Deep Learning for Industrial Applications



Gratianus Wesley Putra Data
Mansfield College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2023

Acknowledgements

First and foremost, I am extremely grateful to my supervisor Prof. Victor Prisacariu for his immense patience, dedication, and guidance in helping me realize this thesis. I am also deeply indebted to Gabriel, without whose unending support would make the completion of this thesis nigh impossible. I also express my gratitude to Prof. David Murray for his valuable guidance and advice early on in this journey. Finally, I thank my examiners Prof. Paul Newman and Prof. Michal Mackiewicz for their constructive input on the penultimate version of this thesis.

Special thanks to Prof. Robin Cleveland, Kristiana, and the Engineering Science staff for their crucial roles in enabling the submission of this thesis.

To those who provided endless food for thought and camaraderie: Ben, Ed, Henry, Jean-Francois, Kate, Marcelo, Michael, Ryan, Shuda, and Theo – thank you so, so much for all the illuminating discussions, cross-cultural exchange, leg-pulling, and debugging hours we've put ourselves through.

To friends at OUMS/OUSinf, especially Tom and Emily, thank you for giving me the privilege of experiencing some of the finest music-making Oxford has to offer.

I am also grateful to – amongst many others past – Arya, Widi, Cheryl, Jack, Ian, Julian, Liiyung, Laura, Billy, and especially Hesti for the significant mark they've made in my life. And last but not least, I can never express enough gratitude to my parents and my sisters Gaby and Gacie for their unwavering belief and unconditional love during this entire journey.

Abstract

Improvement of industrial quality control processes is paramount to support the ever-growing world population. A key part of this process has traditionally relied on classical vision techniques. With the recent success of deep learning in many areas of vision, it is envisioned that its application to this process may bring about significant improvements. The aim of this thesis is to develop and validate how a deep learning based solution may be incorporated into this process.

To this end, we considered how a deep learning solution can be integrated into a widely-used process architecture, and specified a system design that we implemented and validated in the lab.

We then identified data labelling to be a potential bottleneck in extending the solution to other applications, which led us to develop few-shot methods to address this. First, we introduced a method for interpolating batch normalization parameters across class-specialized networks for few-shot image classification. This significantly reduces overfitting compared to fine-tuning and allows for efficient adaptation, which we benchmark against CIFAR10 and ImageNet32. Second, we proposed a cosine-similarity-based method for online few-shot detection. Cosine similarity is used to produce detection scores of unseen exemplars against test candidates without retraining the network. We demonstrate this on the few-shot variant of ImageNet and VOC.

Contents

1 Introduction **1**

1.1 Aim, Scope, and Objectives 3

1.2 Chapter Outline 4

2 Background **6**

2.1 Object Detection 6

2.2 Few-shot Learning 9

2.2.1 Few-shot Classification 11

2.2.2 Few-shot Detection 15

2.2.3 Retrospective 17

3 Design and Implementation of a Deep Learning Engine **19**

3.1 System Review 19

3.2 Design Considerations 21

3.3 On System Architecture 23

3.4 On Image Acquisition 26

3.4.1 Camera and Lighting Hardware 27

3.4.2 Camera and Lighting Calibration 29

3.5 On Data Pipeline 31

3.5.1 Frame Overlap and Stitching 31

3.5.2 Inference and Ejector Mapping 32

3.5.3 Timing Analysis 35

3.6 On Gap Size . . . 38
3.7 Summary . . . 40

4 Experimental and Model Design **42**

4.1 Dataset Creation . . . 42
4.1.1 Labelling . . . 44
4.1.2 Splitting . . . 48
4.1.3 Synthetic Data Generation . . . 49
4.2 Task Framework . . . 51
4.2.1 Real-time Detection . . . 52
4.2.2 Training Techniques . . . 53
4.2.3 Model Selection . . . 55
4.3 Evaluation Method . . . 56
4.3.1 The Overall Task is Binary . . . 57
4.3.2 Non-exclusive Matching by Normalized Intersection . . . 58
4.3.3 Annotation Fusion . . . 60
4.4 Summary . . . 61

5 Experimental Validation **62**

5.1 Determining Deployment Settings . . . 62
5.1.1 Frame Height and Overlap . . . 63
5.1.2 Model, Frame Resolution, and Inference Time . . . 63
5.2 Results . . . 66
5.3 Labelling Efficiency . . . 68
5.4 Improving Label Efficiency . . . 70
5.4.1 Only Using Pure Data? . . . 70
5.4.2 Which Form of Synthetic Data? . . . 72
5.4.3 How Much Synthetic Data? . . . 75

5.5 Summary 75

6 Interpolating Convolutional Neural Networks Using Batch Normalization 77

6.1 Introduction 79

6.2 Related Work 81

6.3 Method 82

6.3.1 Component Generation 83

6.3.2 Binary Dataset Creation 84

6.3.3 Component Selection 84

6.3.4 Interpolating Component Networks 85

6.4 Experiments 87

6.4.1 Learning CIFAR10 from ImageNet Template 88

6.4.2 Few-shot Learning ImageNet32 from CIFAR10 Template . . . 90

6.5 Conclusion 94

7 Cos R-CNN for Online Few-shot Object Detection 96

7.1 Introduction 98

7.2 Related Work 100

7.3 Cos R-CNN 102

7.3.1 Exemplar Embedding Pathway 104

7.3.2 Cosine Comparator Head 104

7.3.3 Supervised Momentum Contrast (Su-MoCo) 106

7.4 Experiments 107

7.4.1 Few-shot ImageNet 108

7.4.2 Few-shot VOC 110

7.4.3 Ablations 113

7.5 Conclusion 116

8 Conclusion **117**

8.1 Summary of Contributions 117

8.2 Future Work 119

Bibliography **121**

List of Figures

2.1 Anatomy of a CNN-based object detector 7

2.2 Episodic training and testing 11

3.1 The system and its main components 20

3.2 System architecture of a standard engine 21

3.3 System architecture of the modified engine 24

3.4 Overview of the CNN Interface Board 25

3.5 Camera sensor spectral responsivity and foreground light power spectrum 28

3.6 Histogram and example capture 30

3.7 Frame overlap and stitching 32

3.8 Illustration of edge filtering criteria 34

3.9 Conversion of inference results into an ejector map 34

3.10 Fusing of overlapping regions in ejector maps 35

3.11 Timing diagram of the processing pipeline 37

3.12 Determination of gap size 39

4.1 Examples of pure and mixed frames 45

4.2 Illustration of steps in data labelling process 47

4.3 Examples of real (mixed) and augmented frames 50

4.4 Example of misestimation from multi-binary evaluation 57

4.5 Example of misestimation from exclusive IoU matching 59

5.1 Example of frames at 65-50 height-overlap 64

5.2 Example of improved detection at larger scales 65
5.3 ROC curves for models trained with varying amounts of training data 69
5.4 ROC curves of applications 71
5.5 Example crops of synthetic frames 73
5.6 ROC curves of applications trained with different combinations of synthetic data 74
5.7 ROC curves of applications trained with different proportions of synthetic data 76

6.1 Intuition for BN interpolation 79
6.2 Overview of BN interpolation method 83
6.3 Mean 1-shot validation accuracy of 3-component ComBN 93
6.4 75% threshold component, ComBN, and PCBN network validation accuracies 94

7.1 Three sets of distinct 3-way, 1-shot detection tasks 99
7.2 Overview of Cos R-CNN 103

List of Tables

3.1 Summary of main system parameters 41

5.1 Validation results of applications across different test image scales and models 64

5.2 Main results of removing defects 68

6.1 Percentage test accuracies on binary CIFAR10 datasets 88

6.2 Mean percentage validation accuracies on the 800 binary ImageNet32 datasets and their differences relative to full fine-tuning 91

7.1 AP50 scores on 5-way ImageNet LOC test 110

7.2 AP50 scores on 20-way VOC test 111

7.3 Ablations on the cosine box head reported as AP_{50} 114

7.4 Cos RPN embedder ablations reported as AP_{50} 114

7.5 Ablations on Su-MoCo AP_{50} performance on Split 1 of few-shot VOC 115

List of Algorithms

4.1 Synthetic data generation 50

Chapter 1

Introduction

[redacted]

[redacted]

[redacted]

[redacted]

Chapter 2

Background

The proliferation of deep learning techniques [4] in computer vision is widely attributed to the breakthrough image classification result [5] achieved in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2012) [27], where the authors were able to nearly halve the error rate of the previous state-of-the-art by using a deep convolutional neural network (CNN). This result kickstarted a major paradigm shift in the development of computer vision systems. In contrast to traditional vision which relied on meticulously hand-crafted descriptors such as Scale Invariant Feature Transforms (SIFT) [28], Histogram of Oriented Gradients (HOG) [29], and the bag-of-words model [30, 31], deep CNNs showed that more powerful feature image features/representations may be learned directly from data.

This chapter aims to provide background literature concerning topics covered in the thesis. Section 2.1 reviews major object detection literature and is relevant for Chapters 3, 4, 5, and 7. Finally, Section 2.2 reviews few-shot literature for Chapters 6 and 7 at the time the research material was written. We then close this section with a brief note on how the field of few-shot learning has progressed since.

2.1 Object Detection

An object detector takes in an input image, and produces predictions of where (localization) and what (classification) objects might be in the input. Numerous ar-

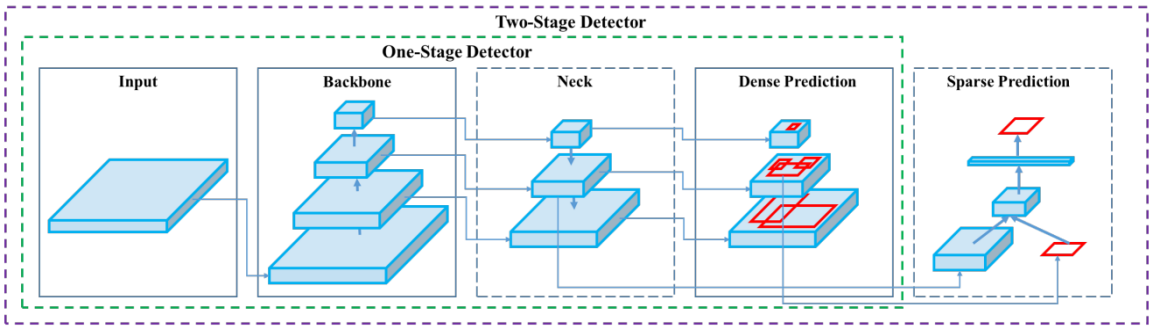


Figure 2.1: Anatomy of a CNN-based object detector. Figure reproduced from [36].

chitectural frameworks for object detection exist in literature [7, 32, 33, 8, 34, 35]. However, almost all of them (except [35]) follow the modular structure outlined in Fig. 2.1. When an input image is received, the model's backbone (and optionally with the neck) generates downsampled feature map(s) of the input image. In the case of a one-stage detector, the resulting feature maps are then passed onto head(s) which directly generate object predictions (class and bounding box dimensions) for each and every pixel[1] in the feature map. As the final predictions are generated from every feature map pixel, one stage detectors can be said to perform 'dense prediction'. Two stage detectors also follow and perform a similar dense prediction process, but instead of obtaining final object predictions, the results are region proposals containing candidate objects. Feature map regions corresponding to these proposals are then pooled and passed on to the final object prediction head(s), which classify and refine the initial region proposal boxes to final object prediction boxes. As only the most promising candidate region proposals are attempted to be predicted, two-stage detectors can be said to perform 'sparse prediction'.

The backbone of an object detector is considered as an integral part as it functions as the detector's primary feature extractor, and is strongly responsible for the performance (speed/accuracy) of the overall system [37, 36], making it an important design choice. Consequently, and given its modular nature, the backbone of a detec-

[1]More specifically, anchor boxes. A single pixel may have a few prior anchor boxes which may each result in a region proposal or object prediction.

tor usually borrows established, high-accuracy CNN designs from image classification literature such as VGG [38], ResNet [6, 39], CSPNet [40], and EfficientNet [41]. For other cases such as edge deployments, architectures which are also optimized for low latency are desirable. For these, backbone designs tend to rely on efficient structures such as ShuffleNet [42, 43], MobileNet [44, 45, 46], and FBNet [47, 48, 49]. Some models directly optimize/implement bespoke backbones specific for detection such as DarkNet [8] in YOLO.

Backbones traditionally output a feature map at a single scale/stride level (e.g., $\frac{1}{16}$ of the original input dimensions). However, a single scale feature map is suboptimal for detection of objects of a varying sizes, especially ones smaller than the feature map's stride. To rectify this and generate features at multiple scales (i.e., akin to a spatial pyramid [50, 51]), detectors may include a neck module, which generally work by pooling and concatenating features from later backbone stages with earlier stages.[2] The neck pathways boost semantic information in earlier stage features and allow them to be utilized as effective spatial pyramid levels. Examples of such modules in literature include SPPNet [52], FPN [53], PANet [54], and BiFPN [55].

As previously mentioned, object detectors may operate in a one-stage or two-stage framework. The heads of one-stage detectors such as SSD [32], YOLO [8, 36], and RetinaNet [33] directly process the backbone/neck features to produce bounding box and class predictions over all candidates (i.e., every anchor box/pixel in the feature map). In two-stage detectors such as Faster R-CNN [7], the backbone/neck features instead are first passed to a region proposal network (RPN). The RPN operates similarly to a one-stage head in that generates dense predictions. However, unlike a one-stage head, these predictions are not final, but instead are proposals of regions-of-interest (RoI) likely to contain objects. The network then takes the high-

[2]In theory, one may also simply collect earlier backbone features directly (without neck processing) to form multiple feature scales. In practice, however, this generally performs worse than utilizing a neck.

est scoring proposals, pools the backbone/neck features (which are cropped to the RoIs), and then resizes them to a pre-defined shape (either through RoIPool [56] or RoIAlign [57]). These RoIs are then passed to heads which then produce the final object predictions. In both one-stage and two-stage, these final predictions are then collected and then post-processed with a non-maximum suppression (NMS) criterion [58, 59] to clean duplicate (i.e., highly overlapping) box predictions.

