31	24.24	8.31	13.3	**9.35**
DTD	Others	−27.67	−4.92	−10.83	−3.44	−11.44	−3.01	−7.33	−1.76	−12.08	−3.71	1.61	0.01	−1	0.19	**1.71**	**0.53**
	DTD	21.44	4.04	16.7	1.33	31.91	4.89	2.13	0.05	26.38	0.9	0.16	2.45	**33.14**	**6.12**	0.74	3.19
EuroSAT	Others	−20.93	−6.7	−14.53	−5.44	−14.87	−5.72	−10.76	−3.75	−3.3	−2.58	−0.82	−0.85	1.56	0.92	**2.13**	**1.32**
	EuroSAT	52.11	3.09	51.96	3.26	**52.37**	3.2	30.93	**3.31**	52.19	2.91	28.04	2.67	52.15	2.98	17.02	2.67
GTSRB	Others	−32.71	−8.5	−16.38	−3.76	−23.89	−5.9	−11.65	−0.94	−14.29	−3.05	1.62	1.18	−0.33	1.07	**2.26**	**1.27**
	GTSRB	61.47	**12.64**	58.8	12.4	66.1	11.96	18.42	11.9	64.74	10.8	9.19	8.27	**66.27**	12.03	19.33	11.2
MNIST	Others	−32.14	−8.64	−24.87	−7.53	−28.80	−8.76	−35.1	−6.02	−5.92	−2.03	**0.54**	1.49	−0.14	1.8	−0.42	**2.64**
	MNIST	51.19	0.88	50.56	0.98	**51.28**	**1.02**	21.34	0.98	51.02	0.8	5.51	0.5	51.18	0.98	8.35	0.83
RESISC45	Others	−17.92	−5.42	−8.47	−3.32	−10.71	−3.79	−4.81	−2.27	−0.8	−0.91	2.46	0.66	2.08	0.41	**3.04**	**0.9**
	RESISC45	33.49	**3.98**	30.4	3.94	**35.1**	3.9	13.65	3.68	33.14	2.19	8.14	2.25	34.76	3.37	10.92	3.25
SVHN	Others	−28.91	−10.74	−21.91	−10.40	−23.44	−11.12	−25.34	−8.73	−7.07	−2.78	−11.78	−2.11	**1.14**	−0.68	−2.72	**−0.29**
	SVHN	65.0	31.92	64.38	31.86	**65.58**	31.79	51.28	**31.98**	64.84	31.02	53.39	31.04	65.25	31.36	50.92	31.38
ImageNet	Others	−5.92	−1.6	−5.13	−1.39	−5.28	−1.26	−4.96	−0.87	−1.2	−0.24	−0.77	−0.1	−0.03	0.13	**0.4**	**0.35**
	ImageNet	13.69	**6.16**	14.52	6.08	**14.93**	5.92	8.97	6.02	12.9	4.8	7.73	4.68	14.82	6.04	9.69	6.11

Table 7.1: Δ_{ZS} and Δ_{LP} for models fine-tuned on 10 image classification datasets. For each dataset, the first row shows Δ_{ZS} and Δ_{LP} averaged over the 8 datasets different from the fine-tuning dataset. The second row shows the same on the respective fine-tuned test sets. Best results in bold.

2. ***Minimal concept forgetting***: *Both the LDIFS baselines, in particular LP-init-LDIFS, significantly minimize concept forgetting over the course of fine-tuning*. This is evident from Figure 7.5 which show their noticeably higher Δ_{LP} on other downstream tasks over the course of fine-tuning. It is also apparent from their consistently high Δ_{ZS} and Δ_{LP} scores in Table 7.1. Particularly, LP-init-LDIFS gets the highest average Δ_{LP} on other tasks for 7 out of 10 fine-tuning cases. This indicates a significantly minimized level of concept forgetting. We have similar conclusions even on a CLIP RN50 model and a FLAVA [271] ViT-B/16 model in Appendix E.3.7.

3. ***Positive Forward Transfer***: From Table 7.1 and Table E.3.3, we also observe that in 8 out of 10 fine-tuning tasks, LP-init-LDIFS results in a positive average Δ_{LP} on other tasks, thereby generally achieving a positive forward transfer. The two exceptions are when

the finetuning tasks $\mathcal{D}_{\text{ft}}$ are Stanford Cars and SVHN, in which case the average Δ_{LP} on other tasks is negative. This observation also highlights how the ordering of fine-tuning tasks can also impact forward transfer. For instance, from Table E.3.3, when fine-tuning on EuroSAT, LP-init-LDIFS achieves a Δ_{LP} of $+3.03$ on SVHN, whereas in the reverse scenario, fine-tuning on SVHN leads to a negative Δ_{LP} of -0.78 on EuroSAT.

7.5 A nudge towards continual fine-tuning

Our results above indicate that fine-tuning on LP-init-LDIFS can make the foundation model learn new downstream information without forgetting pre-trained concepts. A natural question that then arises is: *Can we fine-tune on a sequence of downstream tasks without forgetting concepts?* For an ideal fine-tuning method, the final model should attain state-of-the-art performance on all fine-tuned tasks while still maintaining its pre-trained knowledge. We empirically investigate this question by training on 3 sequences of 3 tasks each: **a)** SVHN $\rightarrow$ C10 $\rightarrow$ R45, **b)** SVHN $\rightarrow$ C100 $\rightarrow$ R45 and **c)** SVHN $\rightarrow$ Cars $\rightarrow$ R45. Note that this setup is similar to continual learning [290–292] but for pre-trained foundation models. Due to their impressive ZS and LP performance on a wide range of downstream tasks, continual fine-tuning of foundation models is relatively unexplored in the literature. Nonetheless, as stated in §7.1, this is an important problem to investigate from the perspective of updating a foundation model with new previously unknown knowledge without forgetting previously trained concepts.

Quantifying concept forgetting in continual setups. Let the sequence of fine-tuning datasets be $\mathcal{D}_1 \rightarrow \mathcal{D}_2, \rightarrow ..., \mathcal{D}_k$ and the corresponding sequence of models be $f_{\theta_0}, ..., f_{\theta_k}$ with f_{θ_0} being the pre-trained model. In order to then quantify concept forgetting on a task $\mathcal{D}$ over this sequence of models, we use intuitions from [291], and extend the Δ_{LP} from Equation (7.2) as follows:

$$\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_k}, \{f_{\theta_0}, f_{\theta_1}, ..., f_{\theta_{k-1}}\}) = \mathcal{A}_{\text{LP}}(f_{\theta_k}, \mathcal{D}) - \max_{i \in \{0,...,k-1\}} \mathcal{A}_{\text{LP}}(f_{\theta_i}, \mathcal{D}). \tag{7.5}$$

Δ_{ZS} can also be defined in a similar manner. Note that we find the difference in ZS/LP performance between the final fine-tuned model f_{θ_k} and the model having the maximum ZS/LP performance in the sequence $\{f_{\theta_0}, f_{\theta_1}, ..., f_{\theta_{k-1}}\}$. This accounts for the possibility of positive forward transfer on $\mathcal{D}$ over the sequence of fine-tuning tasks. It is also trivial to see that on a single task, Equation (7.5) reduces to Equation (7.2).

LP-init-LDIFS minimizes concept forgetting in continual setups. In Table 7.2, we present the Δ_{ZS}, Δ_{LP} and $\mathcal{A}_{\text{LP}}$ for the continual setup for all fine-tuning baselines. For each task sequence, we report the performance on datasets in the sequence in the first three rows and in the fourth row, we report the average performance on all other datasets. From Table 7.2, we observe:

1. For all 3 sequences, LP-init-LDIFS has minimal concept forgetting on tasks which appear earlier in the fine-tuning sequence, as well as other datasets which are not used for fine-tuning.
2. At the same time, LP-init-LDIFS is very competitive on accuracy with the best fine-tuning methods and consistently outperforms L2SP on accuracy on the last fine-tuning task, i.e., RESISC45.

These observations are further complemented in Figure 7.6 where we can see LP-init-LDIFS to have minimal forgetting on prior tasks, SVHN and C10/100 without compromising on performance on the last task.

7.6 Related Work and Discussion

In this section, we first provide a brief discussion on the different types of fine-tuning approaches apart from end-to-end fine-tuning which have been applied on foundation models. We also discuss the relation and differences between our proposed LDIFS regularizer and works on knowledge distillation at the feature level.

Fine-tuning in foundation models: There are many flavours of fine-tuning which improve on CLIP's zero-shot performance. A popular approach is prompt tuning where,

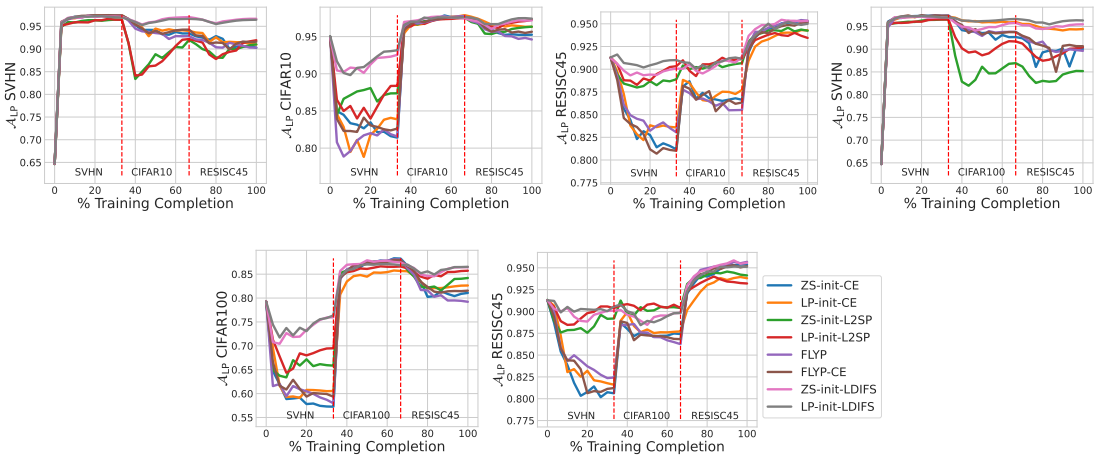


Figure 7.6: $\mathcal{A}_{\text{LP}}$ for sequence: SVHN $\rightarrow$ CIFAR10 $\rightarrow$ RESISC45 (left 3 plots) and ii) SVHN $\rightarrow$ CIFAR100 $\rightarrow$ RESISC45 (right 3 plots). Vertical red line indicates a switch in the fine-tuning tasks.

methods like CoOp [135], CoCoOp [136], TPT [293], Chain of Thought prompt tuning [294], instead of hand-crafting prompts, learn the prompts specific to the fine-tuning task. Another category of methods uses adapters: CLIP-Adapter [137], Tip-Adapter [138], Prompt Adapter [295], SVL-Adapter [296], which works on the principle of combining pre-trained features with a small non-linear adapter network where the adapter is fine-tuned on the downstream task at hand. Finally, along with end-to-end fine-tuning, there are weight space interpolation methods like Wise-FT [150], PAINT [297], task arithmetic [282] and model soups [244] which look at interpolating between pre-trained and fine-tuned models in the weight space in order to achieve the best of both worlds in terms of downstream task performance and robustness to distribution shift. It is worth noting that there are also end-to-end fine-tuning methods like LP-FT [139] (LP-init-CE in our work) which have been built to achieve robustness to distribution-shift and OoD accuracy. Unlike these works, our approach is to tackle the concept forgetting problem and we want to highlight that minimizing concept forgetting should be an additional criterion when designing fine-tuning methods.

In Table E.3.2 of the appendix, we compare linear probing, CoOp, CLIP-Adapter, Tip-Adapter and the classic end-to-end fine-tuning method, ZS-init-CE on downstream task performance on 9 image classification tasks. Not only do we find end-to-end fine-tuning to consistently outperform adapters and prompt tuning, but when there is a big performance

gap between the linear probe and end-to-end fine-tuning, adapters and prompt tuning don't bridge that gap well. This observation reaffirms the importance of end-to-end fine-tuning and thereby necessitates the investigation of better end-to-end fine-tuning methods for tasks where the pre-trained model's performance is sub-par. In Table E.3.1 in the appendix, we also show how concept forgetting in end-to-end fine-tuning methods used in our work can be further improved by combining them with Wise-FT.

Knowledge Distillation & Continual Learning: The LDIFS regularizer can be studied from the lens of knowledge distillation, an approach which has been applied to different ends like calibration [298], pre-training [299], transfer learning [300], robust fine-tuning [301] and continual learning [292, 302–304]. The main difference between these works and ours is that unlike classic distillation methods which mainly use the last layer features [303] or the logits [292, 302] directly for distillation, we concatenate features from shallower layers in the network when computing the LDIFS regularizer. Furthermore, while works like [305–308] develop indirect methods of distilling from intermediate feature representations including additional attention layers [305, 306], probability distribution matching [307], and hidden neuron activation boundaries [308], LDIFS uses a direct L2 distance between intermediate feature representations. This makes it simpler to implement without requiring any additional layers to be trained on intermediate feature representations. Finally, the proposed modifications to distillation in LDIFS turn out to be crucial for performance.

To empirically study this, we first compare performance of LP-init-LDIFS with 5 well-known distillation based continual learning baselines: LwF [302], LFL [303], iCARL [292], Distillation + Retrospection (D+R) [304] and ZSCL [153] on our sequence setup. Results are in Table 7.3. Clearly, LP-init-LDIFS performs better than all other continual learning baselines, both in preventing forgetting as well as obtaining the best performance on the last task. This is evident from its consistently high Δ_{ZS} and Δ_{LP} on all 3 sequences. To further investigate the importance of distilling from earlier features, in Figure 7.7, we compare performance of LDIFS when distilling from just last layer features (we call this ablation LL) during fine-tuning on EuroSAT. These plots provide further evidence that using just the

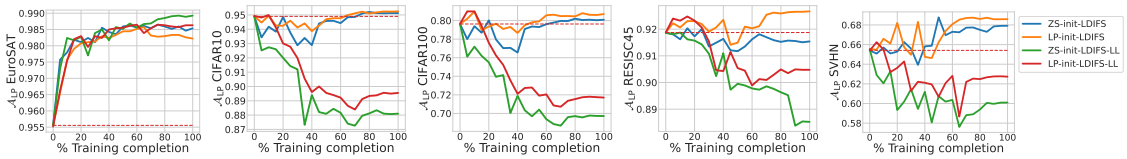


Figure 7.7: $\mathcal{A}_{\text{LP}}$ for models trained with LDIFS on EuroSAT using the full concatenated feature vector vs just the last layer (LL) feature vectors. Full feature vectors are crucial for LDIFS's performance.

last layer features is not nearly as performant as using including the earlier features in the feature vector. This indicates that learned concepts can be encoded in shallower layers of the network which makes distilling from them crucial. Finding which layers contain more pre-trained information for a downstream task is an interesting area of further research.

It is worth noting that there are some continual learning works such as Coda-Prompt [309], L2P [310] and DualPrompt [311] which leverage pre-training to design prompt based methods for continual learning. However, unlike our work, the motivation for these papers are different. They attempt to leverage pre-training as a means to improve performance on classic continual learning benchmarks. In our work, we are rather motivated to design a fine-tuning approach specifically to improve the pre-trained foundation models on a potentially new downstream task while preserving their pre-trained knowledge. Hence, we don't begin our analysis with classic continual learning approaches but rather with state-of-the-art fine-tuning methods for foundation models.

7.7 Conclusion

We explore how end-to-end fine-tuning approaches often applied on foundation models can, in turn, significantly damage the model's ability to recognize concepts outside the fine-tuning task at hand, a phenomenon which we call concept forgetting. Such an effect is undesirable particularly for foundation models which have been specifically trained on millions of samples to encode information on a vast number of real-world concepts. However, we find that among fine-tuning methods, L2SP, which keeps the model near the pre-trained model in its parameter space suffers less from concept forgetting. From insights gained

by analyzing L2SP, we also find that feature space distance provides a better definition for the vicinity of the pre-trained model as the pre-trained concepts are indeed encoded in the feature space. Our proposed new regularizer, LDIFS, encourages the features of fine-tuned model to be in the vicinity of the pre-trained ones. Through extensive evaluation on 10 different downstream classification tasks, as well as a continual fine-tuning setup of 3 different sequences of 3 tasks each, we showed that LDIFS significantly alleviates concept forgetting without impeding downstream task performance.

Dataset Fine-tune	Eval	Fine-tuning baselines																								
		FLYP			FLYP-CE			ZS-init-CE			LP-init-CE (LP-FT)			ZS-init-L2SP			LP-init-L2SP			ZS-init-LDIFS (Ours)			LP-init-LDIFS (Ours)		
		$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{ZS}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$
SVHN → C10 → R45	SVHN	−24.63	−7.06	90.3	−19.45	−5.77	91.61	−14.8	−7.13	90.29	−19.66	−6.46	90.97	−14.37	−5.41	91.01	−11.57	−4.53	91.93	**−0.68**	−0.43	96.66	−1.36	**−0.41**	**96.68**
	CIFAR10	−6.6	−3.16	94.61	−3.4	−1.92	95.65	−5.56	−2.31	95.25	−3.35	−1.57	96.31	−2.39	−1.22	96.33	**−0.23**	−0.25	97.26	−0.61	−0.26	97.18	−0.55	**−0.21**	**97.41**
	RESISC45	29.84	**4.06**	**95.33**	29.54	3.89	95.16	**34.94**	4.0	95.3	−3.78	2.98	94.29	33.43	2.97	94.24	4.59	2.16	93.44	34.68	3.94	95.33	10.98	3.7	95.0
	Others	−23.69	−6.59	78.91	−13.76	−4.9	81.2	−14.2	−5.08	80.91	−16.74	−4.24	82.13	−5.21	−1.82	85.27	−1.83	−0.01	86.89	−2.23	−0.3	86.52	**0.19**	**0.1**	**87.08**
SVHN → C100 → R45	SVHN	−26.67	−7.75	89.64	−18.36	−6.71	90.65	−18.19	−7.28	90.05	−6.96	−2.73	94.42	−27.9	−11.36	85.18	−15.38	−6.12	90.42	−2.74	−1.5	95.47	**−2.26**	**−0.65**	**96.32**
	CIFAR100	−23.0	−8.85	79.22	−16.23	−6.38	81.55	−16.69	−7.18	81.08	−11.56	−3.04	82.63	−7.63	−3.46	84.2	**−1.87**	−0.88	85.72	−2.84	−0.99	86.45	−2.51	**−0.3**	**86.54**
	RESISC45	30.68	**4.38**	**95.68**	30.95	3.9	95.21	**35.0**	4.13	95.4	−2.75	2.51	93.81	33.02	2.79	94.13	2.84	1.9	93.21	34.89	4.29	95.59	9.19	3.83	95.11
	Others	−18.63	−5.23	83.09	−9.04	−4.68	83.97	−10.7	−4.65	83.76	−11.32	−4.02	85.14	−3.19	−1.61	87.52	−2.06	−0.37	89.04	−1.92	−0.56	88.36	**−0.6**	**−0.23**	**89.12**
SVHN → Cars → R45	SVHN	−3.62	−1.26	96.07	−4.79	−1.51	95.76	−2.89	−1.45	95.93	−0.94	−0.76	96.58	−2.2	−1.14	95.42	−0.51	−0.44	95.98	**−0.49**	−0.49	96.56	−2.11	**−0.17**	**96.9**
	Cars	−8.79	−3.61	81.21	−6.74	−4.3	76.87	−9.91	−4.18	76.96	−22.62	−8.36	71.6	−4.14	−0.88	81.18	−7.06	−0.4	81.82	**−1.12**	−0.61	82.89	−2.71	**0.47**	**84.23**
	RESISC45	31.89	**4.13**	**95.43**	29.37	3.81	95.08	34.73	3.89	95.17	−1.16	3.0	94.35	33.22	2.83	94.13	4.03	2.13	93.43	**35.0**	3.94	95.22	9.13	3.73	95.27
	Others	−20.44	−6.68	81.91	−14.66	−5.48	83.2	−12.64	−4.93	83.38	−19.04	−4.51	84.39	−7.06	−2.91	85.74	−8.13	−1.67	87.15	−1.25	−0.1	88.75	**0.9**	**0.23**	**89.39**

Table 7.2: Δ_{ZS}, Δ_{LP} and $\mathcal{A}_{\text{LP}}$ for models fine-tuned on (SVHN, C10, R45), (SVHN, C100, R45) & (SVHN, Cars, R45) sequences. The first 3 rows show performance on fine-tuned tasks and the third row shows averaged performance on the remaining 6 datasets.

Dataset / Fine-tune	Eval	Fine-tuning baselines: LwF		LFL		iCaRL		D+R		ZSCL		LP-init-LDIFS (Ours)	
		$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$
SVHN → C10 → R45	SVHN	−3.81	90.48	−3.21	91.9	−3.67	91.62	−2.78	93.3	−3.23	92.7	**−0.41**	**96.68**
	CIFAR10	−2.9	93.9	−2.32	94.88	−2.1	95.17	−1.9	95.41	−1.6	95.82	**−0.21**	**97.41**
	RESISC45	3.1	94.22	2.98	93.9	2.83	93.72	3.68	94.94	3.62	94.89	**3.7**	**95.0**
	Others	−4.2	80.73	−3.76	81.31	−4.11	80.78	−3.2	81.86	−2.8	83.1	**0.1**	**87.08**
SVHN → C100 → R45	SVHN	−4.34	89.48	−4.08	90.29	−4.31	90.97	−3.23	92.3	−3.92	91.81	**−0.65**	**96.32**
	CIFAR100	−3.25	83.24	−3.01	83.95	−3.13	84.06	−2.6	84.82	−2.13	85.07	**−0.3**	**86.54**
	RESISC45	3.21	93.8	3.62	94.91	3.54	94.87	3.71	95.08	3.65	94.96	**3.83**	**95.11**
	Others	−4.11	81.73	−3.8	82.04	−4.02	81.62	−3.43	82.17	−3.11	82.86	**−0.23**	**89.12**
SVHN → Cars → R45	SVHN	−3.64	91.43	−2.92	92.74	−3.13	91.75	−2.84	92.86	−2.72	92.98	**−0.17**	**96.9**
	Cars	−2.79	81.69	−2.64	81.82	−2.8	81.7	−2.12	82.11	−1.84	82.68	**0.47**	**84.23**
	RESISC45	3.34	93.92	3.55	94.96	3.58	94.97	3.72	95.19	3.63	95.04	**3.73**	**95.27**
	Others	−4.07	81.63	−3.6	82.24	−3.89	81.88	−3.12	82.73	−2.8	83.1	**0.23**	**89.39**

Table 7.3: Δ_{LP} and $\mathcal{A}_{\text{LP}}$ comparing LDIFS with 5 distillation based continual learning methods.

The original question, 'Can machines think' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the [20th] century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.

— Alan Turing

8 Conclusion

Contents

8.1 Summary of Contributions 141
8.1.1 Chapter 3: Calibrating Deep Neural Networks using Focal Loss . 141
8.1.2 Chapter 4: Deep Deterministic Uncertainty: A New Simple Baseline 142
8.1.3 Chapter 5: Raising the Bar on the Evaluation of Out-of-Distribution Detection 144
8.1.4 Chapter 6: Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning 145
8.1.5 Chapter 7: The Crippling Effect of Fine-tuning on Foundation Models 146
8.2 Future Directions 147
8.3 Concluding Remarks 149

This thesis proposed methods to mitigate some of the major challenges in safely scaling modern neural networks. On the front of safety, the thesis proposed simple ways to reliably estimate uncertainty from deterministic single forward-pass neural networks. Uncertainty estimation is one of the core challenges of AI safety and is particularly relevant with the growing capabilities of deep learning models, making them applicable to several potentially safety-critical downstream tasks. On the front of scaling neural networks, the thesis proposed fine-tuning methods for foundation models which can mitigate dependence on expensive

annotations in vision as well as preserve pre-trained knowledge over the course of fine-tuning. The proposed methods are thus directed towards reducing dependence on data as well as facilitating continual or lifelong learning. In this concluding chapter, we first summarize the contributions of Chapters 3-7 in Section 8.1, following which we discuss some of the major avenues for future research from this thesis in Section 8.2. We finally provide our concluding remarks in Section 8.3.

8.1 Summary of Contributions

8.1.1 Chapter 3: Calibrating Deep Neural Networks using Focal Loss

In this chapter, we study the problem of calibration in multi-class classification. As mentioned in [56], modern neural networks are overconfident and miscalibrated and this is due to overfitting on the NLL or cross-entropy loss. We first analyze the NLL overfitting phenomenon drawing insights on how NLL loss during training can be further reduced even after the model has achieved 100% training accuracy and how the model does this through magnifying its weights. We then study an alternative loss function, focal loss [154], which can mitigate the NLL overfitting phenomenon to a great extent. For a training sample with one-hot ground-truth distribution, focal loss adds an extra factor, $\mathcal{L}_{\text{focal}} = -(1-\hat{p})^{\gamma}\log\hat{p}$, compared to cross-entropy which is $\mathcal{L}_{\text{CE}} = -\log\hat{p}$, where $\hat{p}$ is the softmax probability assigned to by the model to the ground-truth correct class for the sample and γ is the focal loss hyperparameter. Our initial analysis on focal loss reveals that it forms an upper-bound on a regularized KL divergence between the predicted softmax distribution and the ground-truth distribution over classes. The regularizer is the entropy of the softmax distribution and the regularization coefficient is γ. Thus, while focal loss tries to minimize the KL divergence, it also tries to maximise the entropy of the predicted softmax distribution, thereby mitigating the overconfidence problem.

When evaluated on the same setting compared to cross-entropy loss, focal loss with $\gamma = 1, 2$ and 3 produces significantly more calibrated models. Further analysis reveals an

interesting relation between the gradient of focal loss and the gradient of cross-entropy. We find the ratio of gradients between focal loss and cross-entropy to be a function of a) the probability $\hat{p}$ assigned to the ground-truth class and b) γ, the focal loss hyperparameter. In the initial stages of training, when the network is generally underconfident on its predictions, this ratio has a value > 1 which provides impetus to magnify weights and produce more confident predictions. However, in the later stages of training, with increase in the value of $\hat{p}$, this ratio goes down to < 1 which indicates focal loss gradients to have lesser norms than cross-entropy gradients, thereby providing an implicit weight regularization effect on the model being trained. This analysis not only provides insights into why models trained on focal loss are more calibrated, it also helps us develop a method to choose ideal values of γ for focal loss to optimally calibrate a model. Following this, we propose schedules of γ during training which further improves calibration results on focal loss models without affecting test accuracy. Our extensive experiments on multiple datasets and multiple model architectures show that focal loss provides calibrated models from scratch and can also be beneficial for detecting out-of-distribution (OoD) samples.

8.1.2 Chapter 4: Deep Deterministic Uncertainty: A New Simple Baseline

In this chapter, we propose a simple method to estimate and disentangle epistemic and aleatoric uncertainty in deterministic single forward-pass neural networks. Our method has two main components. First, we encourage a constrained feature space with smoothness and sensitivity properties to ensure that distance between input samples are preserved in the model's feature space. This is achieved through the use of spectral normalization [181] and architectures with residual connections [15]. Second, once the model is trained, we fit a Gaussian Discriminant Analysis (GDA) on the feature space as the density model to estimate epistemic uncertainty. The GDA fits a single mean and covariance matrix per class. This can be done using one pass through the dataset and requires no further training. We find the feature space density from the GDA to be a reliable estimate of the model's

epistemic uncertainty and the entropy of the softmax distribution as a reliable estimate of the model's aleatoric uncertainty. This leads us to propose a simple algorithm for disentangling epistemic and aleatoric uncertainty using a single forward pass. Compared to previous efforts on deterministic uncertainty like DUQ [57] and SNGP [58], our proposed DDU requires minimal changes to the model architecture and training setup. This makes it easily applicable on large scale vision tasks like semantic segmentation.

We present several evaluations for DDU. First, to evaluate uncertainty estimates on real-world scenarios using a toy setup on MNIST [14], we generate ambiguous MNIST samples using a VAE [312] based approach and combine it clean samples to create the Dirty MNIST dataset. We train models on the Dirty MNIST dataset and use Fashion MNIST as OoD [216]. This is done to represent real-world scenarios where noisy or ambiguous samples may very well be present in training sets as opposed to clean curated datasets like MNIST or CIFAR-10 [89]. Second, we use active learning as a downstream task to evaluate DDU's uncertainty estimates. We use MNIST, CIFAR-10 as well as Dirty MNIST as pool sets in active learning and compare with both deterministic baselines like softmax and SNGP as well as deep ensembles [49]. Particularly on Dirty MNIST, we find that methods which are able to specifically measure epistemic uncertainty without confounding with aleatoric are able to acquire informative clean samples. Two baselines proved to be performant in this setup: an ensemble's mutual information and DDU's feature density. Third, we use several well-known OoD detection benchmarks on CIFAR-10, CIFAR-100 and ImageNet to evaluate the uncertainty baselines. Finally, we use semantic segmentation as a computationally intensive downstream task to highlight the value of DDU as a deterministic uncertainty method. DDU's uncertainty can be obtained in similar latencies as a deterministic softmax model. In all these setups, DDU is competitive with deep ensembles, can outperform DUQ and SNGP, and provides the computational benefits of a deterministic model.

8.1.3 Chapter 5: Raising the Bar on the Evaluation of Out-of-Distribution Detection

Out-of-Distribution (OoD) detection is often used as the go-to application to evaluate uncertainty estimation methods. In this chapter, we study and critique the conventional approach to evaluation and benchmarking followed in OoD detection. Most commonly used benchmarks are based on pairs of in-distribution vs OoD datasets where the OoD datasets are arbitrarily chosen, thereby lacking a proper definition of what forms a good OoD sample. Moreover, a lot of the existing benchmarks are nearing saturation with state-of-the-art methods [94] achieving close to 100% AUROC scores on such benchmarks. In this chapter, we thus propose an alternative way to evaluate OoD detection. First, we identify desirable properties of an OoD sample, using concepts of perceptual and semantic similarity with in-distribution samples. Perceptual similarity captures visual similarity and answers the question as to whether an OoD sample "looks" similar to an in-distribution sample. Semantic similarity captures similarity in meaning and answers the question as to whether an OoD sample represents an in-distribution concept. Based on these similarity concepts, we define two main types of OoD samples, specifically Shifted samples which are semantically similar and perceptually dissimilar to in-distribution and Near OoD samples which are perceptually similar and semantically dissimilar to in-distribution. To obtain such OoD samples, we encode semantic and perceptual similarity as regularizers in the loss function of GAN based generative models, which are then able to generate desirable OoD samples from in-distribution datasets.

We perform extensive analyses on the generated OoD samples. First, we use quantitative methods to sanity-check that the Shifted and Near OoD samples indeed follow the semantic and perceptual similarity constraints which we desire. Second, for both Shifted and Near OoD sets, we perform an extensive evaluation across training sets, model architecture families and OoD detection methods. In all our analyses we observe the generated samples to provide a significantly more difficult benchmark for OoD detection methods. At the same time, we observe that the order of performance between OoD detection methods

is preserved between conventional benchmarks and our generated benchmarks in nearly all cases. Assuming that conventional OoD benchmarks are representative of real-world performance, this observation indicates that our proposed method can provide a reliable yet stronger benchmark to evaluate OoD detection baselines.

8.1.4 Chapter 6: Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning

In this chapter, we propose a new contrastive loss which can be used to train a patch level alignment between pre-trained image and text encoders of the CLIP model. We start with a pre-trained CLIP model and first make the observation that such a patch level alignment does not exist, as is, between the pre-trained vision and text encoders of CLIP. However, we find the patch representations of a CLIP's image encoder to exhibit an interesting emergent property. Semantically similar regions of the image have similar patch-level representations in CLIP's image encoder. This enables us to design a simple adapter and a modified compatibility function for contrastive loss which can be used to train an alignment between the CLS token representation from the text encoder and the patch representations from the image encoder. Specifically, we compute the cosine similarity between the text CLS token and the vision patch tokens and use these cosine similarity scores as coefficients to compute the weighted sum of patch representations as the final output of the image encoder. The cosine similarity between the CLS text output and the image output as mentioned above is then used as the contrastive compatibility function.

Once the model is trained, the modified compatibility function between a given image and text input can be used to make both image level predictions, same as in normal CLIP. At the same time, the patch level cosine similarity can be used to identify regions in an image corresponding to a given text input. This enables the model to be transferred to the task of open vocabulary semantic segmentation. As we just use image-text data to train the adapter, our method does not require any additional pixel-level annotation or binary pixel-level mask for semantic segmentation. Moreover, our method can leverage frozen CLIP

encoders and therefore brings the scale of CLIP's pre-training to the downstream task of semantic segmentation. We compare our method with other state-of-the-art baselines and achieve a significant improvement over all such baselines on 4 segmentation benchmarks. Furthermore, we also find a general improvement in zero-shot image classification over a vanilla pre-trained CLIP across 12 datasets.

8.1.5 Chapter 7: The Crippling Effect of Fine-tuning on Foundation Models

In this chapter, we investigate the effect of end-to-end fine-tuning on pre-trained foundation models like CLIP. We perform a comprehensive analysis across 10 downstream image-classification tasks and 6 state-of-the-art fine-tuning methods to show that all methods lead to the forgetting of pre-trained concepts, an undesirable effect which we call concept forgetting. We discuss the similarities and differences of concept forgetting to the well-known catastrophic forgetting [55] phenomenon and propose a simple way to quantify concept forgetting using zero-shot and linear probe prediction in CLIP. While all fine-tuning methods suffer from concept forgetting, we also find a consistent order of performance between fine-tuning methods, where methods like L2SP [283] which try to regularize fine-tuned models to remain in the vicinity of pre-trained models in weight space lead to less forgetting. Further analysis reveals that feature space distance, which captures a model's input-output behaviour, can be an even better quantifier of vicinity as compared to weight space distance. This leads us to propose a simple new regularizer which uses l_2 distance in feature space (or LDIFS in short) during fine-tuning. Additionally, we find LDIFS to be particularly performant when using feature vectors combining earlier and last layer features in a network, thereby indicating that concepts can be encoded in earlier features in a network.

We evaluate LDIFS across all 10 downstream tasks and observe significantly lowered concept forgetting compared to all other baselines. This encouraging performance leads us to also evaluate LDIFS on a natural extension of fine-tuning where instead of one downstream task, we use a sequence of multiple tasks. Specifically, we use 3 sequences of 3 downstream

tasks each and in each case, we find LDIFS to suffer from minimal concept forgetting while achieving state-of-the-art performance on the fine-tuned tasks. Therefore, LDIFS provides a scalable approach to continually fine-tune foundation models while preserving their prior knowledge.

8.2 Future Directions

In this section, we discuss some of the avenues of future research from the problems explored in this thesis.

Knowledge gaps in pre-trained foundation models: A key advantage of foundation models lies in their large-scale pre-training using vast amounts of internet-scraped data. However, this advantage can also be a problem. In most cases, the pre-training data is proprietary and inaccessible. Even when the data is open-sourced like LAION [14] or internally available in companies, it is difficult to parse such massive datasets to comprehensively identify the real-world concepts contained in them. This makes it difficult to know what a foundation model does not know. Moreover, while these models have proven to be excellent black-box backbones for several downstream tasks, as discussed in Chapter 7, they also exhibit knowledge-gaps. It may be simple to probe model performance on sample datasets like EuroSAT [16] if such datasets are accessible. However, when deployed in the real-world, an out-of-distribution sample, unless properly detected, could be incorrectly predicted by the model with overconfidence. In addition, identifying such knowledge gaps in pre-trained models is necessary as a first step before fine-tuning or updating them. While uncertainty and robustness for large scale models has been studied in the fine-tuned setting [313], there is not much work which focuses on pre-trained models. There is therefore a research need to identify what is truly out-of-distribution for a pre-trained foundation model, i.e., concepts which lie outside the model's pre-training set. One could also frame this problem as *epistemic uncertainty estimation in pre-trained foundation models* to identify real-world concepts that the model does not know.

Uncertainty for generative models: A large number of state-of-the-art foundation models such as LLMs [51] or text-to-image models [54] are generative in nature, with outputs in the form of generated images or free-form text instead of a structured vector. Uncertainty quantification in such generative models is still an open and active area of research. A key problem with generated content like free-form text is that the model has the opportunity to express uncertainty in the form of text, or verbalize its uncertainty. Whether the verbalized uncertainty is then reflective of the model's true calibrated uncertainty in its output is still relatively unknown with work hinting towards both possibilities [314, 315]. Moreover, even with sampling methods, there is the added possibility of multiple outputs from the model having the same semantic meaning. This calls for estimating uncertainty in the semantic space, which has been attempted in some works using bi-directional entailment [316]. However, such uncertainty methods can also impose significant additional overhead. There is thus further scope for interesting research in this area.

Forgetting beyond Classification: In Chapter 7, we proposed a simple way to quantify concept forgetting particularly for downstream classification tasks. However, pre-trained foundation models are designed to be applicable on a wide variety of downstream tasks. Ways to quantify concept forgetting for specific downstream tasks beyond classification is yet unexplored. For instance, first, it is not clear how we can quantify forgetting for a CLIP model which has been downstream fine-tuned for the task of object detection, since there is no apparent zero-shot or linear probe evaluation for a pre-trained CLIP on object detection. Second, there are many categories of foundation models with fine-tuning being a heavily used technique in all of them. While we studied forgetting for models like CLIP, quantifying the forgetting effect for other families of foundation models like LLMs is an open research question. Thus, a similar study as Chapter 7 which analyzes and benchmarks different fine-tuning methods for LLMs, such as Instruction Finetuning [41] or Reinforcement Learning with Human Feedback (RLHF) [317], from the perspective of concept forgetting could be valuable and interesting to the research and industrial communities.

The Many Faces of Safety: In this thesis, we approached AI safety from the core problem of uncertainty quantification. However, the field of safety has become multi-faceted with specialist models being replaced by generalist models capable of solving a wide variety of problems. While the ways in which a specialist downstream model can be unsafe are relatively simpler to define by domain experts, safety in generalist models is a far more under-specified and multi-faceted problem, requiring insights from interdisciplinary expertise to even have a complete and correct definition. However, sidestepping this critical issue of AI safety can lead to potentially catastrophic consequences [25]. Thus, it is crucial to implement proper governance around powerful AI models to make them safe for deployment and this opens avenues for future technical research. One of the main agendas for such research should be to build safety standards for benchmarking foundation models. Current models, LLMs for instance, are mostly evaluated on their performance on various downstream tasks like MMLU (Massive Multitask Language Understanding) [318]. However, standardized benchmarks and quantifiable metrics on evaluating the safety of such models are currently missing from the research literature. Thus, building such standardized safety evaluation measures and benchmarks is an important and open problem for the research community.

8.3 Concluding Remarks

The field of deep learning has truly experienced a boom in recent years with deep neural networks being able to solve several tasks and bring immense value in many real-world applications. With their widespread applicability, ensuring that they are both safe for real-world deployment while also developing scalable methods to expand their abilities is of key interest to deep learning research. This thesis proposes practical methods on both fronts to safely scale modern neural networks. On the front of safety, the thesis proposes simple and computationally efficient methods to ensure that deep neural networks are reliably uncertain on their predictions, thereby knowing what they don't know. On the front of scaling the abilities of neural networks, this thesis develops fine-tuning techniques that leverage powerful

pre-training to reduce dependence on training data, expensive annotations and preserve prior knowledge on pre-trained concepts. While there still remain many unsolved problems and challenges in deep learning research, we hope that this thesis plays a part in advancing the field towards making modern neural networks safe, reliable and performant.

Appendices

A

Appendix: Calibrating Deep Neural Networks using Focal Loss

Contents

A.1 Reliability Plots 153
A.2 Relation between Focal Loss and Entropy Regularised KL Divergence 154
A.3 Focal Loss and Cross-Entropy on a Linear Model 156
A.4 Proofs 158
A.5 Dataset Description and Implementation Details 161
A.6 Additional Results 163
A.7 Bar plots 167
A.8 Focal Loss is Confident and Calibrated 168
A.9 Ordering of Feature Norms 171
A.10 Early stopping 172
A.11 Machine Translation: A Downstream Task 173

In §A.1, we provide some empirical evidence for the observation made in §3.3 in the main paper using reliability plots. In §A.2, we discuss the relation between focal loss and a regularised KL divergence, where the regulariser is the entropy of the predicted distribution. In §A.3, we discuss the regularisation effect of focal loss on a simple setup, i.e. a generalised linear model trained on a simple data distribution. In §A.4, we show the proofs of the two

propositions formulated in the main text. We then describe all the datasets and implementation details for our experiments in §A.5. In §A.6, we discuss additional approaches for training using focal loss, and also the results we get from these approaches. We also provide the Top-5 accuracies of several models as we speculate that calibrated models should have a higher softmax probability on the correct class even when they misclassify, as compared to models which are less calibrated. We further provide the results of evaluating our models using various metrics other than ECE (like AdaECE, Classwise-ECE, MCE and NLL). Next, in §A.7, we provide additional results related to the confidence interval experiments performed in §3.5 of the main paper. In §A.8, we provide empirical and qualitative results to show that models trained using focal loss are calibrated, whilst maintaining their confidence on correct predictions. In §A.9, we provide a brief extension of our discussion about Figure 3.2(e) in the main paper, with a plot of L_2 norms of features obtained from the last ResNet block during training. In §A.10, we provide some empirical evidence to support the claims we make in §3.4 of the main paper about early stopping. Finally, in §A.11, we discuss the performance of focal loss on the downstream task of machine translation with beam search. We choose machine translation as the downstream task because in machine translation, softmax vectors from a model are directly fed into the beam search algorithm, and hence more calibrated probability vectors should intuitively produce better translations

A.1 Reliability Plots

In this section, we provide some empirical evidence to support the observation made in §3.3 of the main paper that a model, even after attaining perfect training accuracy, can reduce the training NLL (loss) further by increasing the prediction confidences to match the ground-truth one-hot encoding. To empirically observe this, we use the ResNet-50 network used for the analysis in §3.3. We divide the confidence range $[0, 1]$ into 25 bins, and present reliability plots computed on the training set at training epochs 100, 200, 300 and 350 (see the top row of Figure A.1.1). In Figure A.1.1, we also show the percentage of samples in each confidence

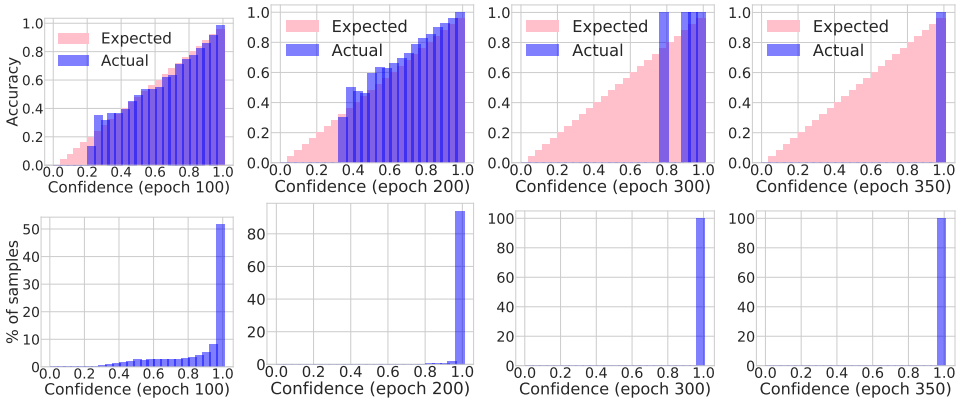


Figure A.1.1: The confidence values for training samples at different epochs during the NLL training of a ResNet-50 on CIFAR-10 (see §3.3). Top row: reliability plots using 25 confidence bins; bottom row: % of samples in each bin. As training progresses, the model gradually shifts all training samples to the highest confidence bin. Notably, it continues to do so even after achieving 100% training accuracy by the 300 epoch point.

bin. It is quite clear from these plots that over time, the network gradually pushes all of the training samples towards the highest confidence bin. Furthermore, even though the network has achieved 100% accuracy on the training set by epoch 300, it still pushes some of the samples lying in lower confidence bins to the highest confidence bin by epoch 350.