Other than the generic framework outlined above, alternative detection paradigms also exist. For example, anchor-less networks such as CornerNet [60], CenterNet [61], and FCOS [34] omit anchor box priors and instead predict box keypoints directly from backbone/neck feature map pixels, which they argue simplifies training as this foregoes the need for tuning hyperparameters related to anchor box priors. Finally, in a radical departure from the established paradigm, there exists transformer-based [62] detectors such as DETR [35, 63]. In DETR, backbone/neck features are instead concatenated with pre-defined positional encodings, and are then passed through the model's transformer encoder. The transformer decoder then takes in as input a small and fixed number of learned positional encodings[3] (object queries) and attends to output of the transformer encoder. The resulting decoder output embeddings are then passed to a feed-forward module to generate object detections. In general, the accuracy (as measured by standard benchmarks such as COCO [64]) of these alternative frameworks are generally comparable to the generic framework.

2.2 Few-shot Learning

Deep CNNs typically require large datasets to train them, limiting their application to problem domains where labelled data is plentiful. Thus, the ability to train/adapt CNNs using as few data as possible, or few-shot learning [65], is desirable to broaden its applicability. While we identified one of the earliest works tackling this as far

[3]These are a separate set of encodings; i.e., not the pre-defined positional encodings assigned to the backbone/neck features.

back as 2006 [66], it was not until 2015 that interest began rising again, when it was shown that a hierarchical Bayesian program learning approach for few-shot character recognition is competitive with human-level performance [67]. Since then, a number of works, both within and outside the framework of CNNs, have attempted to solve few-shot learning.

A chief strategy deployed to tackle few-shot learning is meta-learning, i.e., learning to learn [68, 69]. This is often realized by optimizing a learning algorithm (the meta-learner) that optimizes the training of the model (the learner) to perform well on a variety of downstream tasks. The form of the learning algorithm could be a meta-optimizer that modulates the gradients of the learner (e.g., as in [68]), or it could be a parameter generator that modulates the learner weights to the few-shot task (e.g., as in [70]). Contrast this to traditional learning, where the learner is optimized using a fixed learning algorithm (e.g., SGD) for a single downstream task.

A common procedure used in training and evaluating meta-learning algorithms for few-shot learning is episodic training/testing. This is best explained by analogy. In standard training, the learner parameters are fitted to data points which are randomly and iteratively sampled from a dataset (representing a single task), such that the learner is able to generalize on new data points sampled from the same data distribution (i.e., same task). In episodic training, the meta-learner parameters are fitted to episodes which are randomly and iteratively sampled from a meta-dataset, such that the meta-learner is able to generalize the learner on new tasks from the same task distribution (i.e., same family of tasks). This is achieved by defining an episode as a proxy to a task: it contains training, validation, and test samples to evaluate that particular task episode.[4] In few-shot learning, each episode is a few-shot task (i.e., only few data points in the training split). Fig. 2.2 illustrates the construction of episodes for episodic training and testing.

[4]The training and validation/test samples within an episode are also referred to as the support and query samples, respectively

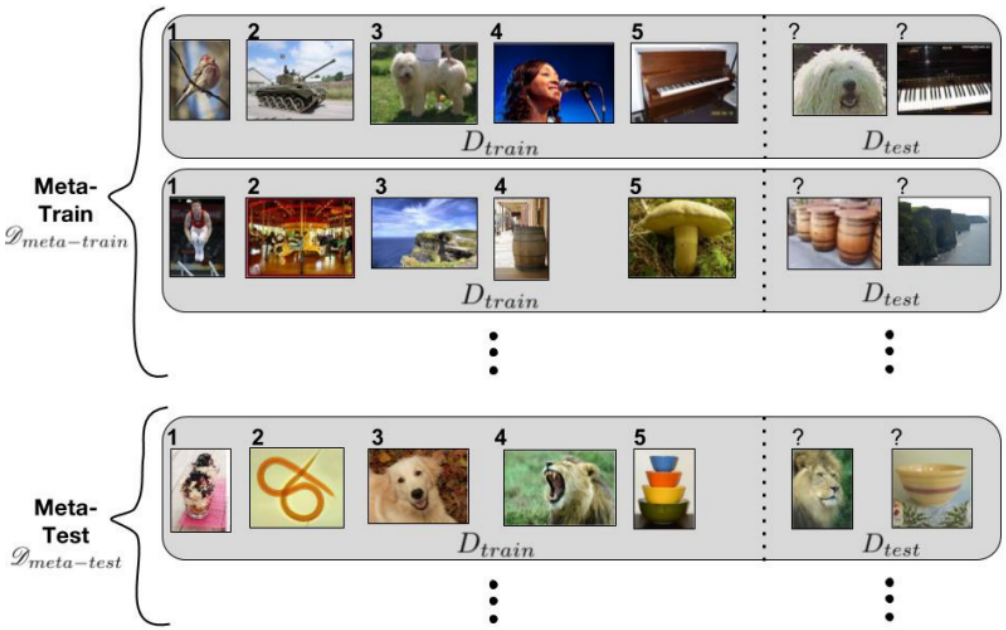


Figure 2.2: Overview of episodic training and testing for few-shot learning. A meta-learner is trained on a meta-dataset, which are a collection of episodes that are split into meta-train/val/test sets. An episode is a collection of data points that themselves are split into train/val/test sets. Figure reproduced from [69].

2.2.1 Few-shot Classification

Meta-learning for few-shot classification may be categorized into four main paradigms: metric-based learning, memory-based learning, learning optimizations, and learning parameterizations.

Metric Learning

In this paradigm, methods rely on learning a mapping of examples to an embedding space, using a distance function within that embedding to evaluate how examples relate to each other.

Koch *et al.* [71] trained a Siamese neural network to classify whether a pair of examples are of the same class. The network consequently learns to output a similarity score between the image pair. This capability is then exploited for one-shot learning by comparing a test image to images that are each single exemplars of a particular label class, where the test image is classified to the same class as the most similar

exemplar image.

Like the previous method, the matching networks of Vinyals *et al.* [72] propose embedding the test image and exemplar images using a neural network, and then matching the test image to the closest exemplar. However, unlike Koch *et al.* [71], proximity is quantified by cosine distance, which is akin to using a nearest-neighbour classifier on the embeddings. Also, the embedding functions of the exemplar images and test image are now decoupled, and are both conditioned on their counterpart inputs. One final important insight (that is seemingly obvious in hindsight) is the matching network training procedure: the network iterates through episodes of few-shot exemplar set/test image pairs, essentially going through what we know now as the meta-learning procedure of few-shot tasks.

In a similar vein, Snell *et al.* propose prototypical networks [73], which, unlike matching networks, specifically attempt to learn an embedding that clusters examples of a certain class around a central prototype. Few-shot classification then proceeds by matching the test example to the nearest prototype, conditioned on the support exemplars. We also note that the distance function is not restricted to cosine, and the authors found that using squared Euclidean resulted in better performance.

Relation networks by Sung *et al.* [74] extend the few-shot metric learning framework by learning the distance function itself using a CNN, which they term the relation module. This parameterization allows non-linear comparisons of the support and test image embeddings, which improved results over the previous two methods.

Memory-based Learning

Santoro *et al.* [75] have introduced Memory-Augmented Neural Networks (MANN) for few-shot classification. These are a variant of Neural Turing Machines (NTM) [76] with a modified memory writer module Santoro *et al.* termed Least Recently Used Access (LRUA). Unlike the NTM memory writer which can use both location- and content-based addressing, LRUA is constrained to just content-based addressing. This

is done as firstly, the non-sequential nature of few-shot classification benefits more from content-based retrieval, and secondly, reduction of complexity enables MANN to be trained more easily than NTM, which is a necessity in the few-shot setting.

Mishra *et al.* [77] proposed the Simple Neural Attentive Learner (SNAIL) architecture, a meta-learner which consists of interwoven layers of temporal convolution and soft attention blocks. Within a meta-learning perspective, temporal convolutions enable local, high-bandwidth access to learned information while soft attention blocks enable learned information to be extracted from a larger (potentially infinite) context. The authors argue that the complimentary functions of these two blocks enable SNAIL to be applied to a wide range of problem domains (i.e. not just few-shot learning).

Learning Optimizations

As an alternative to metric learning, few-shot problems may also be approached from the perspective of learning an appropriate optimizer. Ravi and Larochelle [69] showed that the formulation of LSTM cell state updates allude that of gradient descent. They then propose to train an LSTM learner network to produce gradient descent updates for a main classifier network. Using a similar meta-training framework to [72, 73], the gradient updates were conditioned such that the main classifier performs well on few-shot classification. Using this approach, they were able to improve upon the results of [72].

In another approach, Model-Agnostic Meta-Learning (MAML) by Finn *et al.* [68] proposes to train a learner that produces initializations for a network such that the network generalizes to any target task in a single gradient update. The learner is trained within a meta-learning framework using also gradient descent. They were able to obtain slightly better results than [69].

Hariharan and Girschick [78] proposed a novel loss function that, similarly to MAML, encourages a CNN to produce representations that generalize better to new

tasks. However, unlike MAML, their set-up does not require a meta-learning to train the network. As such, they were able to evaluate their method on the entirety of ILSVRC-2012 by holding out some classes to be used in a few-shot manner. They also proposed a way of hallucinating additional examples of the few-shot task as a form of data augmentation, which improves the few-shot accuracy of their method.

Learning Parameterizations

Bertinetto *et al.* [70] devised a learner that dynamically predicts convolutional parameters of another network, where the predicted parameters are conditioned on an exemplar fed into the learner. They demonstrated the flexibility of the learner by having it dynamically predict the convolutional parameters of both branches in a Siamese network in the style of [71], and having an asymmetric Siamese variant where the one branch assumes the role of the learner and predicts convolution parameters for the other branch.

Our work in Chapter 6 may also be viewed as trying to learn parameterizations. Inspired by the demonstration that batch normalization (BN) can be used to efficiently adapt a classifier to multiple domains [79, 80], we examine whether the same idea can be applied to object classes. We proposed building a dictionary of BN components representing specific object classes, and interpolated between them to efficiently learn unseen classes. Two options for interpolation were presented: ComBN, where interpolation is formulated as a linear combination of BN components, and PCBN, where BN components are utilized to deduce principal directions where optimize BN parameters for the new task are optimized in. We showed that utilizing either ComBN or PCBN allowed us to obtain about 4-17% better accuracy than straightforward fine-tuning on few-shot datasets.

2.2.2 Few-shot Detection

Chen *et al.* introduced LSTD [81], which was a pioneering method in few-shot detection. They investigate the few-shot transfer learning setup, where there exists a source base set of classes with abundant data, and a novel set with few data. The ideal few-shot detector then would perform well on both base and novel classes. To address this, the authors introduce a hybrid SSD/Faster R-CNN model framework where an SSD-style backbone generates a feature pyramid (and box coordinate proposals) that is passed to Faster R-CNN RPN (for proposal objectness classification) and ROI heads, which they empirically show to be better than the individual frameworks on their own. Next, they propose a two-stage training process where the detector is first trained normally on the abundant base classes, and then fine-tuned on the few novel classes. To alleviate overfitting on the few-shot classes, they introduced two regularization terms into the overall loss function in the fine-tuning stage: Background-Depression (BD) regularization and Transfer-Knowledge (TK) regularization. BD regularization aims to suppress non-object activations by performing ridge regression over background features (i.e., L2 norm loss over non-object ROI features). TK regularization aims to recover the base class classifier weights during fine-tuning, by distilling [82] classifier logits (using novel data) obtained from the pre-fine-tuned detector.

In a follow-up work, Kang *et al.* [83] proposed Fewshot-YOLOv2 using YOLOv2 as a base detector. They propose to achieve few-shot adaptation by meta-learning a channel-wise feature modulator that re-weights incoming features of a query image according to a set of few-shot support classes, which is similar to TADAM [84] for few-shot classification. Like LSTD [81], a two-stage training procedure is followed where the detector is first trained on abundant base classes before being fine-tuned on the scarce novel classes. However, unlike LSTD, Fewshot-YOLOv2 splits the base classes into episodes of support and query sets. This enables them to meta-learn the feature modulator. During the few-shot fine-tuning stage, both the base and novel classes are

used to construct episodes, but the total number of training iterations is limited to prevent overfitting. To mark the object location in the support image, the input to the feature modulator is concatenated with a binary mask highlighting the support object location. Additionally, unlike in YOLOv2 where the classification is multi-label (i.e., n binary classifiers), Fewshot-YOLOv2 performs multi-class classification by applying a softmax function over the class logits (i.e., a single n-way classifier), as they found softmax to perform better in the few-shot regime.

Around the same time, Yan *et al.* [85] proposed Meta R-CNN. The authors apply meta-learning to a Faster R-CNN framework by introducing a feature modulator they dubbed the Predictor-head Remodelling Network (PRN). Similar to FewShot-YOLOv2's feature modulator, the PRN modulates ROI proposal features conditioned on a support set. However, the PRN shares most of the backbone weights of the original detector instead of being its own separate module. Additionally, they add a cross-entropy loss term over the modulator output, which they show to induce the output to cluster semantically, improving performance for both base and novel classes.

Simultaneously and similarly, Wang *et al.* [86] also propose a meta-learning framework called MetaDet for object detection. However, unlike the two previous works, they argue for a framework-agnostic formulation where detectors may be classified into class-agnostic and class-specific parts. The class-agnostic parts (e.g., Faster R-CNN backbone and RPN) are to be trained using all available base class data. Then, they perform episodic training on the class-specific parts (e.g., Faster R-CNN heads), which also additionally includes a weight prediction meta-model. This definition allows them to demonstrate their method on both one-stage (YOLO) and two-stage (Faster R-CNN) detectors since the meta-model may simply be attached to the existing frameworks. Additionally, instead of modulating features, the meta-model they proposed directly predicts category-specific weights for both the localization and classification of objects (i.e., it is a parameter-generating meta-model).

More recently, Wang *et al.* in FSDet [87] demonstrated that a simple, non-meta-learning strategy of few-shot fine-tuning only the last detector layers (i.e., the box classifier/regressor) of a Faster R-CNN is competitive or better than previously introduced meta-learning-based few-shot object detection methods. This applies to either a fully-connected or cosine similarity classifier layer. They also investigate previous benchmarks and demonstrate high variance due to inherent susceptibility to randomness of support example selection in the meta-testing stage, and proposed a revised benchmark with repeated meta-testing runs to reduce this variance.

In slight contrast to previous methods, Karlinsky *et al.* propose a metric-learning based approach to few-shot detection with RepMet [88]. Using a Faster R-CNN framework as a base, they propose to learn an embedding module that models object class distributions by mixtures of Gaussians, and employ a triplet loss to learn to generate the Gaussian parameters. Naturally, this is also trained in an episodic manner. At inference, the query object class is then simply the maximum (i.e., nearest) posterior class probability amongst the support set mixtures generated by the embedding module.

Our work in Chapter 7 also address the few-shot detection problem, but with the added constraint of online adaptation (i.e., no fine-tuning during meta-testing), as we envision online detection to be more suitable for resource-constrained edge deployments. Within the body of previous work, only RepMet's [88] design directly addresses this setup. In comparison to RepMet, we find that our simpler cosine-distance based formulation obtains better results on the ImageNet 5-way and VOC few-shot detection benchmarks.

2.2.3 Retrospective

Since our contributions outlined above, numerous successive works have further investigated and improved few-shot classification with more novel adaptation meth-

ods, e.g., [84, 89, 90, 91, 92, 93, 94]. With improved methods, the community has also progressively introduced more challenging benchmarks. For example, tieredImageNet [95] is a larger and more challenging version of miniImageNet introduced in pioneering works [72], and Meta-Dataset [96] contains a set of datasets from disparate domains. More recently, few-shot literature has expanded to explore the transductive [97] evaluation setting, where during meta-testing, both support set data and labels are exploited in tandem with the query set data. This generally enables better predictions to be made as the query set data distribution is then known at test time.

Another factor for improving few-shot performance is, perhaps unsurprisingly, stronger feature representations (i.e., bigger network backbones). For example, Chen *et al.* [98] show that simple classifiers have competitive few-shot performance with more sophisticated methods as backbones get larger. A more extreme demonstration of this exists in foundational model literature [99, 100], where extreme large vision and language models have been demonstrated to possess remarkable few-shot or even zero-shot capacity.

In few-shot detection, there is still ongoing exploration for both transfer learning and meta-learning approaches [101, 102, 103, 104, 105]. Sun *et al.* [106] incorporate a self-supervised contrastive loss on ROI features which improve few-shot classification on object proposals. Qiao *et al.* [107] propose a module to decouple gradient flow between RPN and ROI head modules during training as the class-agnostic and class-specific nature of the respective modules result in suboptimal features for the few-shot setting. Kaul *et al.* [108] introduce a pseudo-labelling method to enable sourcing and re-use of unlabelled training instances present in the base set data for training novel classes.

Chapter 3

Design and Implementation of a Deep Learning Engine



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]



[redacted]



[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

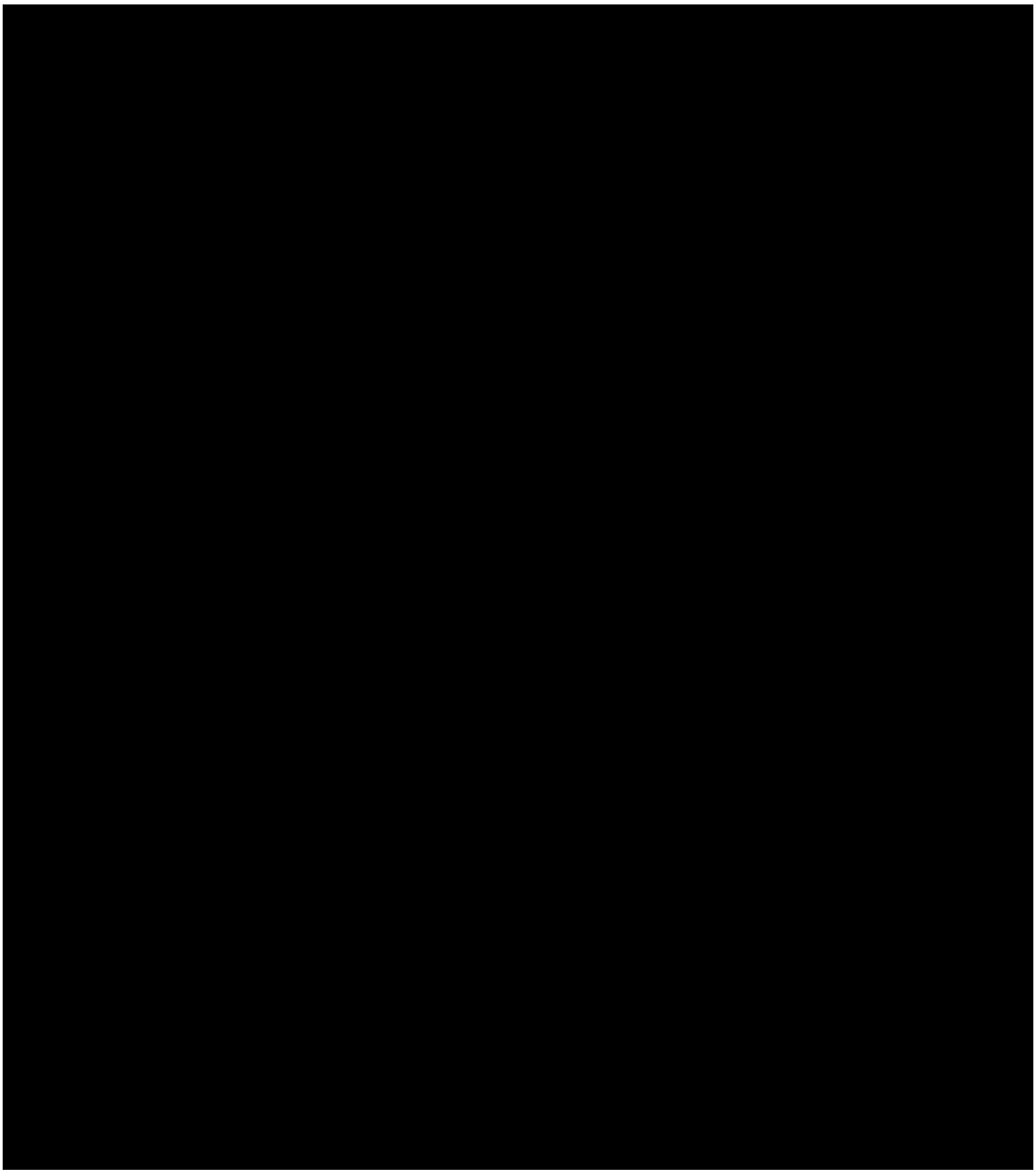
[redacted]

Chapter 4

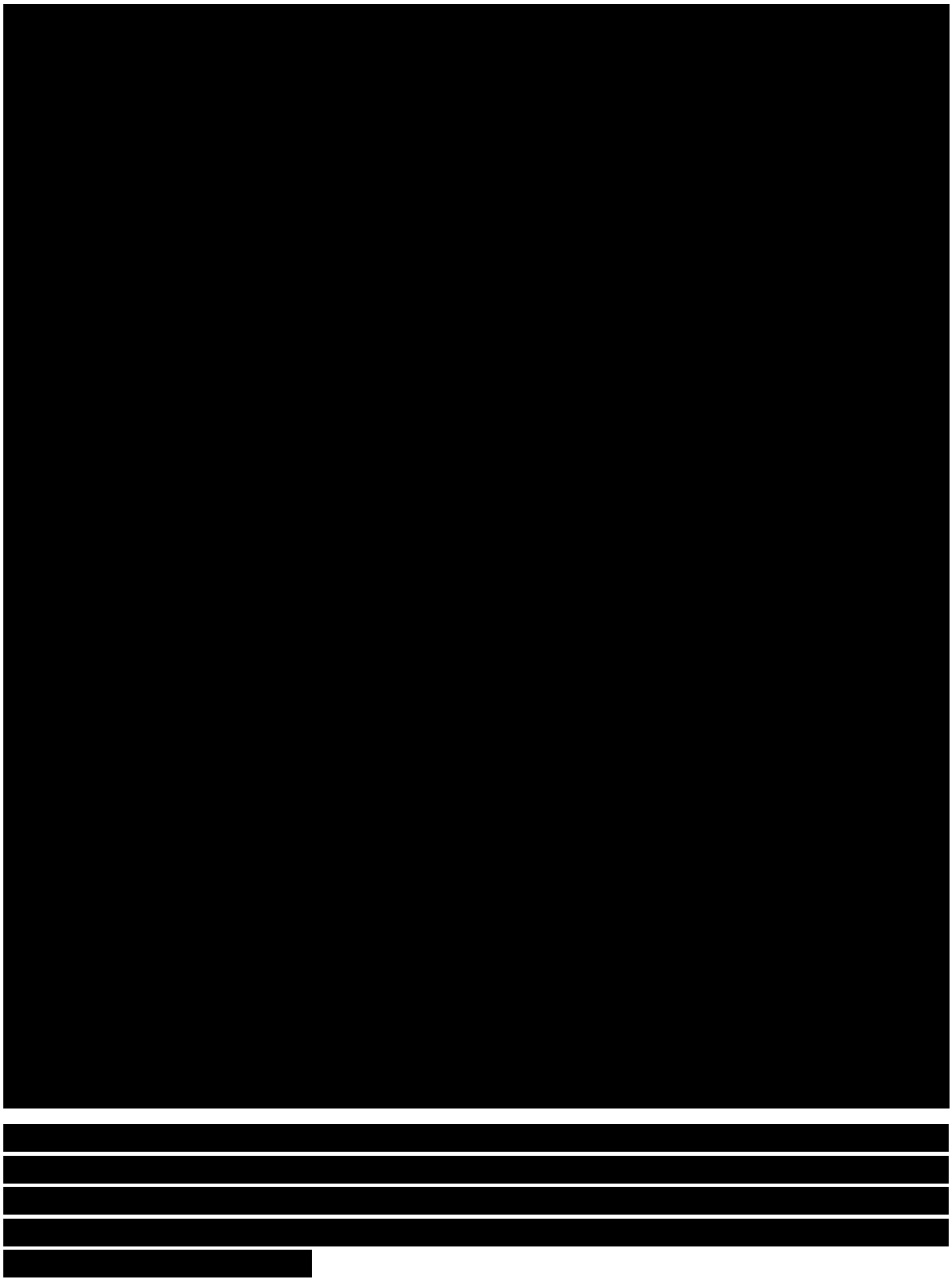
Experimental and Model Design

[redacted]

[redacted]

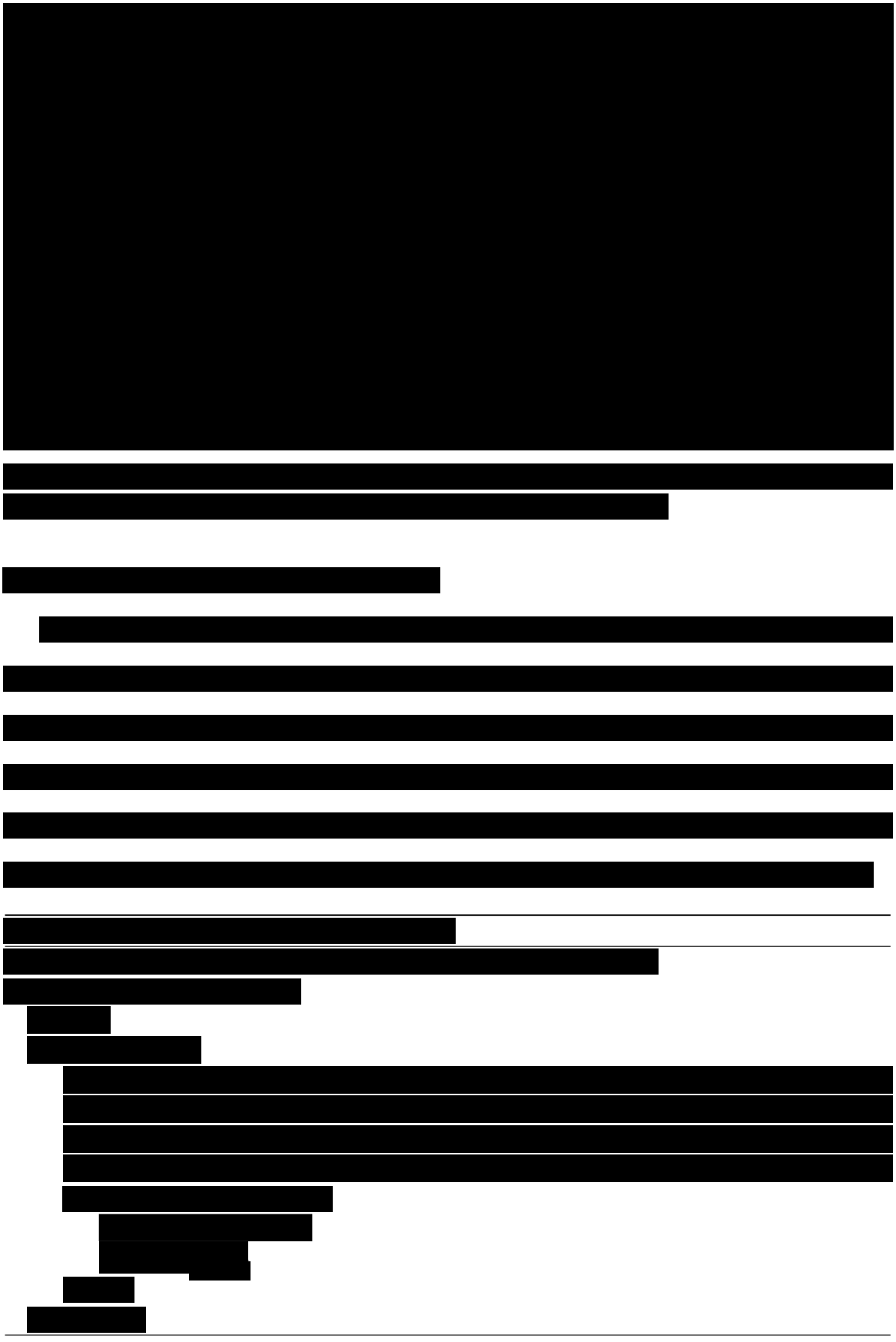


[redacted]