A.2 Relation between Focal Loss and Entropy Regularised KL Divergence

Here we show why focal loss favours accurate but relatively less confident solutions. We show that it inherently provides a trade-off between minimizing the KL-divergence and maximizing the entropy, depending on the strength of γ. We use $\mathcal{L}_f$ and $\mathcal{L}_c$ to denote the focal loss with parameter γ and cross entropy between $\hat{p}$ and q, respectively. K denotes the number of classes and q_y denotes the ground-truth probability assigned to the y^{th} class

(similarly for $\hat{p}_y$). We consider the following simple extension of focal loss:

$$
\begin{aligned}
\mathcal{L}_f &= -\sum_{y=1}^{K}(1-\hat{p}_y)^{\gamma} q_y \log \hat{p}_y \\
&\geq -\sum_{y=1}^{K}(1-\gamma\hat{p}_y) q_y \log \hat{p}_y && \text{By Bernoulli's inequality } \forall\gamma \geq 1, \text{ since } \hat{p}_y \in [0,1] \\
&= -\sum_{y=1}^{K} q_y \log \hat{p}_y - \gamma \left| \sum_{y=1}^{K} q_y \hat{p}_y \log \hat{p}_y \right| && \forall y, \log \hat{p}_y \leq 0 \\
&\geq -\sum_{y=1}^{K} q_y \log \hat{p}_y - \gamma \max_j q_j \sum_{y=1}^{K} |\hat{p}_y \log \hat{p}_y| && \text{By Hölder's inequality } ||fg||_1 \leq ||f||_\infty ||g||_1 \\
&\geq -\sum_{y=1}^{K} q_y \log \hat{p}_y + \gamma \sum_{y=1}^{K} \hat{p}_y \log \hat{p}_y && \forall j, q_j \in [0,1] \\
&= \mathcal{L}_c - \gamma \mathbb{H}[\hat{p}].
\end{aligned}
$$

We know that $\mathcal{L}_c = \mathrm{KL}(q||\hat{p}) + \mathbb{H}[q]$. Combining this equality with the above inequality leads to:

$$
\mathcal{L}_f \geq \mathrm{KL}(q||\hat{p}) + \underbrace{\mathbb{H}[q]}_{constant} - \gamma \mathbb{H}[\hat{p}].
$$

In the case of one-hot encoding (Delta distribution for q), focal loss will maximize $-\hat{p}_y \log \hat{p}_y$ (let y be the ground-truth class index), the component of the entropy of $\hat{p}$ corresponding to the ground-truth index. Thus, it will prefer learning $\hat{p}$ such that $\hat{p}_y$ is assigned a higher value (because of the KL term), but not too high (because of the entropy term), and will ultimately avoid preferring overconfident models (by contrast to cross-entropy loss). Experimentally, we found the solution of the cross-entropy and focal loss equations, i.e. the value of the predicted probability $\hat{p}$ which minimises the loss, for various values of q in a binary classification problem (i.e. $K = 2$), and plotted it in Figure A.2.1. As expected, focal loss favours a more entropic solution $\hat{p}$ that is closer to 0.5. In other words, as

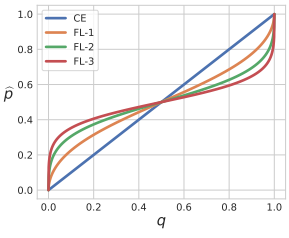
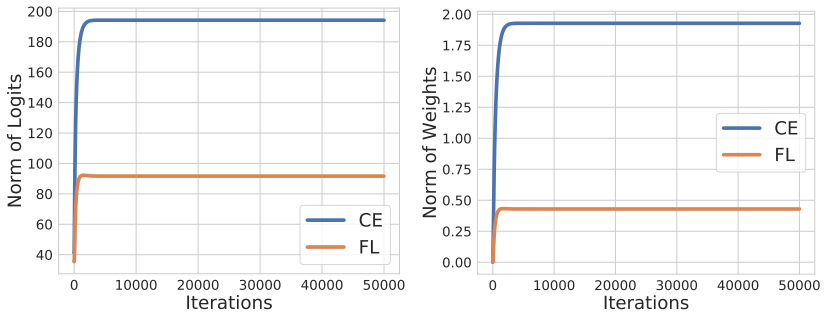


Figure A.2.1: Optimal $\hat{p}$ for various values of q.

Figure A.2.1 shows, solutions to focal loss (Equation A.1) will always have higher entropy than those of cross-entropy, depending on the value of γ.

$$\hat{p} = \text{argmin}_x \; -(1-x)^\gamma q \log x - x^\gamma (1-q) \log (1-x) \quad 0 \leq x \leq 1 \tag{A.1}$$

A.3 Focal Loss and Cross-Entropy on a Linear Model



(a) **(b)**

Figure A.3.1: (a): Norm of logits (b): Norm of weights.

The behaviour of deep neural networks is generally quite different from linear models and the problem of calibration is more pronounced in the case of deep neural networks, hence we focus on analysing the calibration of deep networks in the paper. However, weight norm analysis for the initial layers is complex due to batchnorm and weight decay. Hence, to see the effect of weight magnification on miscalibration, here we use a simple network without batchnorm or weight decay, which is a generalised linear model, and a simple data distribution.

Setup We consider a binary classification problem. The data matrix $\mathbf{X} \in \mathbb{R}^{2 \times N}$ is created by assigning each class, two normally distributed clusters such that the mean of the clusters are linearly separable. The mean of the clusters are situated on the vertices of a two-dimensional hypercube of side length 4. The standard deviation for each cluster is 1 and the samples are randomly linearly combined within each cluster in order to add covariance. Further, for 10% of the data points, the labels were flipped. 4000 samples are used for

training and 1000 samples are used for testing. The model consists of a simple 2-parameter logistic regression model. $f_{\mathbf{w}}(\mathbf{x}) = \sigma(w_1 x_1 + w_2 x_2)$. We train this model using both cross-entropy and focal loss with $\gamma = 1$.

Weight Magnification We have argued that focal loss implicitly regularizes the weights of the model by providing smaller gradients as compared to cross-entropy. This helps in calibration as, if all the weights are large, the logits are large and thus the confidence of the network is large on all test points, even on the misclassified points. When the model misclassifies, it misclassifies with a high confidence. Figure A.3.1 shows, for a generalised linear model, that the norm of the logits and the weights of a network blows for Cross Entropy as compared to Focal Loss.

High Confidence for mistakes Figures A.3.2 (b) and (c) show that running gradient descent with cross-entropy (CE) and focal loss (FL) both gives the same decision regions i.e. the weight vector points in the same region for both FL and CE. However, as we have seen that the norm of the weights is much larger for CE as compared to FL, we would expect the confidence of misclassified test points to be large for CE as compared to FL. Figure A.3.2 (a) plots a histogram of the confidence of the misclassified points and it shows that CE misclassifies almost always with greater than 90% confidence whereas FL misclassifies with much lower confidence.

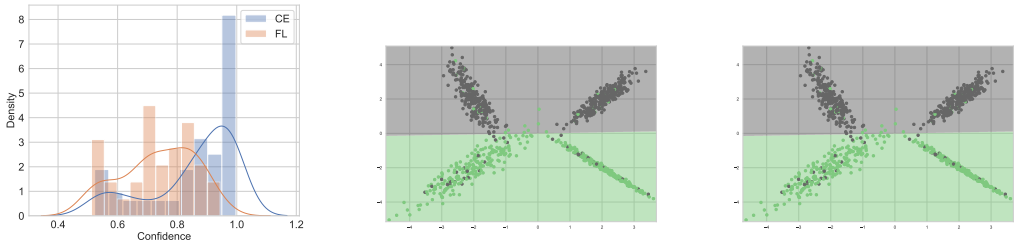


Figure A.3.2: (a): Confidence of mis-classifications (b): Decision boundary of linear classifier trained using cross entropy (c): Decision boundary of linear classifier trained using focal loss

A.4 Proofs

Here we provide the proofs of both the propositions presented in the main text. While Proposition 1 helps us understand the regularization effect of focal loss, Proposition 2 provides us the γ values in a principled way such that it is sample-dependent. Implementing the sample-dependent γ is very easy as implementation of the Lambert-W function [168] is available in standard libraries (e.g. python scipy).

Proposition 1. For focal loss $\mathcal{L}_f$ and cross-entropy $\mathcal{L}_c$, the gradients $\frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} g(\hat{p}_{i,y_i}, \gamma)$, where $g(p, \gamma) = (1-p)^\gamma - \gamma p (1-p)^{\gamma-1} \log(p)$, $\gamma \in \mathbb{R}^+$ is the focal loss hyperparameter, and $\mathbf{w}$ denotes the parameters of the last linear layer. Thus $\left\| \frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} \right\| \leq \left\| \frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} \right\|$ if $g(\hat{p}_{i,y_i}, \gamma) \in [0, 1]$.

Proof. Let $\mathbf{w}$ be the linear layer parameters connecting the feature map to the logit s. Then, using the chain rule, $\frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} = \left(\frac{\partial s}{\partial \mathbf{w}}\right)\left(\frac{\partial \hat{p}_{i,y_i}}{\partial s}\right)\left(\frac{\partial \mathcal{L}_f}{\partial \hat{p}_{i,y_i}}\right)$. Similarly, $\frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} = \left(\frac{\partial s}{\partial \mathbf{w}}\right)\left(\frac{\partial \hat{p}_{i,y_i}}{\partial s}\right)\left(\frac{\partial \mathcal{L}_c}{\partial \hat{p}_{i,y_i}}\right)$. The derivative of the focal loss with respect to $\hat{p}_{i,y_i}$, the softmax output of the network for the true class y_i, takes the form

$$\begin{aligned}\frac{\partial \mathcal{L}_f}{\partial \hat{p}_{i,y_i}} &= -\frac{1}{\hat{p}_{i,y_i}}\Big((1-\hat{p}_{i,y_i})^\gamma - \gamma \hat{p}_{i,y_i}(1-\hat{p}_{i,y_i})^{\gamma-1}\log(\hat{p}_{i,y_i})\Big) \\ &= \frac{\partial \mathcal{L}_c}{\partial \hat{p}_{i,y_i}} g(\hat{p}_{i,y_i}, \gamma),\end{aligned}$$

in which $g(\hat{p}_{i,y_i}, \gamma) = (1-\hat{p}_{i,y_i})^\gamma - \gamma \hat{p}_{i,y_i}(1-\hat{p}_{i,y_i})^{\gamma-1}\log(\hat{p}_{i,y_i})$ and $\frac{\partial \mathcal{L}_c}{\partial \hat{p}_{i,y_i}} = -\frac{1}{\hat{p}_{i,y_i}}$. It is thus straightforward to verify that if $g(\hat{p}_{i,y_i}, \gamma) \in [0, 1]$, then $\left\| \frac{\partial \mathcal{L}_f}{\partial \hat{p}_{i,y_i}} \right\| \leq \left\| \frac{\partial \mathcal{L}_c}{\partial \hat{p}_{i,y_i}} \right\|$, which itself implies that $\left\| \frac{\partial \mathcal{L}_f}{\partial \mathbf{w}} \right\| \leq \left\| \frac{\partial \mathcal{L}_c}{\partial \mathbf{w}} \right\|$. □

Proposition 2. Given a p_0, for $1 \geq p \geq p_0 > 0$, $g(p, \gamma) \leq 1$ for all $\gamma \geq \gamma^* = \frac{a}{b} + \frac{1}{\log a} W_{-1}\Big(-\frac{a^{(1-a/b)}}{b}\log a\Big)$, where $a = 1 - p_0$, $b = p_0 \log p_0$, and W_{-1} is the Lambert-W function [168]. Moreover, for $p \geq p_0 > 0$ and $\gamma \geq \gamma^*$, the equality $g(p, \gamma) = 1$ holds only for $p = p_0$ and $\gamma = \gamma^*$.

Proof. We derive the value of $\gamma > 0$ for which $g(p_0, \gamma) = 1$ for a given $p_0 \in [0, 1]$. From Proposition 4.1, we already know that

$$\frac{\partial \mathcal{L}_f}{\partial \hat{p}_{i,y_i}} = \frac{\partial \mathcal{L}_c}{\partial \hat{p}_{i,y_i}} g(\hat{p}_{i,y_i}, \gamma), \tag{A.2}$$

where $\mathcal{L}_f$ is focal loss, $\mathcal{L}_c$ is cross entropy loss, $\hat{p}_{i,y_i}$ is the probability assigned by the model to the ground-truth correct class for the i^{th} sample, and

$$g(\hat{p}_{i,y_i}, \gamma) = (1 - \hat{p}_{i,y_i})^{\gamma} - \gamma \hat{p}_{i,y_i}(1 - \hat{p}_{i,y_i})^{\gamma - 1} \log(\hat{p}_{i,y_i}). \tag{A.3}$$

For $p \in [0, 1]$, if we look at the function $g(p, \gamma)$, then we can clearly see that $g(p, \gamma) \to 1$ as $p \to 0$, and that $g(p, \gamma) = 0$ when $p = 1$. To observe the behaviour of $g(p, \gamma)$ for intermediate values of p, we first take its derivative with respect to p:

$$\frac{\partial g(p, \gamma)}{\partial p} = \gamma(1 - p)^{\gamma - 2}\Big[- 2(1 - p) - (1 - p) \log p + (\gamma - 1)p \log p\Big] \tag{A.4}$$

In Equation A.4, $\gamma(1-p)^{\gamma-2} > 0$ except when $p = 1$ (in which case $\gamma(1-p)^{\gamma-2} = 0$). Thus, to observe the sign of the gradient $\frac{\partial g(p,\gamma)}{\partial p}$, we focus on the term

$$-2(1 - p) - (1 - p) \log p + (\gamma - 1)p \log p. \tag{A.5}$$

Dividing Equation A.5 by $(-\log p)$, the sign remains unchanged and we get

$$k(p, \gamma) = \frac{2(1 - p)}{\log p} + 1 - \gamma p. \tag{A.6}$$

We can see that $k(p, \gamma) \to 1$ as $p \to 0$ and $k(p, \gamma) \to -(1 + \gamma)$ as $p \to 1$ (using l'Hôpital's rule). Furthermore, $k(p, \gamma)$ is monotonically decreasing for $p \in [0, 1]$. Thus, as the gradient $\frac{\partial g(p,\gamma)}{\partial p}$ is positive initially starting from $p = 0$ and negative later till $p = 1$, we can say that $g(p, \gamma)$ first monotonically increases starting from 1 (as $p \to 0$) and then monotonically decreases down to 0 (at $p = 1$). Thus, if for some threshold $p_0 > 0$ and for some $\gamma > 0$, $g(p, \gamma) = 1$, then $\forall p > p_0$, $g(p, \gamma) < 1$. We now want to find a γ such that $\forall p \geq p_0$,

$g(p, \gamma) \leq 1$. First, let $a = (1 - p_0)$ and $b = p_0 \log p_0$. Then:

$$\begin{aligned}
& g(p_0, \gamma) = (1 - p_0)^\gamma - \gamma p_0 (1 - p_0)^{\gamma - 1} \log p_0 \leq 1 \\
\implies & (1 - p_0)^{\gamma - 1} [(1 - p_0) - \gamma p_0 \log p_0] \leq 1 \\
\implies & a^{\gamma - 1}(a - \gamma b) \leq 1 \\
\implies & (\gamma - 1) \log a + \log(a - \gamma b) \leq 0 \\
\implies & \left(\gamma - \frac{a}{b}\right) \log a + \log(a - \gamma b) \leq \left(1 - \frac{a}{b}\right) \log a \\
\implies & (a - \gamma b) e^{(\gamma - a/b) \log a} \leq a^{(1 - a/b)} \\
\implies & \left(\gamma - \frac{a}{b}\right) e^{(\gamma - a/b) \log a} \leq -\frac{a^{(1 - a/b)}}{b} \\
\implies & \left(\left(\gamma - \frac{a}{b}\right) \log a\right) e^{(\gamma - a/b) \log a} \geq -\frac{a^{(1 - a/b)}}{b} \log a
\end{aligned} \tag{A.7}$$

where $a = (1 - p_0)$ and $b = p_0 \log p_0$. We know that the inverse of $y = xe^x$ is defined as $x = W(y)$, where W is the Lambert-W function [168]. Furthermore, the r.h.s. of the inequality in Equation A.7 is always negative, with a minimum possible value of $-1/e$ that occurs at $p_0 = 0.5$. Therefore, applying the Lambert-W function to the r.h.s. will yield two real solutions (corresponding to a principal branch denoted by W_0 and a negative branch denoted by W_{-1}). We first consider the solution corresponding to the negative branch (which is the smaller of the two solutions):

$$\begin{aligned}
& \left((\gamma - \frac{a}{b}) \log a\right) \leq W_{-1}\left(-\frac{a^{(1 - a/b)}}{b} \log a\right) \\
\implies & \gamma \geq \frac{a}{b} + \frac{1}{\log a} W_{-1}\left(-\frac{a^{(1 - a/b)}}{b} \log a\right)
\end{aligned} \tag{A.8}$$

If we consider the principal branch, the solution is

$$\gamma \leq \frac{a}{b} + \frac{1}{\log a} W_0\left(-\frac{a^{(1 - a/b)}}{b} \log a\right), \tag{A.9}$$

which yields a negative value for γ that we discard. Thus Equation A.8 gives the values of γ for which if $p > p_0$, then $g(p, \gamma) < 1$. In other words, $g(p_0, \gamma) = 1$, and for any $p < p_0$, $g(p, \gamma) > 1$. □

A.5 Dataset Description and Implementation Details

We use the following image and document classification datasets in our experiments:

1. **CIFAR-10** [89]: This dataset has 60,000 colour images of size 32×32, divided equally into 10 classes. We use a train/validation/test split of 45,000/5,000/10,000 images.

2. **CIFAR-100** [89]: This dataset has 60,000 colour images of size 32×32, divided equally into 100 classes. (Note that the images in this dataset are not the same images as in CIFAR-10.) We again use a train/validation/test split of 45,000/5,000/10,000 images.

3. **Tiny-ImageNet** [12]: Tiny-ImageNet is a subset of ImageNet with 64 x 64 dimensional images, 200 classes and 500 images per class in the training set and 50 images per class in the validation set. The image dimensions of Tiny-ImageNet are twice that of CIFAR-10/100 images.

4. **20 Newsgroups** [171]: This dataset contains 20,000 news articles, categorised evenly into 20 different newsgroups based on their content. It is a popular dataset for text classification. Whilst some of the newsgroups are very related (e.g. rec.motorcycles and rec.autos), others are quite unrelated (e.g. sci.space and misc.forsale). We use a train/validation/test split of 15,098/900/3,999 documents.

5. **Stanford Sentiment Treebank (SST)** [172]: This dataset contains movie reviews in the form of sentence parse trees, where each node is annotated by sentiment. We use the dataset version with binary labels, for which 6,920/872/1,821 documents are used as the training/validation/test split. In the training set, each node of a parse tree is annotated as positive, neutral or negative. At test time, the evaluation is done based on the model classification at the root node, i.e. considering the whole sentence, which contains only positive or negative sentiment.

All our experiments required a single 12 GB TITAN Xp GPU. For training networks on CIFAR-10 and CIFAR-100, we use SGD with a momentum of 0.9 as our optimiser, and train the networks for 350 epochs, with a learning rate of 0.1 for the first 150 epochs, 0.01 for the next 100 epochs, and 0.001 for the last 100 epochs. We use a training batch size of 128. Furthermore, we augment the training images by applying random crops and random horizontal flips. For Tiny-ImageNet, we train for 100 epochs with a learning rate of 0.1 for the first 40 epochs, 0.01 for the next 20 epochs and 0.001 for the last 40 epochs. We use a training batch size of 64. It should be noted that for Tiny-ImageNet, we saved 50 samples per class (i.e., a total of 10000 samples) from the training set as our own validation set to fine-tune the temperature parameter (hence, we trained on 90000 images) and we use the Tiny-ImageNet validation set as our test set.

For 20 Newsgroups, we train the Global Pooling Convolutional Network [173] using the Adam optimiser, with learning rate 0.001, and betas 0.9 and 0.999. The code is a PyTorch adaptation of [319]. We used Glove word embeddings [320] to train the network. We trained all the models for 50 epochs and used the models with the best validation accuracy.

For the SST Binary dataset, we train the Tree-LSTM [174] using the AdaGrad optimiser with a learning rate of 0.05 and a weight decay of 10^{-4}, as suggested by the authors. We used the constituency model, which considers binary parse trees of the data and trains a binary Tree-LSTM on them. The Glove word embeddings [320] were also tuned for best results. The code framework we used is inspired by [321]. We trained these models for 25 epochs and used the models with the best validation accuracy.

For all our models, we use the PyTorch framework, setting any hyperparameters not explicitly mentioned to the default values used in the standard models. For MMCE, we used $\lambda = 2$ for all our experiments as we found it to perform better over all the values we tried. A calibrated model which does not generalise well to an unseen test set is not very useful. Hence, for all the experiments, we set the training parameters in a way such that we get best test set accuracies on all datasets for each model.

A.6 Additional Results

In addition to the sample-dependent γ approach, we try the following focal loss approaches as well:

Focal Loss (Fixed γ): We trained models on focal loss with γ fixed to $1, 2$ and 3. We found $\gamma = 3$ to produce the best ECE among models trained using a fixed γ. This corroborates the observation we made in §4 of the main paper that $\gamma = 3$ should produce better results than $\gamma = 1$ or $\gamma = 2$, as the regularising effect for $\gamma = 3$ is higher.

Focal Loss (Scheduled γ): As a simplification to the sample-dependent γ approach, we also tried using a schedule for γ during training, as we expect the value of $\hat{p}_{i,y_i}$ to increase in general for all samples over time. In particular, we report results for two different schedules: (a) Focal Loss (scheduled γ 5,3,2): $\gamma = 5$ for the first 100 epochs, $\gamma = 3$ for the next 150 epochs, and $\gamma = 2$ for the last 100 epochs, and (b) Focal Loss (scheduled γ 5,3,1): $\gamma = 5$ for the first 100 epochs, $\gamma = 3$ for the next 150 epochs, and $\gamma = 1$ for the last 100 epochs. We also tried various other schedules, but found these two to produce the best results on the validation sets.

Finally, for the sample-dependent γ approach, we also found the policy: Focal Loss (sample-dependent γ 5,3,2) with $\gamma = 5$ for $\hat{p}_{i,y_i} \in [0, 0.2)$, $\gamma = 3$ for $\hat{p}_{i,y_i} \in [0.2, 0.5)$ and $\gamma = 2$ for $\hat{p}_{i,y_i} \in [0.5, 1]$ to produce competitive results.

In Tables A.6.1 and A.6.2, we present the AdaECE and Classwise-ECE scores for all the baselines discussed in Table 3.1 of the main paper.

In Table 3.2 of the main paper, we present the classification errors on the test datasets for all the major loss functions we considered. Here we also report the classification errors for the different focal loss approaches in Table A.6.6. We also report the ECE, AdaECE and Classwise-ECE for all the focal loss approaches in Table A.6.3, Table A.6.4 and Table A.6.5 respectively.

Finally, calibrated models should have a higher logit score (or softmax probability) on the correct class even when they misclassify, as compared to models which are less calibrated.

Dataset	Model	Cross-Entropy		Brier Loss		MMCE		LS-0.05		FL-3 (Ours)		FLSD-53 (Ours)	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	17.52	3.42(2.1)	6.52	3.64(1.1)	15.32	2.38(1.8)	7.81	4.01(1.1)	5.08	2.02(1.1)	**4.5**	**2.0(1.1)**
	ResNet-110	19.05	5.86(2.3)	**7.73**	4.53(1.2)	19.14	4.85(2.3)	11.12	8.59(1.1)	8.64	4.14(1.2)	8.55	**3.96(1.2)**
	Wide-ResNet-26-10	15.33	2.89(2.2)	4.22	2.81(1.1)	13.16	4.25(1.9)	5.1	5.1(1)	**2.08**	2.08(1)	2.75	**1.63(1.1)**
	DenseNet-121	20.98	5.09(2.3)	5.04	2.56(1.1)	19.13	3.07(2.1)	12.83	8.92(1.2)	4.15	**1.23(1.1)**	**3.55**	**1.24(1.1)**
CIFAR-10	ResNet-50	4.33	2.14(2.5)	1.74	**1.23(1.1)**	4.55	2.16(2.6)	3.89	2.92(0.9)	1.95	1.83(1.1)	**1.56**	1.26(1.1)
	ResNet-110	4.4	1.99(2.8)	2.6	1.7(1.2)	5.06	2.52(2.8)	4.44	4.44(1)	**1.62**	**1.44(1.1)**	2.07	1.67(1.1)
	Wide-ResNet-26-10	3.23	1.69(2.2)	1.7	1.7(1)	3.29	1.6(2.2)	4.27	2.44(0.8)	1.84	1.54(0.9)	**1.52**	**1.38(0.9)**
	DenseNet-121	4.51	2.13(2.4)	2.03	2.03(1)	5.1	2.29(2.5)	4.42	3.33(0.9)	**1.22**	1.48(0.9)	1.42	**1.42(1)**
Tiny-ImageNet	ResNet-50	15.23	5.41(1.4)	4.37	4.07(0.9)	13.0	5.56(1.3)	15.28	6.29(0.7)	1.88	1.88(1)	**1.42**	**1.42(1)**
20 Newsgroups	Global Pooling CNN	17.91	**2.23(3.4)**	13.57	3.11(2.3)	15.21	6.47(2.2)	**4.39**	2.63(1.1)	8.65	3.78(1.5)	6.92	2.35(1.5)
SST Binary	Tree-LSTM	7.27	3.39(1.8)	8.12	2.84(2.5)	**5.01**	4.32(1.5)	5.14	4.23(1.2)	16.01	2.16(0.5)	9.15	**1.92(0.7)**

Table A.6.1: Adaptive ECE (%) computed for different approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets.

Dataset	Model	Cross-Entropy		Brier Loss		MMCE		LS-0.05		FL-3 (Ours)		FLSD-53 (Ours)	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	0.38	0.22(2.1)	0.22	0.20(1.1)	0.34	0.21(1.8)	0.23	0.21(1.1)	**0.20**	**0.20(1.1)**	**0.20**	**0.20(1.1)**
	ResNet-110	0.41	0.21(2.3)	0.24	0.23(1.2)	0.42	0.22(2.3)	0.26	0.22(1.1)	**0.24**	0.22(1.2)	**0.24**	**0.21(1.2)**
	Wide-ResNet-26-10	0.34	0.20(2.2)	0.19	0.19(1.1)	0.31	0.20(1.9)	0.21	0.21(1)	**0.18**	**0.18(1)**	**0.18**	0.19(1.1)
	DenseNet-121	0.45	0.23(2.3)	0.20	0.21(1.1)	0.42	0.24(2.1)	0.29	0.24(1.2)	0.20	0.20(1.1)	**0.19**	**0.20(1.1)**
CIFAR-10	ResNet-50	0.91	0.45(2.5)	0.46	0.42(1.1)	0.94	0.52(2.6)	0.71	0.51(0.9)	0.43	0.48(1.1)	**0.42**	**0.42(1.1)**
	ResNet-110	0.91	0.50(2.8)	0.59	0.50(1.2)	1.04	0.55(2.8)	0.66	0.66(1)	**0.44**	**0.41(1.1)**	0.48	0.44(1.1)
	Wide-ResNet-26-10	0.68	0.37(2.2)	0.44	0.44(1)	0.70	0.35(2.2)	0.80	0.45(0.8)	0.44	0.36(0.9)	**0.41**	**0.31(0.9)**
	DenseNet-121	0.92	0.47(2.4)	0.46	0.46(1)	1.04	0.57(2.5)	0.60	0.50(0.9)	0.43	0.41(0.9)	**0.41**	**0.41(1)**
Tiny-ImageNet	ResNet-50	0.22	0.16(1.4)	0.16	0.16(0.9)	0.21	0.16(1.3)	0.21	0.17(0.7)	0.16	0.16(1)	**0.16**	**0.16(1)**
20 Newsgroups	Global Pooling CNN	1.95	0.83(3.4)	1.56	**0.82(2.3)**	1.77	1.10(2.2)	**0.93**	0.91(1.1)	1.31	1.05(1.5)	1.40	1.19(1.5)
SST Binary	Tree-LSTM	5.81	3.76(1.8)	6.38	2.48(2.5)	**3.82**	**2.70(1.5)**	3.99	3.20(1.2)	6.35	2.81(0.5)	4.84	3.24(0.7)

Table A.6.2: Classwise-ECE (%) computed for different approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets.

Thus, intuitively, such models should have a higher Top-5 accuracy. In Table A.6.7, we report the Top-5 accuracies for all our models on datasets where the number of classes is relatively high (i.e., on CIFAR-100 with 100 classes and Tiny-ImageNet with 200 classes). We observe focal loss with sample-dependent γ to produce the highest top-5 accuracies on all models trained on CIFAR-100 and the second best top-5 accuracy (only marginally below the highest accuracy) on Tiny-ImageNet.

In addition to ECE, Ada-ECE and Classwise-ECE, we use various other metrics to compare the proposed methods with the baselines (i.e. cross-entropy, Brier loss, MMCE and Label Smoothing). We present the test NLL % before and after temperature scaling in Tables A.6.8 and A.6.9, respectively. We report the test set MCE % before and after temperature scaling in Tables A.6.10 and A.6.11, respectively.

We use the following abbreviation to report results on different varieties of Focal Loss.

FL-1 refers to Focal Loss (fixed γ 1), FL-2 refers to Focal Loss (fixed γ 2), FL-3 refers to Focal Loss (fixed γ 3), FLSc-531 refers to Focal Loss (scheduled γ 5,3,1), FLSc-532 refers to Focal Loss (scheduled γ 5,3,2), FLSD-532 refers to Focal Loss (sample-dependent γ 5,3,2) and FLSD-53 refers to Focal Loss (sample-dependent γ 5,3).

Dataset	**Model**	**FL-1**		**FL-2**		**FL-3**		**FLSc-531**		**FLSc-532**		**FLSD-532**		**FLSD-53**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	12.86	2.3(1.5)	8.61	2.24(1.3)	5.13	1.97(1.1)	11.63	2.09(1.4)	8.47	2.13(1.3)	9.09	1.61(1.3)	4.5	2.(1.1)
	ResNet-110	15.08	4.55(1.5)	11.57	3.73(1.3)	8.64	3.95(1.2)	14.99	4.56(1.5)	11.2	3.43(1.3)	11.74	3.64(1.3)	8.56	4.12(1.2)
	Wide-ResNet-26-10	8.93	2.53(1.4)	4.64	2.93(1.2)	2.13	2.13(1)	9.36	2.48(1.4)	4.98	1.94(1.2)	4.98	2.55(1.2)	3.03	1.64(1.1)
	DenseNet-121	14.24	2.8(1.5)	7.9	2.33(1.2)	4.15	1.25(1.1)	13.05	2.08(1.5)	7.63	1.96(1.2)	8.14	2.35(1.3)	3.73	1.31(1.1)
CIFAR-10	ResNet-50	3.42	1.08(1.6)	2.36	0.91(1.2)	1.48	1.42(1.1)	4.06	1.53(1.6)	2.97	1.53(1.2)	2.52	0.88(1.3)	1.55	0.95(1.1)
	ResNet-110	3.46	1.2(1.6)	2.7	0.89(1.3)	1.55	1.02(1.1)	4.92	1.5(1.7)	3.33	1.36(1.3)	2.82	0.97(1.3)	1.87	1.07(1.1)
	Wide-ResNet-26-10	2.69	1.46(1.3)	1.42	1.03(1.1)	1.69	0.97(0.9)	2.81	0.96(1.4)	1.82	1.45(1.1)	1.31	0.87(1.1)	1.56	0.84(0.9)
	DenseNet-121	3.44	1.63(1.4)	1.93	1.04(1.1)	1.32	1.26(0.9)	4.12	1.65(1.5)	2.22	1.34(1.1)	2.45	1.31(1.2)	1.22	1.22(1)
Tiny-ImageNet	ResNet-50	7.61	3.29(1.2)	3.02	3.02(1)	1.87	1.87(1)	7.77	3.07(1.2)	3.62	2.54(1.1)	2.81	2.57(1.1)	1.76	1.76(1)
20 Newsgroups	Global Pooling CNN	15.06	2.14(2.6)	12.1	3.22(1.6)	8.67	3.51(1.5)	13.55	4.32(1.7)	12.13	2.47(1.8)	12.2	2.39(2)	6.92	2.19(1.5)
SST Binary	Tree-LSTM	6.78	3.29(1.6)	3.05	3.05(1)	16.05	1.78(0.5)	4.66	3.36(1.4)	3.91	2.64(0.9)	4.47	2.77(0.9)	9.19	1.83(0.7)

Table A.6.3: ECE (%) computed for different focal loss approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets.

Dataset	**Model**	**FL-1**		**FL-2**		**FL-3**		**FLSc-531**		**FLSc-532**		**FLSD-532**		**FLSD-53**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	12.86	2.54(1.5)	8.55	2.44(1.3)	5.08	2.02(1.1)	11.58	2.01(1.4)	8.41	2.25(1.3)	9.08	1.94(1.3)	4.39	2.33(1.1)
	ResNet-110	15.08	5.16(1.5)	11.57	4.46(1.3)	8.64	4.14(1.2)	14.98	4.97(1.5)	11.18	3.68(1.3)	11.74	4.21(1.3)	8.55	3.96(1.2)
	Wide-ResNet-26-10	8.93	2.74(1.4)	4.65	2.96(1.2)	2.08	2.08(1)	9.2	2.52(1.4)	5	2.11(1.2)	5	2.58(1.2)	2.75	1.63(1.1)
	DenseNet-121	14.24	2.71(1.5)	7.9	2.36(1.2)	4.15	1.23(1.1)	13.01	2.18(1.5)	7.61	2.04(1.2)	8.04	2.1(1.3)	3.55	1.24(1.1)
CIFAR-10	ResNet-50	3.42	1.51(1.6)	2.37	1.69(1.2)	1.95	1.83(1.1)	4.06	2.43(1.6)	2.95	2.18(1.2)	2.5	1.23(1.3)	1.56	1.26(1.1)
	ResNet-110	3.42	1.57(1.6)	2.69	1.29(1.3)	1.62	1.44(1.1)	4.91	2.61(1.7)	3.32	1.92(1.3)	2.78	1.58(1.3)	2.07	1.67(1.1)
	Wide-ResNet-26-10	2.7	1.71(1.3)	1.64	1.47(1.1)	1.84	1.54(0.9)	2.75	1.85(1.4)	2.04	1.9(1.1)	1.68	1.49(1.1)	1.52	1.38(0.9)
	DenseNet-121	3.44	1.85(1.4)	1.8	1.39(1.1)	1.22	1.48(0.9)	4.11	2.2(1.5)	2.19	1.64(1.1)	2.44	1.6(1.2)	1.42	1.42(1)
Tiny-ImageNet	ResNet-50	7.56	2.95(1.2)	3.15	3.15(1)	1.88	1.88(1)	7.7	2.9(1.2)	3.76	2.4(1.1)	2.81	2.6(1.1)	1.42	1.42(1)
20 Newsgroups	Global Pooling CNN	15.06	2.22(2.6)	12.1	3.33(1.6)	8.65	3.78(1.5)	13.55	4.58(1.7)	12.13	2.49(1.8)	12.19	2.37(2)	6.92	2.35(1.5)
SST Binary	Tree-LSTM	6.27	4.59(1.6)	3.69	3.69(1)	16.01	2.16(0.5)	4.43	3.57(1.4)	3.37	2.46(0.9)	4.42	2.96(0.9)	9.15	1.92(0.7)

Table A.6.4: AdaECE (%) computed for different focal loss approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets.

Dataset	Model	FL-1		FL-2		FL-3		FLSc-531		FLSc-532		FLSD-532		FLSD-53	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-100	ResNet-50	0.30	0.21(1.5)	0.24	0.22(1.3)	0.20	0.20(1.1)	0.28	0.21(1.4)	0.24	0.21(1.3)	0.25	0.21(1.3)	0.20	0.20(1.1)
	ResNet-110	0.34	0.21(1.5)	0.28	0.20(1.3)	0.24	0.22(1.2)	0.34	0.22(1.5)	0.28	0.21(1.3)	0.28	0.21(1.3)	0.24	0.21(1.2)
	Wide-ResNet-26-10	0.23	0.19(1.4)	0.18	0.21(1.2)	0.18	0.18(1)	0.24	0.19(1.4)	0.19	0.20(1.2)	0.19	0.19(1.2)	0.18	0.19(1.1)
	DenseNet-121	0.33	0.22(1.5)	0.23	0.21(1.2)	0.20	0.20(1.1)	0.31	0.22(1.5)	0.23	0.20(1.2)	0.23	0.22(1.3)	0.19	0.20(1.1)
CIFAR-10	ResNet-50	0.73	0.40(1.6)	0.54	0.45(1.2)	0.43	0.48(1.1)	0.85	0.51(1.6)	0.64	0.48(1.2)	0.56	0.42(1.3)	0.42	0.42(1.1)
	ResNet-110	0.74	0.44(1.6)	0.61	0.42(1.3)	0.44	0.41(1.1)	1.02	0.62(1.7)	0.73	0.51(1.3)	0.63	0.44(1.3)	0.48	0.44(1.1)
	Wide-ResNet-26-10	0.59	0.39(1.3)	0.37	0.36(1.1)	0.44	0.36(0.9)	0.61	0.37(1.4)	0.42	0.42(1.1)	0.37	0.33(1.1)	0.41	0.31(0.9)
	DenseNet-121	0.74	0.44(1.4)	0.45	0.39(1.1)	0.43	0.41(0.9)	0.88	0.47(1.5)	0.52	0.43(1.1)	0.56	0.47(1.2)	0.41	0.41(1)
Tiny-ImageNet	ResNet-50	0.18	0.17(1.2)	0.16	0.16(1)	0.16	0.16(1)	0.18	0.17(1.2)	0.16	0.16(1.1)	0.16	0.17(1.1)	0.16	0.16(1)
20 Newsgroups	Global Pooling CNN	1.73	0.98(2.6)	1.51	1.00(1.6)	1.31	1.05(1.5)	1.59	1.15(1.7)	1.54	1.10(1.8)	1.50	0.99(2)	1.40	1.19(1.5)
SST Binary	Tree-LSTM	5.46	3.63(1.6)	3.34	3.34(1)	6.35	2.81(0.5)	4.32	3.60(1.4)	3.95	3.90(0.9)	3.52	3.39(0.9)	4.84	3.24(0.7)

Table A.6.5: Classwise-ECE (%) computed for different focal loss approaches both pre and post temperature scaling (cross-validating T on ECE). Optimal temperature for each method is indicated in brackets.

Dataset	Model	FL-1	FL-2	FL-3	FLSc-531	FLSc-532	FLSD-532	FLSD-53
CIFAR-100	ResNet-50	22.8	23.15	22.75	23.49	23.24	23.55	23.22
	ResNet-110	22.36	22.53	22.92	22.81	22.96	22.93	22.51
	Wide-ResNet-26-10	19.61	20.01	19.69	20.13	20.13	19.71	20.11
	DenseNet-121	23.82	23.19	23.25	23.69	23.72	22.41	22.67
CIFAR-10	ResNet-50	4.93	4.98	5.25	5.66	5.63	5.24	4.98
	ResNet-110	4.78	5.06	5.08	6.13	5.71	5.19	5.42
	Wide-ResNet-26-10	4.27	4.27	4.13	4.11	4.46	4.14	4.01
	DenseNet-121	5.09	4.84	5.33	5.46	5.65	5.46	5.46
Tiny-ImageNet	ResNet-50	50.06	47.7	49.69	50.49	49.83	48.95	49.06
20 Newsgroups	Global Pooling CNN	26.13	28.23	29.26	29.16	28.16	27.26	27.98
SST Binary	Tree-LSTM	12.63	12.3	12.19	12.36	13.07	12.3	12.8

Table A.6.6: Error (%) computed for different focal loss approaches.

Dataset	Model	Cross-Entropy		Brier Loss		MMCE		LS-0.05		FLSD-53 (Ours)	
		Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
CIFAR-100	ResNet-50	76.7	93.77	76.61	93.24	76.8	93.69	76.57	92.86	76.78	**94.44**
	ResNet-110	77.27	93.79	74.9	92.44	76.93	93.78	76.57	92.27	77.49	**94.78**
	Wide-ResNet-26-10	79.3	93.96	79.41	94.56	79.27	94.11	78.81	93.18	79.89	**95.2**
	DenseNet-121	75.48	91.33	76.25	92.76	76	91.96	75.95	89.51	77.33	**94.49**
Tiny-ImageNet	ResNet-50	50.19	74.24	46.8	70.34	48.69	73.52	52.88	**76.15**	50.94	76.07

Table A.6.7: Top-1 and Top-5 accuracies computed for different approaches.

Dataset	Model	Cross-Entropy	Brier Loss	MMCE	LS-0.05	FL-1	FL-2	FL-3	FLSc-531	FLSc-532	FLSD-532	FLSD-53
CIFAR-100	ResNet-50	153.67	99.63	125.28	121.02	105.61	92.82	87.52	100.09	92.66	94.1	88.03
	ResNet-110	179.21	110.72	180.54	133.11	114.18	96.74	90.9	112.46	95.85	97.97	89.92
	Wide-ResNet-26-10	140.1	84.62	119.58	108.06	87.56	77.8	74.66	88.61	78.52	78.86	76.92
	DenseNet-121	205.61	98.31	166.65	142.04	115.5	93.11	87.13	107.91	93.12	91.14	85.47
CIFAR-10	ResNet-50	41.21	18.67	44.83	27.68	22.67	18.6	18.43	25.32	20.5	18.69	17.55
	ResNet-110	47.51	20.44	55.71	29.88	22.54	19.19	17.8	32.77	22.48	19.39	18.54
	Wide-ResNet-26-10	26.75	15.85	28.47	21.71	17.66	14.96	15.2	18.5	15.57	14.78	14.55
	DenseNet-121	42.93	19.11	52.14	28.7	22.5	17.56	18.02	27.41	19.5	20.14	18.39
Tiny-ImageNet	ResNet-50	232.85	240.32	234.29	235.04	219.07	202.92	207.2	217.52	211.42	204.71	204.97
20 Newsgroups	Global Pooling CNN	176.57	130.41	158.7	90.95	140.4	115.97	109.62	128.75	123.72	124.03	109.17
SST Binary	Tree-LSTM	50.2	54.96	37.28	44.34	53.9	47.72	50.29	50.25	53.13	45.08	49.23

Table A.6.8: NLL $(\%)$ computed for different approaches pre temperature scaling.

Dataset	Model	Cross-Entropy	Brier Loss	MMCE	LS-0.05	FL-1	FL-2	FL-3	FLSc-531	FLSc-532	FLSD-532	FLSD-53
CIFAR-100	ResNet-50	106.83	99.57	101.92	120.19	94.58	91.80	87.37	92.77	91.58	92.83	88.27
	ResNet-110	104.63	111.81	106.73	129.76	94.65	91.24	89.92	93.73	91.30	92.29	88.93
	Wide-ResNet-26-10	97.10	85.77	95.92	108.06	83.68	80.44	74.66	84.11	80.01	80.40	78.14
	DenseNet-121	119.23	98.74	113.24	136.28	100.81	91.35	87.55	98.16	91.55	90.57	86.06
CIFAR-10	ResNet-50	20.38	18.36	21.58	27.69	17.56	17.67	18.34	19.93	19.25	17.28	17.37
	ResNet-110	21.52	19.60	24.61	29.88	17.32	17.53	17.62	23.79	20.21	17.78	18.24
	Wide-ResNet-26-10	15.33	15.85	16.16	21.19	15.48	14.85	15.06	15.81	15.38	14.69	14.23
	DenseNet-121	21.77	19.11	24.88	28.95	18.71	17.21	18.10	21.65	19.04	19.27	18.39
Tiny-ImageNet	ResNet-50	220.98	238.98	226.29	214.95	217.51	202.92	207.20	215.37	211.57	205.42	204.97
20 Newsgroups	Global Pooling CNN	87.95	93.11	99.74	90.42	87.24	93.60	94.69	97.89	93.66	91.73	93.98
SST Binary	Tree-LSTM	41.05	38.27	36.37	43.45	45.67	47.72	45.96	45.82	54.52	45.36	49.69

Table A.6.9: NLL $(\%)$ computed for different approaches post temperature scaling (cross-validating T on ECE).

Dataset	Model	Cross-Entropy	Brier Loss	MMCE	LS-0.05	FL-1	FL-2	FL-3	FLSc-531	FLSc-532	FLSD-532	FLSD-53
CIFAR-100	ResNet-50	44.34	36.75	39.53	26.11	33.22	21.03	13.02	26.76	23.56	22.4	16.12
	ResNet-110	55.92	24.85	50.69	36.23	40.49	32.57	26	37.24	29.56	34.73	22.57
	Wide-ResNet-26-10	49.36	14.68	40.13	23.79	27	15.14	9.96	27.81	17.59	13.64	10.17
	DenseNet-121	56.28	15.47	49.97	43.59	35.45	21.7	11.61	38.68	18.91	21.34	9.68
CIFAR-10	ResNet-50	38.65	31.54	60.06	35.61	31.75	25	21.83	30.54	23.57	25.45	14.89
	ResNet-110	44.25	25.18	67.52	45.72	73.35	25.92	25.15	34.18	30.38	30.8	18.95
	Wide-ResNet-26-10	48.17	77.15	36.82	24.89	29.17	30.17	23.86	37.57	30.65	18.51	74.07
	DenseNet-121	45.19	19.39	43.92	45.5	38.03	29.59	77.08	33.5	16.47	17.85	13.36
Tiny-ImageNet	ResNet-50	30.83	8.41	26.48	25.48	20.7	8.47	6.11	16.03	9.28	8.97	3.76
20 Newsgroups	Global Pooling CNN	36.91	31.35	34.72	8.93	34.28	24.1	18.85	26.02	25.02	24.29	17.44
SST Binary	Tree-LSTM	71.08	92.62	68.43	39.39	95.48	86.21	22.32	76.28	86.93	80.85	73.7

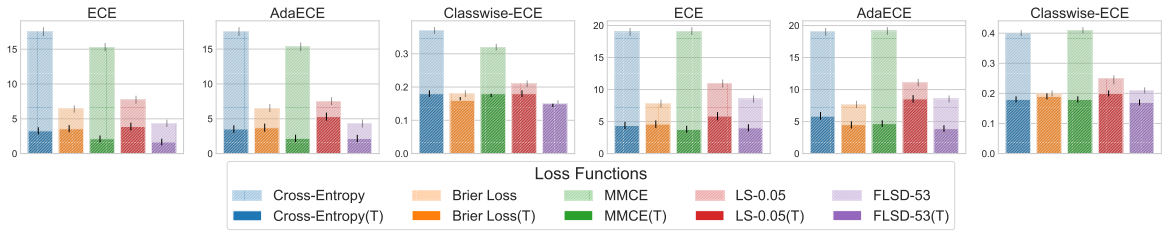
Table A.6.10: MCE $(\%)$ computed for different approaches pre temperature scaling.

A.7 Bar plots

In this section, we present additional results in reference to Figure 3.4 in the main paper. In particular, we compute 90% confidence intervals for ECE, AdaECE and Classwise-

Dataset	Model	Cross-Entropy	Brier Loss	MMCE	LS-0.05	FL-1	FL-2	FL-3	FLSc-531	FLSc-532	FLSD-532	FLSD-53
CIFAR-100	ResNet-50	12.75	21.61	11.99	18.58	8.92	8.86	6.76	7.46	6.76	5.24	27.18
	ResNet-110	22.65	13.56	19.23	30.46	20.13	12	13.06	18.28	13.72	15.89	10.94
	Wide-ResNet-26-10	14.18	13.42	16.5	23.79	10.28	18.32	9.96	13.18	11.01	12.5	9.73
	DenseNet-121	21.63	8.55	13.02	29.95	10.49	11.63	6.17	6.21	6.48	9.41	5.68
CIFAR-10	ResNet-50	20.6	22.46	23.6	40.51	25.86	28.17	15.76	22.05	23.85	24.76	26.37
	ResNet-110	29.98	22.73	31.87	45.72	29.74	23.82	37.61	26.25	25.94	11.59	17.35
	Wide-ResNet-26-10	26.63	77.15	32.33	37.53	74.58	29.58	25.64	28.63	20.23	19.68	36.56
	DenseNet-121	32.52	19.39	27.03	53.57	19.68	22.71	76.27	21.05	32.76	35.06	13.36
Tiny-ImageNet	ResNet-50	13.33	12.82	12.52	17.2	6.5	8.47	6.11	5.97	7.01	5.73	3.76
20 Newsgroups	Global Pooling CNN	36.91	31.35	34.72	8.93	34.28	24.1	18.85	26.02	25.02	24.29	17.44
SST Binary	Tree-LSTM	88.48	91.86	32.92	35.72	87.77	86.21	74.52	54.27	88.85	82.42	76.71

Table A.6.11: MCE (%) computed for different approaches post temperature scaling (cross-validating T on ECE).



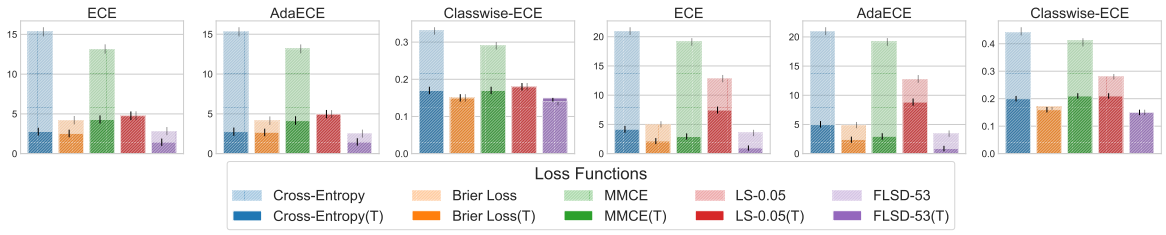
(a) ResNet-50 on CIFAR-100 (b) ResNet-110 on CIFAR-100

Figure A.7.1: Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for ResNet-50 (first 3 figures) and ResNet-110 (last 3 figures) on CIFAR-100.

ECE using 1000 bootstrap samples following [166] and present the resulting confidence intervals as bar plots in Figures A.7.1, A.7.2, A.7.3 and A.7.4. These plots further corroborate the observations made in Section 3.5 of the main paper. We find that FLSD-53 broadly produces the lowest calibration errors, and in quite a few cases (especially before temperature scaling) the differences in calibration errors between cross-entropy and focal loss are statistically significant.

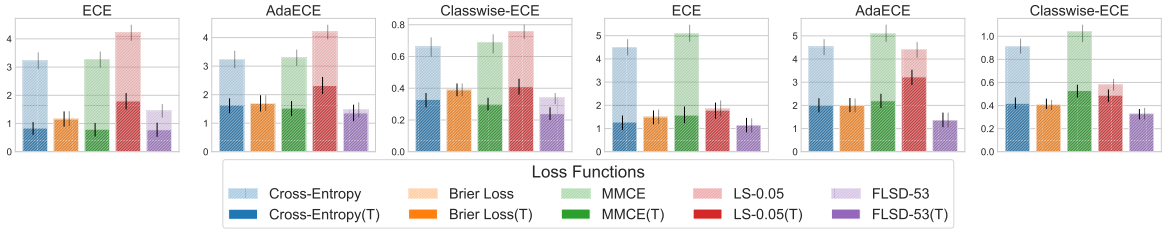
A.8 Focal Loss is Confident and Calibrated

In extension to what we present in Section 5 of the main paper, we also follow the approach adopted in [74], and measure the percentage of test samples that are predicted with a confidence of 0.99 or more (we call this set of test samples $S99$). In Table A.8.1, we report $|S99|$ as a percentage of the total number of test samples, along with the accuracy



(a) Wide-ResNet-26-10 on CIFAR-100 (b) DenseNet-121 on CIFAR-100

Figure A.7.2: Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for Wide-ResNet-26-10 (first 3 figures) and DenseNet-121 (last 3 figures) on CIFAR-100.



(a) Wide-ResNet-26-10 on CIFAR-10 (b) DenseNet-121 on CIFAR-10

Figure A.7.3: Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for Wide-ResNet-26-10 (first 3 figures) and DenseNet-121 (last 3 figures) on CIFAR-10.

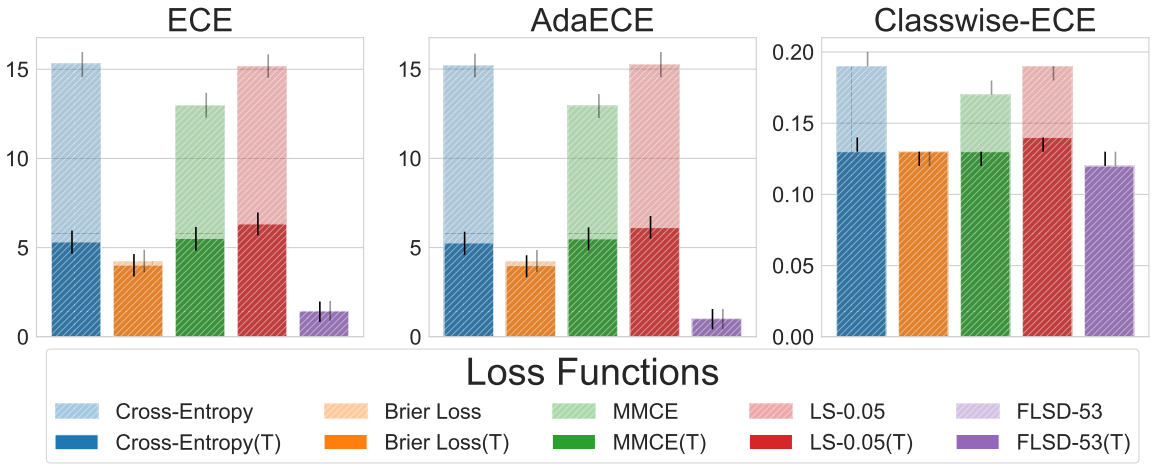


Figure A.7.4: Bar plots with confidence intervals for ECE, AdaECE and Classwise-ECE computed for ResNet-50 on Tiny-ImageNet.

of the samples in $S99$ for ResNet-50 and ResNet-110 trained on CIFAR-10, using cross-entropy loss, MMCE loss, and focal loss. We observe that $|S99|$ for the focal loss model is much lower than for the cross-entropy or MMCE models before temperature scaling. However, after temperature scaling, $|S99|$ for focal loss is significantly higher than for

Dataset	Model	Cross-Entropy (Pre T)		Cross-Entropy (Post T)		MMCE (Pre T)		MMCE (Post T)		Focal Loss (Pre T)		Focal Loss (Post T)	
		|S99|%	Accuracy	|S99|%	Accuracy	|S99|%	Accuracy	|S99|%	Accuracy	|S99|%	Accuracy	|S99|%	Accuracy
CIFAR-10	ResNet-110	97.11	96.33	11.5	97.39	97.65	96.72	10.62	99.83	61.41	99.51	31.10	99.68
CIFAR-10	ResNet-50	95.93	96.72	7.33	99.73	92.33	98.24	4.21	100	46.31	99.57	14.27	99.93

Table A.8.1: Percentage of test samples predicted with confidence higher than 99% and the corresponding accuracy for Cross Entropy, MMCE and Focal loss computed both pre and post temperature scaling (represented in the table as pre T and post T respectively).

both MMCE and cross-entropy. The reason is that with an optimal temperature of 1.1, the confidence of the temperature-scaled model for focal loss does not reduce as much as those of the models for cross-entropy and MMCE, for which the optimal temperatures lie between 2.5 to 2.8. We thus conclude that models trained on focal loss are not only more calibrated, but also better preserve their confidence on predictions, even after being post-processed with temperature scaling.

In Figure A.8.1, we present some qualitative results to support this claim and show the improvement in the confidence estimates of focal loss in comparison to other baselines (i.e., cross entropy, MMCE and Brier loss). For this, we take ResNet-50 networks trained on CIFAR-10 using the four loss functions (cross entropy, MMCE, Brier loss and Focal loss with sample-dependent γ 5,3) and measure the confidence of their predictions for four correctly and four incorrectly classified test samples. We report these confidences both before and after temperature scaling. It is clear from Figure A.8.1 that for all the correctly classified samples, the model trained using focal loss has very confident predictions both pre and post temperature scaling. However, on misclassified samples, we observe a very low confidence for the focal loss model. The ResNet-50 network trained using cross entropy is very confident even on the misclassified samples, particularly before temperature scaling. Apart from focal loss, the only model which has relatively low confidences on misclassified test samples is the one trained using Brier loss. These observations support our claim that focal loss produces not only a calibrated model but also one which is confident on its correct predictions.

A.9 Ordering of Feature Norms

As an extension to the discussion related to Figure 3.2(e) in the main paper, we plot the L_2 norm of the features/activations obtained from the last ResNet block (right before the linear layer is applied on these features to get the logits). We plot these norms throughout the

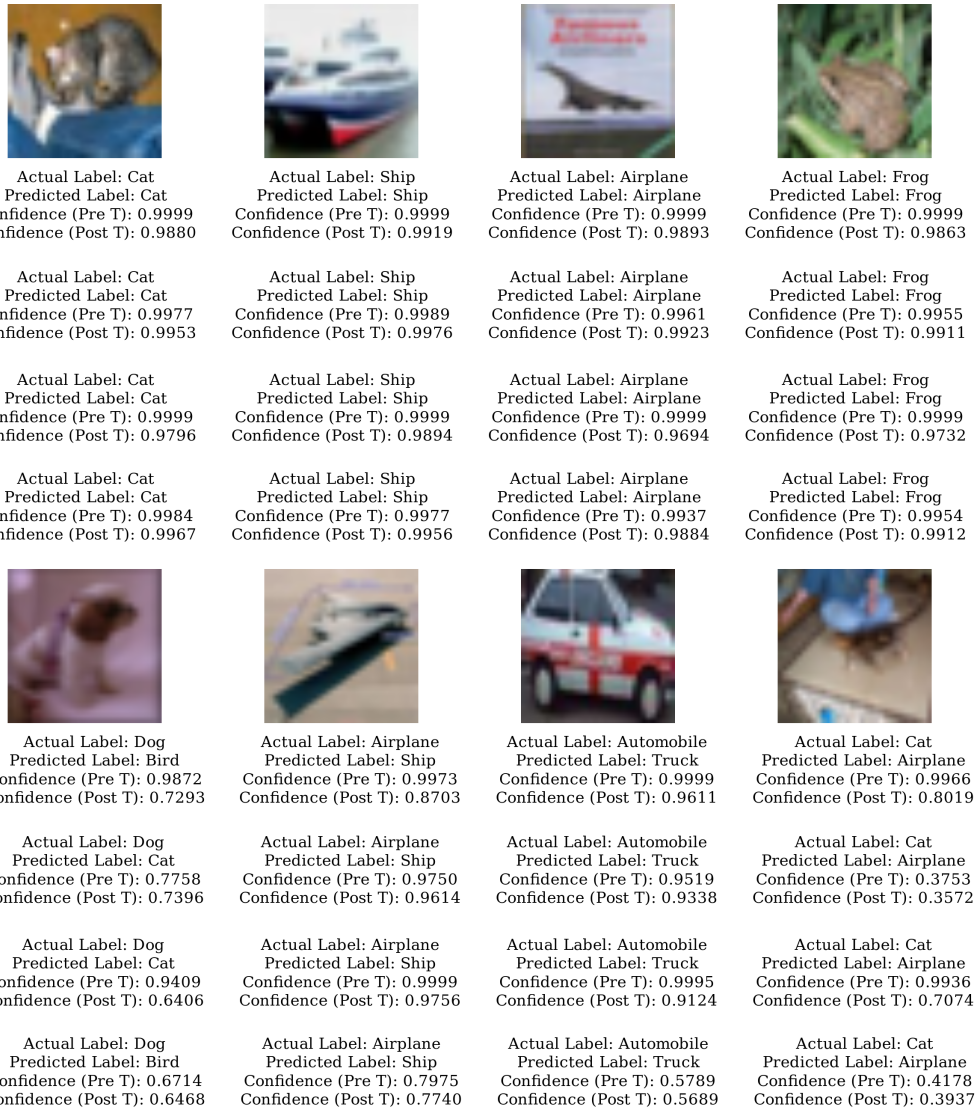


Figure A.8.1: Qualitative results showing the performance of Cross Entropy, Brier loss, MMCE and Focal Loss (sample-dependent γ 5,3) for a ResNet-50 trained on CIFAR-10. The first row of images have been correctly classified by networks trained on all four loss functions and the second row of images have all been incorrectly classified. For each image, we present the actual label, the predicted label and the confidence of the prediction both before and after temperature scaling.

training period for networks trained on cross-entropy and focal loss with γ set to $1, 2$ and 3 in Figure A.9.1. We observe that there is a distinct ordering of feature norms for the four models: cross-entropy has the highest feature norm, followed by focal loss with $\gamma = 1$, followed by focal loss with $\gamma = 2$ and finally focal loss with $\gamma = 3$. Furthermore, this ordering is preserved throughout training. As we saw from Figure 3.2(e) in the main paper, from epoch 150 onwards (i.e., the epoch from which the networks start getting miscalibrated), there is a flip in the ordering of weight norms of the last linear layer. From epoch 150 onwards, the weight norms also follow the exact same ordering that we observe from Figure A.9.1 here. This shows that throughout training the initial layer weights (before last linear layer) of the network trained using focal loss are also regularized to favor lower norm of the output features, thus possibly leading to less peakiness in final prediction as compared to that of cross-entropy loss (see the 'Peak at the wrong place' paragraph of Section 3 of the main paper).

A.10 Early stopping

From Figure 3.2(a), one may think that intermediate models (using early stopping) might provide better accuracy and calibration. However, there is no ideal approach for early stopping. For fair comparison, we train ResNet50, CIFAR-10 using cross-entropy and focal loss with the best (in hindsight) possible early stopping. We train each model for 350 epochs and choose the 3 intermediate models with the best val set ECE, NLL and classification error, respectively. We present the test set performance in Table A.10.1.

From the table, we can observe that: 1) On every early stopping criterion, the model trained on focal loss outperforms the one trained on cross-entropy in both error and ECE, 2) ECE as a stopping criterion provides better test set ECE, but increases the test error significantly, 3) even without early stopping, focal loss achieves consistently better error and ECE compared to cross-entropy using any stopping criterion.

Criterion	Loss	Epoch	Error	ECE %
ECE	CE	151	7.34	1.69
ECE	FLSD-53	257	5.52	0.85
NLL	CE	153	6.69	2.28
NLL	FLSD-53	266	5.34	1.33
Error	CE	344	5.0	4.46
Error	FLSD-53	343	4.99	1.43
Full	CE	350	4.95	4.35
Full	FLSD-53	350	4.98	1.55

Table A.10.1: Classification errors and ECE scores obtained from ResNet-50 models trained using cross-entropy and focal loss with different early stopping criteria (best in hindsight ECE, NLL and classification error on the validation set) applied during training. In the table CE and FL stand for cross-entropy and focal loss respectively and the Full Criterion indicates models where early stopping has not been applied.

A.11 Machine Translation: A Downstream Task

In this section, we explore machine translation with beam search as a relevant downstream task for calibration. Following the setup in [63], we train the Transformer architecture [16] on the WMT 2014 English-to-German translation dataset. The training settings (like optimiser, LR schedule, etc.) are the same as [16]. We chose machine translation as a relevant task because the softmax vectors produced by the transformer model are directly fed into the beam search algorithm, and hence softmax outputs from a calibrated model should intuitively produce better translations and a better BLEU score.

We train three transformer models, one on cross-entropy with hard target labels, the second on cross-entropy with label smoothing (with smoothing factor $\alpha = 0.1$) and the third on focal loss with $\gamma = 1$. In order to compare these models in terms of calibration, we report the test set ECE (%) both before and after temperature scaling in the first row of Table A.11.1. Furthermore, to evaluate their performance on the English-to-German translation task, we also report the test set BLEU score of these models in the second row of Table A.11.1. Finally, to study the variation of test set BLEU score and validation set ECE with temperature, we plot them against temperature for all three models in Figure A.11.1.

We observe from Table A.11.1 that the model trained on focal loss outperforms its competitors on ECE and also has a competitive edge over other methods on BLEU score

as well. The focal loss model also has an optimal temperature of 1, just like the model trained on cross-entropy with label smoothing. From Figure A.11.1, we can see that the models obtain the highest BLEU scores at around the same temperatures at which they obtain low ECEs, thereby confirming our initial notion that a more calibrated model provides better translations. However, since the optimal temperatures are tuned on the validation set, they don't often correspond to the best BLEU scores on the test set. On the test set, the highest BLEU scores we observe are 26.33 for cross-entropy, 26.36 for cross-entropy with label smoothing, and 26.39 for focal loss. Thus, the performance of focal loss on machine translation (a downstream task related to calibration) is also very encouraging.

Metrics	**CE** ($\alpha = 0.0$)	**LS** ($\alpha = 0.1$)	**FL** ($\gamma = 1.0$)
ECE% Pre T / Post T / T	$10.16/2.59/1.2$	$3.25/3.25/1.0$	$\mathbf{1.69}/\mathbf{1.69}/1.0$
BLEU Pre T / Post T	$26.31/26.21$	$26.33/26.33$	$\mathbf{26.39}/\mathbf{26.39}$

Table A.11.1: Test set ECE and BLEU score both pre and post temperature scaling for cross-entropy (CE) with hard targets, cross-entropy with label smoothing (LS) ($\alpha = 0.1$) and focal loss (FL) ($\gamma = 1$).

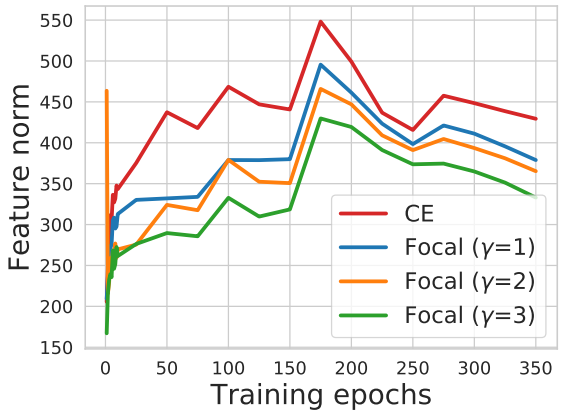


Figure A.9.1: L_2 norm of features obtained from the last ResNet block (before the linear layer) of ResNet-50 averaged over entire training dataset of CIFAR-10 using a batch size of 128.

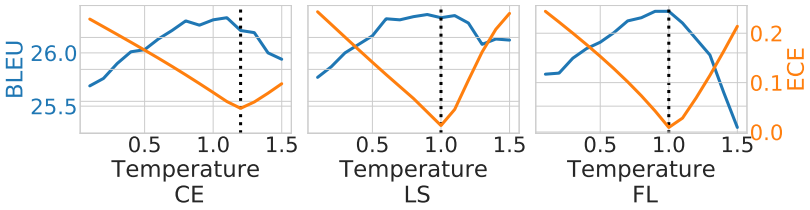


Figure A.11.1: Changes in *test set* BLEU score and *validation set* ECE with temperature, for models trained using (a) cross-entropy with hard targets (CE) (b) cross-entropy with label smoothing (LS) ($\alpha = 0.1$), and (c) focal loss (FL) ($\gamma = 1$).

B

Appendix: Deep Deterministic Uncertainty: A New Simple Baseline

Contents

B.1 Related Work . . . 177
B.1.1 Predictive Entropy and Confidence in Recent Works . . . 178
B.2 Ambiguous- and Dirty-MNIST . . . 178
B.3 Algorithm . . . 179
B.3.1 Increasing sensitivity . . . 179
B.3.2 Algorithm Implementation . . . 180
B.3.3 Computational Complexity . . . 181
B.4 Experimental Details . . . 182
B.4.1 Dirty-MNIST . . . 182
B.4.2 OoD Detection Training Setup . . . 182
B.4.3 Semantic Segmentation Training Setup . . . 183
B.4.4 Compute Resources . . . 183
B.5 Additional Results . . . 184
B.6 Ablations and Toy Experiments . . . 189
B.6.1 Motivational Example in Figure 4.1 . . . 190
B.6.2 Effects of a well-regularized feature space on feature collapse . . . 192
B.6.3 QUBIQ Challenge . . . 193
B.6.4 Additional Baselines . . . 193
B.6.5 Calibration Metrics . . . 194
B.6.6 Two Moons . . . 195
B.6.7 5-Ensemble Visualisation . . . 196

B.6.8 Feature-Space Density &Epistemic Uncertainty vs Softmax Entropy &Aleatoric Uncertainty 197
B.6.9 Objective Mismatch Ablation 199
B.7 Theoretical Results 199
B.7.1 Softmax entropy "cannot" capture epistemic uncertainty because Deep Ensembles "can" 199
B.7.2 Objective Mismatch 200

B.1 Related Work

Several existing approaches model uncertainty using feature-space density but underperform without fine-tuning on OoD data. Our work identifies feature collapse and objective mismatch as possible reasons for this. Among these approaches, [179] uses Mahalanobis distances to quantify uncertainty by fitting a class-wise Gaussian distribution (with shared covariance matrices) on the feature space of a pre-trained ResNet encoder. The competitive results they report require input perturbations, ensembling GMM densities from multiple layers, and fine-tuning on OoD hold-out data. They do not discuss any constraints which the ResNet encoder should satisfy, and therefore, are vulnerable to feature collapse. In Figure 4.1c, for example, the feature density of a LeNet and a VGG are unable to distinguish OoD from iD samples. [186] also propose a density-based estimation of aleatoric and epistemic uncertainty. Similar to [179], they do not constrain their pre-trained ResNet encoder. They do discuss feature collapse though, noting that they do not address this problem. Moreover, they do not consider the objective mismatch that arises (see Observation 4.5.3 below) and use a single estimator for both epistemic and aleatoric uncertainty. Consequently, they report worse epistemic uncertainty: 74% AUROC on CIFAR-10 vs SVHN, which we show to considerably fall behind modern approaches for uncertainty estimation in deep learning in §4.4. Likewise, [195] compute an unnormalized density based on the softmax logits without taking into account the need for inductive biases to ensure smoothness and sensitivity of the feature space.

[187] use contrastive training on the feature extractor before estimating the feature-space density. Our method is orthogonal from this work as we restrict ourselves to the supervised setting and show that the inductive biases that result in bi-Lipschitzness [57, 58] are sufficient for the feature-space density to reliably capture epistemic uncertainty.

Lastly, our method improves upon [57] and [58] by alleviating the need for additional hyperparameters: DDU only needs minimal changes from the standard softmax setup to outperform DUQ and SNGP on uncertainty benchmarks, and our GMM parameters are optimised for the already trained model using the training set. In particular, DDU does not require training or fine-tuning with OoD data. Moreover, our insights in §4.5 explain why [58] found that a baseline that uses *the softmax entropy instead of the feature-space density* of a deterministic network with bi-Lipschitz constraint underperforms.

B.1.1 Predictive Entropy and Confidence in Recent Works

Table B.1.1 shows a selection of recently published papers which use entropy or confidence as OoD score. Only two papers examine using Mutual Information with Deep Ensembles as OoD score at all. None of the papers examines the possible confounding of aleatoric and epistemic uncertainty when using predictive entropy or confidence, or the consistency issues of softmax entropy (and softmax confidence), detailed in §4.5. This list is not exhaustive, of course.

B.2 Ambiguous- and Dirty-MNIST

Each sample in Ambiguous-MNIST is constructed by decoding a linear combination of latent representations of 2 different MNIST digits from a pre-trained VAE [312]. Every decoded image is assigned several labels sampled from the softmax probabilities of an off-the-shelf MNIST neural network ensemble, with points filtered based on an ensemble's MI (to remove 'junk' images) and then stratified class-wise based on their softmax entropy (some classes are inherently more ambiguous, so we "amplify" these; we stratify per-class to try to preserve a wide spread of possible entropy values, and avoid introducing additional

Title	Citation	Softmax Confidence	Predictive Confidence	Predictive Entropy	Mutual Information
A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks	[177]	✓	✗	✗	✗
Deep Anomaly Detection with Outlier Exposure	[98]	✓	✗	✗	✗
Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks	[192]	✓	✗	✗	✗
Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples	[193]	✓	✗	✗	✗
Learning Confidence for Out-of-Distribution Detection in Neural Networks	[194]	✓	✗	✗	✗
Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles	[49]	✗	✓	✓	✗
Predictive Uncertainty Estimation via Prior Networks	[206]	✗	✓	✓	✓
Ensemble Distribution Distillation	[322]	✗	✗	✓	✓
Generalized ODIN: Detecting Out-of-Distribution Image Without Learning From Out-of-Distribution Data	[221]	✓	✗	✗	✗
Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks	[323]	✗	✓	✗	✗

Table B.1.1: *A sample of recently published papers and OoD metrics.* Many recently published papers only use Predictive Entropy or Predictive Confidence (for Deep Ensembles) or Softmax Confidence (for deterministic models) as OoD scores without addressing the possible confounding of aleatoric and epistemic uncertainty, that is ambiguous iD samples with OoD samples. Only two papers examine using Mutual Information with Deep Ensembles as OoD score at all.

ambiguity which will increase all points to have highest entropy). All off-the-shelf MNIST neural networks were then discarded and new models were trained to generate Fig 1 (and as can be seen, the ambiguous points we generate indeed have high entropy regardless of the model architecture used). We create 60K such training and 10K test images to construct Ambiguous-MNIST. Finally, the Dirty-MNIST dataset in this experiment contains MNIST and Ambiguous-MNIST samples in a 1:1 ratio (with 120K training and 20K test samples). In Figure B.2.1, we provide some samples from Ambiguous-MNIST.

B.3 Algorithm

B.3.1 Increasing sensitivity

Using residual connections to enforce sensitivity works well in practice when the layer is defined as $x' = x + f(x)$. However, there are several places in the network where additional spatial downsampling is done in $f(\cdot)$ (through a strided convolution), and in order to compute the residual operation x needs to be downsampled as well. These downsampling operations are crucial for managing memory consumption and generalisation. The way this is traditionally done in ResNets is by introducing an additional function $g(\cdot)$ on the residual



Figure B.2.1: *Samples from Ambiguous-MNIST.*

branch (obtaining $x' = g(x) + f(x)$) which is a strided 1x1 convolution. In practice, the stride is set to 2 pixels, which leads to the output of $g(\cdot)$ only being dependent on the top-left pixel of each 2x2 patch, which reduces sensitivity. We overcome this issue by making an architectural change that improves uncertainty quality without sacrificing accuracy. We use a strided average pooling operation instead of a 1x1 convolution in $g(\cdot)$. This makes the output of $g(\cdot)$ dependent on all input pixels. Additionally, we use leaky ReLU activation functions, which are equivalent to ReLU activations when the input is larger than 0, but below 0 they compute $p * x$ with $p = 0.01$ in practice. These further improve sensitivity as all negative activations still propagate in the network.

B.3.2 Algorithm Implementation

The algorithm is provided in Algorithm 1. Note that in order to compute thresholds for low and high density or entropy, we can simply use the training set containing iD data. We set

Algorithm 1 Deep Deterministic Uncertainty

Definitions:
- Regularized feature extractor $f_\theta : x \to \mathbb{R}^d$
- Softmax output predictions: $p(y|x)$
- GMM density: $q(z) = \sum_y q(z|y = c)\, q(y = c)$
- Dataset (X, Y)

procedure TRAIN
  train NN $p(y|f_\theta(x))$ with (X, Y)
  for each class c with samples $\mathbf{x}_c \subset X$ **do**
    $\mu_c \leftarrow \frac{1}{|\mathbf{x}_c|} \sum_{\mathbf{x}_c} f_\theta(\mathbf{x}_c)$
    $\Sigma_c \leftarrow \frac{1}{|\mathbf{x}_c|-1}(f_\theta(\mathbf{x}_c) - \mu_c)(f_\theta(\mathbf{x}_c) - \mu_c)^T$
    $\pi_c \leftarrow \frac{\sum_{\mathbf{x}_c} 1}{|X|}$
  end for
end procedure

function DISENTANGLE_UNCERTAINTY(sample x)
  compute feature representation $z = f_\theta(x)$
  compute density under GMM: $q(z) = \sum_y \mathrm{q}(z \mid y)\, \mathrm{q}(y)$ with $\mathrm{q}(z \mid y) \sim \mathcal{N}(\mu_y; \sigma_y), \mathrm{q}(y) = \pi_y$
  compute softmax entropy: $H_p[Y|x]$

  if low density $q(z)$ **then**
    return $(-q(z),\ )$
  else if high density $q(z)$ **then**
    if high entropy $H_p[Y|x]$ **then**
      return ambiguous iD
    else if low entropy $H_p[Y|x]$ **then**
      return $(0,\ H_p[Y|x])$
    end if
  end if
end function

all points having density lower than 99% quantile as OoD.

B.3.3 Computational Complexity

Let N be the number of samples; D, the feature space dimensionality; and C, the number of classes with $\approx N/C$ samples per class (balanced). For fitting the GMM via GDA: computing

the covariance matrix per class requires $(C(N/C)D^2)$ = (ND^2) complexity. Computing the inverse and determinant of the covariance matrices via the Cholesky decomposition requires (D^3) per class. Thus, the total computational cost for GDA is $(ND^2 + CD^3)$. Evaluating density of a single point: distance from class means requires (CD), and matrix vector multiplications requires (CD^2). Hence, the total cost for evaluating density on a single point is (CD^2).

B.4 Experimental Details

B.4.1 Dirty-MNIST

We train for 50 epochs using SGD with a momentum of 0.9 and an initial learning rate of 0.1. The learning rate drops by a factor of 10 at training epochs 25 and 40. Following SNGP [58], we apply online spectral normalisation with one step of a power iteration on the convolutional weights. For 1x1 convolutions, we use the exact algorithm, and for 3x3 convolutions, the approximate algorithm from [324]. The coefficient for SN is a hyper-parameter which we set to 3 using cross-validation.

B.4.2 OoD Detection Training Setup

We train the softmax baselines on CIFAR-10/100 for 350 epochs using SGD as the optimiser with a momentum of 0.9, and an initial learning rate of 0.1. The learning rate drops by a factor of 10 at epochs 150 and 250. We train the 5-Ensemble baseline using this same training setup. The SNGP and DUQ models were trained using the setup of SNGP and hyper-parameters mentioned in their respective papers [57, 58]. For models trained on ImageNet, we train for 90 epochs with SGD optimizer, an initial learning rate of 0.1 and a weight decay of 1e-4. We use a learning rate warmup decay of 0.01 along with a step scheduler with step size of 30 and a step factor of 0.1.

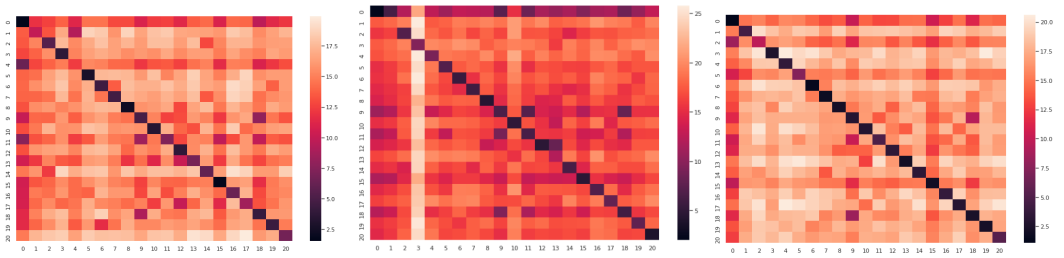


Figure B.4.1: L2 distances between the feature space means of different classes for a pair of distant pixels on the Pascal VOC 2012 val set: (left) Pixels $(10, 255)$ and $(500, 255)$, (middle) Pixels $(234, 349)$ and $(36, 22)$ and (right) Pixels $(300, 500)$ and $(400, 255)$.

B.4.3 Semantic Segmentation Training Setup

In Figure B.4.1, we plot the L2 distance between feature space means of different classes for a pair of randomly chosen distant pixels on the Pascal VOC 2012 val set. We observe that feature space means between pairs of different classes are more distant compared to the same class irrespective of the location of the pixel for the class. This leads us to construct a Gaussian mean and covariance matrix per class as opposed to one mean and one covariance matrix per class per pixel, thereby greatly reducing the computational load of fitting a GMM in semantic segmentation. Similar to classification, we treat each pixel in the training set as a separate sample and fit a single Gaussian mean and covariance matrix per class.

For the semantic segmentation experiment, we use a DeepLab-v3+ [183] model with a ResNet-101 backbone as the architecture of choice. We train each of the models on Pascal VOC for 50 epochs using SGD as the optimizer, with a momentum of 0.9 and a weight decay of $5e-4$. We set the initial learning rate to 0.007 with a polynomial decay during the course of training. Finally, we trained with a batch size of 32 parallelized over 4 GPUs.

B.4.4 Compute Resources

Each model (ResNet-18, Wide-ResNet-28-10, ResNet-50, ResNet-110, DenseNet-121 or VGG-16) used for the large scale active learning, CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet tasks was trained on a single Nvidia Quadro RTX 6000 GPU. Each model (LeNet, VGG-16 and ResNet-18) used to

Train Dataset	Method	Penalty	Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC SVHN (↑)	AUROC CIFAR-100 (↑)	AUROC Tiny-ImageNet (↑)
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	**95.04 ± 0.05**	**0.97 ± 0.04**	93.80 ± 0.41	88.91 ± 0.07	88.32 ± 0.07
	Energy-based [195]	-		Softmax Density			94.48 ± 0.44	88.84 ± 0.08	88.45 ± 0.08
	DUQ [57]	JP	Kernel Distance	Kernel Distance	94.05 ± 0.11	1.71 ± 0.07	93.14 ± 0.43	83.87 ± 0.27	84.28 ± 0.26
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	94.90 ± 0.11	1.01 ± 0.03	93.15 ± 0.85	89.32 ± 0.10	88.96 ± 0.13
	DDU (ours)	SN	Softmax Entropy	GMM Density	94.92 ± 0.06	1 ± 0.04	**94.77 ± 0.35**	**89.98 ± 0.17**	**89.12 ± 0.13**
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	**96.06 ± 0.04**	1.65 ± 0.07	94.75 ± 0.39	89.87 ± 0.06	88.69 ± 0.05
				Mutual Information			94.09 ± 0.20	89.76 ± 0.06	89.04 ± 0.03
					Test Accuracy (↑)	**Test ECE (↓)**	**AUROC SVHN (↑)**		**AUROC Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	77.91 ± 0.09	4.32 ± 0.10	81.32 ± 0.65		79.83 ± 0.07
	Energy-based [195]	-		Softmax Density			82.05 ± 0.69		79.61 ± 0.08
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	74.73 ± 0.22	7.68 ± 0.13	82.50 ± 2.09		77.05 ± 0.16
	DDU (ours)	SN	Softmax Entropy	GMM Density	**79.26 ± 0.16**	**4.07 ± 0.06**	**87.34 ± 0.64**		**82.11 ± 0.20**
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	**81.06 ± 0.07**	**3.54 ± 0.12**	83.42 ± 0.89		77.69 ± 0.12
				Mutual Information			84.24 ± 0.90		81.59 ± 0.05

Table B.4.1: *OoD detection performance of different baselines using a ResNet-50 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalisation, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

Train Dataset	Method	Penalty	Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC SVHN (↑)	AUROC CIFAR-100 (↑)	AUROC Tiny-ImageNet (↑)
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	**95.08 ± 0.04**	1.02 ± 0.04	93.12 ± 0.44	88.7 ± 0.1	88.07 ± 0.11
	Energy-based [195]	-		Softmax Density			93.67 ± 0.47	88.60 ± 0.11	88.13 ± 0.11
	DUQ [57]	JP	Kernel Distance	Kernel Distance	94.32 ± 0.17	1.21 ± 0.07	94.02 ± 0.45	86.17 ± 0.35	85.24 ± 0.21
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	94.85 ± 0.09	1.04 ± 0.02	93.17 ± 0.53	89.23 ± 0.10	88.80 ± 0.12
	DDU (ours)	SN	Softmax Entropy	GMM Density	94.82 ± 0.06	**1.01 ± 0.04**	**95.48 ± 0.30**	**90.08 ± 0.13**	**89.18 ± 0.15**
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	**96.18 ± 0.05**	1.57 ± 0.05	95.07 ± 0.45	**90.23 ± 0.04**	89 ± 0.03
				Mutual Information			94.72 ± 0.34	89.69 ± 0.05	88.35 ± 0.05
					Test Accuracy (↑)	**Test ECE (↓)**	**AUROC SVHN (↑)**		**AUROC Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	78.65 ± 0.10	3.93 ± 0.13	82.04 ± 0.57		80.13 ± 0.07
	Energy-based [195]	-		Softmax Density			82.78 ± 0.60		80.01 ± 0.09
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	76.16 ± 0.27	6.43 ± 0.75	83.94 ± 0.10		78.54 ± 0.28
	DDU (ours)	SN	Softmax Entropy	GMM Density	**78.89 ± 0.17**	**3.79 ± 0.07**	**88.66 ± 0.56**		**82.58 ± 0.24**
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	**81.80 ± 0.10**	**3.67 ± 0.11**	83.68 ± 0.33		81.12 ± 0.13
				Mutual Information			85.11 ± 0.57		81.94 ± 0.06

Table B.4.2: *OoD detection performance of different baselines using a ResNet-110 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalisation, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

get the results in Figure 4.1 and Table B.6.1 was trained on a single Nvidia GeForce RTX 2060 GPU. Each model (ResNet-50, Wide-ResNet-50-2, VGG-16) trained on ImageNet was trained using 8 Nvidia Quadro RTX 6000 GPUs.

B.5 Additional Results

In this section, we provide details of additional results on the OoD detection task using CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C and CIFAR-100 vs SVHN/Tiny-ImageNet for ResNet-50, ResNet-110 and DenseNet-121 architectures. We present results on ResNet-50, ResNet-110 and DenseNet-121 for CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet in Table B.4.1, Table B.4.2 and Ta-

Train Dataset	Method	Penalty	Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC SVHN (↑)	AUROC CIFAR-100 (↑)	AUROC Tiny-ImageNet (↑)
CIFAR-10	Softmax	-	Softmax Entropy	Softmax Entropy	95.16 ± 0.03	1.10 ± 0.04	94 ± 0.44	87.55 ± 0.11	86.99 ± 0.12
	Energy-based [195]	-		Softmax Density			94.07 ± 0.54	86.73 ± 0.15	86.43 ± 0.16
	DUQ [57]	JP	Kernel Distance	Kernel Distance	95.02 ± 0.14	1.08 ± 0.08	94.67 ± 0.41	87.38 ± 0.21	86.72 ± 0.14
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	94.31 ± 0.21	1.08 ± 0.10	94.48 ± 0.34	88.86 ± 0.46	88.40 ± 0.48
	DDU (ours)	SN	Softmax Entropy	GMM Density	$\mathbf{95.21 \pm 0.03}$	$\mathbf{1.05 \pm 0.03}$	$\mathbf{96.21 \pm 0.31}$	$\mathbf{90.84 \pm 0.06}$	$\mathbf{89.70 \pm 0.06}$
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	$\mathbf{96.18 \pm 0.05}$	1.07 ± 0.07	95.78 ± 0.11	90.65 ± 0.03	89.62 ± 0.06
				Mutual Information			95.75 ± 0.10	90.71 ± 0.04	89.34 ± 0.06
					Test Accuracy (↑)	**Test ECE (↓)**	**AUROC SVHN (↑)**		**AUROC Tiny-ImageNet (↑)**
CIFAR-100	Softmax	-	Softmax Entropy	Softmax Entropy	79.02 ± 0.08	4.11 ± 0.08	85.86 ± 0.42		81.10 ± 0.07
	Energy-based [195]	-		Softmax Density			87.09 ± 0.49		80.84 ± 0.08
	SNGP [58]	SN	Predictive Entropy	Predictive Entropy	79.15 ± 0.15	6.73 ± 0.10	85.00 ± 0.12		79.76 ± 0.15
	DDU (ours)	SN	Softmax Entropy	GMM Density	$\mathbf{79.15 \pm 0.07}$	$\mathbf{4.11 \pm 0.06}$	$\mathbf{88.44 \pm 0.55}$		$\mathbf{81.85 \pm 0.11}$
	5-Ensemble [49]	-	Predictive Entropy	Predictive Entropy	$\mathbf{81.01 \pm 0.13}$	4.81 ± 0.05	88.32 ± 0.61		81.45 ± 0.12
				Mutual Information			88.36 ± 0.17		81.73 ± 0.06

Table B.4.3: *OoD detection performance of different baselines using a DenseNet-121 architecture with the CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet and CIFAR-100 vs SVHN/Tiny-ImageNet dataset pairs averaged over 25 runs.* Note: SN stands for Spectral Normalisation, JP stands for Jacobian Penalty. We highlight the best deterministic and best method overall in bold for each metric.

Ablations					Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC SVHN (↑)	AUROC CIFAR-100 (↑)	AUROC Tiny-ImageNet (↑)
Architecture	Ensemble	Residual Connections	SN	GMM							
Wide-ResNet-28-10	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	$\mathbf{95.98 \pm 0.02}$	0.85 ± 0.02	94.44 ± 0.43 94.56 ± 0.51	89.39 ± 0.06 88.89 ± 0.07	88.42 ± 0.05 88.11 ± 0.06
				✓	Softmax Entropy	GMM Density	95.98 ± 0.02	0.85 ± 0.02	96.08 ± 0.25	90.94 ± 0.03	90.62 ± 0.05
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	95.97 ± 0.03	0.85 ± 0.04	94.05 ± 0.26 94.31 ± 0.33	90.02 ± 0.07 89.78 ± 0.08	89.07 ± 0.06 88.96 ± 0.07
				✓	**Softmax Entropy**	**GMM Density**	95.97 ± 0.03	$\mathbf{0.85 \pm 0.04}$	$\mathbf{97.86 \pm 0.19}$	$\mathbf{91.34 \pm 0.04}$	$\mathbf{91.07 \pm 0.05}$
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	$\mathbf{96.59 \pm 0.02}$	$\mathbf{0.76 \pm 0.03}$	97.73 ± 0.31 97.18 ± 0.19	$\mathbf{92.13 \pm 0.02}$ 91.33 ± 0.03	90.06 ± 0.03 90.90 ± 0.03
VGG-16	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	93.63 ± 0.04	1.64 ± 0.03	85.76 ± 0.84 84.24 ± 1.04	82.48 ± 0.14 81.91 ± 0.17	83.07 ± 0.12 82.82 ± 0.14
				✓	Softmax Entropy	GMM Density	93.63 ± 0.04	1.64 ± 0.03	89.25 ± 0.36	86.55 ± 0.10	86.78 ± 0.09
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	93.62 ± 0.04	1.78 ± 0.04	87.54 ± 0.41 86.28 ± 0.51	82.71 ± 0.09 82.15 ± 0.11	83.33 ± 0.08 83.07 ± 0.10
				✓	Softmax Entropy	GMM Density	93.62 ± 0.04	1.78 ± 0.04	89.62 ± 0.37	86.37 ± 0.14	86.63 ± 0.11
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	94.9 ± 0.05	2.03 ± 0.03	92.80 ± 0.18 91 ± 0.22	89.01 ± 0.08 88.43 ± 0.08	87.66 ± 0.08 88.74 ± 0.05

Table B.4.4: *OoD detection performance of different ablations trained on CIFAR-10 using Wide-ResNet-28-10 and VGG-16 architectures with SVHN, CIFAR-100 and Tiny-ImageNet as OoD datasets averaged over 25 runs.* Note: SN stands for Spectral Normalisation. We highlight the best deterministic and best method overall in bold for each metric.

ble B.4.3 respectively. We also present results on individual corruption types for CIFAR-10-C for Wide-ResNet-28-10, ResNet-50, ResNet-110 and DenseNet-121 in Figure B.4.2, Figure B.4.3, Figure B.4.4 and Figure B.4.5 respectively.