[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]



[redacted]

[redacted]

Chapter 5

Experimental Validation

[redacted]





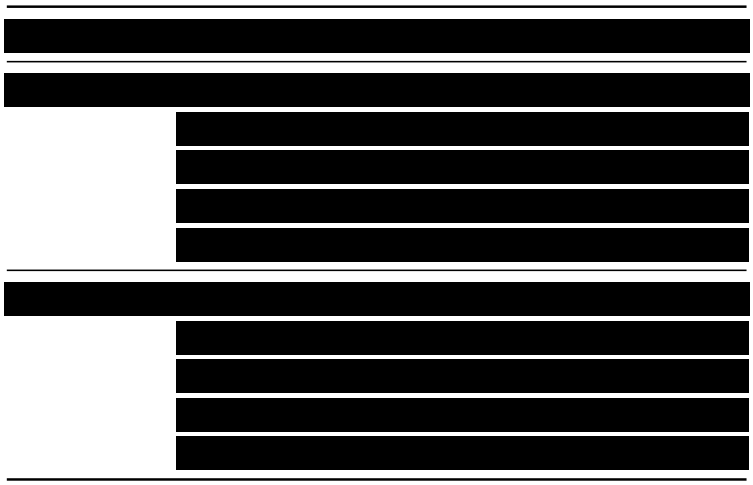
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]





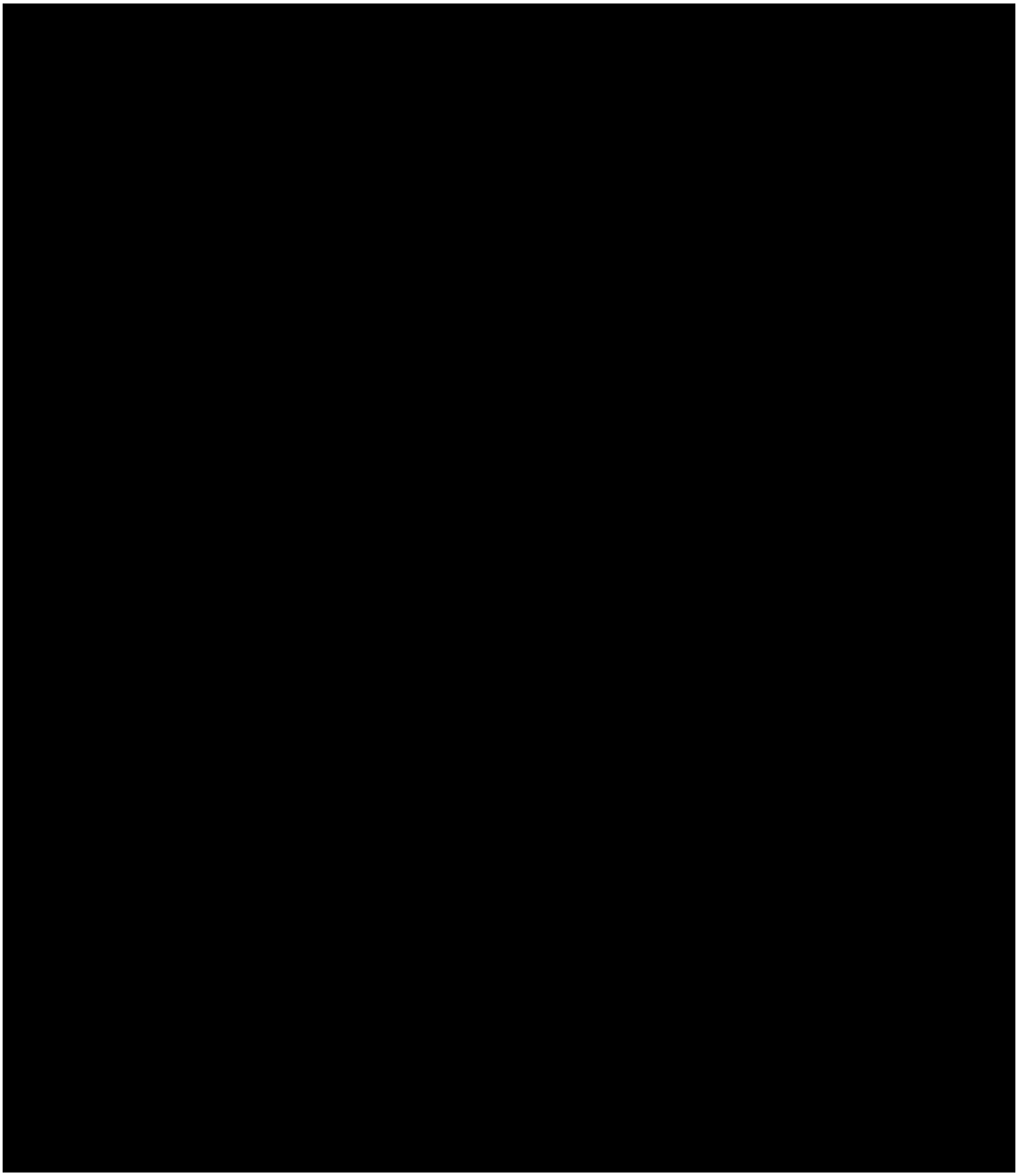
[redacted]



[redacted]

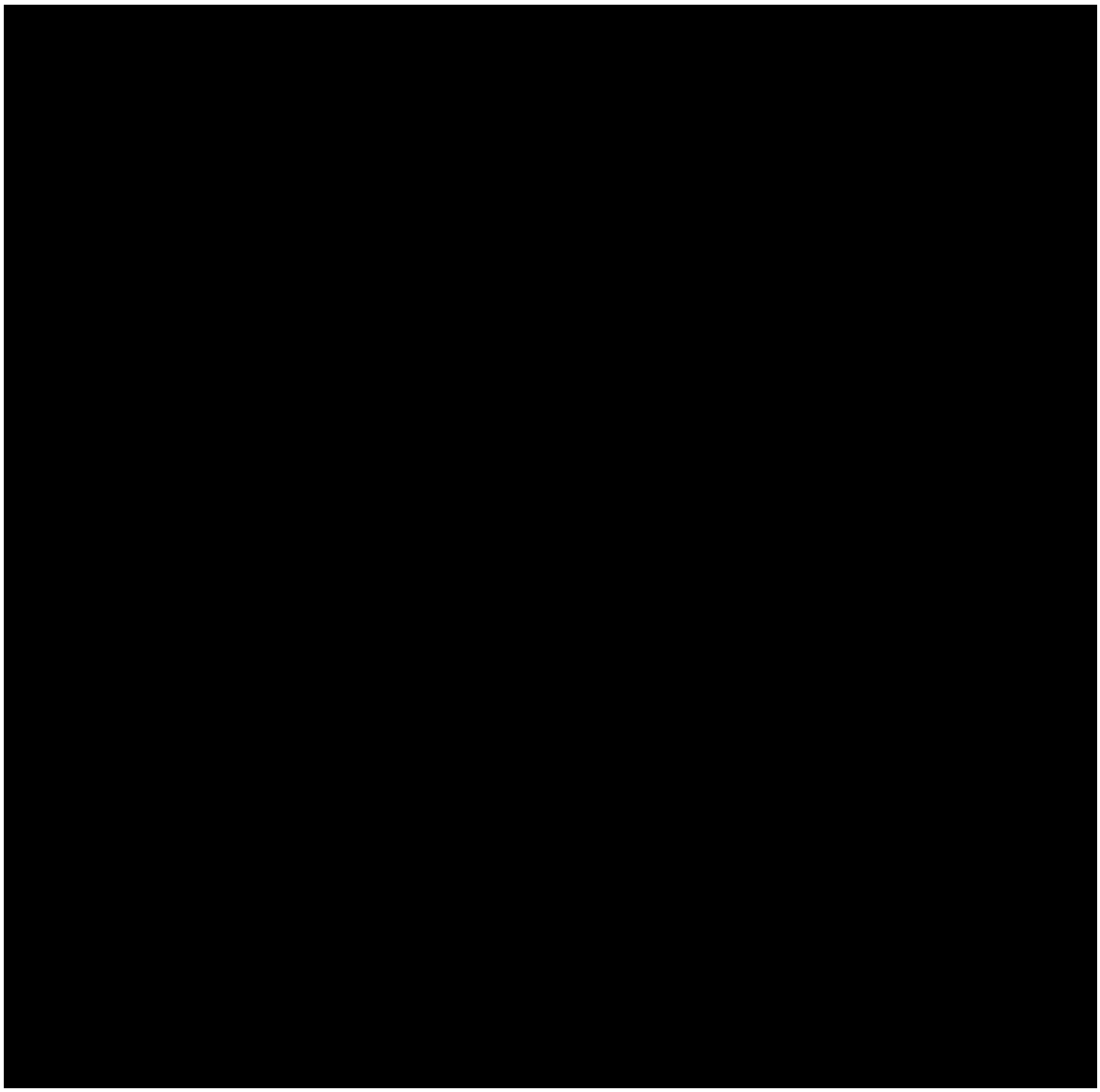


[redacted]





[redacted]



Chapter 6

Interpolating Convolutional Neural Networks Using Batch Normalization

In the previous chapters, we have developed and explored a production-oriented system optimized at specific applications. With data efficiency being a major consideration in building such applications, it is potentially fruitful to look into more cutting edge research to address this. One promising solution is few-shot classification, which we tackle in this chapter with the future aim of being able to apply such methods to improve the production system's data efficiency (Section 5.4). This chapter is the result of such work, where we proposed to utilize batch normalization to regularize fine-tuning of neural networks in the few data regime, and it was originally published in ECCV 2018 [25]. It is thus structured to be self-contained and with a literature review that is relevant at the time of publication. This chapter also lies in the non-confidential part of the thesis, and so will not contain references to work in the preceding chapters.

Abstract

Perceiving a visual concept as a mixture of learned ones is natural for humans, aiding them to grasp new concepts and strengthening old ones. For all their power and recent success, deep convolutional networks do not have this ability. Inspired by recent work on universal representations for neural networks, we propose a simple emulation of this mechanism by purposing batch normalization layers to discriminate visual classes, and formulating a way to combine them to solve new tasks. We show that this can be applied for 2-way few-shot learning where we obtain between 4% and 17% better accuracy compared to straightforward full fine-tuning, and demonstrate that it can also be extended to the orthogonal application of style transfer.

6.1 Introduction

Human visual cognition is remarkable. One of the many things humans do naturally is linking visual concepts to a combination of other concepts. For example, after being shown images of a dog, a cat, and a fox, a child could say that the fox looks like a cross between a cat and a dog (Fig. 6.1). Furthermore, the child will understand much about the concept of a fox given prior knowledge of what cats and dogs look like. A loose mathematical analogy can be expressed as follows: if visual representations of cats and dogs can be encapsulated in the form of functions ϕ_{cat} and ϕ_{dog}, respectively, it should also be possible to build from them a representation for foxes $\phi_{\text{fox}} = f(\phi_{\text{cat}}, \phi_{\text{dog}}, \alpha)$, where f represents how the functions should be combined as parameterized by α. Additionally, it should be easier to deduce the value of α than ϕ_{fox} directly.



Figure 6.1: Intuition for our BN interpolation method. The fox can be seen as a mix between this cat and this dog.

In this chapter we ask if the same ideas could be adapted to deep convolutional neural networks to enable more efficient learning. This is desirable as, despite generally being powerful state-of-the-art models [5, 38, 6], deep networks require a tremendous amount of data to tune millions of parameters that allow it to work so well, limiting its application to tasks where data is plenty or inexpensive.

Recent works [79, 80] have reported that it is possible to prepare a single network that is able to perform visual recognition in multiple domains. This is achieved by training the network to produce universal image representations, relying on (i) the convolutional kernels to extract domain-agnostic information about the world and

on (ii) the batch normalization (BN) layers to transform the internal representations to the relevant target domains. Analogously, within the application domain of style transfer [148], shows that a single network can be equipped with multiple distinct styles by encoding the style information in the network's instance normalization (IN) layers, after which each style can selectively be applied to a target image. These discoveries seem to provide evidence for the ability of normalization layers to encode transforms that can be used to express visual concepts.

In line with our opening exposition, we propose and wish to test the following intuition in this chapter: given that normalization layers (e.g., BN) can be trained to discriminate specific visual classes, it should be possible to combine these normalization layers and interpolate within them to efficiently learn new, unseen classes. In particular, since we will only be manipulating the normalization layers within a network, the number of parameters that we need to tweak will be much lower than full fine-tuning. Fewer parameters also mean less tendency to overfit, enabling training with smaller amounts of data. Focusing on binary classification tasks, we summarize our contributions in this chapter as follows:

1. Defining a procedure that specifies how basis networks that discriminate specific classes are generated and interpolated to discriminate new, unseen classes.