Finally, we provide results for various ablations on DDU. As mentioned in §4.3, DDU consists of a deterministic softmax model trained with appropriate inductive biases. It uses softmax entropy to quantify aleatoric uncertainty and feature-space density to quantify epistemic uncertainty. In the ablation, we try to experimentally evaluate the following scenarios:

1. **Effect of inductive biases (sensitivity + smoothness):** We want to see the effect of removing the proposed inductive biases (i.e. no sensitivity and smoothness constraints)

Ablations					Aleatoric Uncertainty	Epistemic Uncertainty	Test Accuracy (↑)	Test ECE (↓)	AUROC SVHN (↑)	AUROC Tiny-ImageNet (↑)
Architecture	**Ensemble**	**Residual Connections**	**SN**	**GMM**						
Wide-ResNet-28-10	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	80.26 ± 0.06	4.62 ± 0.06	77.42 ± 0.57 78.00 ± 0.63	81.53 ± 0.05 81.33 ± 0.06
				✓	Softmax Entropy	GMM Density	80.26 ± 0.06	4.62 ± 0.06	87.54 ± 0.61	78.13 ± 0.08
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	80.98 ± 0.06	4.10 ± 0.08	85.37 ± 0.36 86.41 ± 0.38	82.57 ± 0.03 82.49 ± 0.04
				✓	**Softmax Entropy**	**GMM Density**	$\mathbf{80.98 \pm 0.06}$	$\mathbf{4.10 \pm 0.08}$	$\mathbf{87.53 \pm 0.62}$	$\mathbf{83.13 \pm 0.06}$
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	$\mathbf{82.79 \pm 0.10}$	$\mathbf{3.32 \pm 0.09}$	79.54 ± 0.91 77.00 ± 1.54	82.95 ± 0.09 82.82 ± 0.04
VGG-16	✗	✓	✗	✗	Softmax Entropy	Softmax Entropy Softmax Density	73.48 ± 0.05	4.46 ± 0.05	76.73 ± 0.72 77.70 ± 0.86	76.43 ± 0.05 74.68 ± 0.07
				✓	Softmax Entropy	GMM Density	73.48 ± 0.05	4.46 ± 0.05	75.65 ± 0.95	74.32 ± 1.73
			✓	✗	Softmax Entropy	Softmax Entropy Softmax Density	73.58 ± 0.06	4.32 ± 0.06	77.21 ± 0.77 77.76 ± 0.90	76.59 ± 0.06 74.86 ± 0.08
				✓	Softmax Entropy	GMM Density	73.58 ± 0.06	4.32 ± 0.06	75.99 ± 1.23	74.06 ± 1.67
	✓	✓	✗	✗	Predictive Entropy	Predictive Entropy Mutual Information	77.84 ± 0.11	5.32 ± 0.10	79.62 ± 0.73 72.07 ± 0.48	78.66 ± 06 76.27 ± 0.05

Table B.4.5: *OoD detection performance of different ablations trained on CIFAR-100 using Wide-ResNet-28-10 and VGG-16 architectures with SVHN and Tiny-ImageNet as the OoD dataset averaged over 25 runs.* Note: SN stands for Spectral Normalisation. We highlight the best deterministic and best method overall in bold for each metric.

Model	WRN-28-10			WRN-28-10		WRN-50-2	RN-50
iD	CIFAR-100			CIFAR-100		ImageNet	
OoD	SVHN	CIFAR-100	TinyImageNet	SVHN	TinyImageNet	ImageNet-O	
LDA (Maha [179])	$\mathbf{98.41 \pm 0.09}$	82.90 ± 0.23	82.48 ± 0.25	$\mathbf{92.53 \pm 0.62}$	68.86 ± 0.13	64.19 ± 0.23	61.68 ± 0.14
GDA (DDU, ours)	97.86 ± 0.19	$\mathbf{91.34 \pm 0.04}$	$\mathbf{91.07 \pm 0.05}$	87.53 ± 0.62	$\mathbf{83.13 \pm 0.06}$	$\mathbf{73.12 \pm 0.19}$	$\mathbf{71.29 \pm 0.08}$

Table B.4.6: *LDA vs GDA ablation for OoD detection performance using Wide-ResNet-50-2, ResNet-50, Wide-ResNet-50-2 architectures (depending on dataset) on CIFAR-10 vs SVHN/CIFAR-100/TinyImageNet, CIFAR-100 vs SVHN/TinyImageNet, and ImageNet vs ImageNet-O [91].* Best AUROC (↑) scores are marked in bold. GDA performs better, except with SVHN as OoD dataset.

on the OoD detection performance of a model. To do this, we train a VGG-16 with and without spectral normalisation. Note that VGG-16 does not have residual connections and hence, a VGG-16 does not follow the sensitivity and smoothness (bi-Lipschitz) constraints.

2. **Effect of sensitivity alone:** Since residual connections make a model sensitive to changes in the input space by lower bounding its Lipschitz constant, we also want to see how a network performs with just the sensitivity constraint alone. To observe this, we train a Wide-ResNet-28-10 without spectral normalisation (i.e. no explicit upper bound on the Lipschitz constant of the model).

3. **Metrics for aleatoric and epistemic uncertainty:** With the above combinations, we try to observe how different metrics for aleatoric and epistemic uncertainty perform. To quantify aleatoric uncertainty, we use the softmax entropy of the model. On the other

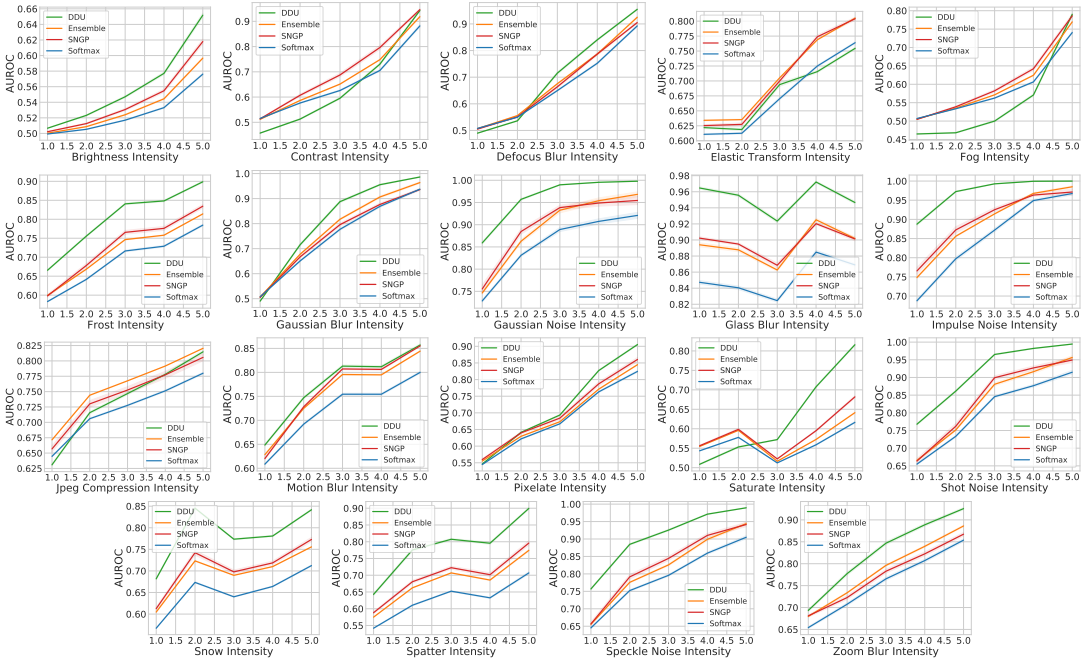


Figure B.4.2: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with Wide-ResNet-28-10 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

hand, to quantify the epistemic uncertainty, we use **i)** the softmax entropy, **ii)** the softmax density [195] or **iii)** the GMM feature density (as described in §4.3).

For the purposes of comparison, we also present scores obtained by a 5-Ensemble of the respective architectures (i.e. Wide-ResNet-28-10 and VGG-16) in Table B.4.4 for CIFAR-10 vs SVHN/CIFAR-100 and in Table B.4.5 for CIFAR-100 vs SVHN. Based on these results, we can make the following observations (in addition to the ones we make in §4.4.2):

Inductive biases are important for feature density. From the AUROC scores in Table B.4.4, we can see that using the feature density of a GMM in VGG-16 without the proposed inductive biases yields significantly lower AUROC scores as compared to Wide-ResNet-28-10 with inductive biases. In fact, in none of the datasets is the feature density of a VGG able to outperform its corresponding ensemble. This provides yet more evidence (in addition to Figure 4.1) to show that the GMM feature density alone cannot estimate epistemic uncertainty in a model that suffers from feature collapse. We need sensitivity

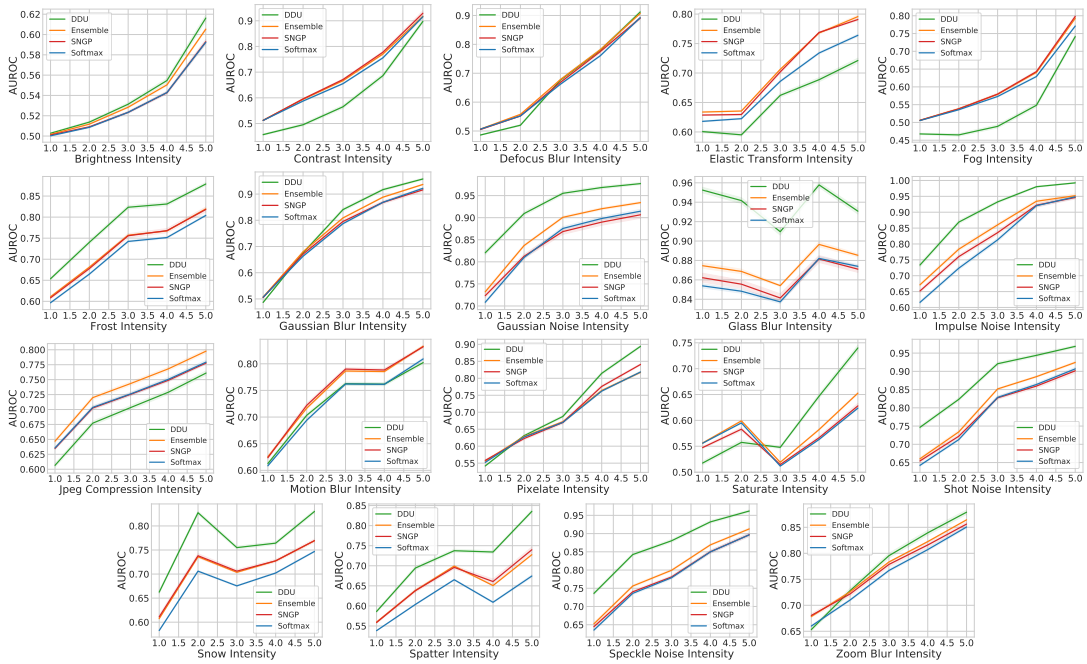


Figure B.4.3: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-50 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

and smoothness conditions (see §4.5) on the feature space of the model to obtain feature densities that capture epistemic uncertainty.

Sensitivity creates a bigger difference than smoothness. We note that the difference between AUROC obtained from feature density between Wide-ResNet-28-10 models with and without spectral normalisation is minimal. Although Wide-ResNet-28-10 with spectral normalisation (i.e. smoothness constraints) still outperforms its counterpart without spectral normalisation, the small difference between the AUROC scores indicates that it might be the residual connections (i.e. sensitivity constraints) that make the model detect OoD samples better. This observation is also intuitive as a sensitive feature extractor should map OoD samples farther from iD ones.

DDU as a simple baseline. In DDU, we use the softmax output of a model to get aleatoric uncertainty. We use the GMM's feature-density to estimate the epistemic uncertainty. Hence, DDU does not suffer from miscalibration as the softmax outputs can be calibrated using

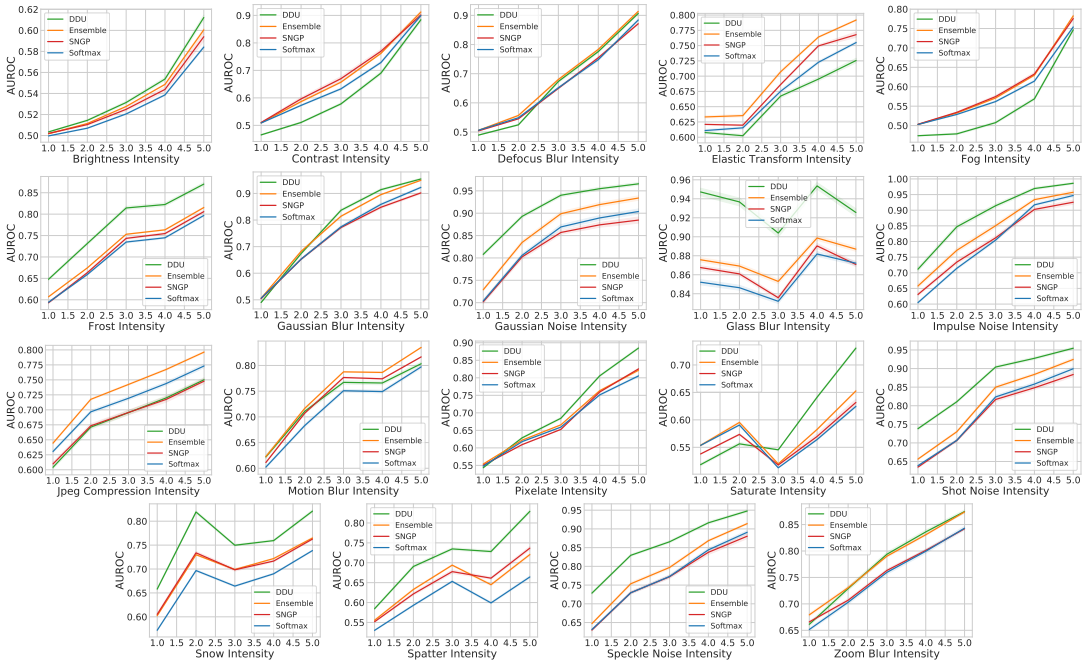


Figure B.4.4: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with ResNet-110 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

Model	ECE	AUROC for Softmax Entropy (↑)	AUROC for Feature Density (↑)
LeNet	2.22	84.23	71.41
VGG-16	**2.11**	84.04	89.01
ResNet-18+SN (DDU)	2.34	83.01	**99.91**

Table B.6.1: *ECE for Dirty-MNIST test set and AUROC for Dirty-MNIST vs Fashion-MNIST as proxies for aleatoric and epistemic uncertainty quality respectively.*

post-hoc methods like temperature scaling. At the same time, the feature-densities of the model are not affected by temperature scaling and capture epistemic uncertainty well.

B.6 Ablations and Toy Experiments

Here, we provide details for toy experiments from the main paper which are visualized in Figure 4.1, Figure B.6.2 and Figure 4.2.

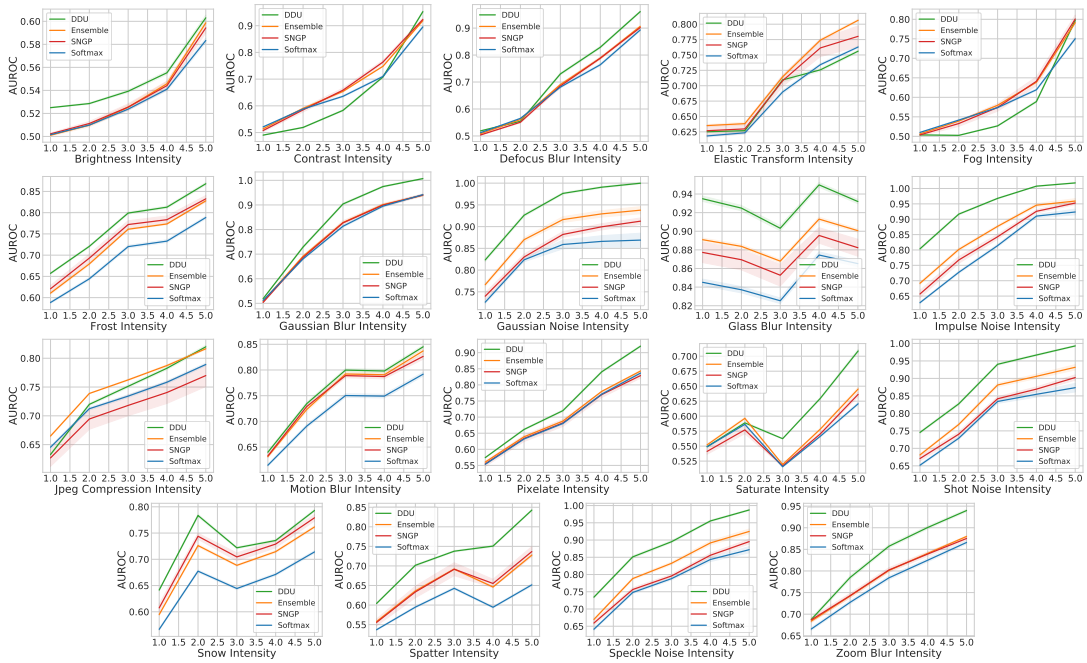


Figure B.4.5: AUROC vs corruption intensity for all corruption types in CIFAR-10-C with DenseNet-121 as the architecture and baselines: Softmax Entropy, Ensemble (using Predictive Entropy as uncertainty), SNGP and DDU feature density.

B.6.1 Motivational Example in Figure 4.1

In Figure 4.1 we train a LeNet [14], a VGG-16 [61] and a ResNet-18 with spectral normalisation [15, 181] (ResNet-18+SN) on *Dirty-MNIST*, a modified version of MNIST [14] with additional ambiguous digits (Ambiguous-MNIST). *Ambiguous-MNIST* contains samples with multiple plausible labels and thus higher aleatoric uncertainty (see Figure 4.1a). We refer to §B.2 for details on how this dataset was generated. With ambiguous data having various levels of aleatoric uncertainty, Dirty-MNIST is more representative of real-world datasets compared to well-cleaned curated datasets, like MNIST and CIFAR-10, commonly used for benchmarking in ML [44, 89]. Moreover, Dirty-MNIST also poses a challenge for recent uncertainty estimation methods, which often confound aleatoric and epistemic uncertainty [57]. Figure 4.1b shows that the softmax entropy of a deterministic model is unable to distinguish between iD (Dirty-MNIST) and OoD (Fashion-MNIST [216]) samples as the entropy for the latter heavily overlaps with the entropy for Ambiguous-MNIST samples.

However, the feature-space density of the model with our inductive biases in Figure 4.1c captures epistemic uncertainty reliably and is able to distinguish iD from OoD samples. The same cannot be said for LeNet or VGG in Figure 4.1c, whose densities are unable to separate OoD from iD samples. This demonstrates the importance of the inductive bias to ensure the sensitivity and smoothness of the feature space as we further argue below. Finally, Figure 4.1b and Figure 4.1c demonstrate that our method is able to separate aleatoric from epistemic uncertainty: samples with low feature density have high epistemic uncertainty, whereas those with both high feature density and high softmax entropy have high aleatoric uncertainty—note the high softmax entropy for the most ambiguous Ambiguous-MNIST samples in Figure 4.1b.

Disentangling Epistemic and Aleatoric Uncertainty

We used a simple example in §4.1 to demonstrate that a single softmax model with a proper inductive bias can reliably capture epistemic uncertainty via its feature-space density and aleatoric uncertainty via its softmax entropy. To recreate the natural characteristics of uncurated real-world datasets, which contain ambiguous samples, we use MNIST [14] as an iD dataset of unambiguous samples, Ambiguous-MNIST as an iD dataset of ambiguous samples and Fashion-MNIST [216] as an OoD dataset (see Figure 4.1a), with more details in §B.2. We train a LeNet [14], a VGG-16 [61] and a ResNet-18 [15] with spectral normalisation (SN) on Dirty-MNIST (a mix of Ambiguous- and standard MNIST) with the training setup detailed in §B.4.1.

Table B.6.1 gives a quantitative evaluation of the qualitative results in §4.1. The AUROC metric reflects the quality of the epistemic uncertainty as it measures the probability that iD and OoD samples can be distinguished, and OoD samples are never seen during training while iD samples are semantically similar to training samples. The ECE metric measures the quality of the aleatoric uncertainty. The softmax outputs capture aleatoric uncertainty well, as expected, and all 3 models obtain similar ECE scores on the Dirty-MNIST test set. However, with an AUROC of around 84% for all the 3 models, on Dirty-MNIST vs Fashion-MNIST, we conclude that softmax entropy is unable to capture epistemic uncertainty

well. This is reinforced in Figure 4.1b, which shows a strong overlap between the softmax entropy of OoD and ambiguous iD samples. At the same time, the feature-space densities of LeNet and VGG-16, with AUROC scores around 71% and 89% respectively, are unable to distinguish OoD from iD samples, indicating that simply using feature-space density without appropriate inductive biases (as seen in [179]) is not sufficient.

Only by fitting a GMM on top of a feature extractor with appropriate inductive biases (DDU) and using its feature density are we able to obtain performance far better (with AUROC of 99.9%*) than the alternatives in the ablation study (see Table B.6.1, also noticeable in Figure 4.1c).* The entropy of a softmax model can capture aleatoric uncertainty, even without additional inductive biases, but it *cannot* be used to estimate epistemic uncertainty (see §4.5). On the other hand, feature-space density can *only* be used to estimate epistemic uncertainty *when the feature extractor is sensitive and smooth*, as achieved by using a ResNet and spectral normalisation in DDU.

B.6.2 Effects of a well-regularized feature space on feature collapse

The effects of a well-regularized feature space on feature collapse are visible in Figure 4.1. In the case of feature collapse, we must have *some* OoD inputs for which the features are mapped on top of the features of iD inputs. The distances of these OoD features to each class centroid must be equal to the distances of the corresponding iD inputs to class centroids, and hence the density for these OoD inputs must be equal to the density of the iD inputs. If the density histograms do not overlap, no feature collapse can be present[1]. We see no overlapping densities in Figure 4.1(c, right), therefore we indeed have no feature collapse. First, the effects of having a well-regularized feature space on feature collapse can be analysed from Figure 4.1. In case of feature collapse we must have *some* OoD features mapped on top of iD features, therefore the distances of at least some OoD features to the class centroids must be equal to iD's distances, hence OoD density must overlap with iD density. We see this

[1]Note though that the opposite ('if the density histograms overlap then there must be feature collapse') needs not hold: the histograms can also overlap due to other reasons.

Softmax	Energy	3-Ensemble PE	DDU
78.4 ± 1.31	77.31 ± 1.5	82.25 ± 0.83	$\mathbf{82.63 \pm 1.08}$

Table B.6.2: Dice scores for the QUBIQ 2021 challenge

in Figure 4.1(c) (left) in the case without a regularized feature space. On the other hand, when we regularize the feature space, we see the densities do not overlap, i.e. the distances of the features of OoD examples to the centroids are larger than the distances of iD examples (Figure 4.1(c) right), hence feature collapse is not present.

B.6.3 QUBIQ Challenge

In this section, we evaluate DDU's performance on the real-world QUBIQ challenge related to biomedical imaging. QUBIQ has a total of 7 binary segmentation tasks in 4 biomedical imaging datasets with multiple annotations per image. The task is to predict the distribution of source labels with a mask of values between 0 and 1. For evaluation, the annotations are averaged to provide a continuous ground-truth. The prediction mask and continuous ground-truth are binarized by thresholding between $[0, 1]$ and a Dice score is computed between the resulting binary masks. The average dice score across thresholds, images and tasks is reported. Note that in the continuous ground-truth, 0.5 indicates maximum uncertainty and values above or below indicate lower uncertainty. Thus, for our comparison, we scale all uncertainty values to the range $u \in [0, 0.5]$ and use $p + u$ if $p = 0$ and $p - u$ if $p = 1$, where p is the binary prediction. We use a UNet model with a ResNet encoder and report the Dice scores averaged over 5 runs in Table B.6.2. Even on this real-world dataset, DDU performs as well as ensembles and outperforms Softmax and Energy baselines.

B.6.4 Additional Baselines

In this section, we provide an ablation with additional baselines on OoD detection for comparison with DDU. In particular, we provide comparisons with Feature Space Singularity (FSSD) [227], Batch Ensemble (BE) [113] and SWAG [325] using Wide-ResNet-28-10 as additional recent baselines. We also use the Wide-ResNet-28-10 feature extractor trained

Train dataset	Method	AUROC (↑) *SVHN*	*CIFAR-100/10*	*Tiny-ImageNet*
C10	FSSD [227]	97.24	89.88	90.23
	BE [113]	95.36	87.63	88.14
	SWAG [325]	96.37	90.33	90.24
	SNGP [58]	94.0 ± 1.3	91.13 ± 0.15	89.97 ± 0.19
	5-Ensemble [49]	97.73 ± 0.31	$\mathbf{92.13 \pm 0.02}$	90.06 ± 0.03
	SNGP + DDU	96.47 ± 0.7	89.97 ± 0.13	90.3 ± 0.12
	DDU (Ours)	$\mathbf{97.86 \pm 0.19}$	91.34 ± 0.04	$\mathbf{91.07 \pm 0.05}$
		SVHN		*Tiny-ImageNet*
C100	FSSD [227]	**87.64**		82.2
	BE [113]	86.44		78.33
	SWAG [325]	81.41		81.67
	SNGP [58]	85.71 ± 0.81		78.85 ± 0.43
	5-Ensemble [49]	79.54 ± 0.91		82.95 ± 0.09
	SNGP + DDU	87.34 ± 0.76		79.62 ± 0.36
	DDU(Ours)	87.53 ± 0.62		$\mathbf{83.13 \pm 0.06}$

Table B.6.3: OoD detection ablation with WRN-28-10 model with additional baselines, FSSD [227], Batch Ensemble (BE) [113] and SWAG [325] as well as using DDU with a feature extractor trained on SNGP. For comparison, we also provide performance for vanilla SNGP, deep ensemble and DDU.

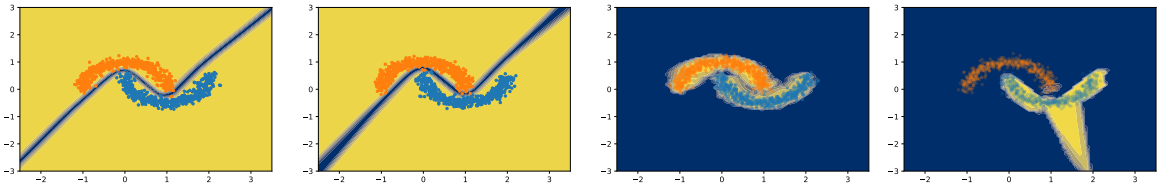
using SNGP loss and fit DDU (i.e., GDA) on its feature space. Since SNGP also uses a sensitive smooth feature space with residual connections and spectral normalization, its feature space makes for a good candidate to apply DDU. In Table B.6.3, we provide the AUROC scores for models trained on CIFAR-10 (C10) and CIFAR-100 (C100). Broadly, DDU outperforms all competitive baselines. Additionally, we observe a broad improvement in AUROC when DDU is applied on the SNGP feature extractor as compared to vanilla SNGP. However, DDU on a feature extractor trained using softmax loss is still superior to DDU on the SNGP feature extractor.

B.6.5 Calibration Metrics

In Table B.6.4, in addition to ECE%, we provide additional calibration error scores: temperature scaled Thresholded Adaptive Calibration Error (TACE) % and Negative Log Likelihood (NLL) for Wide-ResNet-28-10 trained on CIFAR-10/100 (main results for this can be found in Table 4.1. The results for TACE and NLL are consistent with what we see for ECE. Ensembles produce the most calibrated models and among deterministic baselines, DDU is the best calibrated.

Dataset	Metric	Softmax & Energy	DUQ	SNGP	DDU	5-Ensemble
C10	ECE%	$\mathbf{0.85 \pm 0.02}$	1.55 ± 0.08	1.8 ± 0.1	$\mathbf{0.85 \pm 0.04}$	$\mathbf{0.76 \pm 0.03}$
	TACE%	0.63 ± 0.01	0.84 ± 0.03	0.9 ± 0.04	$\mathbf{0.61 \pm 0.01}$	$\mathbf{0.48 \pm 0.01}$
	NLL	0.18 ± 0.06	0.23 ± 0.07	0.27 ± 0.08	$\mathbf{0.16 \pm 0.06}$	$\mathbf{0.11 \pm 0.02}$
C100	ECE%	4.62 ± 0.06	-	4.33 ± 0.01	$\mathbf{4.1 \pm 0.08}$	$\mathbf{3.32 \pm 0.09}$
	TACE%	1.31 ± 0.02	-	1.23 ± 0.04	$\mathbf{1.06 \pm 0.03}$	$\mathbf{0.58 \pm 0.03}$
	NLL	1.17 ± 0.13	-	0.92 ± 0.16	$\mathbf{0.86 \pm 0.14}$	$\mathbf{0.73 \pm 0.09}$

Table B.6.4: Calibration error scores ECE%, TACE% and NLL for WRN-28-10.



(a) Softmax Entropy **(b)** Ensemble PE **(c)** DDU Density **(d)** FC-Net Density

Figure B.6.1: *Uncertainty on Two Moons dataset.* Blue indicates high uncertainty and yellow indicates low uncertainty. Both the softmax entropy of a single model as well as the predictive entropy (PE) of a deep ensemble are uncertain only along the decision boundary whereas the feature-space density of DDU is uncertain everywhere except on the data distribution (the ideal behaviour). However, the feature density of a normal fully connected network (FC-Net) without any inductive biases can't capture uncertainty properly.

B.6.6 Two Moons

In this section, we evaluate DDU's performance on a well-known toy setup: the Two Moons dataset. We use scikit-learn's *datasets* package to generate 2000 samples with a noise rate of 0.1. We use a 4-layer fully connected architecture, ResFFN-4-128 with 128 neurons in each layer and a residual connection, following [58]. As an ablation, we also train using a 4-layer fully connected architecture with 128 neurons in each layer, but *without the residual connection*. We name this architecture FC-Net. The input is 2-dimensional and is projected into the 128 dimensional space using a fully connected layer. Using the ResFFN-4-128 architecture we train 3 baselines:

1. **Softmax:** We train a single softmax model and use the softmax entropy as the uncertainty metric.

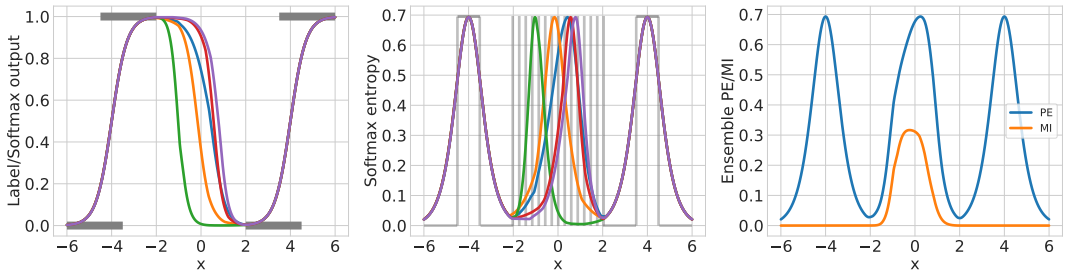
2. **3-ensemble:** We train an ensemble of 3 softmax models and use the predictive entropy of the ensemble as the measure of uncertainty.

3. **DDU:** We train a single softmax model applying spectral normalization on the fully connected layers and using the feature density as the measure of model confidence.

Each model is trained using the Adam optimiser for 150 epochs. In Figure B.6.1, we show the uncertainty results for all the above 3 baselines. It is clear that both the softmax entropy as well as the predictive entropy of the ensemble is uncertain only along the decision boundary between the two classes whereas DDU is confident only on the data distribution and is not confident anywhere else. It is worth mentioning that even DUQ and SNGP perform well in this setup and deep ensembles have been known to underperform in the Two-Moons setup primarily due to the simplicity of the dataset causing all the ensemble components to generalise in the same way. Finally, also note that the feature space density of FC-Net without residual connections is not able to capture uncertainty well (see Figure B.6.1d), thereby reaffirming our claim that proper inductive biases are indeed a necessary component to ensure that feature space density captures uncertainty reliably. Thus, in addition to its excellent performance in active learning, CIFAR-10 vs SVHN/CIFAR-100/Tiny-ImageNet/CIFAR-10-C, CIFAR-100 vs SVHN/Tiny-ImageNet, and ImageNet vs ImageNet-O, we note that DDU captures uncertainty reliably even in a small 2D setup like Two Moons.

B.6.7 5-Ensemble Visualisation

In Figure B.6.2, we provide a visualisation of a 5-ensemble (with five deterministic softmax networks) to see how softmax entropy fails to capture epistemic uncertainty precisely because the mutual information (MI) of an ensemble does not (see §4.5). We train the networks on 1-dimensional data with binary labels 0 and 1. The data is shown in Figure B.6.2a. From Figure B.6.2a and Figure B.6.2b, we find that the softmax entropy is high in regions of ambiguity where the label can be both 0 and 1 (i.e. x between -4.5 and -3.5, and between 3.5 and 4.5). This indicates that softmax entropy can capture aleatoric uncertainty. Furthermore, in the x interval $(-2, 2)$, we find that the deterministic softmax networks disagree in their predictions (see Figure B.6.2a) and have softmax entropies which can be



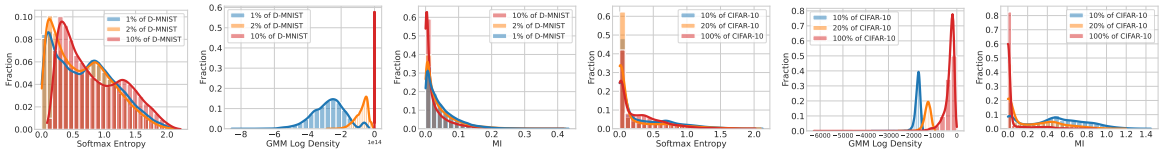
(a) Softmax output **(b)** Softmax entropy **(c)** 5-Ensemble

Figure B.6.2: *Softmax outputs & entropies for 5 softmax models along with the predictive entropy (PE) and mutual information (MI) for the resulting 5-Ensemble.* a and b show that the softmax entropy is only reliably high for ambiguous iD points (±3.5–4.5), whereas it can be low or high for OoD points (-2–2). The different colors are the different ensemble components. Similarly, c shows that the MI of the ensemble is only high for OoD, whereas the PE is high for both OoD and for regions of ambiguity. See §B.6.7.

high, low or anywhere in between (see Figure B.6.2b) following our claim in §4.5. In fact, this disagreement is the very reason why the MI of the ensemble is high in the interval $(-2, 2)$, thereby reliably capturing epistemic uncertainty. Finally, note that the predictive entropy (PE) of the ensemble is high both in the OoD interval $(-2, 2)$ as well as at points of ambiguity (i.e. at -4 and 4). This indicates that the PE of a Deep Ensemble captures both epistemic and aleatoric uncertainty well. From these visualisations, we draw the conclusion that the softmax entropy of a deterministic softmax model cannot capture epistemic uncertainty precisely because the MI of a Deep Ensemble can.

B.6.8 Feature-Space Density & Epistemic Uncertainty vs Softmax Entropy & Aleatoric Uncertainty

To empirically verify the connection between feature-space density and epistemic uncertainty on the one hand and the connection between softmax entropy and aleatoric uncertainty on the other hand, we train ResNet-18+SN models on increasingly large subsets of Dirty-MNIST and CIFAR-10 and evaluate the epistemic and aleatoric uncertainty on the corresponding test sets using the feature-space density and softmax entropy, respectively. Moreover, we also train a 5-ensemble on the same subsets of data and use the ensemble's mutual information



(a) SE DMN **(b)** Density DMN **(c)** MI DMN **(d)** SE C10 **(e)** Density C10 **(f)** MI C10

Figure B.6.3: *Comparison of epistemic and aleatoric uncertainty captured b(ResNet-18+SN) on increasingly large subsets of Dirty-MNIST (DMN) and CIFAR-10 (C10). Clearly, feature density captures epistemic uncertainty which reduces when the model is trained on increasingly large subsets of training data, whereas softmax entropy (SE) does not. For comparison, we also plot a deep-ensemble's epistemic uncertainty, through mutual information (MI) for the same settings. For more details, see Table B.6.5.*

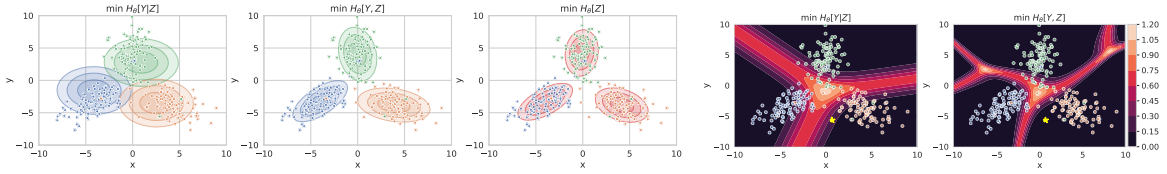
Training Set	Avg Test SE (≈)	Avg Test Log GMM Density (↑)	Avg Test MI	Correlation(SE ‖ MI)	Correlation(-Log GMM Density ‖ MI)
1% of D-MNIST	0.7407	$-2.7268e+14$	0.0476		
2% of D-MNIST	0.6580	$-7.8633e+13$	0.0447	-0.79897	0.8132
10% of D-MNIST	0.8295	-1279.1753	0.0286		
10% of CIFAR-10	0.3189	-1715.3516	0.4573		
20% of CIFAR-10	0.2305	-1290.1726	0.2247	0.5663	0.9556
100% of CIFAR-10	0.2747	-324.8040	0.0479		

Table B.6.5: *Average softmax entropy (SE) and feature-space density of the test set for models trained on different amounts of the training set (Dirty-MNIST and CIFAR-10) behave consistently with aleatoric and epistemic uncertainty.* Aleatoric uncertainty for individual samples does not change much as more data is added to the training set while epistemic uncertainty decreases as more data is added. This is also consistent with Table 3 in [78]. Finally, we observe a consistent strong positive correlation between the negative log feature space density and the mutual information (MI) of a deep ensemble trained on the same subsets of data for both Dirty-MNIST and CIFAR-10. However, the correlation between softmax entropy and MI is not consistent.

as a baseline measure of epistemic uncertainty.

In Figure 4.2, Figure B.6.3 and Table B.6.5, we see that with larger training sets, the average feature-space density increases which is consistent with the epistemic uncertainty decreasing as more data is available as reducible uncertainty. This is also evident from the consistent strong positive correlation between the negative log density and mutual information of the ensemble.

On the other hand, the softmax entropy stays roughly the same which is consistent with aleatoric uncertainty as irreducible uncertainty, which is independent of the training data. Importantly, all of this is also consistent with the experiments comparing epistemic and aleatoric uncertainty on increasing training set sizes in Table 3 of [78].



(a) *Density.* Contours at 68.26%, 95.44%, and 99.7%. **(b)** *Entropy.* Darker is lower.

Figure B.6.4: *3-component GMM fitted to a synthetic dataset with 3 different classes (differently colored) with 4% label noise using different objectives.* **a:** The optimas for conditional log-likelihood $\mathbb{H}_\theta[Y \mid Z]$, joint log-likelihood $\mathbb{H}_\theta[Y, Z]$, and marginalised log-likehood $\mathbb{H}_\theta[Z]$ all differ. Hence, the best calibrated model ($\mathbb{H}_\theta[Y \mid Z]$) will not provide the best density estimate ($\mathbb{H}_\theta[Z]$), and vice-versa. **b:** A mixture model that optimizes $\mathbb{H}_\theta[Y, Z]$ (GDA) does not have calibrated decision boundaries for aleatoric uncertainty: the ambiguous sample (due to label noise) marked by the yellow star has no aleatoric uncertainty under the GDA model.

Model	Prediction Source	Accuracy in % (↑)	ECE (↓)
WideResNet-28-10	Softmax	95.98 ± 0.02	2.29 ± 0.02
	GMM	95.86 ± 0.02	4.13 ± 0.02
WideResNet-28-10+SN	Softmax	95.97 ± 0.03	2.23 ± 0.03
	GMM	95.88 ± 0.02	4.12 ± 0.02

Table B.6.6: *Objective Mismatch Ablation with WideResNet-28-10 models with and without spectral normalisation on CIFAR-10.* While GMMs perform much better than Softmax Entropy for feature-space density/epistemic uncertainty estimation, they underperform for aleatoric uncertainty estimation: both accuracy and in particular ECE are much worse than a regular softmax layer.

B.6.9 Objective Mismatch Ablation

We further validate Observation 4.5.3 by running an ablation on Wide-ResNet-28-10 on CIFAR-10. Table B.6.6 shows that the feature-space density estimator indeed performs worse than the softmax layer for aleatoric uncertainty (accuracy and ECE).

B.7 Theoretical Results

B.7.1 Softmax entropy "cannot" capture epistemic uncertainty because Deep Ensembles "can"

Qualitative Statement

We start with a proof of Observation 4.5.2, which quantitatively examines the qualitative statements that given the same predictive entropy, higher epistemic uncertainty for one point

than another will cause some ensemble members to have lower softmax entropy.

*

Proof. From Equation (4.1), we obtain

$$\begin{aligned} &| \mathbb{I}[Y_1; \omega \mid x_1, \mathcal{D}] + \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_1 \mid x_1, \omega] \right] \\ &- \mathbb{I}[Y_2; \omega \mid x_2, \mathcal{D}] - \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_2 \mid x_2, \omega] \right] | \leq \epsilon. \end{aligned} \tag{B.1}$$

and hence we have

$$\begin{aligned} &\mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_1 \mid x_1, \omega] \right] - \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_2 \mid x_2, \omega] \right] \\ &+ \underbrace{(\mathbb{I}[Y_1; \omega \mid x_1, \mathcal{D}] - \mathbb{I}[Y_2; \omega \mid x_2, \mathcal{D}])}_{>\delta} \leq \epsilon. \end{aligned} \tag{B.2}$$

We rearrange the terms:

$$\mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_1 \mid x_1, \omega] \right] < \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_2 \mid x_2, \omega] \right] - (\delta - \epsilon). \tag{B.3}$$

Now, the statement follows by contraposition: if $\mathbb{H}[Y_1 \mid x_1, \omega] \geq \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_2 \mid x_2, \omega] \right] - (\delta - \epsilon)$ for all ω, the monotonicity of the expectation would yield $\mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_1 \mid x_1, \omega] \right] \geq \mathbb{E}_{\mathrm{p}(\omega|\mathcal{D})} \left[\mathbb{H}[Y_2 \mid x_2, \omega] \right] - (\delta - \epsilon)$. Thus, there is a non-null-set Ω' with $\mathrm{p}(\Omega') > 0$, such that

$$\mathbb{H}[Y_1 \mid x_1, \omega] < \mathbb{H}[Y_2 \mid x_2, \omega] - (\delta - \epsilon), \tag{B.4}$$

for all $\omega \in \Omega'$. □

B.7.2 Objective Mismatch

In §4.5, we noted that the objectives that lead to optimal estimators for aleatoric and epistemic uncertainty via softmax entropy and feature-space density do not match, and DDU therefore uses the softmax layer as a discriminative classifier (implicit LDA) to estimate the predictive entropy, while it is using a GMM as generative classifier to estimate the feature-space density. Here we prove this.

Preliminaries

Before we prove Observation 4.5.3, we will introduce some additional notation following [326].

Definition B.7.1. 1. $\hat{\mathrm{p}}(y, z)$ is the data distribution of the $\mathcal{D}$ in feature space with class labels y and feature representation z.

2. $\mathrm{p}_\theta(\cdot)$ is a probability distribution parameterized by θ.