2. Demonstrating how basis interpolation can be applied for few-shot learning.

The remainder of the chapter proceeds as follows: we first mention several works that are related to our method in Section 6.2. Afterwards, we describe and elaborate procedures for creating and interpolating between component BNs in Section 6.3. We validate the key idea of interpolating BNs on CIFAR10 [149], apply the same procedures to few-shot learn ImageNet32 [150] using CIFAR10-trained kernels, and also apply it to the orthogonal task of style transfer (by replacing BNs with INs) in Section 6.4. Finally, we conclude our chapter in Section 6.5.

6.2 Related Work

To our knowledge, we are the first to tackle the problem of neural network interpolation. Our main reason for attempting this is to reduce the number of parameters required to train the neural network and to achieve faster convergence with fewer images. We therefore believe our approach is most related to other works that (i) try to reduce the number of trained/tested parameters, (ii) dictionary learning, and (iii) few-shot learning/meta-learning approaches.

Parameter Reduction

Within the realm of parameter reduction, there have been attempts to compress and distil knowledge in neural networks [82], and novel designs for efficient architectures which reduce the number of parameters during inference [151, 152]. These assume the neural network is trained traditionally and provide methods by which the post-training parameters can be reduced, e.g., through some form of sparsity. Reduction of training parameters is however much less studied, with the traditional approach looking at training only a subset of the complete network, e.g., the last few layers. However, recent work has shown an alternative strategy of training a neural network (rather than just retuning the last layers): to adapt the network's batch normalization parameters. This proved to be effective when training and adapting domain-agnostic neural networks in [79, 80] and, relatedly, when aiming to adapt existing neural networks to new types of style transfer as in [148].

Dictionary Learning

The aim of dictionary learning [153] is to learn fundamental representations from data that can be combined linearly to construct sparse codings of the data. A collection of these fundamental representations (*atoms*) form a *dictionary*. A few well-known algorithms that perform dictionary learning include the method of optimal directions [154]

and K-SVD [155].

Few-shot Learning/Meta-learning

The application of deep convolutional networks for few-shot learning has recently seen a resurgence. Naming just a few methods, Koch *et al.* [71] used Siamese networks to quantify distances between samples, and then used a non-parametric classifier such as k-nearest neighbours to perform one-shot learning. Bertinetto *et al.* [70] modified the Siamese architecture to enable the first network to predict suitable weights for the other in the one-shot regime. Hariharan and Girschick [78] proposed the SGM loss and hallucination as data augmentation to perform n-way few-shot learning where n is large. Luo *et al.* [156] suggested a network framework that is able to learn transferrable representations in a label-efficient manner.

Within the framework of meta-learning, Vinyals *et al.* [72] introduced the concept of episodic training to ensure few-shot training and testing conditions match, and used a cosine similarity metric on network embeddings to perform the classification. Ravi and Larochelle [69] used an LSTM meta-learner to directly perform episodic weight updates on a few-shot learner network, made possible by the similarity between LSTM and gradient descent update formulation. Snell *et al.* [73] utilized a network to learn embeddings which cluster classes around prototypes, which then classifies new examples by proximity to learned prototypes. Finn *et al.* [68] proposed a simple meta-learning training algorithm that aims to generate good initialization parameters for a classifier network, which was then able to achieve good performance after a single parameter update step.

6.3 Method

The following subsections develop and explain the BN interpolation method. Fig. 6.2 shows an overview of the whole method.

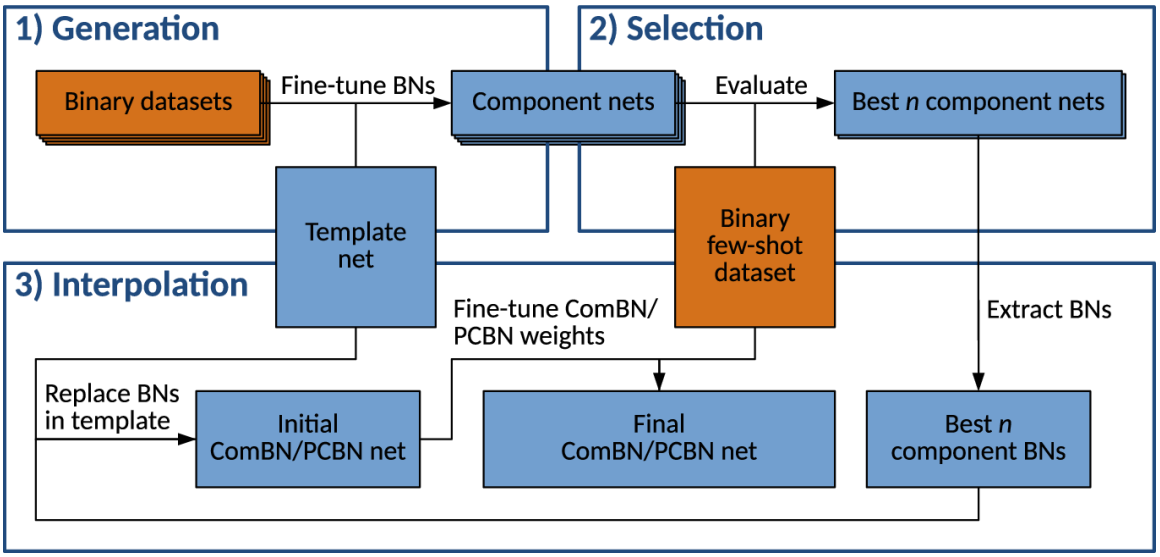


Figure 6.2: An overview of the BN interpolation method.

First, we briefly review the batch normalizing (BN) transform [24]. Let x_i be the activations of a single example i inside a mini-batch of size m. The BN transform is defined as the operation,

$$\mathrm{BN}(x_i) \equiv \gamma \widehat{x}_i + \beta, \tag{6.1}$$

given the mean $\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$, variance $\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$, and the normalized input $\widehat{x}_i = (x_i - \mu_{\mathcal{B}})/\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}$. The scale γ and shift β parameters are learnable, while ϵ is a small positive constant to prevent division by zero.

Next, we will show how component BNs are constructed, and detail two ways of combining them for the purpose of interpolating new classes. This section focuses on the binary classification scenario, but the principles presented can translate to other application domains, as we show later in Section 6.4.

6.3.1 Component Generation

Given that component BNs are purposed to be discriminative towards a particular object class, a straightforward way to generate them for that object class would be to extract BNs from a network trained on a corresponding binary classification task.

More specifically, we start from a base pretrained network which we refer to as the

template network. To create BN layers that detect the concept of e.g., cat, we fine-tune the network on a dataset containing examples of cats and non-cats by adjusting only the BN and last classification layer parameters. This fine-tuned network is now a component network that detects cats. We repeat this procedure for other object classes, always starting from the same template network, until the desired number of component networks is obtained.

The number of component networks is a function of the task at hand and the quantity of available data and classes. For example, for our experiments, we create 9 component networks for CIFAR10 and 200 for ImageNet.

6.3.2 Binary Dataset Creation

It is beneficial to generate a number of component BNs so that a good coverage of the target task is achievable. Large multi-class datasets are naturally a suitable source. However, they need to be binarized before they can be used for component generation.

We formalize this as follows. Suppose we have a set of N labelled images $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $y_n \in \{1, \ldots, K\}$ are the labels. Binarizing $\mathcal{D}$ for class k means randomly copying $S/2$ elements of $\mathcal{D}$ where $y_n = k$ and another $S/2$ where $y_n \neq k$ to form a new set $\mathcal{D}_k = \{(x_1, y_1), \ldots, (x_S, y_S)\}$ where $y_s \in \{0, 1\}$ after applying the binary label transform

$$\phi(y_n) = y_s = \begin{cases} 1 & \text{if } y_n = k, \\ 0 & \text{if } y_n \neq k. \end{cases} \tag{6.2}$$

6.3.3 Component Selection

Once a number of component BNs have been generated, it is important to select the components that will be relevant for the task at hand (e.g., aeroplanes are probably not a good component to include when trying to detect foxes). Although there may exist sophisticated selection methods, we propose two straightforward criteria which we demonstrate in more detail in Section 6.4.2. The first criterion involves selecting m

component networks with the lowest cross-entropy loss on the target binary task. The second criterion does the same thing, except that it ranks based on highest accuracy on the target binary task. Naturally, the first criterion is more amenable to tasks with few examples to evaluate.

6.3.4 Interpolating Component Networks

After the BN components have been computed and selected for a specific novel target class, we propose two approaches for interpolation:

1. Composite Batch Normalization (ComBN), providing a linear combination of BN components.

2. Principal Component Batch Normalization (PCBN), providing a PCA-based latent space interpolation.

The interpolation weights for both approaches are learned through standard neural network optimization techniques, i.e., backpropagation and stochastic gradient descent (SGD).

Other methods could be used to interpolate the BN components, such as more complex non-linear dimensionality reduction techniques like Gaussian Process Latent Variable Models [157], but, as we show in the results section, the simpler linear models already achieve very good results.

Composite Batch Normalization (ComBN)

Using a similar notation to the one above, we propose the ComBN transform as a linear combination of generated BN components,

$$\text{ComBN}_{\alpha}(x_i) \equiv \sum_{j=1}^{J} \alpha_j \text{BN}_j(x_i), \tag{6.3}$$

where J is the number of BN components that make up a ComBN, and α_j are learnable scalar coefficients that represent the interpolation weights, all initialized to $1/J$.

In practice, after the component networks have been generated, each BN layer in the original template network is replaced with a ComBN, which is constructed from BN layers of selected component networks originating from the same depth-wise layer position. Afterwards, we train the ComBN network by optimizing α_j and the last layer to the target task using standard techniques (i.e., backpropagation and SGD).

Note that the component BNs in the ComBN network are always utilized in inference mode; i.e., their γ, β, running mean, and running variance are frozen, and the running mean and variance are used in place of the mini-batch mean $\mu_{\mathcal{B}}$ and variance $\sigma^2_{\mathcal{B}}$ when evaluating Eq. 6.1.

Additionally, this formulation typically enables a large reduction of the number of parameters, which is helpful in reducing overfitting when the training data is scarce.

Principal Component Batch Normalization (PCBN)

An alternative way to exploit information contained in BN components is to first use them to learn a latent space mapping for its parameters, and then perform optimization in the latent space.

To achieve this using PCA, we first stack row vectors of γ and β parameters that originate from each BN component j to form $J \times C$ matrices $\mathbf{\Gamma}$ and $\mathbf{B}$, respectively, where J is the number of components and C is the number of channels in each component BN layer. We then mean-centre $\mathbf{\Gamma}$ and $\mathbf{B}$ by subtracting from them their column-wise mean vectors μ_γ and μ_β, resulting in $\mathbf{X}_\gamma = \mathbf{\Gamma} - \mu_\gamma$ and $\mathbf{X}_\beta = \mathbf{B} - \mu_\beta$. Afterwards, we apply singular value decomposition to obtain principal axes matrices $\mathbf{V}_\gamma^\top$ and $\mathbf{V}_\beta^\top$,

$$\mathbf{U}_\gamma \mathbf{S}_\gamma \mathbf{V}_\gamma^\top \leftarrow \mathbf{X}_\gamma, \tag{6.4}$$

$$\mathbf{U}_\beta \mathbf{S}_\beta \mathbf{V}_\beta^\top \leftarrow \mathbf{X}_\beta. \tag{6.5}$$

The number of dimensions of our latent space is set to the maximum possible, i.e., $\min(J, C)$. We then train latent space parameter vectors g and b (initialized

by transforming existing BN weights of the template network to latent space), and transform these back to parameter space using the principal axes matrices,

$$\gamma = g\mathbf{V}_{\gamma}^{\top} + \mu_{\gamma}, \tag{6.6}$$

$$\beta = b\mathbf{V}_{\beta}^{\top} + \mu_{\beta}. \tag{6.7}$$

This is then finally applied similarly to ComBN by replacing BN layers in the original template network with PCBN (*i.e.,* substituting Eq. 6.6 and 6.7 into Eq. 6.1). In essence, this is like using standard BN except for the optimization of parameters in latent space.

In contrast to ComBN where interpolation is directly performed in the parameter space of the original component class (in the form of frozen BN components), here we attempt to first distil the concepts of class into principal classes in latent space before optimizing them.

6.4 Experiments

For this section, we show results for two application domains: visual classification and style transfer.

We choose to constrain our experiments to binary classification so that the same protocol can be used for both component generation and evaluation. To highlight the contribution of BN layers, we utilize a template network trained on a different dataset to the one we are testing with; first using ImageNet for the template and CIFAR10 for the testing of our approach, and second using CIFAR10 to train the template network and ImageNet32 for testing.

We chose style transfer because (i) we view this as an orthogonal (i.e., related but highly distinct) task to binary classification since it requires utilization of full encoder-decoder networks and replacement of batch normalization with instance normalization, and (ii) it allows us to produce qualitative results.

Positive class	Last	Full	BN	ComBN	PCBN	BN*	ComBN*	PCBN*
aeroplane	90.1	**96.1**	95.8	81.9	93.8	76.8	77.2	95.4
car	92.0	**98.0**	97.3	97.7	97.4	77.7	79.8	97.1
bird	87.5	**95.5**	94.5	93.8	91.4	66.7	68.3	93.5
cat	84.3	91.7	89.6	**91.9**	90.3	70.6	69.8	88.9
deer	86.0	**96.6**	94.8	94.3	94.0	71.7	73.1	93.5
dog	89.1	93.1	93.3	**93.9**	93.7	71.4	70.9	93.1
frog	92.3	97.3	95.9	**97.4**	96.0	77.6	77.8	97.0
horse	91.2	**96.8**	96.0	96.2	95.3	67.4	68.5	95.1
ship	91.2	**97.3**	96.9	97.0	95.8	77.8	78.7	96.0
truck	92.9	**97.3**	96.5	90.1	96.2	76.4	73.7	95.9
mean	89.7	**96.0**	95.1	93.4	94.4	73.4	73.8	94.6

Table 6.1: Percentage test accuracies on binary CIFAR10 datasets. Asterix (*) indicates results that were based on a random template network. Best results are in bold.