3. Entropies and conditional entropies are over the empirical data distribution $\hat{\mathrm{p}}(\cdot)$:

$$\mathbb{H}[\cdot] = \mathbb{H}\left(\hat{\mathrm{p}}(\cdot)\right) = \mathbb{E}_{\hat{\mathrm{p}}(\cdot)}\left[-\log \hat{\mathrm{p}}(\cdot)\right]. \tag{B.5}$$

4. $\mathbb{H}[Y \mid z]$ is the entropy of $\hat{\mathrm{p}}(y \mid z)$ for a given z, whereas $\mathbb{H}[Y \mid Z]$ is the conditional entropy:

$$\mathbb{H}[Y \mid Z] = \mathbb{E}_{\hat{\mathrm{p}}(z)}\, \mathbb{H}[Y \mid z]. \tag{B.6}$$

5. $\mathbb{H}(\mathrm{p}(y, z) \mid\mid \mathrm{q}(y|z|))$ is the cross-entropy of $\mathrm{q}(y \mid z)$ under $\mathrm{p}(y \mid z)$ in expectation over $\mathrm{p}(z)$:

$$\begin{aligned}\mathbb{H}(\mathrm{p}(y, z) \mid\mid \mathrm{q}(y \mid z)) &= \mathbb{E}_{\mathrm{p}(z)}\, \mathbb{H}(\mathrm{p}(y \mid z) \mid\mid \mathrm{q}(y \mid z)) \\ &= \mathbb{E}_{\mathrm{p}(y,z)}\left[-\log \mathrm{q}(y \mid z)\right].\end{aligned}$$

6. Similarly, $D_{\mathrm{KL}}(\mathrm{p}(y, z) \mid\mid \mathrm{q}(y|z|))$ is the Kullback-Leibler divergence of $\mathrm{q}(y \mid z)$ under $\mathrm{p}(y \mid z)$ in expectation over $\mathrm{p}(z)$:

$$\begin{aligned}D_{\mathrm{KL}}(\mathrm{p}(y, z) \mid\mid \mathrm{q}(y|z|)) &= \mathbb{E}_{\mathrm{p}(z)}\, D_{\mathrm{KL}}(\mathrm{p}(y \mid z) \mid\mid \mathrm{q}(y|z|)) \\ &= \mathbb{H}(\mathrm{p}(y, z) \mid\mid \mathrm{q}(y|z|)) - \mathbb{H}[Y \mid Z]\end{aligned}$$

7. For cross-entropies of $\mathrm{p}_\theta(\cdot)$ under $\hat{\mathrm{p}}(z, y)$, we use the convenient short-hand $\mathbb{H}_\theta[\cdot] = \mathbb{H}(\hat{\mathrm{p}}(z, y) \mid\mid \mathrm{p}_\theta(\cdot))$.

Then we can observe the following connection between $\mathbb{H}_\theta[\cdot]$ and $\mathbb{H}[\cdot]$:

Lemma B.7.2. *Cross-entropies upper-bound the respective entropy with equality when* $\mathrm{p}_\theta(\cdot) = \hat{\mathrm{p}}(\cdot)$*, which is important for variational arguments:*

1. $\mathbb{H}_\theta[Y, Z] \geq \mathbb{H}[Y, Z]$,
2. $\mathbb{H}_\theta[Z] \geq \mathbb{H}[Z]$, *and*
3. $\mathbb{H}_\theta[Y \mid Z] \geq \mathbb{H}[Y \mid Z]$.

Proof. 1. $\mathbb{H}_\theta[Y, Z] - \mathbb{H}[Y, Z] = D_{\mathrm{KL}}(\hat{\mathrm{p}}(y, z) \,||\, \mathrm{p}_\theta(y, z)) \geq 0$.

2. follows from Item 1.

3. We expand the expectations and note that inequality commutes with expectations:

$$\mathbb{H}_\theta[Y \mid Z] - \mathbb{H}[Y \mid Z] = \mathbb{E}_{\hat{\mathrm{p}}(z)}\left[\mathbb{H}_\theta[Y \mid z] - \mathbb{H}[Y \mid z]\right] \geq 0,$$

because $\mathbb{H}_\theta[Y \,|\, z] - \mathbb{H}[Y \,|\, z] \geq 0$ for all z. The equality conditions follows from the properties of the Kullback-Leibler divergence as well.

□

We also have:

Lemma B.7.3.

$$\mathbb{H}_\theta[Y, Z] = \mathbb{H}_\theta[Y \mid Z] + \mathbb{H}_\theta[Z] \tag{B.7}$$

$$= \mathbb{H}_\theta[Z \mid Y] + \mathbb{H}_\theta[Y]. \tag{B.8}$$

Proof. We substitute the definitions and obtain:

$$\mathbb{H}_\theta[Y, Z] = \mathbb{E}_{\mathrm{p}(y,z)}\left[-\log \mathrm{q}(y, z)\right] \tag{B.9}$$

$$= \mathbb{E}_{\mathrm{p}(y,z)}\left[-\log \mathrm{q}(y \mid z)\right] + \mathbb{E}_{\mathrm{p}(y,z)}\left[-\log \mathrm{q}(z)\right] \tag{B.10}$$

$$= \mathbb{H}_\theta[Y \mid Z] + \mathbb{H}_\theta[Z]. \tag{B.11}$$

□

The same holds for entropies: $\mathbb{H}[Y, Z] = \mathbb{H}[Y \mid Z] + \mathbb{H}[Z] = \mathbb{H}[Y \mid Z] + \mathbb{H}[Y]$ [327].

Proof

We can now prove the observation. *

Proof. 1. The conditional log-likelihood is a strictly proper scoring rule [182]. The optimization objective can be rewritten as

$$\max_{\theta} \mathbb{E}_{\log \mathrm{p}_\theta(y|z)} = \min_{\theta} \mathbb{H}_\theta[Y \mid Z] \geq \mathbb{H}[Y \mid Z]. \tag{B.12}$$

An optimal discriminative classifier $\mathrm{p}_\theta(y \mid z)$ would thus capture the true (empirical) distribution everywhere: $\mathrm{p}_\theta(y \mid z) = \hat{\mathrm{p}}(y \mid z)$. This means the negative conditional log-likelihood will be equal $\mathbb{H}[Y \mid Z]$ and $\mathbb{H}_\theta[Y \mid z] = \mathbb{H}[Y \mid z]$ for all z.

2. For density estimation $\mathrm{q}(z)$, the maximum likelihood $\mathbb{E}\left[\log \mathrm{q}(z)\right]$ using the empirical data distribution is maximized. We can rewrite this as

$$\max_{\theta} \mathbb{E}_{\hat{\mathrm{p}}(y,z)} \log \mathrm{p}_\theta(z) = \min_{\theta} \mathbb{H}_\theta[Z] \geq \mathbb{H}[Z]. \tag{B.13}$$

We see that the negative marginalized likelihood of the density estimator upper-bounds the entropy of the feature representations $\mathbb{H}[Z]$. We have equality and $\mathrm{p}_\theta(z) = \hat{\mathrm{p}}(z)$ in the optimum case.

3. Using $\mathbb{H}_\theta[Y, Z] = \mathbb{H}_\theta[Y \mid Z] + \mathbb{H}_\theta[Z]$, we can relate the objectives from Equation (B.12) and (B.13) to each other. First, we characterize a shared optimum, and then we show that both objectives are generally not minimized at the same time. For both objectives to be minimized, we have $\nabla\, \mathbb{H}_\theta[Y \mid Z] = 0$ and $\nabla\, \mathbb{H}_\theta[Z] = 0$, and we obtain

$$\nabla\, \mathbb{H}_\theta[Y, Z] = \nabla\, \mathbb{H}_\theta[Y \mid Z] + \nabla\, \mathbb{H}_\theta[Z] = 0. \tag{B.14}$$

From this we conclude that minimizing both objectives also minimizes $\mathbb{H}_\theta[Y, Z]$, and that generally the objectives trade-off with each other at stationary points θ of $\mathbb{H}_\theta[Y, Z]$:

$$\nabla \mathbb{H}_\theta[Y \mid Z] = -\nabla \mathbb{H}_\theta[Z] \quad \text{when } \nabla \mathbb{H}_\theta[Y, Z] = 0. \tag{B.15}$$

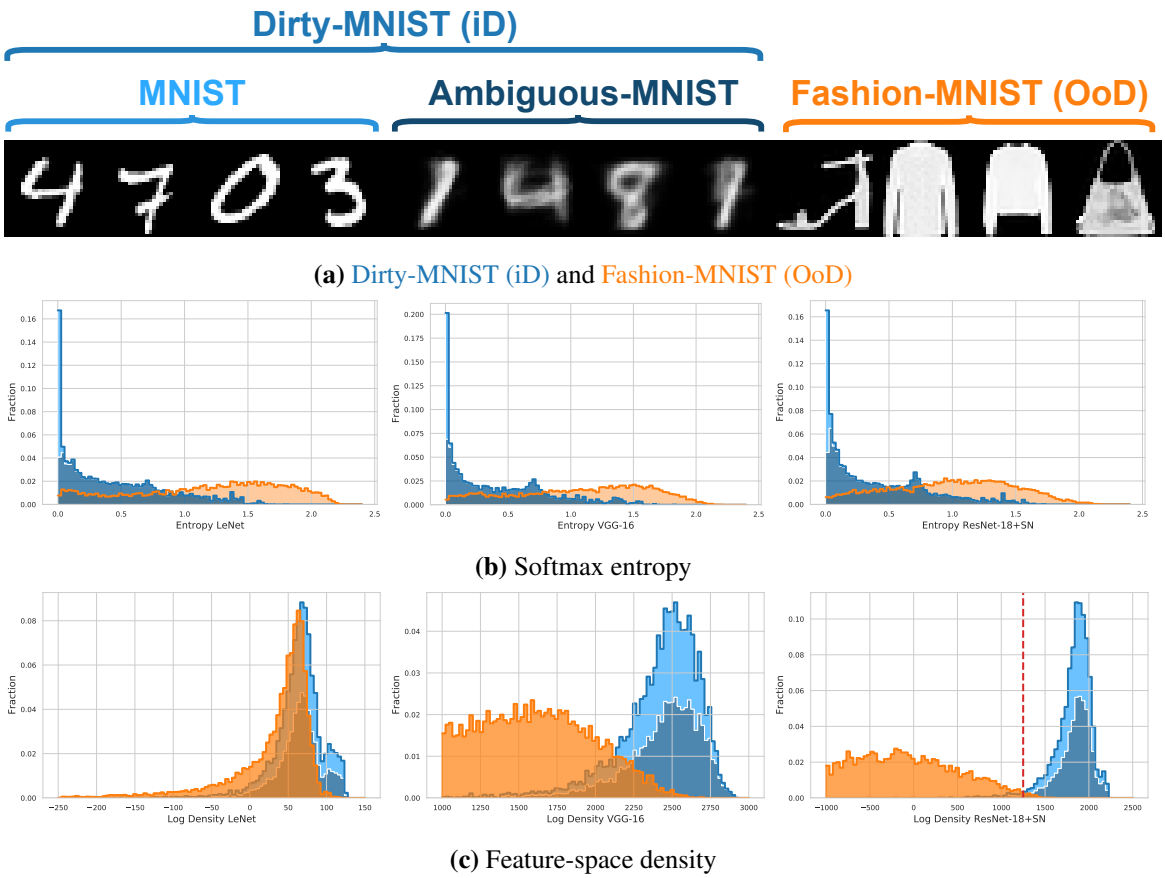
This tells us that to construct a case where the optima do not coincide, discriminative classification needs to be opposed better density estimation.

Specifically, when we have a GMM with one component per class, minimizing $\mathbb{H}_\theta[Y, Z]$ on an empirical data distribution is equivalent to Gaussian Discriminant Analysis, as is easy to check, and minimizing $\mathbb{H}_\theta[Z]$ is equivalent to fitting a density estimator, following Equation (B.13). The difference is that using a GMM as a density estimator does not constrain the component assignment, unlike in GDA.

Consequently, we see that *both objectives can be minimized at the same time exactly when the feature representations of different classes are perfectly separated*, such that a GMM fit as density estimator would assign each class's feature representations to a single component.

By the above, we can construct a simple case: if the samples of different classes are not separated in feature-space, optimas for the objectives will not coincide, so for example if samples were drawn from the same Gaussian and labeled randomly. On the other hand, if we have classes whose features lie in well-separated clusters, GDA will minimize all objectives.

□



(a) Dirty-MNIST (iD) and Fashion-MNIST (OoD)

(b) Softmax entropy

(c) Feature-space density

Figure B.7.1: *Disentangling aleatoric and epistemic uncertainty on Dirty-MNIST (iD) and Fashion-MNIST (OoD) **a** requires using* softmax entropy ***b** and* feature-space density (GMM) ***c** with appropriate inductive biases (*ResNet-18+SN *vs* LeNet *&* VGG-16 *without them). Enlarged version.* **b:** Softmax entropy captures aleatoric uncertainty for iD data (Dirty-MNIST), thereby separating unambiguous MNIST samples and Ambiguous-MNIST samples. However, iD and OoD are confounded: softmax entropy has arbitrary values for OoD, indistinguishable from iD. **c:** With appropriate inductive biases (DDU with ResNet-18+SN), iD and OoD densities do not overlap, capturing epistemic uncertainty. However, without appropriate inductive biases (LeNet & VGG-16), feature density suffers from *feature collapse*: iD and OoD densities overlap. Generally, feature-space density confounds unambiguous and ambiguous iD samples as their densities overlap. **Note:** Unambiguous MNIST samples and Ambiguous-MNIST samples are shown as stacked histograms with the total fractions adding up to 1 for Dirty-MNIST.

C

Appendix: Raising the Bar on the Evaluation of Out-of-Distribution Detection

Contents

C.1 Additional Training Details 206
C.1.1 Classifier Training 206
C.1.2 Training Pix-2-Pix GAN 208
C.1.3 Training GAN 208
C.2 Outlier Exposure 209
C.3 Additional Results 210
C.4 Qualitative Examples of Generated Samples 212

C.1 Additional Training Details

In this section, we provide training details for all models used to report results in §5.4 of the main paper.

C.1.1 Classifier Training

MNIST models: For experiments on MNIST, we use 4 convolutional architectures: LeNet, AlexNet, VGG-11 and ResNet-18. Each model has been trained on a single 12 GB TITAN

Xp GPU for 100 epochs using SGD as the optimizer with a momentum of 0.9. The initial learning rate used was 0.1 and there are learning rate drops by a factor of 10 at training epochs 40 and 60. The training batch size used for MNIST is 256.

CIFAR-10/100 models: All convolutional classifiers: DenseNet-121, ResNet-50/110, VGG-16, are trained using the Pytorch framework with a single 12 GB TITAN Xp GPU. To train models on CIFAR-10/100, we use the SGD optimiser with a momentum of 0.9 and a weight decay of $5e^{-4}$. We train each model for 350 epochs using 0.1 as the learning rate and a learning rate drop by a factor of 10 at training epochs 150 and 250. We use a training batch size of 128 and augment the training set using random crops and random horizontal flips.

For Vision Transformer (ViT) models trained on CIFAR-10/100, we use 4 12 GB TITAN Xp GPUs to train a single model. We train 4 different ViT models: ViT-B-16/32 and ViT-L-16/32, using an image size of 224×224 and other conventional augmentations including random crop and random horizontal flips. All the ViTs are pretrained on ImageNet-21K. We use a SGD with a momentum of 0.9 and a learning rate of $3e^{-2}$ with a cosine learning rate decay. We use 500 warmup steps for each model and train them for a maximum of 10000 steps. We use a training batch size of 256 for the ViT models.

ImageNet models: For ImageNet, we use pre-trained convolutional models available in the `torchvision.models` library. Furthermore, we use pretrained Vision Transformers for all evaluation purposes.[1]

Classifier Suite for computing $\mathcal{L}_{\text{MI}}$ Note that the in order to compute $\mathcal{L}_{\text{MI}}$, we use a single ensemble containing models, each with a different architecture. For MNIST experiments, we use 4 different models: LeNet, AlexNet, VGG-11 and ResNet-18 (one model of each architecture) as the ensemble to compute mutual information over. Similarly, for CIFAR-10/100, we use 6 models with 6 different architectures: DenseNet-121, ResNet-50/110, VGG-16, Wide-ResNet-28-10 and Inception-v3. Finally, for ImageNet, we use a set of pretrained classifiers from the Pytorch `torchvision.models` library. In particular, we get ResNet-18, MobileNet-v3-Large and EfficientNet-B0. Note that the use of ensembles with different

[1] See `github.com/rwightman/pytorch-image-models` for details.

architectures is to encourage higher variability in predictions and representations within the ensemble, thereby encouraging higher mutual information for predictions. Ensembles used for the evaluation of generated samples all have the same architecture. All the classifiers used for computing $\mathcal{L}_{\text{MI}}$ are trained using the same dataset-specific settings as mentioned above.

C.1.2 Training Pix-2-Pix GAN

In order to train a Pix-2-Pix GAN, we use $\mathcal{L}_{\text{Shift}}$ defined in Equation (5.3) as the loss function for the generator of the GAN and there is no change to the loss of the discriminator. However, note that the target image for the Pix-2-Pix discriminator is the same as the input. Thus the loss of the discriminator can be given as:

$$\begin{aligned}\mathcal{L}_{\text{D}_{\text{Pix-2-Pix}}} =& \mathbb{E}_{\mathbf{x}}\left[\log(D(\mathbf{x},\mathbf{x}))\right] - \\ & \mathbb{E}_{\mathbf{x},\mathbf{z}}\left[\log(1 - D(\mathbf{x}, G(\mathbf{x},\mathbf{z})))\right]\end{aligned} \tag{C.1}$$

For $\mathcal{L}_{\text{MI}}$, in Equation (5.3), we use the method specified above. We use a single 12 GB TITAN Xp GPU to train the Pix-2-Pix model on CIFAR-10 and 8 such GPUs to train on ImageNet. We use a training batch size of 256 and train the model for 100 epochs using Adam as the optimizer, a learning rate of 0.0002 and beta values 0.5 and 0.999. All other training settings are the same as specified in the original Pix-2-Pix paper [220].

C.1.3 Training GAN

For generating Near OoD samples (i.e., Near OoD), we use a DCGAN for MNIST and a BigGAN for CIFAR-10/100 and ImageNet. We use a single 12 GB TITAN Xp GPU to train DCGAN for MNIST and BigGAN for CIFAR-10/100. However, we use 8 such GPUs to train a single BigGAN on ImageNet. The loss function for the discriminator of the GAN undergoes no change and is shown as follows:

$$\mathcal{L}_{\text{D}_{\text{GAN}}} = \mathbb{E}_{\mathbf{x}}\left[\log D(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}}\left[\log(1 - D(G(z)))\right] \tag{C.2}$$

Model	Outlier Dataset	# Outlier Classes	Test Accuracy	AUROC					
				SVHN	C10	Tiny-ImageNet	N C100	Places365	Texture
ResNet-50	None	0	79.52	80.97	78.98	79.52	56.38	71.17	68.71
	SVHN	10	78.79	–	79.95	80.02	60.45	72.30	70.07
	C10	10	78.98	82.97	–	83.11	62.60	73.23	72.17
	N C100 (Ours)	100	78.82	87.02	81.65	84.40	–	75.80	74.92
	Tiny-ImageNet	200	79.07	86.46	81.57	–	64.11	75.69	74.75
	ImageNet (Subset)	500	78.61	88.17	82.66	85.01	65.32	76.44	74.87
Wide-ResNet-28-10	None	0	80.46	81.46	80.54	81.84	62.69	75.43	73.62
	SVHN	10	80.13	–	82.97	82.77	64.44	76.21	75.13
	C10	10	79.7	84.45	–	82.98	64.52	78.4	77.21
	N C100 (Ours)	100	79.92	87.23	84.31	85.63	–	80.61	80.12
	Tiny-ImageNet	200	79.83	87.09	84.52	–	65.73	80.34	80.06
	ImageNet (Subset)	500	79.74	89.01	85.6	83.57	66.24	81.7	80.77

Table C.2.1: AUROC% obtained by performing outlier exposure [98] on models trained on CIFAR-100 (C100) with increasingly diverse OoD datasets. Models tuned using Near OoD CIFAR-100 (N C100) as outliers with 100 outlier classes outperform less diverse outlier sets and perform competitively with more diverse outlier sets including Tiny-ImageNet and our ImageNet subset (500 classes).

The loss function for the generator is $\mathcal{L}_{\text{Near OoD}}$ as shown in Equation (5.4). The $\mathcal{L}_{\text{MI}}$ in $\mathcal{L}_{\text{Near OoD}}$ is computed as described above. We train all GANs for 100 epochs and all other training details for the GANs are exactly the same as set out in their respective repositories.[2]

C.2 Outlier Exposure

In [98], exposure to outliers during training was proposed as a way to improve model performance on OoD datasets. In outlier exposure, models are trained on two datasets: i) the training set on which the loss is the usual cross-entropy and ii) the outlier dataset on which the loss is the cross-entropy between the softmax distribution and a uniform distribution over classes. The assumption is that exposure to good outlier datasets will make the model detect any unseen outlier datasets as well. In this experiment, we want to see how models can improve on OoD detection performance once exposed to increasingly diverse OoD samples as outliers and if our Near OoD samples can be used effectively for outlier exposure.

To do this, we train a ResNet-50 and a Wide-ResNet-28-10 on CIFAR-100 using SVHN, CIFAR-10, Near OoD CIFAR-100, Tiny-ImageNet and a subset of ImageNet as outlier datasets. For the ImageNet outliers, we use a subset of 500 classes from ImageNet which are

[2] See `github.com/ajbrock/BigGAN-PyTorch` for details on training BigGAN.

disjoint from CIFAR-100 and with 100 randomly chosen images from each class. Note that the outlier sets are increasingly diverse with Near OoD CIFAR-100 having the same number of classes as CIFAR-100 and Tiny-ImageNet and the ImageNet subset having 200 and 500 classes respectively. The training procedure is the same as set out in §C.1.1. However, in the loss, following [98], in addition to the cross-entropy term, we also have an additional regulariser which computes the cross-entropy of the output with a uniform distribution for outlier samples. For both architectures, we also compare with a baseline with no exposure to outliers. Finally, we use Places365 [328] Texture [241] as independent OoD datasets which are not used for any outlier exposure. We present the test accuracy and AUROC scores for all models in Table C.2.1. *We observe that models trained using Near OoD CIFAR-100 as outliers consistently outperform models trained with relatively less diverse outlier sets like CIFAR-10 and SVHN. Additionally, we find that these models also broadly outperform Tiny-ImageNet and are competitive with models trained using the ImageNet subset which are more diverse outlier sets. All models outperform the ones trained without any outliers.*

The above observation provides additional evidence to support the use of Near OoD samples, not just to benchmark OoD detection baselines, but also to improve them through outlier exposure. It indicates that our generated samples are not obtained from random transformations of in-distribution images but indeed capture desirable properties which make them effective outliers for training more robust models. This corroborates our previous observation that even an image which does not represent any real world object can be very useful if it captures desirable properties in terms of semantic and perceptual similarity.

C.3 Additional Results

In this section, we present additional results to support the results in the main paper.

MI overlap: In Figure C.3.1, we show the mutual information of the training ensemble on real CIFAR-10 samples along with Near OoD generated samples on CIFAR-10. We choose samples which minimise MI overlap between real and generated samples without

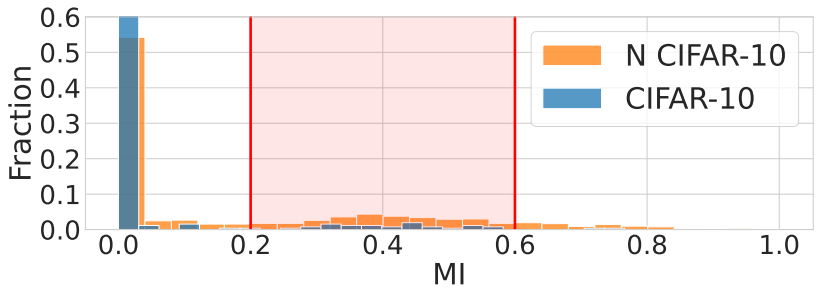


Figure C.3.1: MI of the ensemble for Near OoD (N) CIFAR-10 and real CIFAR-10 samples. We use this to find plausible thresholds of MI.

having a very high MI as that leads to generated samples losing their perceptual similarity with iD. For CIFAR-10, we choose $[0.2, 0.6]$ as the MI interval for generated samples.

OoD Detection on Near OoD Datasets Table C.3.1 presents test set accuracy and AUROC scores of models trained on MNIST on the MNIST vs Fashion-MNIST and MNIST vs Near OoD MNIST. In Table C.3.2, we report the CIFAR-10/100 and ImageNet test set accuracy of all the models we use to evaluate our benchmark. In Table C.4.6 and Table C.4.8, we report the AUROC scores of 6 convolutional models: DenseNet-121, ResNet-50/110, VGG-16, Wide-ResNet-28-10 and Inception-v3 and 4 Vision Transformer models: ViT-B-16/32, ViT-L-16/32 trained on CIFAR-10 and CIFAR-100 respectively. The uncertainty computation method here uses the softmax entropy, softmax confidence and Mahalanobis distance computed from a single deterministic model. We also compute the AUROC scores for a deep ensemble of size 5, using the 6 convolutional architectures and report the corresponding results in Table C.4.1 and Table C.4.3 for models trained on CIFAR-10 and CIFAR-100 respectively. For CIFAR-10, we use SVHN, CIFAR-100 and Near OoD CIFAR-10 as OoD sets and for CIFAR-100, we use SVHN, CIFAR-10 and Near OoD CIFAR-100 as OoD sets. The corresponding AUPRC scores for all models trained on CIFAR-10 and CIFAR-100 are shown in Table C.4.7 and Table C.4.9 for deterministic models and Table C.4.2 and Table C.4.4 for deep ensembles respectively. In addition, we also show the AUPRC scores as plots for deterministic models, deep ensembles and Vision Transformers in Figure C.4.1.

Model	Test Accuracy	AUROC %			
		F-MNIST (SE)	F-MNIST (SC)	N-MNIST (Ours) (SE)	N-MNIST (Ours) (SC)
LeNet	98.97 ± 0.02	98.87 ± 0.05	98.80 ± 0.05	$\mathbf{65.29 \pm 0.12}$	$\mathbf{64.14 \pm 0.11}$
AlexNet	99.04 ± 0.03	99.10 ± 0.05	99.07 ± 0.05	$\mathbf{70.31 \pm 0.15}$	$\mathbf{69.64 \pm 0.13}$
VGG-11	99.35 ± 0.02	99.20 ± 0.04	99.17 ± 0.03	$\mathbf{72.11 \pm 0.13}$	$\mathbf{71.85 \pm 0.13}$
ResNet-18	99.54 ± 0.02	99.16 ± 0.03	99.14 ± 0.04	$\mathbf{73.15 \pm 0.13}$	$\mathbf{72.81 \pm 0.12}$

Table C.3.1: AUROC % on MNIST using softmax entropy (SE) and softmax confidence (SC) with Fashion(F)-MNIST and Near OoD(N)-MNIST as OoD.

Model	Test/Val Set Accuracy		
	CIFAR-10	CIFAR-100	ImageNet
DenseNet-121	95.66 ± 0.05	80.08 ± 0.15	–
ResNet-50	95.34 ± 0.05	78.26 ± 0.33	80.86
ResNet-110	95.50 ± 0.12	79.50 ± 0.27	–
VGG-16	93.81 ± 0.09	74.33 ± 0.18	–
Wide-ResNet-28-10	96.33 ± 0.07	80.60 ± 0.11	–
Wide-ResNet-50-2	–	–	81.60
Inception-v3	95.25 ± 0.10	78.04 ± 0.14	–
ViT-B-16	99.12 ± 0.03	92.73 ± 0.04	85.02
ViT-B-32	98.73 ± 0.01	92.14 ± 0.02	84.72
ViT-L-16	99.18 ± 0.01	93.73 ± 0.04	86.55
ViT-L-32	99.02 ± 0.01	93.29 ± 0.04	85.47

Table C.3.2: CIFAR-10/100 test and ImageNet val accuracy for CNNs and ViTs used in our evaluation.

Evaluation of Shifted Datasets We present the ECE% of 6 architectures: DenseNet-121, ResNet-50/110, VGG-16, Wide-ResNet-28-10 and Inception-v3, all trained on CIFAR-10 on CIFAR-10-C where we use 15 different corruption types at the highest intensity (i.e., 5) and compare with the ECE% of shifted CIFAR-10. Similarly, we present the ECE% of 4 ViTs: ViT-B-16, ViT-B-32, ViT-L-16 and ViT-L-32, trained on ImageNet, evaluated on ImageNet-C. The results are presented in Figure C.4.2. Clearly, the ECE of shifted CIFAR-10 and shifted ImageNet is significantly higher than all corruption types.

C.4 Qualitative Examples of Generated Samples

In Figure C.4.3, Figure C.4.4 and Figure C.4.5, we present additional qualitative samples of shifted and near OoD examples respectively for both CIFAR-10, CIFAR-100 and ImageNet. In Figure C.4.3, on the left column, we show real samples from CIFAR-10 and CIFAR-100. On the right column, we show corresponding shifted samples. Similar examples for ImageNet

Model	AUROC		
	SVHN	CIFAR-100	N CIFAR-10
DenseNet-121	97.52	91.42	**85.67**
ResNet-50	96.24	90.89	**84.66**
ResNet-110	96.75	91.3	**85.55**
VGG-16	91.26	89.16	**81.07**
Wide-ResNet-28-10	96.59	91.78	**86.54**
Inception-v3	96.12	91.31	**87.07**

Table C.4.1: AUROC of ensemble models trained on CIFAR-10 using predictive entropy on SVHN, CIFAR-100 and Near OoD CIFAR-10 (N CIFAR-10).

Model	AUPRC		
	SVHN	CIFAR-10	N CIFAR-100
DenseNet-121	98.83	90.72	**86.92**
ResNet-50	97.87	89.7	**85.83**
ResNet-110	98.19	90.35	**86.7**
VGG-16	94.89	87.97	**83.93**
Wide-ResNet-28-10	98.06	90.94	**88.03**
Inception-v3	97.79	90.14	**88.14**

Table C.4.2: AUPRC of ensemble models trained on CIFAR-10 using predictive entropy on SVHN, CIFAR-100 and Near OoD CIFAR-10 (N CIFAR-10).

Model	AUROC		
	SVHN	CIFAR-10	N CIFAR-100
DenseNet-121	88.75	82.92	**62.33**
ResNet-50	82.66	81.28	**57.18**
ResNet-110	81.07	82.55	**59.7**
VGG-16	78.3	78.87	**52.42**
Wide-ResNet-28-10	83.62	82.65	**62.83**
Inception-v3	83.89	83.3	**64.66**

Table C.4.3: AUROC of ensemble models trained on CIFAR-100 using predictive entropy on SVHN, CIFAR-10 and Near OoD CIFAR-100.

Model	AUPRC		
	SVHN	CIFAR-10	N CIFAR-100
DenseNet-121	93.97	78.96	**67.82**
ResNet-50	90.24	76.78	**64.85**
ResNet-110	89.08	78.67	**66.21**
VGG-16	88.27	74.51	**62.7**
Wide-ResNet-28-10	91.27	78.81	**67.94**
Inception-v3	89.81	79.14	**68.76**

Table C.4.4: AUPRC of ensemble models trained on CIFAR-100 using predictive entropy on SVHN, CIFAR-10 and Near OoD CIFAR-100.

Model	CIFAR-10-C		Shifted CIFAR-10 (Ours)
	Avg ECE %	*Max ECE %*	*ECE %*
DenseNet121	13.69 ± 0.17	25.86 ± 0.40	$\mathbf{51.55 \pm 0.33}$
ResNet-50	13.71 ± 0.48	25.76 ± 1.09	$\mathbf{50.07 \pm 1.24}$
ResNet-110	14.40 ± 0.28	28.03 ± 0.55	$\mathbf{52.16 \pm 0.66}$
VGG-16	17.51 ± 0.22	34.45 ± 0.40	$\mathbf{56.25 \pm 0.41}$
Wide-ResNet-28-10	11.92 ± 0.13	22.87 ± 0.21	$\mathbf{49.64 \pm 0.43}$
Inception-v3	13.47 ± 0.37	25.10 ± 0.71	$\mathbf{52.84 \pm 0.19}$

Table C.4.5: ECE % on CIFAR-10-C compared to Shifted CIFAR-10.

can be found in Figure C.4.4. Finally, in Figure C.4.5, we show examples of near OoD samples for both CIFAR-10, CIFAR-100 and ImageNet.

Model	AUROC SVHN			AUROC CIFAR-100			AUROC Near OoD CIFAR-10		
	Entropy	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*
DenseNet-121	93.12 ± 1.13	92.85 ± 1.11	96.22 ± 0.30	87.23 ± 0.21	87.17 ± 0.22	89.71 ± 0.14	78.81 ± 0.36	79.11 ± 0.34	79.75 ± 0.39
ResNet-50	92.39 ± 0.30	92.17 ± 0.30	92.67 ± 1.35	86.92 ± 0.53	86.78 ± 0.50	88.40 ± 0.33	78.92 ± 0.75	79.09 ± 0.72	79.04 ± 0.57
ResNet-110	91.63 ± 1.82	91.41 ± 1.81	91.94 ± 1.56	87.48 ± 0.09	87.35 ± 0.09	87.91 ± 0.2	78.08 ± 0.49	80.20 ± 0.48	78.14 ± 0.50
VGG-16	86.70 ± 1.05	86.78 ± 1.00	90.93 ± 0.81	83.37 ± 0.22	83.30 ± 0.21	85.94 ± 0.35	73.43 ± 0.55	73.61 ± 0.53	75.46 ± 1.12
Wide-ResNet-28-10	90.98 ± 1.14	90.89 ± 1.09	98.72 ± 0.11	88.60 ± 0.06	88.48 ± 0.06	91.15 ± 0.02	80.56 ± 0.47	81.73 ± 0.46	81.78 ± 0.11
Inception-v3	91.94 ± 0.54	91.77 ± 0.53	93.49 ± 0.79	86.54 ± 0.43	86.42 ± 0.42	89.56 ± 0.28	80.27 ± 0.39	80.41 ± 0.38	83.76 ± 0.43
ViT-B-16	99.65 ± 0.01	99.49 ± 0.01	96.67 ± 0.18	98.33 ± 0.03	98.19 ± 0.03	98.87 ± 0.00	87.00 ± 0.04	$87.08 \pm .04$	86.65 ± 0.22
ViT-B-32	99.65 ± 0.01	99.44 ± 0.02	95.35 ± 0.21	98.10 ± 0.03	97.93 ± 0.03	98.67 ± 0.01	85.33 ± 0.12	$85.44 \pm .12$	86.21 ± 0.23
ViT-L-16	99.76 ± 0.02	99.64 ± 0.01	$97.66 \pm .42$	98.70 ± 0.02	98.61 ± 0.01	99.17 ± 0.01	85.93 ± 0.28	86.15 ± 0.27	89.47 ± 0.25
ViT-L-32	99.78 ± 0.01	99.63 ± 0.02	95.63 ± 0.09	$98.45 \pm .02$	$98.29 \pm .02$	$98.80 \pm .02$	85.25 ± 0.2	85.38 ± 0.20	84.83 ± 0.11

Table C.4.6: AUROC of models trained on CIFAR-10 using softmax entropy (Entropy), softmax confidence (Confidence) and Mahalanobis distance on SVHN, CIFAR-100 and Near OoD CIFAR-10. Near OoD samples are far harder to detect given their consistently low AUROC scores.

Model	AUPRC SVHN			AUPRC CIFAR-100			AUPRC Near OoD CIFAR-10		
	Entropy	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*
DenseNet-121	96.78 ± 0.38	82.89 ± 5.11	94.4 ± 0.41	86.84 ± 0.11	84.7 ± 0.66	89.75 ± 0.15	80.38 ± 0.14	63.6 ± 0.52	68.72 ± 0.75
ResNet-50	95.88 ± 0.13	86.19 ± 1.44	88.41 ± 2.29	85.85 ± 0.39	85.39 ± 0.92	88.45 ± 0.42	80.97 ± 0.33	65.81 ± 1.92	68.86 ± 1.09
ResNet-110	95.58 ± 0.95	85.35 ± 3.25	86.25 ± 2.29	86.29 ± 0.14	86.27 ± 0.23	87.54 ± 0.29	80.8 ± 0.28	67.73 ± 1.26	64.41 ± 1.2
Wide-ResNet-28-10	95.58 ± 0.59	77.81 ± 2.88	97.6 ± 0.15	88.15 ± 0.08	85.83 ± 0.16	91.55 ± 0.04	80.9 ± 0.22	66.03 ± 0.87	72.06 ± 0.2
Inception-v3	95.7 ± 0.31	83.32 ± 1.8	90.91 ± 0.86	85.86 ± 0.41	83.14 ± 0.67	90.13 ± 0.28	81.47 ± 0.22	63.45 ± 0.66	78.11 ± 0.34
ViT-B-16	99.86 ± 0.0	99.08 ± 0.02	93.2 ± 0.37	98.41 ± 0.03	98.18 ± 0.03	98.77 ± 0.01	92.54 ± 0.03	76.39 ± 0.19	76.72 ± 0.43
ViT-B-32	99.86 ± 0.01	99.0 ± 0.04	90.64 ± 0.42	98.19 ± 0.03	97.88 ± 0.02	98.57 ± 0.01	91.53 ± 0.08	73.21 ± 0.15	76.56 ± 0.6
ViT-L-16	99.9 ± 0.01	99.36 ± 0.04	95.42 ± 0.85	98.8 ± 0.01	98.49 ± 0.02	99.02 ± 0.01	92.68 ± 0.12	70.26 ± 0.77	81.19 ± 0.29
ViT-L-32	99.91 ± 0.0	99.33 ± 0.02	90.36 ± 0.18	98.55 ± 0.02	98.25 ± 0.04	98.6 ± 0.03	92.02 ± 0.08	69.81 ± 0.59	73.29 ± 0.14

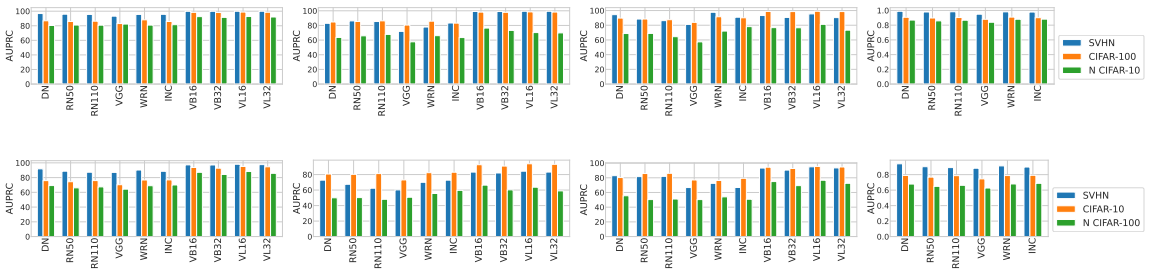
Table C.4.7: AUPRC of models trained on CIFAR-10 using softmax entropy (Entropy), softmax confidence (Confidence) and Mahalanobis distance on SVHN, CIFAR-100 and Near OoD CIFAR-10. Near OoD samples are far harder to detect given their consistently low AUPRC scores.

Model	AUROC SVHN			AUROC CIFAR-100			AUROC Near OoD CIFAR-100		
	Entropy	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*
DenseNet-121	84.52 ± 1.55	83.13 ± 1.44	89.49 ± 0.55	79.73 ± 0.25	79.12 ± 0.24	77.26 ± 0.47	60.32 ± 0.27	60.5 ± 0.26	64.16 ± 0.28
ResNet-50	79.77 ± 0.69	78.82 ± 0.71	86.41 ± 0.11	78.82 ± 0.08	78.26 ± 0.09	82.68 ± 0.18	56.58 ± 0.26	56.67 ± 0.26	58.48 ± 0.67
ResNet-110	77.84 ± 1.56	77.26 ± 1.4	86.62 ± 0.23	79.92 ± 0.17	79.3 ± 0.15	82.9 ± 0.23	58.6 ± 0.56	58.58 ± 0.46	59.73 ± 0.59
VGG-16	76.33 ± 1.12	75.38 ± 0.97	78.01 ± 1.24	74.02 ± 0.14	73.62 ± 0.13	74.99 ± 0.13	51.06 ± 0.14	51.53 ± 0.15	56.41 ± 0.42
Wide-ResNet-28-10	81.85 ± 0.79	80.71 ± 0.7	84.18 ± 1.01	80.82 ± 0.11	80.41 ± 0.12	73.42 ± 0.14	62.19 ± 0.17	62.05 ± 0.14	62.38 ± 0.1
Inception-v3	81.6 ± 1.64	80.95 ± 1.46	81.8 ± 0.57	81.24 ± 0.18	80.89 ± 0.18	79.87 ± 0.22	63.96 ± 0.85	63.39 ± 0.78	60.53 ± 0.97
ViT-B-16	93.31 ± 0.21	91.92 ± 0.19	95.91 ± 0.03	93.29 ± 0.04	92.35 ± 0.05	93.95 ± 0.03	79.47 ± 0.06	79.04 ± 0.06	82.91 ± 0.07
ViT-B-32	92.98 ± 0.13	91.56 ± 0.11	93.78 ± 0.21	91.97 ± 0.2	90.94 ± 0.21	92.22 ± 0.19	75.36 ± 0.16	75.05 ± 0.15	78.97 ± 0.24
ViT-L-16	95.11 ± 0.16	94.29 ± 0.15	97.6 ± 0.04	94.62 ± 0.08	94.04 ± 0.09	95.31 ± 0.09	80.36 ± 0.08	80.23 ± 0.1	84.72 ± 0.21
ViT-L-32	94.01 ± 0.07	92.62 ± 0.06	96.01 ± 0.12	94.09 ± 0.07	93.28 ± 0.06	94.15 ± 0.06	76.87 ± 0.12	76.64 ± 0.12	81.29 ± 0.14

Table C.4.8: AUROC of models trained on CIFAR-100 using softmax entropy (Entropy), softmax confidence (Confidence) and Mahalanobis distance on SVHN, CIFAR-10 and Near OoD CIFAR-100. Near OoD samples are far harder to detect given their consistently low AUROC scores.

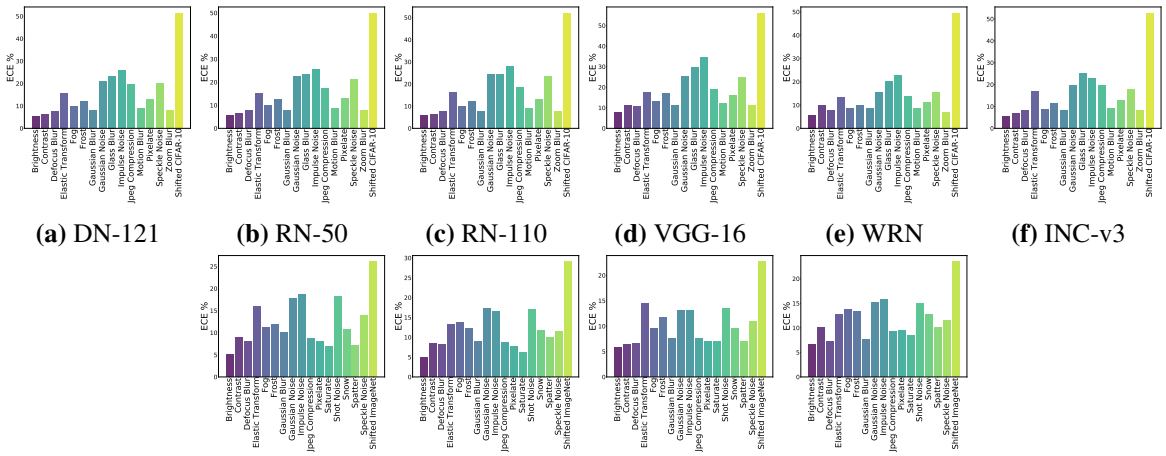
Model	AUPRC SVHN			AUPRC CIFAR-100			AUPRC Near OoD CIFAR-100		
	Entropy	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*	*Entropy*	*Confidence*	*Mahalanobis*
DenseNet-121	91.84 ± 0.93	72.78 ± 2.29	82.85 ± 0.73	75.89 ± 0.3	80.67 ± 0.38	80.31 ± 0.26	69.17 ± 0.13	50.01 ± 0.6	55.53 ± 0.3
ResNet-50	88.69 ± 0.28	67.45 ± 1.23	81.58 ± 0.51	74.4 ± 0.14	80.26 ± 0.18	85.78 ± 0.45	66.07 ± 0.15	50.27 ± 0.6	50.33 ± 0.91
ResNet-110	87.29 ± 1.01	62.34 ± 2.97	81.9 ± 0.78	75.94 ± 0.17	81.25 ± 0.23	86.1 ± 0.55	67.37 ± 0.35	48.07 ± 1.02	51.08 ± 0.99
VGG-16	87.04 ± 0.78	60.21 ± 1.61	66.7 ± 1.54	70.35 ± 0.18	73.07 ± 0.23	77.02 ± 0.25	64.3 ± 0.07	50.72 ± 0.17	50.32 ± 0.39
Wide-ResNet-28-10	90.27 ± 0.55	69.98 ± 1.23	72.42 ± 0.92	76.76 ± 0.17	82.41 ± 0.11	76.36 ± 0.15	68.98 ± 0.09	55.56 ± 0.19	54.03 ± 0.13
Inception-v3	88.54 ± 1.19	72.75 ± 2.25	66.81 ± 2.55	76.89 ± 0.3	82.99 ± 0.1	79.31 ± 0.35	69.98 ± 0.34	59.6 ± 1.21	50.77 ± 1.07
ViT-B-16	97.24 ± 0.08	83.16 ± 0.85	93.22 ± 0.06	93.73 ± 0.03	92.68 ± 0.06	94.4 ± 0.02	87.15 ± 0.02	66.33 ± 0.15	74.84 ± 0.18
ViT-B-32	97.1 ± 0.06	82.0 ± 0.69	90.46 ± 0.28	92.79 ± 0.18	90.93 ± 0.21	92.66 ± 0.25	84.23 ± 0.15	60.24 ± 0.12	69.31 ± 0.4
ViT-L-16	98.08 ± 0.06	84.33 ± 0.89	94.92 ± 0.09	95.11 ± 0.06	93.89 ± 0.12	95.43 ± 0.08	88.25 ± 0.05	63.66 ± 0.21	76.63 ± 0.33
ViT-L-32	97.58 ± 0.01	83.2 ± 0.41	93.4 ± 0.23	94.65 ± 0.06	93.2 ± 0.08	94.66 ± 0.07	85.8 ± 0.06	58.91 ± 0.24	72.46 ± 0.21

Table C.4.9: AUPRC of models trained on CIFAR-100 using softmax entropy (Entropy), softmax confidence (Confidence) and Mahalanobis distance on SVHN, CIFAR-10 and Near OoD CIFAR-100. Near OoD samples are far harder to detect given their consistently low AUPRC scores.