6.4.1 Learning CIFAR10 from ImageNet Template

Here we validate the idea of using BN components for training networks on new, unseen tasks. Lastly, for the experiments in this section, we base our template network on an ILSVRC2012-pretrained ResNet34 [6], and use binarized CIFAR10 datasets to generate our BN components and evaluate the performance of ComBN and PCBN networks.

We begin by creating master training/validation/test splits. These master training/validation splits are created by partitioning the original CIFAR10 training set 40,000/10,000, while the master test split is the same as the original CIFAR10 test set. Afterwards, we generate binarized splits by applying the method in Section 6.3.2 to each of the master splits. Specifically, for each target class, the binary training split is formed by sequentially sampling 1000 positive and 1000 negative examples from the master split, while the binary validation/test splits are formed by sequentially and exhaustively sampling all available examples from their respective master splits such that a balanced dataset is obtained. All experiments pertaining to a particular target class use identical binary training/validation/test splits.

All networks (except the template, which naturally follows [6]) are trained on random batches of size of 8 using SGD with a momentum of 0.9 for 30 epochs. The learning rate is set initially to 10^{-3} and decayed by 0.1 after epoch 20. Images fed into the networks are upscaled to 224 by 224 using bilinear interpolation and normalized without any additional augmentation. We did not perform hyperparameter tuning and while this choice of hyperparameters is arbitrary, we believe it suffices for the tasks at hand as all networks were able to converge.

To generate and evaluate a component network pertaining to a target class, we fine-tune the BN and last layers (BN) of the template network on its binary training set. We then select the network that has the best validation accuracy and finally report on the test set accuracy. This is repeated for each target class in CIFAR10, resulting in 10 component networks. To generate and evaluate the ComBN/PCBN networks, we replace the BN layers in the template network with ComBN/PCBN layers built from the BNs of the 9 component networks that were not trained on the target class, and then apply the same evaluation procedure. One might object to the fact that the 9 component networks might have seen a few examples of the target class as negative examples during their original training, but we find the difference to be negligible even after carefully omitting them. Finally, for completeness, we also report on results of performing full (Full) and last layer (Last) fine-tuning on the same tasks. The results are summarized in Table 6.1.

For this initial set of experiments, we find that in comparison with other methods, fine-tuning just the last layer results in the lowest accuracy. We also find that results obtained by ComBN and PCBN were comparable, if not just slightly worse, with BN and full fine-tuning, suggesting that ComBN and PCBN are potentially valid methods for training networks.

To better understand the role of BN components, we also performed the Comp, ComBN and PCBN experiments using a template network that has randomized (ac-

cording to [6]) convolutional layer weights. Surprisingly, this configuration still manages to achieve respectable accuracies, suggesting that random weights can still manage to map the inputs to representations which can be discriminated by the BN transformations, while also attesting the representational power of BN layers. Even more surprising is the performance that random PCBN obtained, which we leave for future investigation.

6.4.2 Few-shot Learning ImageNet32 from CIFAR10 Template

Motivated by the results in Section 6.4.1, we now attempt the more challenging task of evaluating ComBN and PCBN on ImageNet using a CIFAR10-pretrained template network. Owing to constraints on computational resources, we switch to testing on ImageNet32 [150] (which is ImageNet downsampled to 32×32 images) and use ResNet32 [6] as the template network. Additionally, to align ourselves with the original goal, we will perform the evaluations in terms of 2-way few-shot tasks.

Unlike the previous experiments, we forego the creation of a test split and will instead report on validation accuracy. We use the original ImageNet training/validation sets as our master training/validation splits, and then create binarized splits from the master splits using the same procedure outlined in Section 6.3.2. In order to ensure evaluation is performed on unseen classes, we first randomly sample 200 target classes to construct our component networks and reserve the remaining 800 for few-shot tuning and evaluation. The training/validation splits of these 200 binary datasets exhaust all available positive and negative examples from the master splits that result in a balanced dataset (i.e., about 2000 examples in the training split, and exactly 100 in the validation split). The remaining 800 binary datasets will have training/validation splits that respectively possess $2n/100$ examples per split, where n refers to a particular n-shot task.

Unless stated otherwise, all results are trained on random batches of size 128 ($2n$

			1-shot		5-shot	
Setup	Component selection	No. of components	μ	Δ	μ	Δ
Max	—	—	87.6	25.1	87.6	17.6
Last	—	—	63.0	0.5	69.3	−0.7
Full	—	—	62.5	—	70.1	—
BN	—	—	62.9	0.4	69.6	−0.5
PCBN	—	200	58.2	−4.3	65.3	−4.8
ComBN	Few-shot loss	3	66.3	3.8	73.6	3.5
ComBN	Few-shot loss	5	65.8	3.3	73.3	3.2
ComBN	Few-shot loss	10	65.7	3.2	71.5	1.4
ComBN	Max-shot accuracy	3	77.2	14.7	78.3	8.3
ComBN	Max-shot accuracy	5	76.3	13.8	78.0	7.9
ComBN	Max-shot accuracy	10	72.2	9.7	75.0	4.9
ComBN	Max-shot accuracy	75% threshold*	80.0	17.5	81.6	11.5
PCBN	Few-shot loss	3	64.5	2.0	71.3	1.3
PCBN	Few-shot loss	5	63.3	0.7	70.6	0.5
PCBN	Few-shot loss	10	62.8	2.6	68.4	−1.7
PCBN	Max-shot accuracy	3	71.2	8.7	74.4	4.4
PCBN	Max-shot accuracy	5	70.8	8.3	74.5	4.4
PCBN	Max-shot accuracy	10	67.5	5.0	71.4	1.3
PCBN	Max-shot accuracy	75% threshold*	73.0	10.5	77.5	7.4
SGM	—	—	64.8	2.3	70.6	0.5
L2	—	—	57.0	−5.5	59.7	−10.4

Table 6.2: Mean percentage validation accuracies μ on the 800 binary ImageNet32 datasets and their differences relative to full fine-tuning $\Delta = \mu - \mu_{\text{Full}}$. Asterix (*) indicates evaluation towards a subset (of about 400) of the 800 binary datasets that fulfil the 75% threshold criterion.

during few-shot) using SGD with a momentum of 0.9 for 60 epochs and weight decay of 10^{-4}. The learning rate is set initially to 0.01, and is decayed by 0.1 after epochs 30 and 45. The model with the best validation accuracy during the training procedure is selected. No data augmentation is performed, and this choice of hyperparameters is again not optimized. The template network was trained following [6].

The following discussions refer to Table 6.2, where we report the mean validation accuracy of various networks trained in the few-shot training regime on the 800 binary datasets (except for Max, which uses all training data and is therefore not few-shot;

we include this to illustrate a performance upper bound). As in Section 6.4.1, we also report on the results of fine-tuning the last layer (Last), all layers (Full), and only BN + last layers (BN) of the template network to serve as baselines.

Afterwards, we proceed to generating the 200 component networks by fine-tuning the BN and last layers of the template network using the aforementioned 200 binary datasets, and use all 200 components to construct a PCBN network. However, as we can see from the results, this construction performs worse than the baselines in both 1-shot and 5-shot tasks. This drop in performance might be attributed to the presence of components that are irrelevant for the target task. Furthermore, it is not feasible to create a ComBN network using 200 components due to memory constraints, making it apparent that we need to selectively reduce the number of components used in ComBN/PCBN.

To do so, and as previously hinted in Section 6.3.3, we propose to evaluate the cross-entropy loss of each component network on the training split of one of the 800 datasets that correspond to the unknown target class. We then rank and select m component networks with the lowest loss and use these m networks to construct our ComBN/PCBN networks (i.e., few-shot loss component selection). The results from Table 6.2 seem to indicate that this strategy works, with both ComBN and PCBN outperforming the baselines by about 4%, and $m = 3$ leading to the best results overall.

Furthermore, we also considered the case of having an ideal selection of components by assuming we were able to binarize all available training data from the master split for component selection (i.e., max-shot accuracy component selection). From here, we (i) selected m component networks with the highest validation accuracy on the target task and (ii) selected 3-10 component networks that perform above 75% in terms of validation accuracy on the target task. This resulted in a marked performance increase of about 14-17% when compared to the baseline, sug-

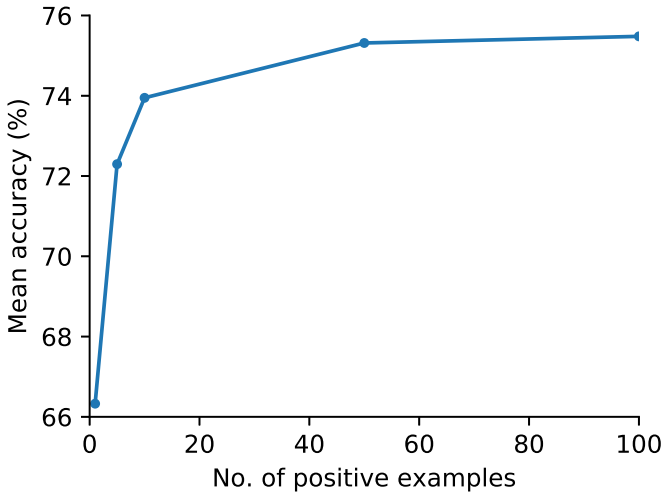
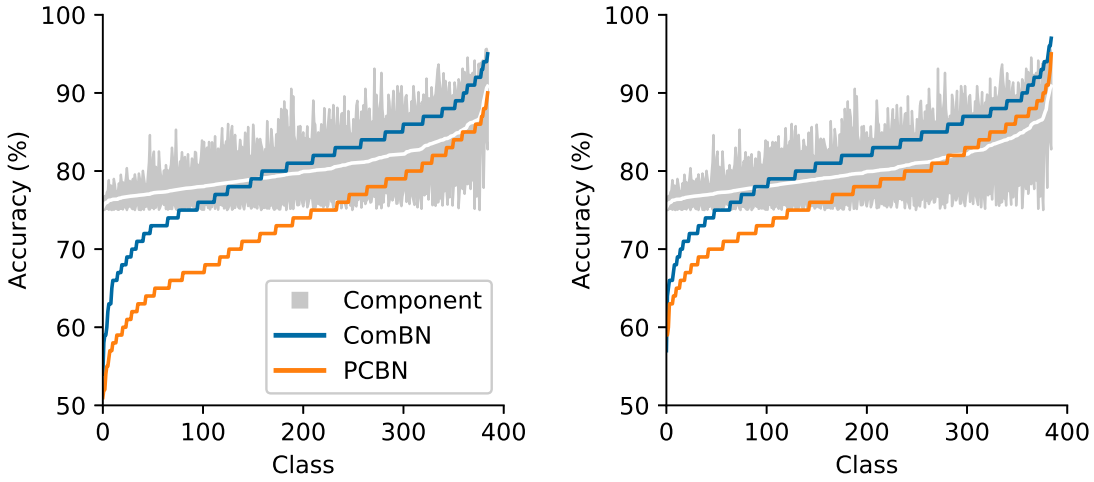


Figure 6.3: Mean 1-shot validation accuracy of 3-component ComBN as a function of number of positive examples in training split used in 1-shot loss component selection.

gesting that component selection is an important procedure that warrants further study. Again, $m = 3$ seems to lead to the best results overall. An illustration of (i) for 3-component ComBN in the 1-shot regime for other as a function of dataset size is plotted in Fig. 6.3.

Additionally, if we plot the validation accuracies of 75% threshold components and ComBN/PCBN networks constructed from them (Fig. 6.4), we notice that for some classes ComBN/PCBN does worse than any of the components. This seems odd given that we know for ComBN, a solution consisting of $\alpha_j = 1$ if $j = 1$, and $\alpha_j = 0$ if $j \neq 1$ should result in an accuracy above 75%. This anomaly might be caused by the tendency of SGD to avoid this particular solution, although further work is necessary to better understand this.

As an extra benchmark, we also attempted to compare our results to [78], which is one of the few works which attempted to tackle few-shot tasks without resorting to meta-learning. We do this by training two additional template networks using SGM and L2 loss functions as described in that work, which are theorized to create features better suited for few-shot learning. As before, we then subjected the two



(a) 1-shot (b) 5-shot

Figure 6.4: 75% threshold component, ComBN, and PCBN network validation accuracies. The filled plot represents the minimum, mean, and maximum accuracies for each set of component networks. Each plot has been independently sorted by mean accuracy to aid visualization.

template networks to fine-tuning in the few-shot regime, and reported their mean validation accuracies. Set up this way, our best-performing method (3-component ComBN) outperformed theirs (SGM) by a margin of 2-3% with the few-shot loss selection, which could go up to 8-13% assuming ideal component selection.

6.5 Conclusion

Based on a recent idea that batch normalization modules could transform inputs to encode class-specific representations, we propose an interpolation method within learned BN layers to efficiently learn new classes. We show that this works for few-shot learning by implementing it as a linear combination of BNs (ComBN) or PCA on BN parameters (PCBN), obtaining an accuracy between 4% to 17% over standard full fine-tuning. We have also shown that good performance is dependent on careful selection of the BN modules, and proposed a simple criterion to achieve this.

This method, however, still requires a fine-tuning in the form of ComBN/PCBN interpolation for the few-shot classes, which might be a limiting factor for deployment

at the edge (which typically has no access to training hardware). Furthermore, the intended industrial application requires object localization in addition to recognition, which is more suited for e.g., under an object detection framework. We shall address both limitations in the next chapter.

Chapter 7

Cos R-CNN for Online Few-shot Object Detection

Following the spirit of the previous chapter, where we explore more cutting edge research with potential to improve the existing production system, we now attempt to address the two limitations outlined in the last chapter by building an online few-shot detector. The aim is an automated system that can be adapted quickly to changing conditions in the field without being needed to go through an offline training loop. This chapter summarizes such work, where we explore the utilization of cosine-based similarity measures for performing online few-shot detection. The work was originally submitted for publication in 2020 [26], and at that time, is one of the first few works on few-shot detection. Like the previous chapter, the literature review in this chapter reflects the period when it was written.

Abstract

We propose Cos R-CNN, a simple exemplar-based R-CNN formulation that is designed for online few-shot object detection. That is, it is able to localize and classify novel object categories in images with few examples without fine-tuning. Cos R-CNN frames detection as a learning-to-compare task: unseen classes are represented as exemplar images, and objects are detected based on their similarity to these exemplars. The cosine-based classification head allows for dynamic adaptation of classification parameters to the exemplar embedding, and encourages the clustering of similar classes in embedding space without the need for manual tuning of distance-metric hyperparameters. This simple formulation achieves best results on the recently proposed 5-way ImageNet few-shot detection benchmark, beating the online 1/5/10-shot scenarios by more than 8/3/1%, as well as performing up to 20% better in online 20-way few-shot VOC across all shots on novel classes.

7.1 Introduction

Deep data-driven feature learning has enabled rapid advances in visual recognition tasks such as image classification [5, 38, 158, 6] and object detection [7, 33, 57, 8] due to its ability to learn good features from data. However, training deep models from scratch requires a large amount of training data, which makes its application in a potentially much larger number of data-scarce tasks prohibitive. To this end, the problem of learning from few labelled annotations, called few-shot learning, needs to be tackled. This is reflected in an ever-growing body of work studying few-shot image classification [67, 72, 70, 73, 68, 69, 159, 160, 74].

Despite rapid progress in few-shot image classification, the analogous task of few-shot object detection has received comparatively less attention [81, 88, 83, 85, 86, 87, 102]. Furthermore, most of existing literature [81, 83, 85, 86, 87] rely on a 2-stage training pipeline that involves 1) training a base detector using abundant examples of base classes, and then 2) fine-tuning the base detector on data-scarce novel class examples. While this approach has been successful, it is not appropriate for deployment in resource-constrained applications such as embedded devices as these devices typically have limited computational capacity, which may prohibit the training of large networks. Furthermore, these settings require base class examples to be available during fine-tuning to prevent catastrophic forgetting [161], which again may prohibit fine-tuning if the devices are memory-constrained.

It is then useful to study few-shot detection in an online setting, i.e., to be able to train a model that can detect unseen classes without having to be fine-tuned as illustrated in Fig. 7.1.

Our approach, Cos R-CNN, follows the general structure of an R-CNN, but is trained to generate locations for objects that are similar to exemplars of known labels. This naturally avoids the need for fine-tuning, as the same model can be used for different sets of detection classes. Inspired by recent approaches from few-show

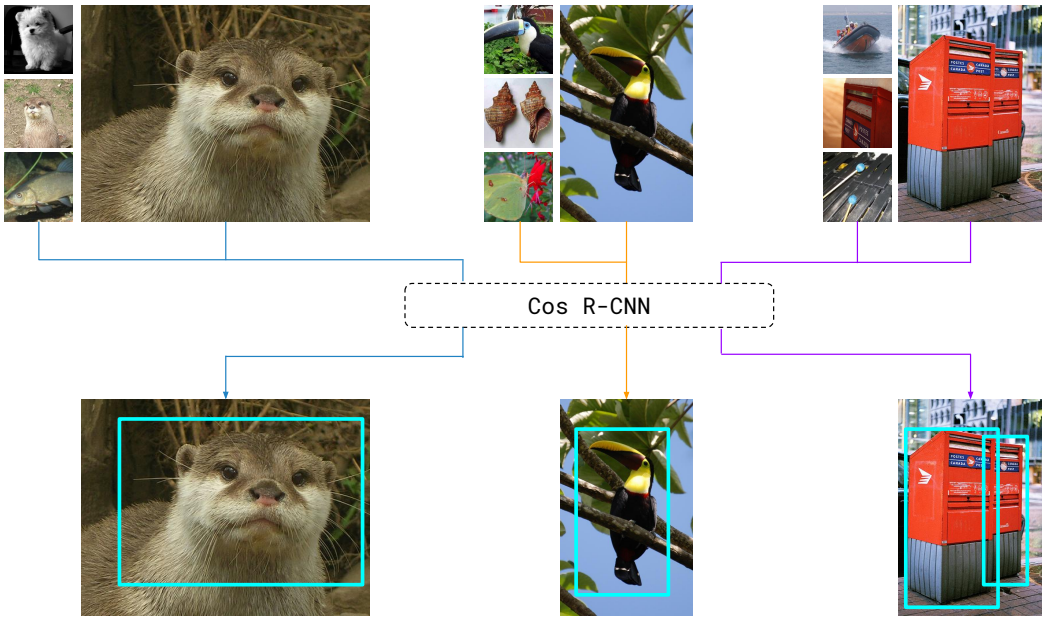


Figure 7.1: Three sets of distinct 3-way, 1-shot detection tasks. For each task, a query image containing a novel object is to be detected by providing exemplar images, at least one of which contains the novel object. Cos R-CNN is an online few-shot object detector, meaning that the same model can perform all 3 tasks having to be fine-tuned.

classification [98, 89, 162] we use cosine distance as our similarity measure, as it scales the logits (i.e., pre-softmax) of any class into the same range. Additionally, we also intuit that cosine-based metric learning may benefit from being shown a large quantity of data in each iteration (i.e. having large batch sizes), but this is usually difficult to obtain as R-CNN based frameworks possess a large memory footprint during training even at small batch sizes (e.g., 1-2). To work around this, we also introduce SuMoCo, a variant of MoCo [139] that enables a much larger corpus of exemplars to be compared to, improving the learning process.

We structure our chapter as follows. In §7.2 we summarize related work on few-shot object detection and classification. In §7.3, we detail each component of Cos R-CNN, and in §7.4 then evaluate our model, benchmarking it on the 5-way detection task introduced in [88], and detail ablations for our design choices. Finally, we summarize our findings in §7.5.

7.2 Related Work

We explore the background of few-shot research that provides context for this chapter. Object detection combines the traditional task of image classification with an additional constraint of localization. In other words, instead of classifying an image as a whole, the class and extent of an arbitrary number of object regions must be predicted. While there have been a number of successful detectors [7, 32, 163, 8, 33], few-shot object detection remains relatively under-explored. Therefore, we initially explore work towards few-shot image classification, which is closely related to object detection. Later, we detail recent methods for few-shot object detection.

Few-shot Classification

Few-shot learning is a form of supervised learning that, in contrast to deep learning, aims to achieve generalization using few examples. There has been increasing interest recently in few-shot learning. Lake *et al.* [67] proposed a Bayesian framework for recognizing unseen handwritten characters by distilling them into strokes and combining them in an example-efficient manner. In the same year, Koch *et al.* [71] used a deep Siamese convolutional network to determine if pairs of known/unknown image exemplars belong to the same class, framing few-shot classification as a matching task.

A concept that arose relatively recently is meta-learning, where models are set up to learn over a distribution of few-shot tasks by making it iterate over 'episodes' of few-shot tasks. This is unlike standard supervised learning where training is performed on batches of examples which make up a single task. By learning distributions of few-shot tasks, the model aims to generalize to any possible future few-shot task. This is first demonstrated by Matching Networks [72], which trained an exemplar matching network based on k-NN cosine similarity using this paradigm.

Similarity or matching-based strategies have since then been featured in a growing

number of works. Prototypical Networks [73] compared Euclidean distances of query embeddings to meta-learned class prototypes and matched the closest pair. Relation Networks [74] expanded on this idea and made the similarity measure learnable by introducing a relation module to do the comparison.

Other works such as MAML [68], Meta-SGD [164], and Reptile [159] expand the meta-learning strategy by inventing solvers to learn a network initialization that is amenable to the few-shot context. In the case of meta-learner LSTM [69], an SGD update rule is additionally also learned to guide few-shot optimization.

More recently, works such as [89, 162, 98] demonstrated that, given a powerful enough feature extractor, a scaled cosine similarity formulation for classification is competitive with many state-of-the-art few-shot methods.

Few-shot Object Detection

The literature in few-shot object detection is sparser when compared to few-shot classification. Most methods propose a 2-stage training pipeline where a base detector is first trained on abundant base classes, and then fine-tuned on scarce novel classes. Among the first to adopt this is LSTD [81], where they proposed to regularize the fine-tuning stage with additional objectives designed to reduce the severity of overfitting to the scarce novel classes. More recently, FSDet [87] showed that a simple strategy of freezing weights and fine-tuning only the last box head layer of an R-CNN detector is competitive with many contemporary few-shot object detection methods.

Still following the general 2-stage pipeline, other works propose to generate network weights for classifying novel classes in the second stage. Fewshot-YOLOv2 [83] and Meta-RCNN [85] aim to modulate base network features with channel-wise attention vectors generated by a subnetwork from the few-shot exemplars. Meta R-CNN additionally introduced a loss function that encourages these vectors to semantically cluster in the attention vector space. Similarly, MetaDet [86] also learns a subnet-

work to predict parameters, but directly treats them instead as classification weights instead of using it to modulate incoming features.

In contrast to previous methods, RepMet [88] proposed the usage of distance metric learning to learn an embedding space for object categories, each of which is represented by a multi-modal distribution. This could then be applied to few-shot detection by replacing the network head in an R-CNN with the learned object embeddings. Classification is performed using the distance of the query from representative embeddings of novel categories to explicitly compute class posteriors.

Similar to RepMet, the cosine formulation of Cos R-CNN can be thought of to induce metric learning, which can then be exploited to perform online detection. Unlike RepMet, however, Cos R-CNN is based on pure pairwise comparison, simplifying the formulation while also surprisingly (as we can see in §7.4) obtaining better performance. Furthermore, Cos R-CNN is also much simpler with less training objectives and having no specific hyperparameters to tune.

7.3 Cos R-CNN

The proposed architecture is shown in Figure 7.2. It roughly follows the form of a generic, two-stage, R-CNN approach. However, in a standard R-CNN detection approach, the model learns to detect object categories over many iterations and, importantly for generalization, with many distinct examples of the same object type. This formulation is ill-suited for few-shot detection: any attempt to train for novel class detection without careful regularization will likely overfit to the small number of specific examples. This overfitting is only exacerbated in our few-shot scenario where there are only few training examples.

Cos R-CNN tackles this issue of single-example generalization within an object category with cosine similarity, which explicitly encourages the network to learn an embedding that clusters similar categories together. Intuitively, this formulation

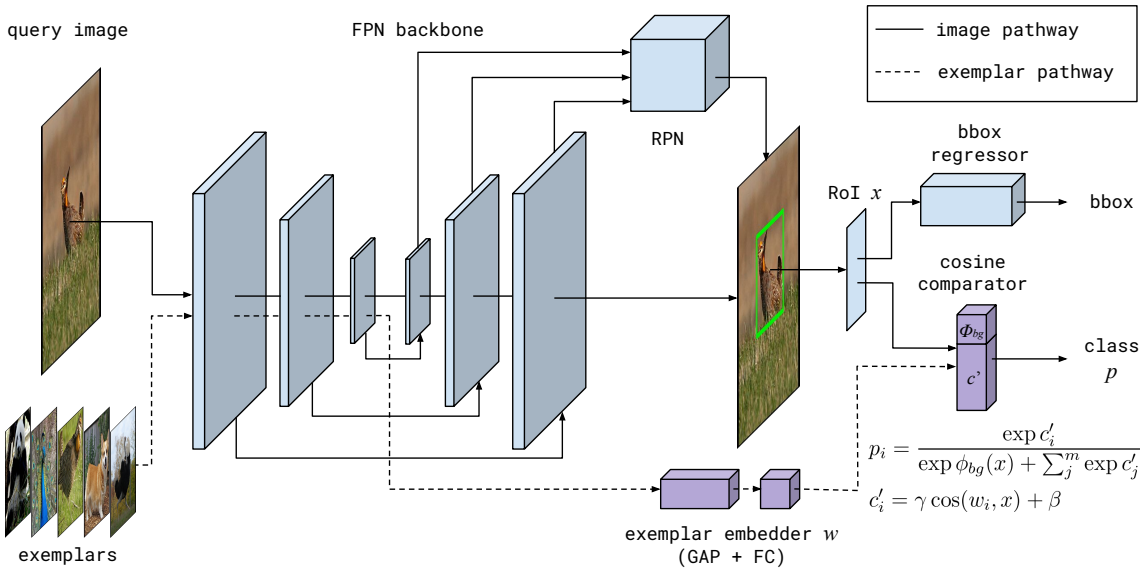


Figure 7.2: Overview of Cos R-CNN. Components which are unaltered from the original Faster R-CNN are coloured in blue, while those modified to enable online few-shot detection are coloured in purple.

achieves a similar objective to a metric learning formulation such as in [88] without having to tune a margin hyperparameter.

Specifically, the structure of Cos R-CNN roughly follows that introduced by Faster R-CNN [7]. The first stage is a subnetwork called the Region Proposal Network (RPN), which proposes candidate objects and their bounding boxes. We refer to the second stage as the network *head*. It is a small subnetwork which aims to classify candidate objects and to refine their bounding boxes. The network head operates on fixed-sized feature maps that are pooled from an image feature map through RoIAlign [57]. The head is therefore essentially Fast R-CNN [56]. Inference time is reduced by having the RPN and head operate on the same image feature map, computed by a convolutional *backbone*. Commonly, ResNets [6] with a Feature Pyramid Network (FPN) [53] are used as the backbone architecture.