(a) Entropy **(b)** Confidence **(c)** Mahalanobis **(d)** Ensemble

Figure C.4.1: AUPRC % for different models, DenseNet-121 (DN), ResNet-50 (RN50), ResNet-110 (RN110), VGG-16, Wide-ResNet-28-10 (WRN) and Inception-v3 (INC), ViT-B-16/32 (VB16/32) and ViT-L-16/32 (VL16/32) trained on CIFAR-10 (first row) and CIFAR-100 (second row) using SVHN, CIFAR-10/100 and Near OoD (N) CIFAR-10/100 as OoD datasets and softmax entropy, confidence, Mahalanobis distance and deep ensemble baselines.



(a) DN-121 **(b)** RN-50 **(c)** RN-110 **(d)** VGG-16 **(e)** WRN **(f)** INC-v3

(g) VB-16 **(h)** VB-32 **(i)** VL-16 **(j)** VL-32

Figure C.4.2: ECE% of models on CIFAR-10-C (top row) and ImageNet-C (bottom row) for different corruption types.



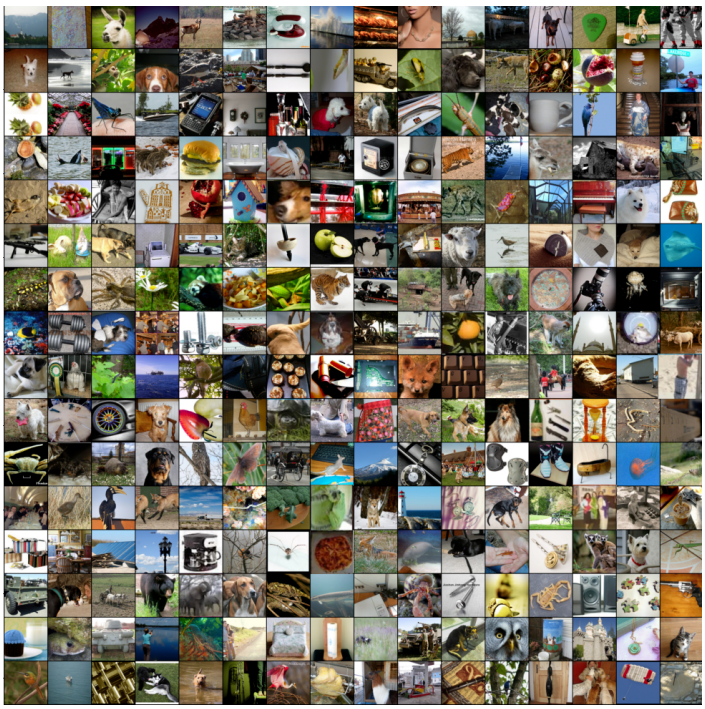
(a) CIFAR-10 Real

(b) CIFAR-10 Shifted

(c) CIFAR-100 Real

(d) CIFAR-100 Shifted

Figure C.4.3: Additional qualitative samples for CIFAR-10 and CIFAR-100 datasets. Left column shows real samples, and the right column shows corresponding shifted/transformed samples.

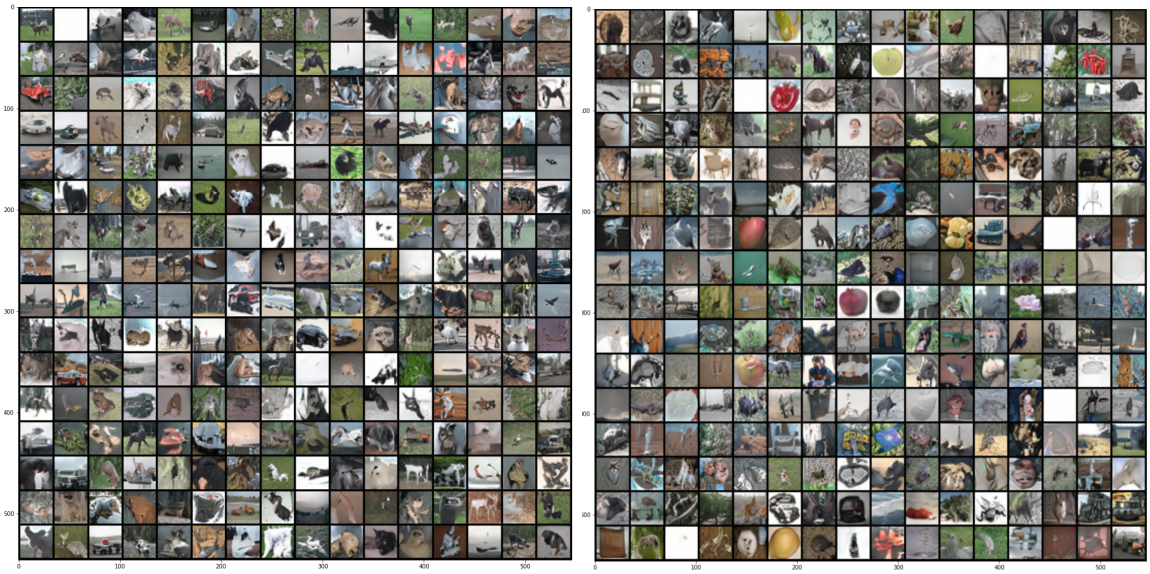


(a) ImageNet Real

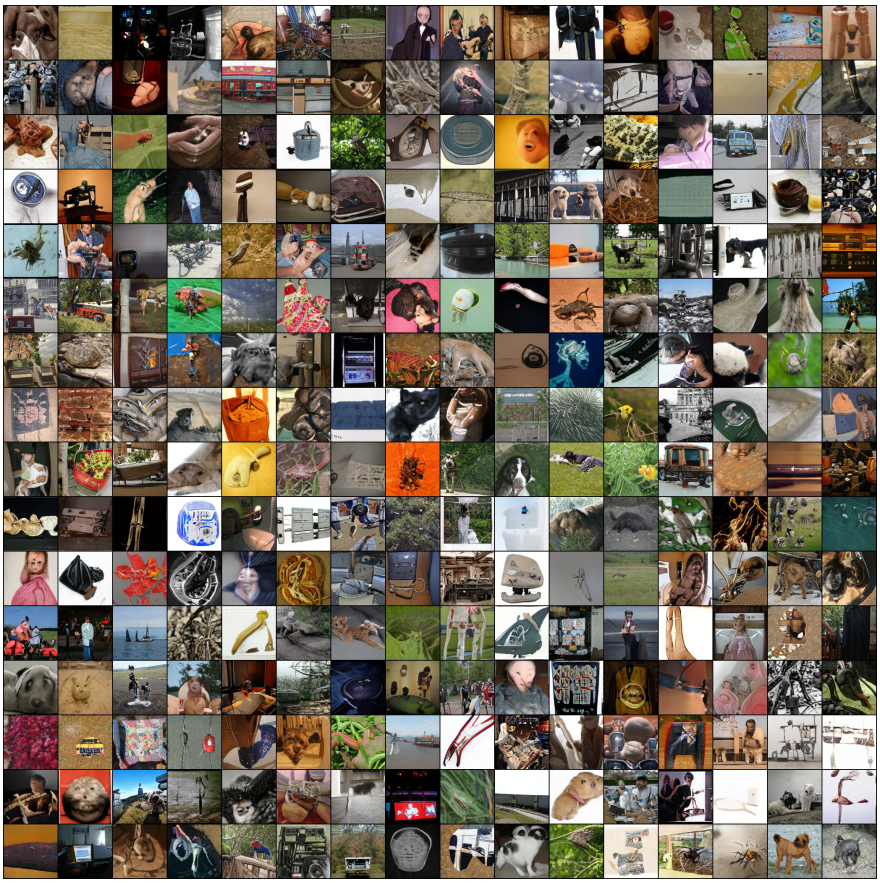


(b) ImageNet Shifted

Figure C.4.4: Additional qualitative samples for ImageNet datasets. Top shows real samples, and bottom shows corresponding shifted/transformed samples.



(a) Near OoD CIFAR-10 **(b)** Near OoD CIFAR-100



(c) Near OoD ImageNet

Figure C.4.5: Additional qualitative samples for CIFAR-10, CIFAR-100 and ImageNet datasets. Samples show Near OoD images.

D

Appendix: Open Vocabulary Semantic Segmentation with Patch Aligned Contrastive Learning

Contents

D.1 Additional Implementation Details 220
D.1.1 Vision Embedder Architecture 221
D.1.2 Training details for Vision Embedder 221
D.1.3 Training and Evaluation Datasets 223
D.2 Additional Results 225
D.2.1 Semantic Coherence in CLIP 225
D.2.2 Qualitative Segmentation Results 226
D.2.3 Zero-shot Image Classification 226

D.1 Additional Implementation Details

In this section, we describe the implementation details for the proposed PACL method. Particularly, in Appendix D.1.1, we describe the architecture of the vision embedder used for training PACL, in Appendix D.1.2, we describe specifics of training including hyperpa-

rameters and prompt engineering details. Finally, in Appendix D.1.3, we describe details of image-text datasets used for training as well as segmentation and image classification datasets used for evaluation.

D.1.1 Vision Embedder Architecture

In §6.6.1, we have discussed that the proposed PACL approach is flexible in the sense that PACL can be applied using pre-trained frozen encoders. Particularly, since CLIP's pre-trained vision encoders have desirable properties (see Semantic Coherence in §6.4), we use these pre-trained encoders from CLIP to train PACL to transfer to the task of zero-shot semantic segmentation. This simplifies the training to just a small vision embedder on top of the vision encoder. In this section and in Figure D.1.1, we present the architecture of the Vision embedder. In particular, we use a single residual block with two linear layers in the main branch and a single linear layer in the residual branch. There is a ReLU non-linearity between the two linear layers in the main branch. The resulting model requires training a mere 1.1M parameters whereas the architecture has a total of 150M parameters for CLIP ViT-B/16. This helps us in scaling up and training on a larger batch size for our experiments as there is no gradient propagation through the frozen image and text encoders.

D.1.2 Training details for Vision Embedder

In Appendix D.1.2, we describe the architecture of pre-trained encoders as well as the hyperparameters used for training the PACL models. In Appendix D.1.2, we provide some details on CLIP's prompt engineering used to derive best results from the text encoder of a pre-trained CLIP model.

Architecture and Hyperparameters

As mentioned above, we only train PACL using a Vision embedder on top of a pre-trained CLIP vision encoder. This allows us the flexibility not only of using multiple pre-trained vision encoders but also combinations of different vision and text encoders. In §6.6.1, we

show an ablation with combinations of different pre-trained vision and text encoders. In particular, we use: a) CLIP ViT-B/16 vision and text encoders, b) CLIP ViT-L/14 vision and text encoders and b) DINO ViT-B/16 vision encoder with CLIP ViT-B/16 text encoder. For each of these combinations, we train a vision embedder as discussed in Appendix D.1.1 and report zero-shot semantic segmentation results in Table 6.3 of the main paper.

All our models are trained on a single node with 4 NVIDIA A100 GPUs with a GPU memory of 40GB in each GPU. We use AdamW as the optimizer with beta values 0.9 and 0.98, an eps value of $1e-6$ and a weight decay of 0.2. We use an initial learning rate of $5e-4$ and reduce the learning rate using a Cosine Annealing schedule where the maximum number of iterations is set as the total number of iterations during training (i.e., number of epochs $\times$ number of iterations per epoch). We use a batch size of 4096 (1024 per GPU) and train the model for a total of 10 epochs on image-text data. *We do not use any segmentation annotations or class-agnostic segmentation masks during training*. We provide details on these image-text datasets in Appendix D.1.3. When the training dataset is a combination of GCC-3M, GCC-12M and YFCC-15M, the model takes 10 days to train on 4 NVIDIA A100 GPUs.

Prompt Engineering

Since we use CLIP's [254] pre-trained text encoders, we follow the prompt engineering guidelines following CLIP's OpenAI repository during inference time. In particular, during inference, we compute the average embedding from the text encoder using a set of 7 prompts: `itap of a ().`, `a bad photo of the ().`, `a origami ().`, `a photo of the large ().`, `a () in a video game.`, `art of the ()`, `a photo of the small ()`, where we put the name of the class within the parenthesis `()`. We use the mean of the embeddings from the prompts for each class in order to compute cosine similarity with the patch representations from the vision encoder. This is similar to the way CLIP performs zero-shot image classification, however CLIP uses only the CLS token from the vision encoder to compute cosine similarity.

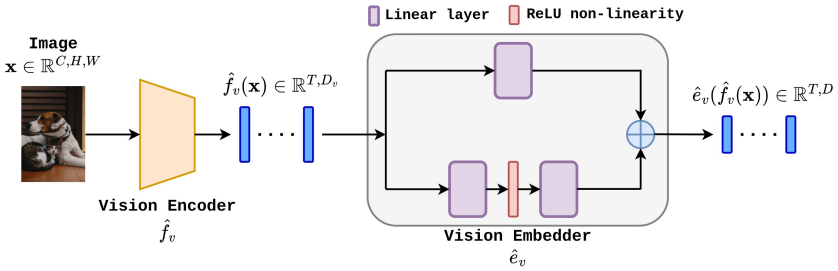


Figure D.1.1: Vision embedder $\hat{e}_v$ architecture for PACL. The image encoder $\hat{f}_v$ produces token/patch-wise representations $\hat{f}_v(\mathbf{x}) \in \mathbb{R}^{T,D_v}$ of an input image $\mathbf{x}$. The vision embedder $\hat{e}_v$ converts the patch-wise representations to the multi-modal shared dimensional space, $\hat{e}_v(\hat{f}_v(\mathbf{x})) \in \mathbb{R}^{T,D}$ where T is the number of tokens or patches.

D.1.3 Training and Evaluation Datasets

Image-text datasets for training: We use primarily 3 different image-text datasets for training all our models. **Firstly,** we use *Google Conceptual Captions (GCC) 3M*, which contains approximately 3 million images, each annotated with a caption. The images are scraped from the web and the corresponding captions are obtained from the Al-text HTML data associated with each image from the web. **Secondly,** we use *Google Conceptual Captions (GCC) 12M*, which is similar to GCC-3M but containing a much larger corpus of image-text pairs with approximately 12 million samples. The primary purpose of GCC-12M is for pre-training whereas GCC-3M is a relatively less noisy dataset meant for fine-tuning pre-trained models. **Thirdly,** we use *YFCC-15M*, a subset of 15 million samples from the popular YFCC-100M [277] dataset, which is one of the largest publicly available datasets containing image-text information obtained from Flickr. The subset of approximately 15 million images is defined by CLIP [254] by filtering images from YFCC-100M with natural language titles and/or descriptions in English.

Semantic segmentation datasets for zero-shot segmentation: We use the following semantic segmentation datasets for zero-shot evaluation on the task of semantic segmentation: **a)** *Pascal VOC* [184]: it has 20 foreground classes and 1 background class with 1449 validation images. We measure performance only on foreground classes and mask predictions with entropy above 1.5 as background, **b)** *Pascal Context* [259]: it has 59 classes with 5k

validation images of indoor and outdoor scenes, **c)** *COCO Stuff* [260]: it has 172 classes categorised into either “thing" classes or “stuff" classes and has 5k validation images, **d)** *ADE20K* [261]: the version we evaluate on is widely used and has 150 classes with 2k validation images. For all datasets, we report the mean intersection over union (mIoU), the most popular evaluation metric for semantic segmentation.

Image classification datasets for zero-shot classification: We evaluate PACL on a suite of 12 image classification datasets which include ImageNet [12], 4 well-known distribution shifts on ImageNet as well as 7 other popular image classification datasets. *ImageNet* is a very popular image classification dataset with 1000 classes relating to concepts contained in the WordNet hierarchy. We use 50000 validation samples in ImageNet for evaluation. The 4 datasets considered to be popular distribution shifts on ImageNet are: **a)** *ImageNet-A* [91] which contains natural real-world images from 200 classes in ImageNet but which are mostly mis-classified by well-known ResNet classifiers, **b)** *ImageNet-R* [86] which contains cartoons, graphics and other art renditions of images from 200 classes in ImageNet, **c)** *ImageNet-Sketch* [87] which contains 50000 validation images, 50 from each of the 1000 ImageNet classes constructed by making the Google search, "sketch of ()" where () is the ImageNet class concerned and **d)** *ImageNet-V2* [242] which has 10000 validation images obtained by following the same collection procedure as ImageNet original images, in order to make the distribution of ImageNet-V2 as similar as possible to ImageNet. The other 7 image classification datasets include: **a)** *CIFAR-10* [89] having 10000 test images from 10 classes including different types of automobiles and animals, **b)** *CIFAR-100* [89] having 10000 test images from 100 classes instead of 10 obtained in a similar fashion as CIFAR-10, **c)** *Stanford Cars* [329] with 8041 test images containing cars of different makes and models, **d)** *Caltech-101* [330] having 101 categories of images with 40-800 images per class, **e)** *Food-101* [331] containing 101 classes of food items organized by the type of food, with approximately 25000 test images, **f)** *Oxford-IIIT Pets* [332], a dataset with 37 categories of pets with approximately 200 images per class, and **g)** *Flower* dataset [333] having 102 different categories of flowers with between 40 and 258 images for each class.

Model	Vision Encoder	Datasets											
		Image Classification								*ImageNet Shifts*			
		ImageNet	*C10*	*C100*	*Cars*	*Caltech101*	*Food101*	*Pets*	*Flowers102*	*ImageNet-A*	*ImageNet-R*	*ImageNet-Sketch*	*ImageNet-V2*
CLIP	VB/16	68.73	91.18	67.88	63.50	85.69	87.52	88.44	61.12	38.88	76.83	48.36	62.21
	ViT-L/14	75.96	95.85	76.94	76.9	86.38	92.69	92.91	69.13	55.44	87.32	59.71	70.26
CLIP + PACL	VB/16	73.61	92.3	69.11	60.7	84.8	89.12	90.1	62.3	42.10	78.1	50.14	65.4
	ViT-L/14	78.2	95.13	74.43	74.2	86.25	93.2	93.05	69.7	59.13	85.6	63.23	72.88

Table D.2.1: Zero-shot Image Classification on 12 different datasets. We compare PACL's performance with vanilla CLIP for both ViT-B/16 and ViT-L/14 encoders. The first 8 datasets are standard image classification datasets: ImageNet, CIFAR-10, CIFAR-100, Stanford Cars, Caltech101, Food101, OxfordIIITPets, and Flowers102. The remaining 4 datasets are standard distribution shifts on ImageNet: ImageNet-A, ImageNet-R, ImageNet-Sketch and ImageNet-V2. PACL + CLIP broadly outperforms vanilla CLIP on most of the classification datasets.

D.2 Additional Results

D.2.1 Semantic Coherence in CLIP

Semantic coherence is a property of ViT based vision encoders where semantically similar regions of the image have similar patch/token level representations in the feature space of the vision encoder. In §6.4 and Figure 6.4, we have shown both quantitative and qualitative results comparing the semantic coherence of a CLIP and a DINO ViT-B/16 vision encoders. Particularly, we had seen that CLIP's ViT-B/16 vision encoder performs better than DINO. In Figure D.2.1, we present additional qualitative examples to further corroborate our observations in §6.4. We show qualitative examples from the Bird, Plane and Sheep classes in Pascal VOC and plot the patch level similarity between a selected patch from the original image (marked using a yellow cross) and all patches from the same image as well as a different image. The similarity is shown using a heatmap where yellow and red shades indicate high similarity and blue shades indicate low similarity. Our observations are similar to the ones in §6.4, and we see that CLIP performs competitively or better than DINO. While DINO seems to cover semantically meaningful regions in the images, it doesn't cover the entirety of the relevant object. CLIP seems to be doing a better job at covering all the patches for the object as highly similar to the marked patch, thereby indicating better semantic coherence.

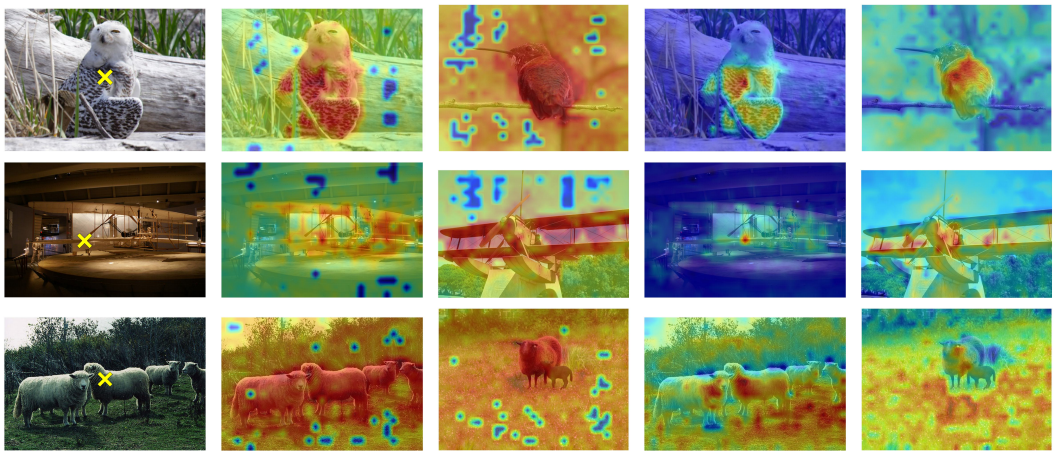


Figure D.2.1: Additional qualitative results on semantic coherence between CLIP and DINO ViT-B/16. a): we show the original image of a class (bird in top row, aeroplane in middle row and sheep in bottom row) with the patch marker (yellow X near the centre). **b, c)**: we show CLIP vision encoder cosine similarity across all patches for the same and a different image of the same class. **d, e)**: we show the same for DINO.

D.2.2 Qualitative Segmentation Results

In Figure 6.6, we showed qualitative results for the task of zero-shot semantic segmentation on both Pascal VOC and ADE20K datasets. In this section, we present more qualitative results on the same. Particularly, in Figure D.2.2, we show additional qualitative results on 8 images from Pascal VOC covering different concepts including bus, cat, dog, bird, potted plant, bottle and plane. Similarly, in Figure D.2.3, we provide qualitative segmentation results from various indoor and outdoor scenes from ADE20K. Similar to our observations in the main paper, we find that the zero-shot segmentation results are decent and our models can recognise a large variety of concepts without ever having been trained on segmentation annotations or masks for any of them. This shows the potential of using the scale of large image-text datasets for zero-shot transfer to semantic segmentation.

D.2.3 Zero-shot Image Classification

In Figure 6.8 of the main paper, we showed the difference in zero-shot classification accuracies for PACL models trained with CLIP backbones as compared to vanilla CLIP

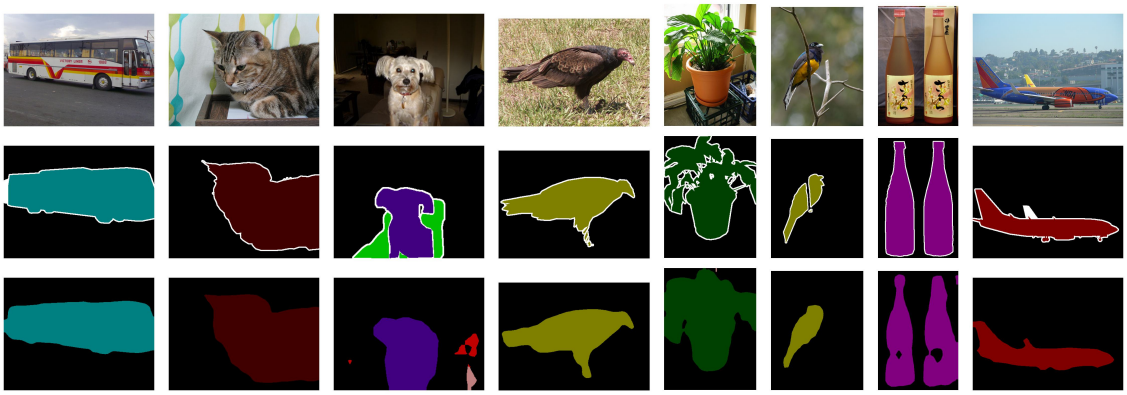


Figure D.2.2: Additional qualitative results on zero-shot semantic segmentation on Pascal VOC. The first row shows the original images, the second row shows the corresponding ground-truth labels and the third row shows the predictions from our best performing model, i.e., PACL trained using a pre-trained CLIP ViT-B/16 encoder on GCC-3M + GCC-12M + YFCC-15M.

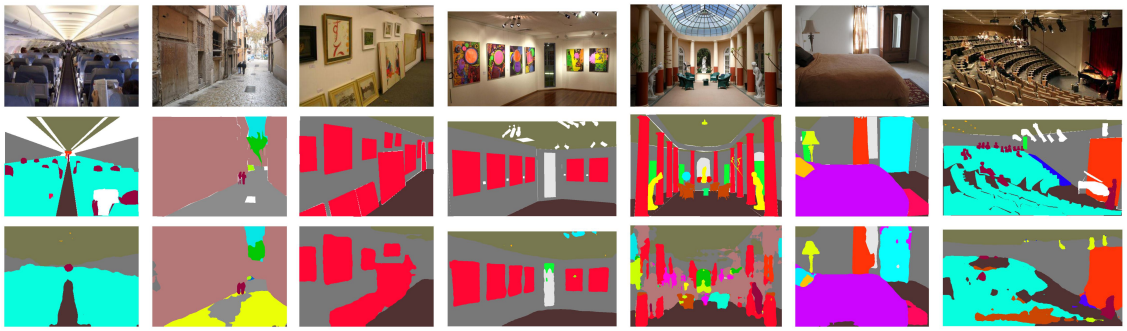


Figure D.2.3: Additional qualitative results on zero-shot semantic segmentation on ADE-20K. The first row shows the original images, the second row shows the corresponding ground-truth labels and the third row shows the predictions from our best performing model, i.e., PACL trained using a pre-trained CLIP ViT-B/16 encoder on GCC-3M + GCC-12M + YFCC-15M.

models on a suite of 12 image classification tasks including ImageNet, 4 datasets considered to be distribution shifts on ImageNet and 7 other well-known image classification datasets. In this section, we provide the exact classification accuracies for all the models on each of the datasets. We present these results in Table D.2.1. As mentioned in the main paper, the PACL models outperform vanilla CLIP on 10 out of 12 datasets for the ViT-B/16 model and 7 out of 12 datasets for the ViT-L/14 backbone, thereby broadly outperforming CLIP on zero-shot image classification.

E

Appendix: The Crippling Effect of Fine-tuning on Foundation Models

Contents

E.1 Training details 229
E.2 Computing $\Phi_\theta(\mathbf{x})$ 230
E.3 Additional results 231
E.3.1 Crippling effect of fine-tuning 231
E.3.2 Analysing L2SP 231
E.3.3 Performance of LDIFS 232
E.3.4 Comparison with Adapters and Prompt Tuning 234
E.3.5 Comparison with Wise-FT 235
E.3.6 Visualizing Table 7.1 235
E.3.7 Generality of Concept Forgetting beyond CLIP ViT-B/32 236

In Appendix E.1, we provide training details for each of our fine-tuning run. In Appendix E.2, we provide details on how to compute the concatenated feature vector $\Phi_\theta(\mathbf{x})$ from a ViT-B/32 CLIP model for LDIFS. In Appendix E.3, we provide additional results on **i)** observing how fine-tuning can lead to concept forgetting, **ii)** our investigations on why LP-init-L2SP is a better baseline compared to others in avoiding concept forgetting and **ii)** performance of our proposed LDIFS regularizer.

E.1 Training details

Training datasets and hyper-parameters: We fine-tune the CLIP ViT-B/32 model on each of the 9 image classification datasets: **a)** Stanford Cars, **b)** CIFAR-10, **c)** CIFAR-100, **d)** DTD, **e)** EuroSAT, **f)** GTSRB, **g)** MNIST, **h)** RESISC45, **i)** SVHN. For each dataset, we have a separate fine-tune run for each of the baselines discussed in §7.2. For each run, we train using the AdamW [334] optimizer, with an initial learning rate of $1e-5$, a weight decay of 0.1 and a cosine learning rate scheduler with a warmup length of 500. For all the runs, we use a batch-size of 128. Following the code of [282], we use the following number of epochs to fine-tune each dataset: **a)** Stanford Cars: 35 epochs, **b)** CIFAR-10/100: 10 epochs, **c)** DTD: 76 epochs, **d)** EuroSAT: 12 epochs, **e)** GTSRB: 11 epochs, **f)** MNIST: 10 epochs, **g)** RESISC45: 15 epochs and **h)** SVHN: 10 epochs. For each dataset, we keep a minimum of 10 epochs for fine-tuning.

Compute and number of experiments: Each of our fine-tuning run is done on a single NVIDIA A100 GPU. As there are a total of 9 classification datasets and a 8 fine-tuning baselines (including our proposed LDIFS), we perform a total of 72 fine-tune training runs from a pre-trained CLIP ViT-B/32. Furthermore, in order to produce the plots capturing $\mathcal{A}_{\text{ZS}}$, $\mathcal{A}_{\text{LP}}$, as well as ℓ_2 distance in parameter and feature space, we store intermediate checkpoints over the course of fine-tuning. Particularly, for each run, we evaluate checkpoints after every 20% of training completion. Finally, in order to obtain the full set of results in Table E.3.3, for the 72 fine-tuned models, we evaluate each of them on 9 datasets (i.e., including the test set of the fine-tuned task), thereby having a total of 648 evaluations. Similar to training, each evaluation is also performed on a single NVIDIA A100 GPU.

Choosing λ_{LDIFS}: One hyper-parameter which the LDIFS regularizer introduces is λ_{LDIFS}. A higher value of λ_{LDIFS} encourages the model to preserve features of the original foundation model and vice versa. For each classification task, we performed a grid search over $\lambda_{\text{LDIFS}} \in \{0.01, 0.05, 0.1, 0.5, 1, 10, 100\}$ and cross-validated this hyper-parameter on a held-out validation set, choosing the value which produces the best performance on the

validation set. We found $\lambda_{\text{LDIFS}} = 10$ to produce the best performance over datasets in general, so all the results we present in this paper are with λ_{LDIFS} set to 10.

E.2 Computing $\Phi_\theta(\mathbf{x})$

In this section, we discuss how we compute the concatenated feature vector $\Phi_\theta(\mathbf{x})$ given input $\mathbf{x}$ and model parameters θ, specifically for a ViT-B/32 model. This is used for computing LDIFS, our proposed regularizer for fine-tuning.

Let the feature output from layer l in the network for input $\mathbf{x}$ be $\Phi_{\theta(l)}(\mathbf{x}) \in \mathbb{R}^l$. Then the normalized feature vector for layer l can be represented as $\frac{\Phi_{\theta(l)}(\mathbf{x})}{||\Phi_{\theta(l)}(\mathbf{x})||}$. In order to form the concatenated feature vector, one can take technically take features from every intermediate layer of the network. However, storing all the features is memory intensive. Thus, we follow a similar approach to the LPIPS [236] implementation for VGG and AlexNet, and choose 5 intermediate points in the ViT-B/32 architecture to collect features from. For a single input image $\mathbf{x} \in \mathbb{R}^{3\times224\times224}$, each of the intermediate feature representations has a dimension of $\mathbb{R}^{50\times768}$ with 50 tokens and 768 dimensional representation for each token. When normalizing the feature vector, we flatten this vector out to a single 38400 dimensional vector. Thus the full concatenated feature vector $\Phi_\theta(\mathbf{x})$ has dimensionality $5 \times 38400 = 192000$. However, note that there can be other ablations to this design and we leave that for future exploration.

LDIFS vs LPIPS [236]: One can find similarities between our proposed LDIFS regulariser and the LPIPS metric [236] used for measuring perceptual similarity between images. However, while LPIPS uses feature space distance on a *pre-trained, frozen* model to find perceptual similarity between pairs or sets of images, LDIFS instead feature space distance between a pair of pre-trained and fine-tuned model for the same image, to preserve the input-output behaviour of the pre-trained model.

E.3 Additional results

In this section, we present additional empirical results to supplement our observations and conclusions in the main paper.

E.3.1 Crippling effect of fine-tuning

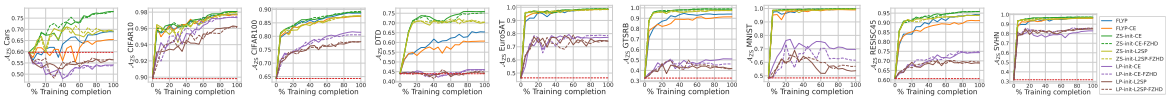
In §7.3, we observed how most existing fine-tuning methods, while gaining state-of-the-art performance on fine-tuned tasks, can lead to concept forgetting in models. In this section, we provide additional empirical results to further strengthen those observations.

In Figure E.3.1 and Figure E.3.8, we present the $\mathcal{A}_{\text{ZS}}$ and $\mathcal{A}_{\text{LP}}$ accuracy for models fine-tuned on each of the 9 classification tasks, on their respective test sets. This shows expected behaviour as models broadly achieve very high test set accuracy, which increases over the course of fine-tuning, on their respective fine-tuned tasks. Note that LP-init baselines seem to have relatively lower performance on $\mathcal{A}_{\text{ZS}}$ accuracy compared to their ZS-init counterparts. However, this reduced performance is only limited to $\mathcal{A}_{\text{ZS}}$ as this does not translate to a reduced performance in $\mathcal{A}_{\text{LP}}$. Moreover, L2SP baselines (both ZS and LP-init) obtain relatively lower fine-tuned accuracy both in case of $\mathcal{A}_{\text{ZS}}$ and $\mathcal{A}_{\text{LP}}$.

In Figure E.3.2 and Figure E.3.9, we present the $\mathcal{A}_{\text{ZS}}$ and $\mathcal{A}_{\text{LP}}$ for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) as captured on 8 classification datasets different from their respective fine-tuned set. Broadly, the ZS and LP performance for all models drops on other datasets as they are fine-tuned, thereby capturing concept forgetting in the fine-tuned models. Among the baselines, we consistently observe LP-init-L2SP to perform better than others in avoiding concept forgetting. This is evident through the distinctly higher $\mathcal{A}_{\text{ZS}}$ and $\mathcal{A}_{\text{LP}}$ accuracies over fine-tuning for the LP-init-L2SP baselines.

E.3.2 Analysing L2SP

In this section, we provide additional results to supplement our observations related to investigating the L2SP baseline in §7.4 of the main paper. In Figure E.3.12, we present the ℓ_2



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)** **(i)**

Figure E.3.1: Test set ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for different fine-tuning methods on 9 image classification tasks: (a) Stanford Cars, (b) CIFAR-10, (c) CIFAR-100, (d) DTD, (e) EuroSAT, (f) GTSRB, (g) MNIST (h) RESISC45 and (i) SVHN. $\mathcal{A}_{\text{ZS}}$ generally rises over the course of fine-tuning. However, for the ZS-init-L2SP and LP-init-L2SP baselines, the gain in $\mathcal{A}_{\text{ZS}}$ is relatively lower. Furthermore, for LP-init baselines, the performance is consistently lower compared to other baselines

distance in parameter space $||\theta - \theta_0||_2^2$ between the pre-trained and current models captured over fine-tuning. For all the datasets, we observe that L2SP baselines while initially diverging slightly in the parameter space, converge back to a model having low ℓ_2 parameter space distance from the original foundation model. Other baselines on the other hand, completely diverge away from the original model in the parameter space.

To further investigate the change in input-output behaviour of the model over fine-tuning, we measure the distance in feature space (see Equation (7.3)) over fine-tuning. In Figure E.3.13, we present the ℓ_2 distance in feature space captured for models fine-tuned on EuroSAT. Again, consistent with our previous observation, we find that unlike other baselines, L2SP first diverges away from the original foundation model and then converges back to the original input-output behaviour, as is indicative through a decreasing L2 feature space distance in the later stages of fine-tuning. This observation is consistent on the EuroSAT train set, the EuroSAT test set as well as on other datasets, thereby providing our motivation for the LDIFS regularizer.

E.3.3 Performance of LDIFS

In this section, we provide additional results to supplement observations related to the performance of the proposed LDIFS regularizer.

Analysing LDIFS on parameter space and feature space distance: In Figure E.3.3 and Figure E.3.4, we plot the ℓ_2 distance in parameter/weight and feature space respectively (same as Figure 7.3 and Figure 7.4 in the main paper), but with the LDIFS baselines added in. In the

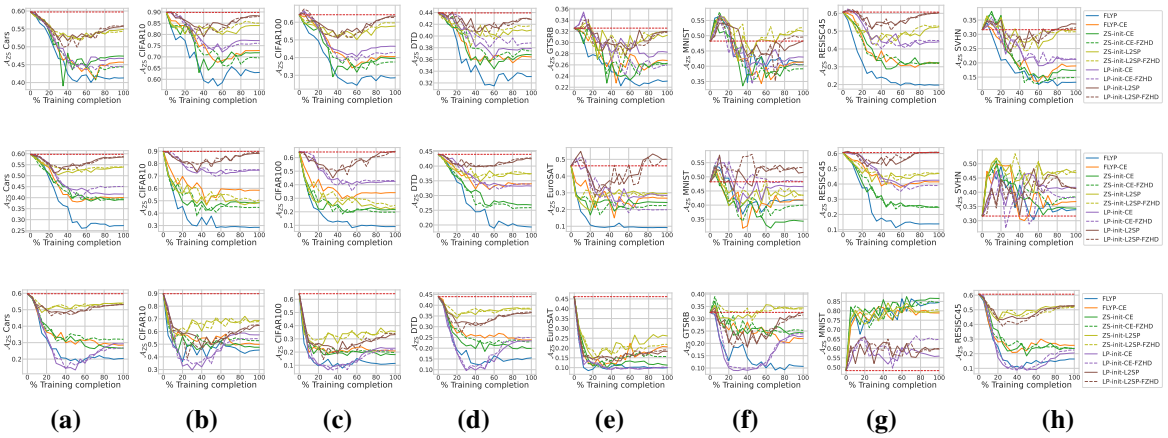


Figure E.3.2: ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for models fine-tuned on EuroSAT (first row), GTSRB (second row), and SVHN (third row) and evaluated on 8 different datasets different from their fine-tuning dataset. Most fine-tuning methods show a drop in $\mathcal{A}_{\text{ZS}}$ performance over the course of fine-tuning indicating a reduction in the model's transferability.

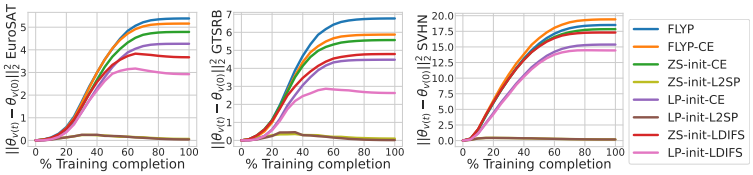


Figure E.3.3: ℓ_2 distance in weight space $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ between the image encoder fine-tuned to the current time-step $f_{\theta_{v(t)}}$ and the pre-trained image encoder $f_{\theta_{v(0)}}$ over the course of fine-tuning.

weight space, LDIFS, while lower than other fine-tuning methods, has a relatively higher ℓ_2 distance from the pre-trained model compared to L2SP. On the contrary, in the feature space, LDIFS consistently gets the lowest distance from the pre-trained model. This is expected and was indeed the purpose behind LDIFS's design which is to minimize the difference in input-output behaviour, captured through feature space distance between the fine-tuned and pre-trained models and not necessarily constrain the parameter space of the fine-tuned model to lie close to the pre-trained model. Furthermore, note from Figure E.3.4, that although we are applying the LDIFS regularizer only on the EuroSAT samples during fine-tuning, the preservation of features extends even to other datasets like CIFAR10, CIFAR100 and GTSRB, as is indicated by the low feature space distance even on these datasets.

Results and observations: Firstly, in Figure E.3.6 and Figure E.3.10, we present the

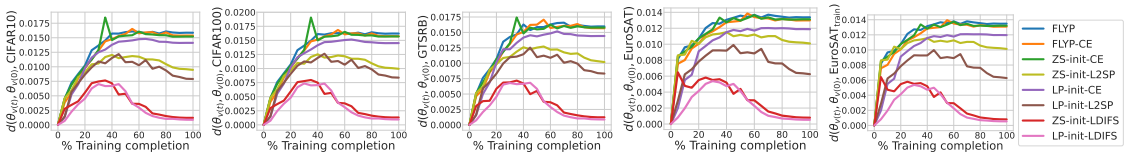


Figure E.3.4: ℓ_2 distance in feature space $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$, between image encoders, $f_{\theta_{v(t)}}$ and $f_{\theta_{v(0)}}$ computed over different fine-tuning methods for models fine-tuned on EuroSAT.

test set accuracy of all fine-tuning methods including LDIFS on all 9 classification tasks. We see that LDIFS performs competitively with all other baselines and improves on L2SP consistently. In Figure E.3.7 and Figure E.3.11, we present the $\mathcal{A}_{\text{ZS}}$ and $\mathcal{A}_{\text{LP}}$ accuracy respectively, for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) over the course of fine-tuning. Furthermore, we also provide the Δ_{ZS}, Δ_{LP} and the $\mathcal{A}_{\text{LP}}$ values for fully fine-tuned models on all 9 classification tasks and all baselines in Table E.3.3.

As is crystal clear from our results, LP-init-LDIFS consistently minimizes concept forgetting without compromising on performance on the fine-tuned task. This is evident from its high $\mathcal{A}_{\text{ZS}}$, $\mathcal{A}_{\text{LP}}$ (see Figure E.3.7 and Figure E.3.11) and high Δ_{ZS} and Δ_{LP} (see Table E.3.3) when evaluated on tasks other than the fine-tuning task at hand. Note from Table E.3.3 that even when LP-init-LDIFS is not achieving the best performance on a certain dataset pair, it is often a close second with very little difference compared to the best performing baseline. In addition to this advantage, its $\mathcal{A}_{\text{LP}}$ accuracy on the fine-tuned task itself is very competitive with the top scores obtained by other fine-tuning baselines and provides a consistent improvement on L2SP.

E.3.4 Comparison with Adapters and Prompt Tuning

In Table E.3.2, we present results of fine-tuning a CLIP ViT-B/32 model using linear probing, CoOp, CLIP-Adapter, Tip-Adapter and full end-to-end fine-tuning using ZS-init-CE loss on 9 downstream image classification tasks. We consistently observe end-to-end fine-tuning to produce the best test set accuracy across all 9 tasks. Furthermore, we also note that in case of tasks where the gap between linear probe performance and end-to-end fine-tuning

performance is significant (e.g. SVHN), adapter and prompt tuning approaches don't cover this performance gap. Hence, if the pre-trained encoder is not performant on a downstream task, the only way to achieve state-of-the-art results on the task requires end-to-end fine-tuning the model on the task itself. This observation thus necessitates study into better end-to-end fine-tuning methods for foundation models.

E.3.5 Comparison with Wise-FT

In this section, we perform an ablation to see the effect of Wise-FT [150] on preventing concept forgetting. Wise-FT is a weight interpolation method which forms a linear combination between the pre-trained θ_0 and fine-tuned θ_f model parameters:

$$\theta_{\text{wse}} = \alpha\theta_0 + (1-\alpha)\theta_f \tag{E.1}$$

Since this can work with any fine-tuned model, we can apply it on all the end-to-end fine-tuning baselines in this work. Specifically, we tune the hyperparameter α on a held-out validation set of the downstream task at hand to maximum validation accuracy. We evaluate concept forgetting on 3 downstream tasks: CIFAR-10, EuroSAT and SVHN using the Δ_{LP} metric and report the results in Table E.3.1. For each dataset, we evaluate the mean Δ_{LP} on 5 other downstream tasks: Cars, DTD, GTSRB, RESISC45 and MNIST. Our observations show a consistent reduction in concept forgetting across all fine-tuning baselines. However, the order of performance between the fine-tuning baselines does not change and LDIFS still maintains superior performance over other baselines both pre and post application of Wise-FT. Thus, the results show that Wise-FT can be combined with any E2E fine-tuning method to further minimize concept forgetting.

E.3.6 Visualizing Table 7.1

In this section, we present the results of Table 7.1 using an alternate visualization. For each of the 9 downstream datasets, we plot the LP accuracy on the dataset itself on the x-axis and the average LP accuracy on all other datasets on the y-axis. The best baselines should lie on the

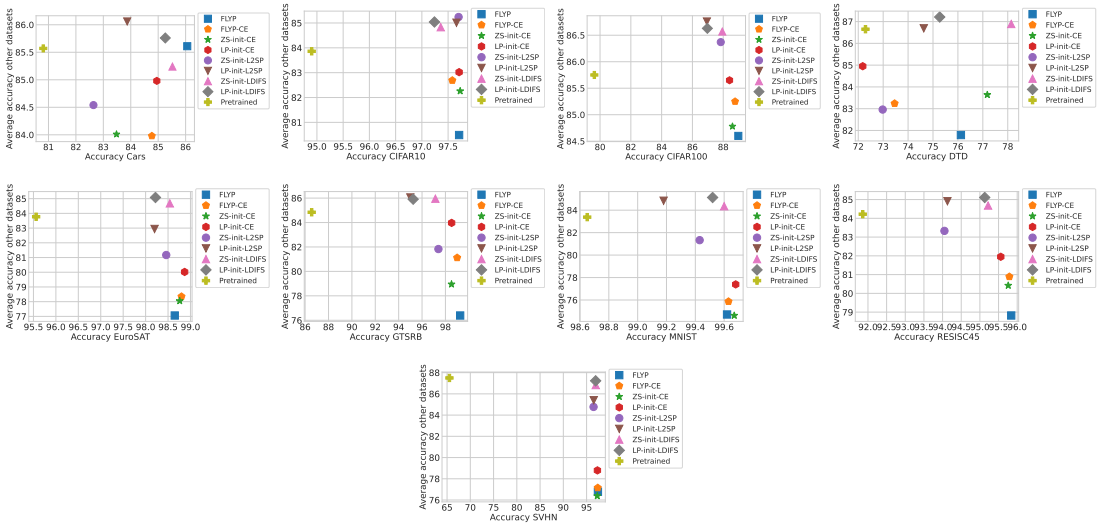


Figure E.3.5: Visualizing test LP accuracy of fine-tuning baselines on different datasets. For each dataset, we plot its test set LP accuracy on the x-axis and the average test set LP accuracy of the remaining datasets on the y-axis.

Dataset	Fine-tuning baselines															
	FLYP		**FLYP-CE**		**ZS-init-CE**		**LP-init-CE (LP-FT)**		**ZS-init-L2SP**		**LP-init-L2SP**		**ZS-init-LDIFS (Ours)**		**LP-init-LDIFS (Ours)**	
	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)	Δ_{LP}	Δ_{LP}(+wse)
CIFAR10	−5.14	−0.11	−1.33	−0.16	−1.67	−0.1	−1.49	−0.01	1.58	1.64	1.49	1.96	1.53	1.99	**1.72**	**2.01**
EuroSAT	−6.5	−1.32	−5.53	−1.04	−5.62	−1.23	−4.08	−0.76	−1.77	−0.12	−0.87	0.08	0.83	1.2	**1.24**	**1.96**
SVHN	−10.82	−6.24	−10.18	−6.37	−10.98	−7.2	−9.2	−4.11	−2.65	−0.66	−2.13	−0.23	−0.57	0.12	**−0.18**	**0.33**

Table E.3.1: Δ_{LP} without and with Wise-FT [150] for all fine-tuning baselines on CIFAR-10, EuroSAT and SVHN.

top right corner of this plot as that indicates that as the model is fine-tuned on the dataset, it still retains performance on other datasets. Results are in Figure E.3.5. Clearly, we consistently see the LDIFS baselines to lie at the top right, again indicative of their performance in not just minimizing concept forgetting but also obtaining excellent downstream task performance.

E.3.7 Generality of Concept Forgetting beyond CLIP ViT-B/32

To study the relevance of concept forgetting beyond a CLIP ViT model, in Table E.3.4 and Table E.3.5, we apply the end-to-end fine-tuning methods to fine-tune a CLIP RN-50 model and a FLAVA [271] ViT-B/16 model respectively on EuroSAT. From both these tables, we observe both concept forgetting to exist and LP-init-LDIFS to minimize it, thereby validating our assumptions on models beyond a CLIP ViT model.

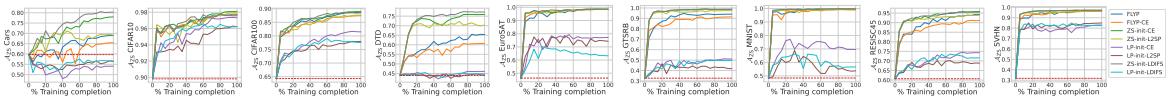
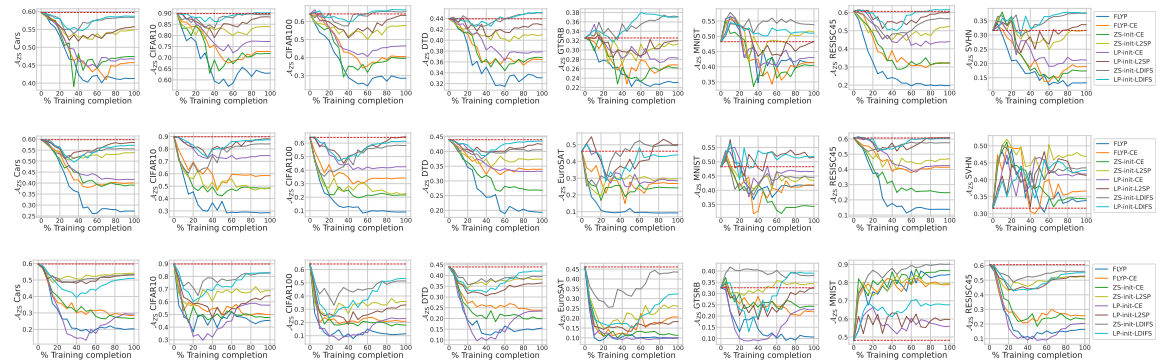
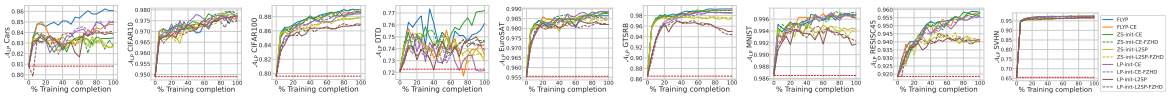


Figure E.3.6: Comparing test set ZS Accuracy $\mathcal{A}_{\text{ZS}}$ for different fine-tuning methods with LDIFS on 9 image classification tasks. While LP-init baselines generally underperform on $\mathcal{A}_{\text{ZS}}$, we find ZS-init-LDIFS to be competitive with the best baselines.



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)**

Figure E.3.7: Comparing ZS Accuracy $\mathcal{A}_{\text{ZS}}$ of different fine-tuning methods with LDIFS for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) and evaluated on 8 datasets different from their fine-tuning dataset. LP-init-LDIFS almost consistently beats all other baselines including LP-init-L2SP in preserving the model's transferability.



(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)** **(i)**

Figure E.3.8: Test set LP Accuracy $\mathcal{A}_{\text{LP}}$ for different fine-tuning methods on 9 image classification tasks: (a) Stanford Cars, (b) CIFAR-10, (c) CIFAR-100, (d) DTD, (e) EuroSAT, (f) GTSRB, (g) MNIST (h) RESISC45 and (i) SVHN. $\mathcal{A}_{\text{LP}}$ generally rises over the course of fine-tuning. However, for the ZS-init-L2SP and LP-init-L2SP baselines, the gain in $\mathcal{A}_{\text{LP}}$ is relatively lower. Furthermore, for LP-init baselines, the performance is consistently lower compared to other baselines

Dataset	LP	CoOp	CLIP-Adapter	Tip-Adapter	E2E Fine-tuning
Cars	80.80	80.88	81.24	81.32	**83.48**
CIFAR10	94.92	95.02	94.87	95.13	**97.73**
CIFAR100	79.62	80.12	80.06	80.82	**88.6**
DTD	72.08	72.01	71.87	72.62	**77.18**
EuroSAT	95.56	95.14	95.38	96.23	**98.76**
GTSRB	86.70	86.81	87.45	88.04	**98.52**
MNIST	98.65	98.84	99.03	99.11	**99.67**
RESISC45	91.86	91.79	91.82	91.80	**95.76**
SVHN	65.47	67.28	69.76	69.23	**97.3**

Table E.3.2: Test set accuracy obtained from linear probing (LP), CoOp, CLIP-Adapter, Tip-Adapter and end-to-end (E2E) fine-tuning using ZS-init-CE loss.

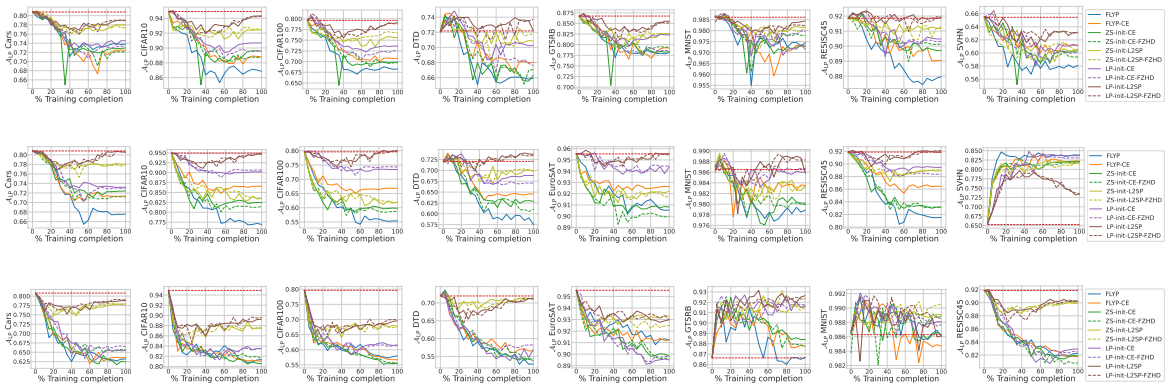


Figure E.3.9: LP Accuracy $\mathcal{A}_{\text{LP}}$ for models fine-tuned on EuroSAT (first row), GTSRB (second row), and SVHN (third row) and evaluated on 8 different datasets different from their fine-tuning dataset. Most fine-tuning methods show a drop in $\mathcal{A}_{\text{LP}}$ performance over the course of fine-tuning indicating a reduction in the model's transferability.

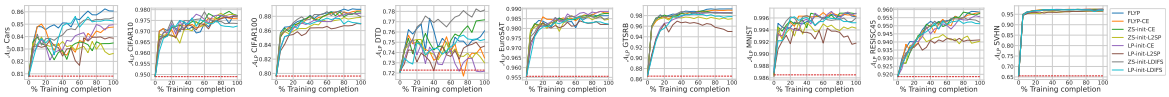


Figure E.3.10: Comparing test set LP Accuracy $\mathcal{A}_{\text{LP}}$ for different fine-tuning methods with LDIFS on 9 image classification tasks.

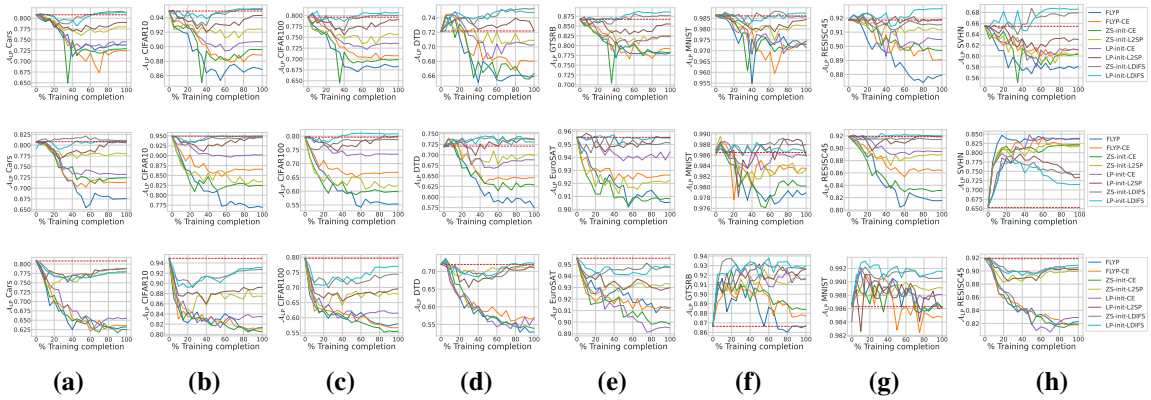


Figure E.3.11: Comparing LP Accuracy $\mathcal{A}_{\text{LP}}$ of different fine-tuning methods with LDIFS for models fine-tuned on EuroSAT (first row), GTSRB (second row) and SVHN (third row) and evaluated on 8 datasets different from their fine-tuning dataset. LP-init-LDIFS almost consistently beats all other baselines including LP-init-L2SP in preserving the model's transferability.

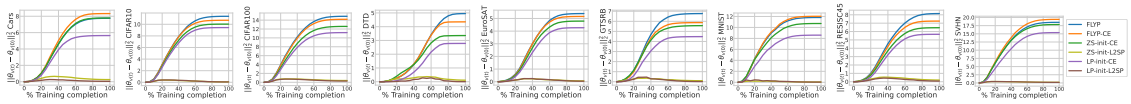


Figure E.3.12: ℓ_2 **distance in weight space** $||\theta - \theta_0||_2^2$ between pre-trained image encoder f_{θ_0} and fine-tuned image encoder f_θ over the course of fine-tuning. Apart from ZS-init-L2SP and LP-init-L2SP, all fine-tuning baselines diverge in weight space over the course of fine-tuning.

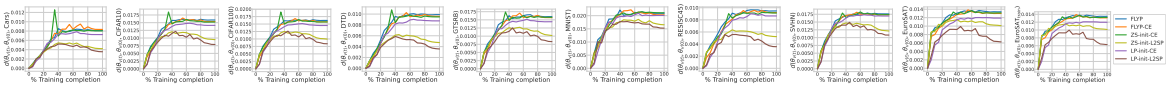


Figure E.3.13: ℓ_2 **distance in feature space** $d(\theta, \theta_0, \mathcal{D})$, between fine-tuned and pre-trained image encoders, f_θ and f_{θ_0} computed over different fine-tuning methods for models fine-tuned on EuroSAT.

Dataset / Fine-tune	Eval	FLYP			FLYP-CE			ZS-init-CE			LP-init-CE (LP-FT)			ZS-init-L2SP			LP-init-L2SP			ZS-init-LDIFS (Ours)			LP-init-LDIFS (Ours)		
		$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$
Cars	CIFAR10	−3.55	−0.68	94.24	−9.0	−1.74	93.18	−9.81	−1.52	93.4	−10.42	−0.93	94.0	−2.78	−0.89	94.02	**−0.26**	**0.06**	**94.97**	−2.98	−0.47	94.45	−2.85	−0.22	94.69
	CIFAR100	−6.91	−0.76	78.89	−7.41	−2.74	76.91	−8.62	−3.58	76.04	−6.84	−2.12	77.51	−3.4	−2.28	77.34	**0.22**	**0.17**	**79.81**	−1.6	−0.37	79.26	−1.76	0.07	79.71
	DTD	−2.61	1.28	73.46	−4.2	−1.44	70.9	−3.99	0.53	72.5	−3.99	2.13	74.26	−3.09	−0.64	71.44	**−1.17**	**2.39**	**74.63**	−1.54	−0.32	71.76	−2.34	0.43	72.55
	EuroSAT	−16.94	−0.2	95.33	−5.89	−0.54	95.0	−6.69	−0.8	94.74	2.74	−0.02	95.52	−4.8	−0.56	94.98	−2.46	**0.02**	**95.56**	1.0	−0.13	95.41	**4.41**	**0.02**	**95.56**
	GTSRB	−3.62	0.55	87.09	−4.28	−1.39	85.24	−1.1	−0.66	85.96	−3.58	−1.01	85.62	2.19	−0.4	86.21	1.85	**0.64**	**87.21**	**2.36**	−0.02	86.56	0.74	0.44	87.18
	MNIST	4.28	−0.19	98.56	2.26	−0.52	98.13	**4.29**	−0.43	98.24	2.21	−0.14	98.48	1.46	−0.34	98.33	2.45	−0.07	98.59	2.33	0.01	98.71	2.15	**0.06**	**98.62**
	RESISC45	−6.19	−1.21	90.67	−7.16	−1.57	90.3	−10.7	−2.13	89.75	−4.81	−0.87	91.0	−4.48	−1.4	90.48	**−0.21**	−0.25	91.62	−4.13	−0.48	91.4	−2.7	**−0.13**	**91.75**
	SVHN	0.33	**1.14**	**66.63**	−5.4	−3.38	62.15	−1.63	−3.91	61.47	−0.7	−2.09	63.49	**2.93**	−2.01	63.51	1.54	0.78	66.08	−0.05	−1.13	64.35	1.03	0.53	66.01
	Cars	9.2	**5.25**	**86.06**	5.73	3.95	84.77	18.24	2.67	83.48	−5.62	4.14	84.95	9.44	1.83	82.64	−3.31	3.06	83.87	**20.52**	4.71	85.52	−3.15	4.45	85.26
CIFAR10	Cars	−58.0	−14.58	66.24	−45.32	−7.04	73.75	−46.08	−5.68	75.12	−36.86	−8.27	72.53	−15.81	0.52	81.32	−4.19	−0.55	80.26	−13.78	**0.53**	**81.33**	**−3.95**	−1.31	79.51
	CIFAR100	−19.22	0.97	80.62	−0.33	1.86	81.52	−2.39	1.86	81.48	0.23	2.85	82.46	8.79	4.81	84.45	**8.97**	**5.22**	**84.85**	3.08	0.28	79.92	3.31	2.12	81.74
	DTD	−14.31	−7.93	64.15	−9.52	−4.52	67.71	−9.73	−6.06	66.01	−9.52	−4.36	67.77	−1.86	**1.38**	**73.46**	−2.77	−0.74	71.33	−0.8	0.27	72.34	**−0.69**	0.05	72.18
	EuroSAT	−16.93	−0.07	95.46	−14.76	−0.3	95.24	−18.06	−0.19	95.35	−17.04	−0.33	95.2	**4.2**	−0.07	95.46	−1.09	0.44	95.98	0.17	0.07	95.61	2.83	**0.46**	**96.0**
	GTSRB	−2.72	−1.24	85.3	−0.8	0.35	87.05	−2.16	−0.48	86.25	−0.33	1.03	87.63	1.63	1.15	87.82	−0.87	1.58	88.23	**5.81**	2.94	89.45	5.38	**3.21**	**89.99**
	MNIST	−9.31	−0.45	98.23	−10.42	−0.07	98.59	−12.21	−0.1	98.54	−4.02	**0.22**	**98.73**	**2.74**	−0.1	98.55	2.25	−0.13	98.52	−0.4	0.04	98.62	0.21	−0.17	98.49
	RESISC45	−26.32	−4.83	87.05	−14.44	−2.54	89.33	−15.49	−2.51	89.37	−16.32	−1.97	89.9	−0.29	−0.25	91.62	−1.49	−0.1	91.78	0.25	**0.17**	**92.05**	**1.03**	0.13	92.0
	SVHN	−3.49	1.36	66.84	−7.19	2.9	68.33	−5.45	0.4	65.98	0.62	4.34	69.95	7.15	3.71	69.22	7.08	3.59	69.13	6.17	3.97	69.34	**8.13**	**4.92**	**70.4**
	CIFAR10	7.87	2.79	97.71	7.8	2.65	97.58	**8.25**	**2.83**	**97.73**	7.5	2.79	97.71	8.05	2.8	97.7	6.38	2.74	97.66	7.72	2.46	97.36	6.39	2.33	97.24
CIFAR100	Cars	−27.76	−3.15	77.66	−12.57	−3.36	77.44	−21.22	−3.58	77.23	−12.42	−4.38	76.42	−6.74	−0.12	80.69	**−1.32**	**0.15**	**80.96**	−12.29	−0.72	80.08	−6.32	−1.07	79.73
	CIFAR10	−8.17	1.05	95.96	0.22	1.33	96.25	−0.64	1.5	96.41	1.06	1.52	96.42	2.65	**1.96**	**96.87**	**3.24**	1.62	96.53	0.33	1.34	96.25	0.43	0.88	95.79
	DTD	−10.37	−6.44	65.64	−8.83	−5.53	66.54	−9.36	−5.11	66.97	−6.54	−3.3	68.83	−4.52	−0.74	71.33	−2.39	**0.16**	**72.23**	−4.41	−1.44	70.64	**−1.97**	−1.97	70.11
	EuroSAT	−23.19	−0.06	95.48	−12.96	−0.26	95.28	−16.59	−0.48	95.06	−7.11	−0.17	95.37	**−0.46**	0.19	95.72	−7.13	0.46	96.0	−3.33	0.26	95.8	−2.19	**0.56**	**96.09**
	GTSRB	−6.46	0.17	86.79	−0.43	2.52	89.11	−4.64	0.99	87.62	2.98	2.69	89.3	1.27	1.8	88.42	3.2	2.65	89.18	4.88	2.98	89.65	**6.08**	**4.04**	**90.7**
	MNIST	−8.99	−0.35	98.36	−7.26	−0.15	98.51	−1.37	−0.3	98.36	−6.36	0.11	98.79	3.09	0.08	98.52	2.85	0.04	98.75	**8.1**	−0.07	98.68	0.25	**0.2**	**98.95**
	RESISC45	−20.65	−3.54	88.33	−16.51	−3.56	88.32	−17.65	−3.3	88.57	−14.14	−1.95	89.92	−5.78	−1.3	90.59	**−2.81**	−0.57	91.3	−7.68	−1.1	90.78	−6.89	**−0.44**	**91.43**
	SVHN	2.46	3.06	68.59	6.05	5.04	70.58	2.43	2.57	68.03	5.7	4.64	70.18	7.89	3.3	68.84	6.98	3.7	69.12	6.02	**5.32**	**70.67**	**8.24**	4.68	70.21
	CIFAR100	24.4	9.34	88.98	23.16	9.12	88.77	**24.63**	8.97	88.6	17.26	8.78	88.41	23.33	8.2	87.84	13.77	7.31	86.94	24.24	8.31	87.94	13.3	**9.35**	**88.99**
DTD	Cars	−18.09	−3.36	77.44	−10.19	−2.87	77.94	−9.85	−2.54	78.25	−5.97	−2.43	78.36	−11.63	−4.15	76.66	**−0.06**	0.21	81.01	−1.94	**0.77**	**81.58**	−0.2	0.36	81.17
	CIFAR10	−36.94	−6.55	88.37	−13.28	−4.32	90.59	−13.4	−4.74	90.17	−9.33	−1.81	93.11	−12.64	−4.99	89.92	**0.57**	−0.44	94.48	−0.76	−0.22	94.7	0.44	**0.21**	**95.12**
	CIFAR100	−38.48	−10.03	69.61	−18.82	−7.08	72.57	−20.99	−8.23	71.41	−8.12	−3.3	76.34	−17.73	−8.45	71.18	**2.39**	0.8	80.45	−3.41	−0.72	78.93	1.61	**0.84**	**80.49**
	EuroSAT	−32.67	−0.57	94.96	−20.46	−1.54	94.0	−12.56	−0.48	95.06	−11.02	−0.52	95.02	−22.26	−1.33	94.2	**4.87**	0.15	95.69	−7.19	**0.43**	**95.96**	2.76	0.0	95.54
	GTSRB	−18.13	−7.02	79.69	−7.89	−3.52	83.15	−13.89	−2.21	84.51	−4.41	−1.61	85.01	−5.76	−2.62	83.97	0.34	−0.39	86.18	**2.11**	0.69	87.32	1.43	**1.3**	**87.93**
	MNIST	−24.64	−0.95	97.75	3.59	−0.51	98.16	−7.74	0.02	98.35	−5.74	−0.32	98.34	−7.92	−0.14	98.49	**5.03**	−0.02	98.69	−2.84	**0.17**	**98.86**	2.6	0.15	98.84
	RESISC45	−34.51	−5.0	86.87	−11.0	−2.52	89.37	−13.37	−2.65	89.22	−10.52	−1.48	90.4	−19.27	−4.48	87.38	0.33	−0.13	91.75	−1.86	−0.38	91.49	**1.21**	**0.14**	**92.02**
	SVHN	−17.89	−5.9	59.64	−8.6	−5.14	60.18	0.31	−3.28	62.14	−3.49	−2.61	63.03	0.55	−3.54	61.9	1.84	−0.11	65.3	**7.88**	0.76	66.31	3.81	**1.23**	**66.69**
	DTD	21.44	4.04	76.12	16.7	1.33	73.46	31.91	4.89	77.18	2.13	0.05	72.18	26.38	0.9	72.98	0.16	2.45	74.63	**33.14**	**6.12**	**78.14**	0.74	3.19	75.27
EuroSAT	Cars	−18.44	−7.0	73.8	−14.04	−8.34	72.45	−12.34	−7.85	72.95	−13.0	−6.27	74.53	−4.89	−2.79	78.01	−3.95	−1.84	78.97	−1.37	0.42	81.22	**−1.08**	**0.53**	**81.33**
	CIFAR10	−26.75	−8.04	86.86	−17.0	−6.07	88.84	−17.99	−5.32	89.59	−12.57	−4.24	90.68	−5.87	−2.49	92.41	−1.59	−0.6	94.31	−0.53	0.23	95.13	**0.39**	**0.34**	**95.25**
	CIFAR100	−35.66	−11.37	68.26	−23.68	−8.87	70.76	−24.64	−9.78	69.85	−17.79	−6.03	73.59	−6.43	−4.0	75.62	−0.83	−0.56	79.08	1.23	0.46	80.09	**2.27**	**1.06**	**80.7**
	DTD	−10.85	−6.17	65.9	−7.5	−4.04	68.03	−7.13	−5.85	66.33	−6.12	−1.81	70.27	−2.93	−1.22	70.85	−1.17	0.05	72.23	1.01	**3.19**	**75.27**	**1.12**	2.66	74.73
	GTSRB	−9.49	−8.53	78.14	−5.78	−7.54	79.16	−6.39	−8.39	78.35	−4.29	−4.35	82.31	−1.27	−4.05	82.6	−0.58	−1.2	85.53	**4.56**	0.94	87.67	4.51	**1.98**	**88.6**
	MNIST	−6.85	−1.2	97.45	−6.92	−1.35	97.32	−7.76	−1.32	97.29	−5.24	−1.48	97.14	3.21	−0.54	98.16	0.04	−0.2	98.44	**5.54**	−0.07	98.57	2.82	**0.13**	**98.74**
	RESISC45	−40.95	−3.89	87.97	−28.6	−2.83	89.05	−28.44	−2.16	89.7	−16.65	−1.4	90.48	−8.29	−0.7	91.16	−0.46	0.05	91.9	−4.08	−0.33	91.54	**0.9**	**0.79**	**92.67**
	SVHN	−18.42	−7.38	58.09	−12.7	−4.44	61.17	−14.24	−5.12	60.32	−10.42	−4.45	61.13	0.07	−4.86	60.56	1.99	−2.51	63.06	**6.08**	2.5	67.92	6.07	**3.03**	**68.59**
	EuroSAT	52.11	3.09	98.65	51.96	3.26	98.8	**52.37**	3.2	98.76	30.93	**3.31**	**98.87**	52.19	2.91	98.46	28.04	2.67	98.2	52.15	2.98	98.54	17.02	2.67	98.22
GTSRB	Cars	−32.45	−13.23	67.57	−19.66	−9.45	71.36	−20.69	−8.39	72.42	−18.07	−7.61	73.2	−5.77	−2.75	78.05	−1.24	−0.25	80.55	−2.66	0.31	80.94	**−0.88**	**1.82**	**81.11**
	CIFAR10	−61.3	−17.99	76.92	−31.36	−8.35	86.57	−41.57	−12.51	82.4	−15.19	−4.71	90.21	−60.89	−11.41	83.51	−1.77	−0.2	94.72	−1.71	−0.38	94.53	**−0.14**	**0.0**	**94.92**
	CIFAR100	−55.13	−24.28	55.36	−30.12	−12.83	66.83	−42.57	−19.81	59.82	−21.7	−6.2	73.43	−41.4	−17.66	61.97	0.46	0.39	80.01	−2.16	−0.91	78.73	**1.81**	**1.21**	**80.85**
	DTD	−24.68	−14.63	57.45	−10.0	−7.5	64.57	−17.18	−9.2	62.87	−10.74	−3.51	68.83	−6.6	−2.13	69.84	−1.54	**1.76**	**73.72**	−2.55	1.65	73.62	**−0.32**	1.17	73.03
	EuroSAT	−36.94	−5.0	90.56	−19.31	−2.91	92.63	−21.91	−4.72	90.81	−17.7	−1.11	94.43	−16.5	−3.31	92.22	**3.94**	**0.0**	**95.54**	1.17	−0.43	95.11	2.57	−0.06	95.48
	MNIST	−6.53	−0.81	97.89	−6.43	−0.25	98.35	−13.95	−0.63	98.02	−1.72	−0.09	98.6	−4.72	−0.39	98.33	3.33	−0.06	98.58	−2.23	**0.13**	**98.8**	**6.78**	0.02	98.66
	RESISC45	−46.86	−10.35	81.52	−19.24	−5.44	86.43	−35.92	−8.73	83.14	−18.17	−2.43	89.44	−13.68	−2.92	88.95	0.1	−0.02	91.86	−2.6	−0.48	91.4	**0.32**	0.11	91.98
	SVHN	2.22	**18.35**	**83.82**	5.07	16.66	82.25	2.74	16.81	82.08	10.11	18.12	83.65	**15.22**	16.15	81.75	9.68	7.79	73.39	10.08	8.67	74.17	7.94	5.9	71.45
	GTSRB	61.47	**12.64**	**99.26**	58.8	12.4	99.0	66.1	11.96	98.52	18.42	11.9	98.53	64.74	10.8	97.4	9.19	8.27	95.0	**66.27**	12.03	98.45	19.33	11.2	97.81
MNIST	Cars	−29.96	−17.16	63.64	−29.3	−16.44	64.38	−29.55	−18.58	62.24	−44.11	−15.91	64.92	−4.56	−2.18	78.65	**−2.47**	**−0.41**	**80.38**	−3.89	−0.9	79.93	−3.25	−0.58	80.23
	CIFAR10	−59.44	−15.91	79.0	−39.13	−13.25	81.66	−58.04	−15.62	79.29	−50.68	−10.78	84.14	−15.31	−5.61	89.3	**−1.19**	**0.32**	**95.23**	−1.49	−0.93	93.98	−3.11	−0.26	94.66
	CIFAR100	−56.3	−22.92	56.72	−42.41	−18.2	61.44	−50.72	−21.05	58.57	−55.09	−16.71	62.94	−23.77	−9.44	70.19	−5.18	0.11	79.76	−5.64	−1.26	78.37	**−4.43**	**0.46**	**80.11**
	DTD	−28.03	−7.5	64.41	−15.21	−10.59	61.49	−21.91	−9.95	62.13	−35.59	−11.81	60.37	−3.14	−0.32	71.7	−3.3	1.55	73.57	−4.95	0.11	72.18	**−0.37**	**1.6**	**73.62**
	EuroSAT	−32.02	−3.09	92.44	−34.48	−3.44	92.09	−27.09	−2.98	92.56	−34.72	−1.94	93.59	**−2.46**	−0.72	94.81	−4.15	−0.17	95.37	**−2.46**	**0.3**	**95.83**	−12.63	0.02	95.56
	GTSRB	−25.71	−6.02	80.59	−15.46	−4.22	82.45	−20.58	−6.25	80.39	−23.1	−0.02	86.63	−5.57	−2.45	84.19	1.26	2.77	89.33	−0.91	1.85	88.47	**3.71**	**2.89**	**89.49**
	RESISC45	−42.9	−6.68	85.19	−39.11	−7.29	84.59	−37.94	−7.05	84.84	−51.0	−5.94	85.94	−5.46	−1.56	90.32	**−1.68**	0.24	92.11	−4.68	−0.06	91.81	−3.19	**0.26**	**92.13**
	SVHN	17.26	10.15	75.71	16.11	13.21	78.82	15.42	11.42	76.94	13.47	14.99	80.57	12.9	6.06	71.45	12.41	7.51	72.85	**22.94**	15.31	80.84	19.95	**16.76**	**82.31**
	MNIST	51.19	0.88	99.62	50.56	0.98	99.63	**51.28**	**1.02**	**99.67**	21.34	0.98	99.68	51.02	0.8	99.43	5.51	0.5	99.18	51.18	0.98	99.6	8.35	0.83	99.52
RESISC45	Cars	−17.17	−5.3	75.51	−10.88	−6.02	74.78	−12.42	−4.5	76.3	−11.42	−6.12	74.7	−4.32	−0.21	80.6	−0.68	0.49	81.28	−1.17	0.56	81.37	**0.26**	**0.66**	**81.47**
	CIFAR10	−33.88	−8.43	86.49	−15.74	−5.1	89.81	−18.47	−5.84	89.08	−10.76	−3.44	91.47	−7.58	−2.04	92.88	−0.89	0.0	94.91	−0.6	−0.3	94.61	**−0.2**	**0.13**	**95.04**
	CIFAR100	−35.9	−11.43	68.19	−17.78	−6.75	72.89	−23.07	−8.85	70.78	−12.54	−5.25	74.38	−7.25	−3.25	76.39	1.55	0.57	80.19	0.37	−0.43	79.2	**1.57**	**0.63**	**80.26**
	DTD	−12.66	−4.47	67.77	−6.44	−2.39	69.52	−8.56	−3.24	68.83	−3.51	−0.48	71.6	−3.4	−0.74	71.33	−0.59	2.07	74.15	−1.81	1.01	73.4	**−0.43**	**2.98**	**74.79**
	EuroSAT	2.5	1.06	96.61	8.85	1.07	96.61	5.22	0.74	96.28	9.15	1.07	96.61	12.3	1.13	96.67	13.11	**1.26**	**96.8**	**13.74**	0.91	96.44	12.76	0.76	96.3
	GTSRB	−9.58	−6.32	80.42	−6.98	−3.82	82.76	−11.79	−4.01	82.62	−4.41	−1.08	85.51	−1.76	−1.06	85.66	0.1	0.34	86.95	−2.09	0.81	87.51	**1.96**	**0.85**	**87.55**
	MNIST	−15.45	−0.85	97.75	−7.76	−0.37	98.33	−8.0	−0.07	98.57	−0.94	−0.23	98.44	2.03	−0.19	98.49	2.19	−0.12	98.58	4.04	−0.35	98.37	**5.47**	**−0.1**	**98.6**
	SVHN	−21.2	−7.58	57.82	−11.01	−3.14	62.38	−8.6	−4.57	60.93	−4.06	−2.6	62.88	3.55	−0.88	64.61	**4.9**	0.68	66.23	4.16	1.04	66.39	2.9	**1.27**	**66.89**
	RESISC45	33.49	**3.98**	**95.84**	30.4	3.94	95.79	**35.1**	3.9	95.76	13.65	3.68	95.56	33.14	2.19	94.05	8.14	2.25	94.13	34.76	3.37	95.22	10.92	3.25	95.13
SVHN	Cars	−39.37	−17.56	63.23	−31.22	−17.22	63.59	−33.06	−18.38	62.43	−30.44	−15.28	65.51	**−5.66**	−2.92	77.9	−6.63	**−2.0**	**78.8**	−6.5	−2.04	78.76	−8.73	−2.89	77.91
	CIFAR10	−44.46	−13.57	81.34	−39.98	−13.84	81.06	−42.24	−14.41	80.5	−32.49	−11.44	83.49	−21.11	−7.47	87.45	−24.95	−5.66	89.25	−7.25	−2.16	92.75	**−6.83**	**−1.86**	**93.06**
	CIFAR100	−52.73	−21.64	58.0	−43.38	−22.93	56.72	−46.16	−24.14	55.51	−41.28	−18.07	61.59	−28.52	−11.84	67.8	−30.95	−10.09	69.56	−12.88	−5.26	74.38	**−11.03**	**−2.73**	**76.9**
	DTD	−28.51	−19.2	52.87	−20.11	−15.48	56.6	−24.1	−18.09	53.88	−20.48	−15.11	56.97	−5.8	−1.06	71.17	−7.45	−0.9	71.12	−4.47	−0.48	71.76	**−1.86**	**0.21**	**72.29**
	EuroSAT	−36.17	−4.31	91.22	−25.59	−4.3	91.24	−34.87	−5.7	89.83	−36.09	−6.04	89.5	−19.98	−2.26	93.28	−27.78	−2.56	92.98	**−2.61**	**−0.78**	**94.76**	−13.74	**−0.78**	**94.76**
	GTSRB	−21.91	0.0	86.68	−10.66	0.95	87.72	−8.27	1.78	88.41	−9.62	4.95	91.57	1.8	5.04	91.7	−0.06	6.03	92.64	5.49	6.03	92.66	**6.45**	**6.23**	**92.9**
	MNIST	36.19	−0.04	98.61	30.61	−0.19	98.47	38.35	0.0	98.63	7.66	0.15	98.81	31.19	0.23	98.91	11.39	−0.06	98.61	**41.91**	**0.66**	**99.15**	19.55	0.49	99.06
	RESISC45	−44.29	−9.59	82.29	−34.94	−10.1	81.78	−37.16	−10.0	81.87	−40.17	−8.97	82.9	−8.44	−1.95	89.92	−7.84	−1.65	90.22	**−4.56**	−1.38	90.49	−5.56	**−1.0**	**90.87**
	SVHN	65.0	31.92	97.44	64.38	31.86	97.4	**65.58**	31.79	97.3	51.28	**31.98**	**97.5**	64.84	31.02	96.5	53.39	31.04	96.54	65.25	31.36	96.97	50.92	31.38	96.95

Table E.3.3: Δ_{ZS}, Δ_{LP} and $\mathcal{A}_{LP}$ for models fully fine-tuned on 9 different classification datasets. LP-init-LDIFS outperforms other fine-tuning baselines and even LP-init-L2SP in minimizing concept forgetting, while also outperforming LP-init-L2SP on the fine-tuned task performance.

Dataset / Fine-tune	Eval	FLYP			FLYP-CE			ZS-init-CE			LP-init-CE (LP-FT)			ZS-init-L2SP			LP-init-L2SP			ZS-init-LDIFS (Ours)			LP-init-LDIFS (Ours)		
		$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{ZS}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$
EuroSAT	CIFAR10	−57.93	−23.21	65.15	−49.37	−23.02	65.46	−53.46	−23.74	65.01	−46.95	−22.55	66.13	−40.33	−14.04	74.4	−31.21	−8.9	79.55	−20.3	−8.21	80.15	**−14.1**	**−3.67**	**84.69**
	SVHN	−20.75	−19.91	47.77	−14.33	−24.64	43.08	−14.14	−22.23	45.54	−18.34	−16.54	51.14	−2.63	−12.81	54.91	−3.17	−6.92	60.73	**0.14**	−7.23	60.45	0.06	**−2.17**	**65.51**
	EuroSAT	56.46	**3.04**	**98**	55.29	3.09	97.76	**56.74**	2.83	97.52	3.74	2.81	97.80	55.78	1.56	96.46	−11.91	2.89	96.70	56.31	3.01	97.97	−1.6	3.02	97.98

Table E.3.4: Δ_{ZS}, Δ_{LP} and $\mathcal{A}_{LP}$ for CLIP RN50 fully fine-tuned on EuroSAT.

Dataset / Fine-tune	Eval	FLYP		FLYP-CE		ZS-init-CE		LP-init-CE (LP-FT)		ZS-init-L2SP		LP-init-L2SP		ZS-init-LDIFS (Ours)		LP-init-LDIFS (Ours)	
		$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$	$\Delta_{LP}(\uparrow)$	$\mathcal{A}_{LP}(\uparrow)$
EuroSAT	CIFAR10	−9.44	83.43	−8.13	84.74	−6.01	86.86	−4.76	88.11	−2.32	90.55	−0.11	92.76	0.07	92.8	**0.16**	**93.03**
	SVHN	−8.92	54.24	−8.16	55.0	−6.63	56.53	−5.10	58.06	−1.87	61.29	−1.21	61.95	1.37	64.53	**2.68**	**65.84**
	EuroSAT	1.63	98.89	**1.65**	**98.91**	1.41	98.67	1.53	98.79	0.95	98.21	1.42	98.68	1.47	98.73	1.6	98.86

Table E.3.5: Δ_{LP} and $\mathcal{A}_{LP}$ for FLAVA ViT-B/16 fully fine-tuned on EuroSAT.

Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution.

— Albert Einstein

References

[1] Richard L Hunter. *The Argonautica of Apollonius*. Cambridge University Press, 2004 (cit. on p. 2).

[2] B Jack Copeland. *The essential turing*. Clarendon Press, 2004 (cit. on p. 2).

[3] Alan Mathison Turing et al. "On computable numbers, with an application to the Entscheidungsproblem". In: *J. of Math* 58.345-363 (1936), p. 5 (cit. on p. 2).

[4] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133 (cit. on p. 2).

[5] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009 (cit. on p. 2).

[6] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957 (cit. on p. 2).

[7] Seppo Linnainmaa. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors". PhD thesis. Master's Thesis (in Finnish), Univ. Helsinki, 1970 (cit. on p. 2).

[8] Paul Werbos. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math. Harvard University". PhD thesis. Jan. 1974 (cit. on p. 2).

[9] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536 (cit. on pp. 2, 16).

[10] Rina Dechter. "Learning while searching in constraint-satisfaction problems". In: (1986) (cit. on p. 2).

[11] Igor Aizenberg, Naum N Aizenberg, and Joos PL Vandewalle. *Multi-valued and universal binary neurons: Theory, learning and applications*. Springer Science & Business Media, 2013 (cit. on p. 2).

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on pp. 3, 21, 24, 42, 63, 76, 81, 114, 126, 161, 224).

[13] Jared Kaplan et al. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020) (cit. on pp. 3, 25).

[14] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 3, 24, 53, 76, 81, 85, 126, 143, 190, 191).

[15] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778 (cit. on pp. 3, 14, 24, 42, 56, 63, 70, 85, 142, 190, 191).

[16] Ashish Vaswani et al. “Attention Is All You Need”. In: *Conference on Neural Information Processing Systems*. 2017 (cit. on pp. 3, 14, 24, 173).

[17] Junyi Chai et al. “Deep learning in computer vision: A critical review of emerging techniques and application scenarios”. In: *Machine Learning with Applications* 6 (2021), p. 100134 (cit. on p. 3).

[18] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018 (cit. on p. 3).

[19] Kuniaki Noda et al. “Audio-visual speech recognition using deep learning”. In: *Applied intelligence* 42 (2015), pp. 722–737 (cit. on p. 3).

[20] Kai Arulkumaran et al. “Deep reinforcement learning: A brief survey”. In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38 (cit. on p. 3).

[21] Rohan Gupta et al. “Artificial intelligence to deep learning: machine intelligence approach for drug discovery”. In: *Molecular diversity* 25 (2021), pp. 1315–1360 (cit. on p. 3).

[22] Shuai Zhang et al. “Deep learning based recommender system: A survey and new perspectives”. In: *ACM computing surveys (CSUR)* 52.1 (2019), pp. 1–38 (cit. on p. 3).

[23] Rishi Bommasani et al. “On the opportunities and risks of foundation models”. In: *arXiv preprint arXiv:2108.07258* (2021) (cit. on pp. 3, 4, 7, 25, 118).

[24] Ben Goertzel and Cassio Pennachin. *Artificial general intelligence*. Vol. 2. Springer, 2007 (cit. on p. 3).

[25] Yoshua Bengio et al. “Managing ai risks in an era of rapid progress”. In: *arXiv preprint arXiv:2310.17688* (2023) (cit. on pp. 3, 149).

[26] Jindong Wang and Yiqiang Chen. “Pre-Training and Fine-Tuning”. In: *Introduction to Transfer Learning: Algorithms and Practice*. Springer, 2022, pp. 125–140 (cit. on p. 3).

[27] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. “What makes ImageNet good for transfer learning?” In: *arXiv preprint arXiv:1608.08614* (2016) (cit. on p. 3).

[28] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on pp. 4, 24).

[29] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660 (cit. on pp. 4, 24, 97, 98, 100, 112).

[30] Zhenda Xie et al. “Simmim: A simple framework for masked image modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9653–9663 (cit. on p. 4).

[31] Kaiming He et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009 (cit. on pp. 4, 24).

[32] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 4, 24).

[33] Christoph Schuhmann et al. “Laion-400m: Open dataset of clip-filtered 400 million image-text pairs”. In: *arXiv preprint arXiv:2111.02114* (2021) (cit. on pp. 4, 25).

[34] Together Computer. *RedPajama: an Open Dataset for Training Large Language Models*. 2023. URL: https://github.com/togethercomputer/RedPajama-Data (cit. on pp. 4, 25).

[35] Jason Wei et al. “Emergent abilities of large language models”. In: *arXiv preprint arXiv:2206.07682* (2022) (cit. on p. 4).

[36] Lingxi Xie et al. “Image classification and retrieval are one”. In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. 2015, pp. 3–10 (cit. on p. 5).

[37] Zhong-Qiu Zhao et al. “Object detection with deep learning: A review”. In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232 (cit. on pp. 5, 27).

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. June 2015, pp. 3431–3440 (cit. on pp. 5, 6, 27, 65, 100).

[39] Hamid Rezatofighi et al. “Generalized intersection over union: A metric and a loss for bounding box regression”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 658–666 (cit. on p. 5).

[40] Qingxiu Dong et al. “A survey on in-context learning”. In: *arXiv preprint arXiv:2301.00234* (2022) (cit. on p. 5).

[41] Jason Wei et al. “Finetuned language models are zero-shot learners”. In: *arXiv preprint arXiv:2109.01652* (2021) (cit. on pp. 5, 148).

[42] Konrad Jacobs and Konrad Jacobs. “Independent identically distributed (iid) random variables”. In: *Discrete Stochastics* (1992), pp. 65–101 (cit. on p. 6).

[43] Elad Hoffer, Itay Hubara, and Daniel Soudry. “Train longer, generalize better: closing the generalization gap in large batch training of neural networks”. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 6).

[44] Angelos Filos et al. “A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks”. In: *arXiv preprint arXiv:1912.10481* (2019) (cit. on pp. 6, 14, 50, 190).

[45] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *nature* 542.7639 (2017), pp. 115–118 (cit. on pp. 6, 14, 50).

[46] Yu Huang and Yue Chen. “Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies”. In: *arXiv preprint arXiv:2006.06091* (2020) (cit. on pp. 6, 14, 50).

[47] Thomas Fischer and Christopher Krauss. “Deep learning with long short-term memory networks for financial market predictions”. In: *European journal of operational research* 270.2 (2018), pp. 654–669 (cit. on p. 6).

[48] Yarin Gal. "Uncertainty in Deep Learning". PhD thesis. University of Cambridge, 2016 (cit. on pp. 6, 14, 19, 55, 80, 82, 83).

[49] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 6402–6413. URL: https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf (cit. on pp. 6, 9, 23, 50, 52, 55, 63, 67, 71, 75, 78–80, 88, 143, 179, 184, 185, 194).

[50] Will Knight. "OpenAI's CEO says the age of giant AI models is already over". In: *Wired, April* 17 (2023), p. 2023 (cit. on p. 7).

[51] OpenAI. *ChatGPT: An AI Language Model by OpenAI*. https://openai.com. 2020 (cit. on pp. 7, 25, 28, 119, 148).

[52] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on pp. 7, 13, 25, 26).

[53] Lei Huang et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". In: *arXiv preprint arXiv:2311.05232* (2023) (cit. on p. 7).

[54] Jaemin Cho, Abhay Zala, and Mohit Bansal. "Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3043–3054 (cit. on pp. 7, 148).

[55] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165 (cit. on pp. 7, 124, 146).

[56] Chuan Guo et al. "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1321–1330 (cit. on pp. 8, 14, 17, 18, 31, 32, 35, 36, 55, 59, 64, 80, 82, 141).

[57] Joost van Amersfoort et al. "Uncertainty estimation using a single deep deterministic neural network". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9690–9700 (cit. on pp. 9, 23, 50–52, 55–57, 63, 64, 70, 75, 80, 143, 178, 182, 184, 185, 190).

[58] Jeremiah Zhe Liu et al. "Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness". In: *NeurIPS*. 2020 (cit. on pp. 9, 23, 50, 52, 53, 55, 56, 63, 64, 70, 71, 75, 80, 143, 178, 182, 184, 185, 194, 195).

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on pp. 14, 85).

[60] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998 (cit. on p. 14).

[61] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*. 2015 (cit. on pp. 14, 24, 85, 190, 191).

[62] John Bridle. "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf (cit. on p. 15).

[63] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. "When Does Label Smoothing Help?" In: *Conference on Neural Information Processing Systems*. 2019 (cit. on pp. 15, 18, 32, 43, 173).

[64] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning". In: *AAAI Conference on Artificial Intelligence*. 2015 (cit. on pp. 17, 31, 33, 34).

[65] Jeremy Nixon et al. "Measuring Calibration in Deep Learning." In: *CVPR workshops*. Vol. 2. 7. 2019 (cit. on p. 17).

[66] Alexandru Niculescu-Mizil and Rich Caruana. "Predicting Good Probabilities With Supervised Learning". In: *International Conference on Machine Learning*. 2005 (cit. on pp. 17, 34).

[67] Bianca Zadrozny and Charles Elkan. "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *International Conference on Machine Learning*. 2001 (cit. on pp. 18, 31).

[68] Bianca Zadrozny and Charles Elkan. "Transforming Classifier Scores into Accurate Multiclass Probability Estimates". In: *ACM International Conference on Knowledge Discovery and Data Mining*. 2002 (cit. on pp. 18, 31).

[69] Juozas Vaicenavicius et al. "Evaluating model calibration in classification". In: *International Conference on Artificial Intelligence and Statistics*. 2019 (cit. on pp. 18, 34).

[70] John Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers* 10.3 (1999), pp. 61–74 (cit. on pp. 18, 31).

[71] Azadeh Sadat Mozafari et al. "A New Loss Function for Temperature Scaling To Have Better Calibrated Deep Networks". In: *arXiv preprint arXiv:1810.11586v1* (2018) (cit. on p. 18).

[72] Tom Joy et al. "Sample-dependent adaptive temperature scaling for improved calibration". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 12. 2023, pp. 14919–14926 (cit. on p. 18).

[73] Gabriel Pereyra et al. "Regularizing Neural Networks by Penalizing Confident Output Distributions". In: *arXiv preprint arXiv:1701.06548* (2017) (cit. on pp. 18, 37).

[74] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. "Trainable Calibration Measures For Neural Networks From Kernel Mean Embeddings". In: *International Conference on Machine Learning*. 2018 (cit. on pp. 18, 31, 42, 168).

[75] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. "Meta-Calibration: Learning of Model Calibration Using Differentiable Expected Calibration Error". In: *arXiv preprint arXiv:2106.09613* (2021) (cit. on p. 18).

[76] Sunil Thulasidasan et al. “On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks”. In: *Conference on Neural Information Processing Systems*. 2019 (cit. on pp. 18, 45).

[77] Jishnu Mukhoti et al. “Calibrating deep neural networks using focal loss”. In: *arXiv preprint arXiv:2002.09437* (2020) (cit. on pp. 19, 80).

[78] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems*. 2017, pp. 5574–5584 (cit. on pp. 19, 54, 67, 80, 198).

[79] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep bayesian active learning with image data”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1183–1192 (cit. on pp. 20, 50, 60).

[80] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. “BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning”. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on pp. 20, 24, 54).

[81] Yonatan Geifman and Ran El-Yaniv. “Selective classification for deep neural networks”. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 20).

[82] Owen Lockwood and Mei Si. “A review of uncertainty for deep reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 18. 1. 2022, pp. 155–162 (cit. on p. 20).

[83] Jingkang Yang et al. “Openood: Benchmarking generalized out-of-distribution detection”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 32598–32611 (cit. on pp. 20, 80).

[84] Steffen Bickel, Michael Brückner, and Tobias Scheffer. “Discriminative learning under covariate shift.” In: *Journal of Machine Learning Research* 10.9 (2009) (cit. on p. 20).

[85] Dan Hendrycks and Thomas Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *arXiv preprint arXiv:1903.12261* (2019) (cit. on pp. 21, 45).

[86] Dan Hendrycks et al. “The many faces of robustness: A critical analysis of out-of-distribution generalization”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8340–8349 (cit. on pp. 21, 80, 91, 114, 224).

[87] Haohan Wang et al. “Learning Robust Global Representations by Penalizing Local Predictive Power”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 10506–10518 (cit. on pp. 21, 80, 91, 114, 224).

[88] Tom Fawcett. “An introduction to ROC analysis”. In: *Pattern recognition letters* 27.8 (2006), pp. 861–874 (cit. on p. 21).

[89] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009) (cit. on pp. 21, 24, 42, 63, 76, 125, 143, 161, 190, 224).

[90] Yuval Netzer et al. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *Conference on Neural Information Processing Systems Workshops*. 2011 (cit. on pp. 21, 45, 63, 76, 126).

[91] Dan Hendrycks et al. “Natural adversarial examples”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15262–15271 (cit. on pp. 21, 63, 81, 88, 91, 114, 186, 224).

[92] Haoqi Wang et al. “Vim: Out-of-distribution with virtual-logit matching”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4921–4930 (cit. on pp. 21, 80, 88).

[93] Dan Hendrycks et al. “Scaling out-of-distribution detection for real-world settings”. In: *arXiv preprint arXiv:1911.11132* (2019) (cit. on pp. 21, 81, 88).

[94] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. “Exploring the limits of out-of-distribution detection”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7068–7081 (cit. on pp. 21, 75–78, 80, 81, 144).

[95] Shu Kong and Deva Ramanan. “Opengan: Open-set recognition via open data generation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 813–822 (cit. on pp. 21, 81).

[96] Martin Mundt et al. “Unified probabilistic deep continual learning through generative replay and open set recognition”. In: *arXiv preprint arXiv:1905.12019* (2019) (cit. on pp. 21, 75).

[97] Guangyao Chen et al. “Adversarial reciprocal points learning for open set recognition”. In: *arXiv preprint arXiv:2103.00953* (2021) (cit. on pp. 21, 81).

[98] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. “Deep Anomaly Detection with Outlier Exposure”. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 21, 55, 87, 179, 209, 210).

[99] Chu Wang, Yan-Ming Zhang, and Cheng-Lin Liu. “Anomaly detection via minimum likelihood generative adversarial networks”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 1121–1126 (cit. on pp. 21, 81).

[100] Phuc Cuong Ngo et al. “Fence GAN: Towards better anomaly detection”. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE. 2019, pp. 141–148 (cit. on pp. 21, 81).

[101] David JC MacKay. “Bayesian methods for adaptive models”. PhD thesis. California Institute of Technology, 1992 (cit. on pp. 22, 54).

[102] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012 (cit. on pp. 22, 54, 80, 82).

[103] Tianqi Chen, Emily Fox, and Carlos Guestrin. “Stochastic gradient hamiltonian monte carlo”. In: *International conference on machine learning*. PMLR. 2014, pp. 1683–1691 (cit. on p. 22).

[104] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688 (cit. on p. 22).

[105] Michael I Jordan et al. “An introduction to variational methods for graphical models”. In: *Machine learning* 37 (1999), pp. 183–233 (cit. on p. 22).

[106] Alex Graves. "Practical variational inference for neural networks". In: *Advances in neural information processing systems* 24 (2011) (cit. on p. 22).

[107] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526 (cit. on pp. 22, 124).

[108] Hippolyt Ritter, Aleksandar Botev, and David Barber. "A scalable laplace approximation for neural networks". In: *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. Vol. 6. International Conference on Representation Learning. 2018 (cit. on p. 22).

[109] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059 (cit. on pp. 22, 50, 53, 55, 60, 80).

[110] Yarin Gal, Jiri Hron, and Alex Kendall. "Concrete dropout". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 22).

[111] Charles Blundell et al. "Weight uncertainty in neural network". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622 (cit. on pp. 22, 50, 55, 80).

[112] Andrew G Wilson and Pavel Izmailov. "Bayesian deep learning and a probabilistic perspective of generalization". In: *Advances in neural information processing systems* 33 (2020), pp. 4697–4708 (cit. on pp. 23, 67, 80, 83).

[113] Yeming Wen, Dustin Tran, and Jimmy Ba. "BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Sklf1yrYDr (cit. on pp. 23, 50, 55, 80, 193, 194).

[114] Nikita Durasov et al. "Masksembles for uncertainty estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13539–13548 (cit. on pp. 23, 80).

[115] Olivier Laurent et al. "Packed-Ensembles for Efficient Uncertainty Estimation". In: *arXiv preprint arXiv:2210.09184* (2022) (cit. on p. 23).

[116] Jishnu Mukhoti et al. "Deep Deterministic Uncertainty: A New Simple Baseline". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 24384–24394 (cit. on pp. 23, 80).

[117] Jishnu Mukhoti et al. "Deep deterministic uncertainty for semantic segmentation". In: *arXiv preprint arXiv:2111.00079* (2021) (cit. on p. 24).

[118] Neil Houlsby et al. "Bayesian active learning for classification and preference learning". In: *arXiv preprint arXiv:1112.5745* (2011) (cit. on p. 24).

[119] Sayna Ebrahimi et al. "Uncertainty-guided continual learning with bayesian neural networks". In: *arXiv preprint arXiv:1906.02425* (2019) (cit. on p. 24).

[120] Hongjoon Ahn et al. "Uncertainty-based continual learning with adaptive regularization". In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 24).

[121] Chen Sun et al. “Revisiting unreasonable effectiveness of data in deep learning era”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852 (cit. on pp. 24, 54).

[122] Pablo Villalobos et al. “Machine learning model sizes and the parameter gap”. In: *arXiv preprint arXiv:2207.02852* (2022) (cit. on p. 24).

[123] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 24, 98).

[124] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 24).

[125] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456 (cit. on p. 24).

[126] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016) (cit. on p. 24).

[127] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on p. 25).

[128] Rohan Anil et al. “Palm 2 technical report”. In: *arXiv preprint arXiv:2305.10403* (2023) (cit. on p. 25).

[129] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision”. In: *arXiv preprint arXiv:2304.07193* (2023) (cit. on p. 25).

[130] Alexander Kirillov et al. “Segment anything”. In: *arXiv preprint arXiv:2304.02643* (2023) (cit. on pp. 25, 27).

[131] Jiahui Yu et al. “Coca: Contrastive captioners are image-text foundation models”. In: *arXiv preprint arXiv:2205.01917* (2022) (cit. on pp. 25, 97, 98, 100, 118).

[132] Zhan Shi et al. “Improving image captioning with better use of captions”. In: *arXiv preprint arXiv:2006.11807* (2020) (cit. on p. 25).

[133] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695 (cit. on p. 25).

[134] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 26, 105, 106).

[135] Kaiyang Zhou et al. “Learning to prompt for vision-language models”. In: *International Journal of Computer Vision* 130.9 (2022), pp. 2337–2348 (cit. on pp. 26, 28, 118, 134).

[136] Kaiyang Zhou et al. “Conditional prompt learning for vision-language models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16816–16825 (cit. on pp. 26, 118, 134).

[137] Peng Gao et al. “Clip-adapter: Better vision-language models with feature adapters”. In: *arXiv preprint arXiv:2110.04544* (2021) (cit. on pp. 26, 28, 118, 134).

[138] Renrui Zhang et al. “Tip-adapter: Training-free clip-adapter for better vision-language modeling”. In: *arXiv preprint arXiv:2111.03930* (2021) (cit. on pp. 26, 118, 134).

[139] Ananya Kumar et al. “Fine-tuning can distort pretrained features and underperform out-of-distribution”. In: *arXiv preprint arXiv:2202.10054* (2022) (cit. on pp. 26, 118, 122, 134).

[140] Sachin Goyal et al. “Finetune like you pretrain: Improved finetuning of zero-shot vision models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 19338–19347 (cit. on pp. 26, 28, 118, 122).

[141] Agrim Gupta, Piotr Dollar, and Ross Girshick. “Lvis: A dataset for large vocabulary instance segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5356–5364 (cit. on p. 27).

[142] Haotian Zhang et al. “GLIPv2: Unifying Localization and Vision-Language Understanding”. In: *arXiv preprint arXiv:2206.05836* (2022) (cit. on pp. 27, 97, 100).

[143] Aishwarya Kamath et al. “MDETR-modulated detection for end-to-end multi-modal understanding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1780–1790 (cit. on pp. 27, 100).

[144] Boyi Li et al. “Language-driven Semantic Segmentation”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=RriDjddCLN (cit. on pp. 27, 98, 99, 101, 110, 111).

[145] Golnaz Ghiasi et al. “Scaling open-vocabulary image segmentation with image-level labels”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 540–557 (cit. on pp. 27, 98, 99, 101, 110, 111).

[146] Jiarui Xu et al. “GroupViT: Semantic Segmentation Emerges from Text Supervision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18134–18144 (cit. on pp. 27, 98, 99, 101, 109–111).

[147] Patrick Helber et al. “Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019) (cit. on pp. 28, 125).

[148] Bastiaan S Veeling et al. “Rotation Equivariant CNNs for Digital Pathology”. In: (June 2018). arXiv: 1806.03962 [cs.CV] (cit. on p. 28).

[149] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474 (cit. on p. 28).

[150] Mitchell Wortsman et al. “Robust fine-tuning of zero-shot models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7959–7971 (cit. on pp. 28, 118, 134, 235, 236).

[151] Edward J Hu et al. “Lora: Low-rank adaptation of large language models”. In: *arXiv preprint arXiv:2106.09685* (2021) (cit. on p. 28).

[152] Tim Dettmers et al. “Qlora: Efficient finetuning of quantized llms”. In: *arXiv preprint arXiv:2305.14314* (2023) (cit. on p. 28).

[153] Zangwei Zheng et al. “Preventing Zero-Shot Transfer Degradation in Continual Learning of Vision-Language Models”. In: *arXiv preprint arXiv:2303.06628* (2023) (cit. on pp. 28, 135).

[154] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *International Conference on Computer Vision*. 2017 (cit. on pp. 30, 32, 37, 141).

[155] Chensi Cao et al. “Deep Learning and Its Applications in Biomedicine”. In: *Genomics, Proteomics & Bioinformatics* 16.1 (2018), pp. 17–32 (cit. on p. 31).

[156] Hongming Chen et al. “The rise of deep learning in drug discovery”. In: *Drug Discovery Today* 23.36 (2018), pp. 1241–1250 (cit. on p. 31).

[157] Andreas Kamilaris and Francesc X Prenafeta-Boldú. “Deep learning in agriculture: A survey”. In: *Computers and Electronics in Agriculture* 147 (2018), pp. 70–90 (cit. on p. 31).

[158] Justin Ker et al. “Deep Learning Applications in Medical Image Analysis”. In: *IEEE Access* 6 (2018), pp. 9375–9389 (cit. on p. 31).

[159] Jinjiang Wang et al. “Deep learning for smart manufacturing: Methods and applications”. In: *Journal of Manufacturing Systems* 48 (2018), pp. 144–156 (cit. on p. 31).

[160] Mahdi Pakdaman Naeini and Gregory F Cooper. “Binary Classifier Calibration using an Ensemble of Near Isotonic Regression Models”. In: *IEEE International Conference on Data Mining*. 2016 (cit. on p. 31).

[161] Yaniv Ovadia et al. “Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift”. In: *Conference on Neural Information Processing Systems*. 2019, pp. 13991–14002 (cit. on pp. 31, 45, 50, 55, 81, 82).

[162] Glenn W Brier. “Verification of forecasts expressed in terms of probability”. In: *Monthly weather review* (1950) (cit. on pp. 33, 42).

[163] Khanh Nguyen and Brendan O’Connor. “Posterior calibration and exploratory analysis for natural language processing models”. In: *arXiv preprint arXiv:1508.05154* (2015) (cit. on p. 34).

[164] Meelis Kull et al. “Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration”. In: *Conference on Neural Information Processing Systems*. 2019 (cit. on p. 34).

[165] David Widmann, Fredrik Lindsten, and Dave Zachariah. “Calibration tests in multi-class classification: A unifying framework”. In: *Conference on Neural Information Processing Systems*. 2019 (cit. on p. 34).

[166] Ananya Kumar, Percy S Liang, and Tengyu Ma. “Verified Uncertainty Calibration”. In: *Conference on Neural Information Processing Systems*. 2019 (cit. on pp. 34, 45, 168).

[167] *Train CIFAR10 with PyTorch*. Available online at https://github.com/kuangliu/pytorch-cifar (as of 22nd May 2019). 2019 (cit. on p. 35).

[168] Robert M Corless et al. “On the Lambert W Function”. In: *Advances in Computational Mathematics* 5.1 (1996), pp. 329–359 (cit. on pp. 41, 158, 160).

[169] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: *British Machine Vision Conference*. 2016 (cit. on pp. 42, 63).

[170] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708 (cit. on pp. 42, 63, 85).

[171] Ken Lang. “NewsWeeder: Learning to Filter Netnews”. In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 331–339 (cit. on pp. 42, 161).

[172] Richard Socher et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2013 (cit. on pp. 42, 161).

[173] Min Lin, Qiang Chen, and Shuicheng Yan. “Network In Network”. In: *International Conference on Learning Representations*. 2014 (cit. on pp. 42, 162).

[174] Kai Sheng Tai, Richard Socher, and Christopher D Manning. “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. In: *Association for Computational Linguistics (ACL)*. 2015 (cit. on pp. 42, 162).

[175] Morris H DeGroot and Stephen E Fienberg. “The Comparison and Evaluation of Forecasters”. In: *Journal of the Royal Statistical Society: Series D (The Statistician)* (1983) (cit. on p. 43).

[176] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural safety* 31.2 (2009), pp. 105–112 (cit. on p. 50).

[177] Dan Hendrycks and Kevin Gimpel. “A baseline for detecting misclassified and out-of-distribution examples in neural networks”. In: *arXiv preprint arXiv:1610.02136* (2016) (cit. on pp. 50, 55, 75, 79, 88, 179).

[178] Michael Dusenberry et al. “Efficient and scalable bayesian neural nets with rank-1 factors”. In: *International conference on machine learning*. PMLR. 2020, pp. 2782–2792 (cit. on pp. 50, 55).

[179] Kimin Lee et al. “A simple unified framework for detecting out-of-distribution samples and adversarial attacks”. In: *Advances in neural information processing systems* 31 (2018) (cit. on pp. 50, 52, 55, 58, 65, 70, 71, 75, 88, 177, 186, 192).

[180] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *NeurIPS*. 2017 (cit. on pp. 50, 56).

[181] Takeru Miyato et al. “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 50, 56, 142, 190).

[182] Tilmann Gneiting and Adrian E Raftery. “Strictly proper scoring rules, prediction, and estimation”. In: *Journal of the American statistical Association* 102.477 (2007), pp. 359–378 (cit. on pp. 52, 203).

[183] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017) (cit. on pp. 53, 66, 100, 183).

[184] M. Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338 (cit. on pp. 53, 65, 99–101, 110, 111, 223).

[185] Tim Pearce, Alexandra Brintrup, and Jun Zhu. *Understanding Softmax Confidence and Uncertainty*. 2021. arXiv: 2106.04972 [cs.LG] (cit. on p. 53).

[186] Janis Postels et al. "Quantifying Aleatoric and Epistemic Uncertainty Using Density Estimation in Latent Space". In: *arXiv preprint arXiv:2012.03082* (2020) (cit. on pp. 53–55, 70, 177).

[187] Jim Winkens et al. "Contrastive training for improved out-of-distribution detection". In: *arXiv preprint arXiv:2007.05566* (2020) (cit. on pp. 53, 75, 76, 178).

[188] Laurent Dinh, David Krueger, and Yoshua Bengio. *NICE: Non-linear Independent Components Estimation*. 2015. arXiv: 1410.8516 [cs.LG] (cit. on p. 54).

[189] Lewis Smith and Yarin Gal. "Understanding Measures of Uncertainty for Adversarial Example Detection". In: *UAI*. 2018 (cit. on p. 55).

[190] Geoffrey E Hinton and Drew Van Camp. "Keeping the neural networks simple by minimizing the description length of the weights". In: *Proceedings of the sixth annual conference on Computational learning theory*. 1993, pp. 5–13 (cit. on p. 55).

[191] José Miguel Hernández-Lobato and Ryan Adams. "Probabilistic backpropagation for scalable learning of bayesian neural networks". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1861–1869 (cit. on p. 55).

[192] Shiyu Liang, Yixuan Li, and R Srikant. "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks". In: *International Conference on Learning Representations*. 2018 (cit. on pp. 55, 75, 80, 179).

[193] Kimin Lee et al. "Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples". In: *International Conference on Learning Representations*. 2018 (cit. on pp. 55, 80, 81, 179).

[194] Terrance DeVries and Graham W Taylor. "Learning confidence for out-of-distribution detection in neural networks". In: *arXiv preprint arXiv:1802.04865* (2018) (cit. on pp. 55, 179).

[195] Weitang Liu et al. "Energy-based Out-of-distribution Detection". In: *Advances in Neural Information Processing Systems* 33 (2020) (cit. on pp. 55, 63, 64, 80, 88, 177, 184, 185, 187).

[196] Joost van Amersfoort et al. "Improving Deterministic Uncertainty Estimation in Deep Learning for Classification and Regression". In: *arXiv preprint arXiv:2102.11409* (2021) (cit. on p. 56).

[197] Mihaela Rosca et al. "A case for new neural network smoothness constraints". In: *arXiv preprint arXiv:2012.07969* (2020) (cit. on p. 56).

[198] Jens Behrmann et al. "Invertible residual networks". In: *International conference on machine learning*. PMLR. 2019, pp. 573–582 (cit. on p. 57).

[199] Lewis Smith et al. *Can convolutional ResNets approximately preserve input distances? A frequency analysis perspective*. 2021. arXiv: 2106.02469 [cs.LG] (cit. on p. 57).

[200] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122 (cit. on p. 59).

[201] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. "Active learning with statistical models". In: *Journal of artificial intelligence research* 4 (1996), pp. 129–145 (cit. on pp. 59, 60).

[202] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR*. 2015 (cit. on p. 61).

[203] Dan Hendrycks and Thomas Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *Proceedings of the International Conference on Learning Representations* (2019) (cit. on pp. 63, 76, 80, 87, 91).

[204] Jishnu Mukhoti and Yarin Gal. "Evaluating bayesian deep learning methods for semantic segmentation". In: *arXiv preprint arXiv:1811.12709* (2018) (cit. on p. 66).

[205] Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. "Bayesian Deep Ensembles via the Neural Tangent Kernel". In: *Advances in neural information processing systems*. 2020 (cit. on p. 67).

[206] Andrey Malinin and Mark JF Gales. "Predictive Uncertainty Estimation via Prior Networks". In: *NeurIPS*. 2018 (cit. on pp. 69, 179).

[207] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012 (cit. on p. 70).

[208] Vincent Fortuin et al. "Deep classifiers with label noise modeling and distance awareness". In: *arXiv preprint arXiv:2110.02609* (2021) (cit. on p. 71).

[209] Angelos Filos et al. "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3145–3153 (cit. on p. 75).

[210] Abhijit Guha Roy et al. "Does Your Dermatology Classifier Know What It Doesn't Know? Detecting the Long-Tail of Unseen Conditions". In: *arXiv preprint arXiv:2104.03829* (2021) (cit. on p. 75).

[211] Jishnu Mukhoti et al. "Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty". In: *arXiv preprint arXiv:2102.11582* (2021) (cit. on p. 75).

[212] Francesco Pinto et al. "RegMixup: Mixup as a Regularizer Can Surprisingly Improve Accuracy and Out Distribution Robustness". In: *arXiv: 2206.14502*. 2022 (cit. on p. 75).

[213] Ziwei Liu et al. "Large-scale long-tailed recognition in an open world". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2537–2546 (cit. on pp. 75, 79).

[214] Kai Xiao et al. "Noise or signal: The role of image backgrounds in object recognition". In: *arXiv preprint arXiv:2006.09994* (2020) (cit. on pp. 76, 80).

[215] Pang Wei Koh et al. "Wilds: A benchmark of in-the-wild distribution shifts". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5637–5664 (cit. on pp. 76, 80).

[216] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017) (cit. on pp. 76, 81, 143, 190, 191).

[217] Thomas Deselaers and Vittorio Ferrari. “Visual and semantic similarity in imagenet”. In: *CVPR 2011*. IEEE. 2011, pp. 1777–1784 (cit. on p. 77).

[218] Clemens-Alexander Brust and Joachim Denzler. “Not just a matter of semantics: the relationship between visual similarity and semantic similarity”. In: *arXiv preprint arXiv:1811.07120* (2018) (cit. on p. 77).

[219] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 78).

[220] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134 (cit. on pp. 78, 83, 208).

[221] Yen-Chang Hsu et al. “Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data”. In: *CVPR*. 2020, pp. 10951–10960 (cit. on pp. 80, 179).

[222] Yiyou Sun, Chuan Guo, and Yixuan Li. “React: Out-of-distribution detection with rectified activations”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 144–157 (cit. on p. 80).

[223] Hongxin Wei et al. “Mitigating Neural Network Overconfidence with Logit Normalization”. In: *arXiv preprint arXiv:2205.09310* (2022) (cit. on p. 80).

[224] Stephan Mandt, Matthew D Hoffman, and David M Blei. “Stochastic gradient descent as approximate bayesian inference”. In: *arXiv preprint arXiv:1704.04289* (2017) (cit. on p. 80).

[225] Jeremiah Zhe Liu et al. “A Simple Approach to Improve Single-Model Deep Uncertainty via Distance-Awareness”. In: *arXiv preprint arXiv:2205.00403* (2022) (cit. on p. 80).

[226] Chandramouli Shama Sastry and Sageev Oore. “Detecting out-of-distribution examples with gram matrices”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8491–8501 (cit. on p. 80).

[227] Haiwen Huang et al. “Feature Space Singularity for Out-of-Distribution Detection”. In: *Proceedings of the Workshop on Artificial Intelligence Safety 2021 (SafeAI 2021)*. 2021 (cit. on pp. 80, 193, 194).

[228] Ishaan Gulrajani and David Lopez-Paz. “In search of lost domain generalization”. In: *arXiv preprint arXiv:2007.01434* (2020) (cit. on p. 80).

[229] Nikolaos Dionelis. “OMASGAN: Out-of-Distribution Minimum Anomaly Score GAN for Sample Generation on the Boundary”. In: *arXiv preprint arXiv:2110.15273* (2021) (cit. on p. 81).

[230] Muhammad Zaigham Zaheer et al. “Old is gold: Redefining the adversarially learned one-class classifier training paradigm”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14183–14193 (cit. on p. 81).

[231] Xuefeng Du et al. “Vos: Learning what you don’t know by virtual outlier synthesis”. In: *arXiv preprint arXiv:2202.01197* (2022) (cit. on p. 81).

[232] Jianhao Yuan et al. “Not just pretty pictures: Text-to-image generators enable interpretable interventions for robust representations”. In: *arXiv preprint arXiv:2212.11237* (2022) (cit. on p. 81).

[233] Sebastian Bosse et al. “Deep Neural Networks for No-Reference and Full-Reference Image Quality Assessment”. In: *IEEE Transactions on Image Processing* 27.1 (2018), pp. 206–219 (cit. on p. 82).

[234] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612 (cit. on p. 82).

[235] Lin Zhang et al. “FSIM: A feature similarity index for image quality assessment”. In: *IEEE transactions on Image Processing* 20.8 (2011), pp. 2378–2386 (cit. on p. 82).

[236] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595 (cit. on pp. 82, 119, 129, 230).

[237] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826 (cit. on p. 85).

[238] Andrew Howard et al. “Searching for mobilenetv3”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1314–1324 (cit. on p. 85).

[239] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114 (cit. on p. 85).

[240] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 86).

[241] Mircea Cimpoi et al. “Describing textures in the wild”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 3606–3613 (cit. on pp. 88, 125, 210).

[242] Benjamin Recht et al. “Do imagenet classifiers generalize to imagenet?” In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5389–5400 (cit. on pp. 91, 114, 224).

[243] Zihang Dai et al. “Coatnet: Marrying convolution and attention for all data sizes”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3965–3977 (cit. on p. 97).

[244] Mitchell Wortsman et al. “Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998 (cit. on pp. 97, 134).

[245] Yixuan Wei et al. “Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation”. In: *arXiv preprint arXiv:2205.14141* (2022) (cit. on p. 97).

[246] Wenhui Wang et al. “Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks”. In: *arXiv preprint arXiv:2208.10442* (2022) (cit. on p. 97).

[247] Hao Zhang et al. “Dino: Detr with improved denoising anchor boxes for end-to-end object detection”. In: *arXiv preprint arXiv:2203.03605* (2022) (cit. on p. 97).

[248] Ze Liu et al. “Swin transformer v2: Scaling up capacity and resolution”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12009–12019 (cit. on p. 97).

[249] Lu Yuan et al. “Florence: A new foundation model for computer vision”. In: *arXiv preprint arXiv:2111.11432* (2021) (cit. on pp. 97, 98, 100).

[250] Feng Li et al. “Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation”. In: *arXiv preprint arXiv:2206.02777* (2022) (cit. on p. 97).

[251] Zhe Chen et al. “Vision Transformer Adapter for Dense Predictions”. In: *arXiv preprint arXiv:2205.08534* (2022) (cit. on p. 97).

[252] Jang Hyun Cho et al. “Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16794–16804 (cit. on pp. 97, 100).

[253] Mark Hamilton et al. “Unsupervised Semantic Segmentation by Distilling Feature Correspondences”. In: *arXiv preprint arXiv:2203.08414* (2022) (cit. on pp. 97, 98, 100, 103, 104).

[254] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763 (cit. on pp. 97, 98, 100, 101, 109, 111, 118–121, 123, 125, 222, 223).

[255] Chao Jia et al. “Scaling up visual and vision-language representation learning with noisy text supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4904–4916 (cit. on pp. 98, 100, 101, 118).

[256] Mengde Xu et al. “A simple baseline for zero-shot semantic segmentation with pre-trained vision-language model”. In: *arXiv preprint arXiv:2112.14757* (2021) (cit. on p. 98).

[257] Quande Liu et al. “Open-World Semantic Segmentation via Contrasting and Clustering Vision-Language Embedding”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 275–292 (cit. on pp. 98, 99, 101, 110, 111).

[258] Tanmay Gupta et al. “Contrastive learning for weakly supervised phrase grounding”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 752–768 (cit. on pp. 98, 105, 108).

[259] Roozbeh Mottaghi et al. “The Role of Context for Object Detection and Semantic Segmentation in the Wild”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014 (cit. on pp. 99, 110, 111, 223).

[260] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. “COCO-Stuff: Thing and stuff classes in context”. In: *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE. 2018, pp. 1209–1218 (cit. on pp. 99, 110, 111, 224).

[261] Bolei Zhou et al. “Scene parsing through ade20k dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 633–641 (cit. on pp. 99, 100, 110, 111, 224).

[262] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223 (cit. on p. 100).

[263] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 100).

[264] Jingdong Wang et al. "Deep high-resolution representation learning for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3349–3364 (cit. on p. 100).

[265] Hengshuang Zhao et al. "Pyramid scene parsing network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890 (cit. on p. 100).

[266] Jyh-Jing Hwang et al. "Segsort: Segmentation by discriminative sorting of segments". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7334–7344 (cit. on p. 100).

[267] Wouter Van Gansbeke et al. "Unsupervised semantic segmentation by contrasting object mask proposals". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10052–10062 (cit. on p. 100).

[268] Luke Melas-Kyriazi et al. "Deep Spectral Methods: A Surprisingly Strong Baseline for Unsupervised Semantic Segmentation and Localization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8364–8375 (cit. on p. 100).

[269] Yangtao Wang et al. "TokenCut: Segmenting Objects in Images and Videos with Self-supervised Transformer and Normalized Cut". In: *arXiv preprint arXiv:2209.00383* (2022) (cit. on p. 100).

[270] Wonjae Kim, Bokyung Son, and Ildoo Kim. "Vilt: Vision-and-language transformer without convolution or region supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5583–5594 (cit. on p. 100).

[271] Amanpreet Singh et al. "Flava: A foundational language and vision alignment model". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15638–15650 (cit. on pp. 100, 131, 236).

[272] Feilong Chen et al. "Vlp: A survey on vision-language pre-training". In: *arXiv preprint arXiv:2202.09061* (2022) (cit. on p. 100).

[273] Kaiye Wang et al. "A comprehensive survey on cross-modal retrieval". In: *arXiv preprint arXiv:1607.06215* (2016) (cit. on p. 100).

[274] Stanislaw Antol et al. "Vqa: Visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433 (cit. on p. 100).

[275] Piyush Sharma et al. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning". In: *Proceedings of ACL*. 2018 (cit. on pp. 109, 111).

[276] Soravit Changpinyo et al. "Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3558–3568 (cit. on pp. 109, 111).

[277] Bart Thomee et al. "YFCC100M: The new data in multimedia research". In: *Communications of the ACM* 59.2 (2016), pp. 64–73 (cit. on pp. 109, 111, 223).

[278] Yongqin Xian et al. “Semantic projection network for zero-and few-label semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8256–8265 (cit. on pp. 110, 111).

[279] Maxime Bucher et al. “Zero-shot semantic segmentation”. In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on pp. 110, 111).

[280] Xinlei Chen et al. “Microsoft coco captions: Data collection and evaluation server”. In: *arXiv preprint arXiv:1504.00325* (2015) (cit. on p. 111).

[281] Jordi Pont-Tuset et al. “Connecting Vision and Language with Localized Narratives”. In: *ECCV*. 2020 (cit. on p. 111).

[282] Gabriel Ilharco et al. “Editing Models with Task Arithmetic”. In: *arXiv preprint arXiv:2212.04089* (2022) (cit. on pp. 118, 134, 229).

[283] LI Xuhong, Yves Grandvalet, and Franck Davoine. “Explicit inductive bias for transfer learning with convolutional networks”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2825–2834 (cit. on pp. 118–120, 122, 146).

[284] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 121).

[285] Ronald Kemker et al. “Measuring catastrophic forgetting in neural networks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 2018 (cit. on p. 124).

[286] Gabriel Ilharco et al. *OpenCLIP*. Zenodo, July 2021. URL: https://doi.org/10.5281/zenodo.5143773 (cit. on p. 125).

[287] Jonathan Krause et al. “Collecting a large-scale dataset of fine-grained cars”. In: (2013) (cit. on p. 125).

[288] Johannes Stallkamp et al. “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition”. In: *Neural networks* 32 (2012), pp. 323–332 (cit. on p. 126).

[289] Gong Cheng, Junwei Han, and Xiaoqiang Lu. “Remote sensing image scene classification: Benchmark and state of the art”. In: *Proceedings of the IEEE* 105.10 (2017), pp. 1865–1883 (cit. on p. 126).

[290] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 127, 132).

[291] Arslan Chaudhry et al. “Riemannian walk for incremental learning: Understanding forgetting and intransigence”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 532–547 (cit. on pp. 127, 132).

[292] Sylvestre-Alvise Rebuffi et al. “icarl: Incremental classifier and representation learning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 2001–2010 (cit. on pp. 132, 135).

[293] Manli Shu et al. “Test-time prompt tuning for zero-shot generalization in vision-language models”. In: *arXiv preprint arXiv:2209.07511* (2022) (cit. on p. 134).

[294] Jiaxin Ge et al. “Chain of Thought Prompt Tuning in Vision Language Models”. In: *arXiv preprint arXiv:2304.07919* (2023) (cit. on p. 134).

[295] Jingchen Sun et al. “Prompt Tuning based Adapter for Vision-Language Model Adaption”. In: *arXiv preprint arXiv:2303.15234* (2023) (cit. on p. 134).

[296] Omiros Pantazis et al. “SVL-Adapter: Self-Supervised Adapter for Vision-Language Pretrained Models”. In: *arXiv preprint arXiv:2210.03794* (2022) (cit. on p. 134).

[297] Gabriel Ilharco et al. “Patching open-vocabulary models by interpolating weights”. In: *arXiv preprint arXiv:2208.05592* (2022) (cit. on p. 134).

[298] Guande He, Jianfei Chen, and Jun Zhu. “Preserving Pre-trained Features Helps Calibrate Fine-tuned Language Models”. In: *arXiv preprint arXiv:2305.19249* (2023) (cit. on p. 135).

[299] Seanie Lee et al. “Self-Distillation for Further Pre-training of Transformers”. In: *arXiv preprint arXiv:2210.02871* (2022) (cit. on p. 135).

[300] Yang Zhou et al. “Preserve Pre-trained Knowledge: Transfer Learning With Self-Distillation For Action Recognition”. In: *arXiv preprint arXiv:2205.00506* (2022) (cit. on p. 135).

[301] Xiaofeng Mao et al. “Context-Aware Robust Fine-Tuning”. In: *arXiv preprint arXiv:2211.16175* (2022) (cit. on p. 135).

[302] Zhizhong Li and Derek Hoiem. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947 (cit. on p. 135).

[303] Heechul Jung et al. “Less-forgetting learning in deep neural networks”. In: *arXiv preprint arXiv:1607.00122* (2016) (cit. on p. 135).

[304] Saihui Hou et al. “Lifelong learning via progressive distillation and retrospection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 437–452 (cit. on p. 135).

[305] Sergey Zagoruyko and Nikos Komodakis. “Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer”. In: *arXiv preprint arXiv:1612.03928* (2016) (cit. on p. 135).

[306] Peyman Passban et al. “Alp-kd: Attention-based layer projection for knowledge distillation”. In: *Proceedings of the AAAI Conference on artificial intelligence*. Vol. 35. 15. 2021, pp. 13657–13665 (cit. on p. 135).

[307] Nikolaos Passalis and Anastasios Tefas. “Learning deep representations with probabilistic knowledge transfer”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 268–284 (cit. on p. 135).

[308] Byeongho Heo et al. “Knowledge transfer via distillation of activation boundaries formed by hidden neurons”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3779–3787 (cit. on p. 135).

[309] James Seale Smith et al. “Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 11909–11919 (cit. on p. 136).

[310] Zifeng Wang et al. “Learning to prompt for continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 139–149 (cit. on p. 136).

[311] Zifeng Wang et al. “Dualprompt: Complementary prompting for rehearsal-free continual learning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 631–648 (cit. on p. 136).

[312] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *ICLR*. 2014 (cit. on pp. 143, 178).

[313] Dustin Tran et al. “Plex: Towards reliability using pretrained large model extensions”. In: *arXiv preprint arXiv:2207.07411* (2022) (cit. on p. 147).

[314] Adam Tauman Kalai and Santosh S Vempala. “Calibrated language models must hallucinate”. In: *arXiv preprint arXiv:2311.14648* (2023) (cit. on p. 148).

[315] Katherine Tian et al. “Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback”. In: *arXiv preprint arXiv:2305.14975* (2023) (cit. on p. 148).

[316] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. “Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation”. In: *arXiv preprint arXiv:2302.09664* (2023) (cit. on p. 148).

[317] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744 (cit. on p. 148).

[318] Dan Hendrycks et al. “Measuring massive multitask language understanding”. In: *arXiv preprint arXiv:2009.03300* (2020) (cit. on p. 149).

[319] *20 Newsgroups*. https://github.com/aviralkumar2907/MMCE. Accessed: 2019-05-22. 2018 (cit. on p. 162).

[320] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014 (cit. on p. 162).

[321] *TreeLSTM*. https://github.com/ttpro1995/TreeLSTMSentiment. Accessed: 2019-05-22. 2015 (cit. on p. 162).

[322] Andrey Malinin, Bruno Mlodozeniec, and Mark John Francis Gales. “Ensemble Distribution Distillation”. In: *ArXiv* abs/1905.00076 (2020) (cit. on p. 179).

[323] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. “Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks”. In: *ICML*. 2020 (cit. on p. 179).

[324] Henry Gouk et al. “Regularisation of neural networks by enforcing lipschitz continuity”. In: *Machine Learning* 110.2 (2021), pp. 393–416 (cit. on p. 182).

[325] Wesley J Maddox et al. “A simple baseline for bayesian uncertainty in deep learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 13153–13164 (cit. on pp. 193, 194).

[326] Andreas Kirsch, Clare Lyle, and Yarin Gal. "Unpacking information bottlenecks: Unifying information-theoretic objectives in deep learning". In: *arXiv preprint arXiv:2003.12537* (2020) (cit. on p. 201).

[327] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999 (cit. on p. 203).

[328] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017) (cit. on p. 210).

[329] Jonathan Krause et al. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013, pp. 554–561 (cit. on p. 224).

[330] Fei-Fei Li et al. *Caltech 101*. en. 2022. URL: https://data.caltech.edu/records/20086 (cit. on p. 224).

[331] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests". In: *European Conference on Computer Vision*. 2014 (cit. on p. 224).

[332] Omkar M. Parkhi et al. "Cats and Dogs". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012 (cit. on p. 224).

[333] Maria-Elena Nilsback and Andrew Zisserman. "Automated Flower Classification over a Large Number of Classes". In: *Indian Conference on Computer Vision, Graphics and Image Processing*. Dec. 2008 (cit. on p. 224).

[334] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017) (cit. on p. 229).