7.3.1 Exemplar Embedding Pathway

As the cosine similarity formulation necessitates comparisons of the query image with exemplars, the typical R-CNN pipeline needs to be modified to generate these exemplars. We choose to generate exemplar features by piping images of cropped exemplar instances through the ResNet-FPN backbone, followed by an exemplar embedder as shown in Figure 7.2. The exemplar embedder consists of a global average pooling (GAP) layer followed by a fully-connected layer that maps the pooled feature vector of channels 2048 $\rightarrow$ 1024 so that it matches the channel dimensions of RoI features from the query image, making them comparable by cosine similarity. We also note that sharing the backbone in this manner allows exemplar features to be generated without substantially increasing network size/memory consumption (as it would be if a separate backbone were to be used).

7.3.2 Cosine Comparator Head

Cosine similarity/distance is defined as,

$$c_i = \cos(\vec{w}_i, \vec{x}) = \frac{\vec{w}_i}{\|\vec{w}_i\|} \cdot \frac{\vec{x}}{\|\vec{x}\|}, \tag{7.1}$$

where $i = 1, 2, ..., m$ refers to the i-th class in an m-way classification task. $\vec{w}_i$ is dynamically computed from exemplar instances via the exemplar pathway in Cos R-CNN's backbone, whereas $\vec{x}$ is the embedded image feature. In other words, classification is performed in terms of comparison of query to exemplar, and the model can easily adapt to novel classes just by comparing to a novel class exemplar. The L2 normalization term ensures that the classification weights for both base and novel classes have similar magnitudes as they are bounded to $[-1, 1]$, which allows few-shot classification by exemplars without requiring the model to be fine-tuned. Note that the exemplar embedding module/embedder can alternatively be interpreted as a few-shot weight generator, transforming exemplar features into classification weight

vectors (albeit one that is conditional on the input exemplar instead of being a model parameter).

In practice, we replace the fully-connected classification layer in the box class head with a cosine comparator layer, which computes a scaled cosine similarity on query and exemplars,

$$c'_i = \gamma \cdot \cos(\vec{w}_i, \vec{x}) + \beta, \tag{7.2}$$

where γ and β are learnable scalar scale and bias parameters. This form allows the model to learn to control the peakiness of the class logits distribution by expanding the domain to $[-\infty, \infty]$ while still having the advantages of a cosine formulation as reported in other works [84, 98].

As the open background class cannot be represented by exemplars, we still compute the background logits from RoI features using a learnable function, ϕ_{bg}, which is implemented as a fully-connected layer. We concatenate background logits with the logits computed from exemplars. The combined logits are then softmaxed and passed to a cross-entropy loss objective, which can simply then be backpropagated to train the model. In all, this leads to a definition of the posterior for class i in the m-way, 1-shot task as,

$$p(y = i|\vec{x}) = \frac{\exp(c'_i)}{\exp(\phi_{bg}(\vec{x})) + \sum_j^m \exp(c'_j)}. \tag{7.3}$$

Few-shot Inference

So far, we have formulated Cos R-CNN's detection as a comparison between a single example from each class, i.e., the one-shot scenario. One of the ways Cos R-CNN can be extended to the m-way, n-shot detection task is as follows. We provide n exemplars for each of the m classes. Inference then follows as $n \cdot m$-way, 1-shot detection. However, after applying softmax to the cosine-similarity logits across the $n \cdot m$ exemplars, the detection score for each class is computed as the sum of the scores from the n exemplars belonging to the same class. The posterior for class i in

the m-way, n-shot task is thus given by,

$$p(y = i|\vec{x}) = \frac{\sum_k^n \exp(c'_{ik})}{\exp(\phi_{bg}(\vec{x})) + \sum_j^m \sum_k^n \exp(c'_{jk})}. \tag{7.4}$$

It has been found for comparator-based classification that it is beneficial to train with the same number of shots, n, as used in testing [73]. However, we later evaluate how this formulation of few-shot detection allows for performance improvements that come from additional examples of novel classes, without requiring Cos R-CNN to be re-trained or even fine-tuned.

7.3.3 Supervised Momentum Contrast (Su-MoCo)

It has recently been shown that instance-based contrastive self-supervised learning benefit from being able to iterate on larger batch sizes [139]. As Cos R-CNN is also trained in a contrastive manner (comparison of query to exemplars), we posit that it may also benefit despite it being trained in a supervised manner. However, it is not straightforward to simply increase the batch size as R-CNN architectures use a substantially larger amount of GPU memory due to RoI sampling heuristics (e.g., batch sizes of 1 or 2 are commonly used).

One workaround is to serialize exemplar features from previous iterations in a memory bank after detaching these exemplars from the computational graph. Afterwards, in the current training iteration, randomly sample features from this bank and append them as additional instances to the current batch [165]. The serialized exemplars consume very little memory and many of them may be used to increase the effective 'batch size'.

However, a drawback to the memory bank is that randomly sampled exemplars might not be coherent as they might have been generated at very different stages of the network evolution during training. MoCo (Momentum Contrast) [139] solves this by adopting a queue strategy (FIFO) for sampling from the memory bank, and using a duplicate of the main network to generate exemplar features that is updated more

slowly than the main network. This ensures that the sampled exemplars remain as coherent as possible. We build on this work and introduce a variant for supervised object detection which we call Su-MoCo.

Like MoCo, Su-MoCo also utilizes a slowly-evolving duplicate model for the exemplar pathway and update its weights according to a momentum rule. Namely, we use the same update rule to compute the duplicate model weights,

$$\theta_k \leftarrow \alpha\theta_k + (1-\alpha)\theta_q, \tag{7.5}$$

where θ_q are the weights of the latest main network in training, and α is the momentum value.

Unlike MoCo, which uses one unsupervised loss for such as InfoNCE [166], our supervised loss for Su-MoCo is an average of negative log-likelihoods from different queue samples, namely

$$\mathcal{L} = -\frac{\sum_{\ell}^{q}(\log p_\ell)}{q} \tag{7.6}$$

where p_ℓ is the posterior in Eq. 7.3 computed for a particular queue ℓ and q is the queue size. Note that because of the R-CNN architecture, each batch of queue posteriors in one iteration of Su-MoCo are sampled from different RoI batches.

Using Su-MoCo allows us to considerably increase the amount of exemplars being compared in one iteration which is beneficial to Cos R-CNN performance as can be seen in §7.4.3.

7.4 Experiments

We now outline experiments that evaluate the performance of Cos R-CNN, and show ablations on its design. Specifically, we benchmark against 2 distinct tests: 5-way ImageNet [88] and 20-way Pascal VOC [83]. The 2 tests allow us to benchmark our method in both an episodic (which is common in few-shot classification literature) and non-episodic manner.

7.4.1 Few-shot ImageNet

We first test our method on the ImageNet 5-way, n-shot detection benchmark proposed by [88]. The benchmark is based on a 314-class subset (containing mostly animal classes) of the ImageNet CLS-LOC dataset [27], partitioned into 100 base and 214 novel classes. After training on the 100 base class examples, the model is tested on the unseen 214 novel classes by evaluating 500 episodes of 5-way, n-shot few-shot tasks. Each episode consists of 50 query images (10 images for each of the 5 randomly sampled novel classes for this episode) and $5n$ exemplar images (n exemplars for each of the 5 novel classes). Detection metrics are then reported by jointly evaluating test images from all 500 episodes.

Model

Unless otherwise stated, we base our detection model on Faster R-CNN with a ResNet50-FPN backbone [53]. We keep the original RPN for region proposal, and instead only modify the classification box head with the cosine classifier outlined in §7.3.2 to perform few-shot detection. Consequently, we also use a class-agnostic bounding box regressor in the box head to be able to adapt to any number of classes. For region-of-interest (RoI) pooling and warping, we utilize the more recent RoIAlign instead of RoIPool [57].

Training

We initialize the model parameters by sampling from a Gaussian distribution with $\mu = 0$ and $\sigma = 0.01$ except for the ResNet50 backbone which was pre-trained on COCO [64], and the cosine comparator where we set $\gamma = 1$ and $\beta = 0$ (see Eq. 7.2). We also freeze all the batch normalization layers of the pretrained model and backbone until the second ResNet stage.

We train our model using SGD with a momentum of 0.9 and weight decay of 10^{-4} for 240,000 iterations, starting by warming up the learning rate linearly to 0.0375 at

iteration 500. The learning rate is then decayed by a factor of 10 at iterations 160,000 and 213,333.

We utilize image-centric episodic training on 3 GPUs with a batch size of 1 image per GPU, where each iteration in each GPU consists of an episode comprising 1 query image and 5 exemplar instances. Then, we resize query images and exemplars so that their shorter side is at least 600 pixels, and obtain exemplar instances after by cropping them from their ground truth bounding boxes. We also flip the images/exemplars horizontally with a probability of 0.5 as a form of data augmentation. Finally, we do not use Su-MoCo here to expand the effective exemplar batch size, as the episodic training format means that exemplars of the previous batch are obsolete for the current iteration.

Inference

Taking the model at the end of the training schedule, we use the identical 500 episodes (query/exemplar image sets) used by RepMet [88] to evaluate the novel subset, and sample our own episodes for the base subset as the exact episodes were not provided. Evaluation on shots > 1 uses per-channel averaging of the exemplar features at the comparison layer.

Results

Our results are shown in Table 7.1, where we outperform state-of-the-art RepMet [88] on both base and novel class detections. Base class detections are evaluated in an episodic manner like novel classes (i.e., by comparing to exemplars). Note that we adopt the COCO convention and write our results as AP_{50}, which is equivalent to ILSVRC mAP as reported in RepMet.

Table 7.1 also shows the evaluation of our extension into few-shot detection. We demonstrate that Cos R-CNN is able to improve detection through the combination of multiple examples of each class. It is important to note that Cos R-CNN has only

	Base AP_{50}			Novel AP_{50}		
Method/Shot	1	5	10	1	5	10
RepMet	64.5	79.4	82.6	56.9	68.8	71.5
Cosine	**84.5**	**88.4**	**89.4**	**65.1**	**72.3**	**72.9**

Table 7.1: AP50 scores on 5-way ImageNet LOC test proposed in RepMet [88]. We obtain improved results on 1-shot, 5-shot, and 10-shot compared to their results without fine-tuning on novel classes. Both models use the same episodes to evaluate the novel subset performance.

been trained in the one-shot scenario and few-shot detection is performed online (i.e., without any fine-tuning or re-training to the new shot number).

7.4.2 Few-shot VOC

Next, we evaluate against the 20-way, n-shot detection benchmark in [83]. The benchmark is based on the PASCAL VOC object detection dataset [146] and consists of 3 different splits consisting of 15 base and 5 novel classes each. Unlike few-shot ImageNet (which can form many episodes of different class combinations), this testing is non-episodic, and we report the results on each split directly: after training on the 15 base class examples, the model is tested on the unseen 5 novel classes. Again, we report AP_{50} as our evaluation metric.

Training

The same training hyperparameters and schedules were used as in §7.4.1, except that a) we now pre-train the backbone on ImageNet as COCO and PASCAL VOC classes overlap, and b) each iteration consists of 1 query image and 15 exemplar instances. Additionally, we utilize Su-MoCo with a queue size of 100 and momentum of 0.999 to boost the effective number of exemplars seen in each iteration by a factor of 100. As the ablations in §7.4.3 show, using Su-MoCo helps improve the final performance of the model.

VOC Split 1		Base AP_{50}/shot					Novel AP_{50}/shot				
Method	Backbone	1	2	3	5	10	1	2	3	5	10
FS-YOLOv2	DarkNet	67.3	69.4	69.5	69.5	69.6	2.0	8.2	8.7	11.8	11.9
Meta R-CNN	R-101-FPN	68.9	68.9	69.1	69.1	69.1	2.9	3.6	3.2	3.2	3.4
RepMet	R-101-DCN	68.4	67.0	70.5	70.7	72.5	29.0	32.8	30.9	35.6	40.2
RepMet	R-50-FPN	65.3	68.9	67.6	69.1	70.1	14.9	11.1	13.6	19.3	21.3
L2	R-50 FPN	65.8	63.9	66.0	66.4	66.4	**30.2**	21.9	24.2	31.4	32.0
Cosine	R-50-FPN	63.4	67.2	67.6	67.7	67.5	27.9	**33.0**	**32.1**	**36.2**	**33.6**
VOC Split 2		Base AP_{50}/shot					Novel AP_{50}/shot				
Method	Backbone	1	2	3	5	10	1	2	3	5	10
FS-YOLOv2	DarkNet	66.2	71.2	71.4	70.1	70.0	11.9	1.0	4.6	4.1	1.1
Meta R-CNN	R-101-FPN	69.7	64.5	70.1	71.2	71.5	1.5	1.1	1.0	3.5	4.6
RepMet	R-101-DCN	68.1	66.6	69.6	69.7	71.9	11.1	10.8	16.8	18.0	21.5
RepMet	R-50-FPN	65.2	67.0	65.9	68.3	68.9	15.3	**14.0**	12.3	15.7	18.9
L2	R-50-FPN	64.8	64.9	66.6	66.7	66.4	14.3	10.0	14.2	13.4	18.7
Cosine	R-50-FPN	62.5	66.8	67.2	67.4	67.3	**19.4**	12.6	**14.4**	**19.1**	**21.9**
VOC Split 3		Base AP_{50}/shot					Novel AP_{50}/shot				
Method	Backbone	1	2	3	5	10	1	2	3	5	10
FS-YOLOv2	DarkNet	68.9	70.0	70.1	70.1	70.1	6.0	7.6	8.5	9.9	9.5
Meta R-CNN	R-101-FPN	68.9	68.9	69.1	69.1	69.2	1.7	2.9	1.4	0.8	1.1
RepMet	R-101-DCN	63.3	64.0	67.2	68.5	72.2	21.1	21.7	24.9	26.5	27.9
RepMet	R-50-FPN	63.4	62.8	55.1	66.1	68.4	16.0	12.4	2.0	15.8	14.4
L2	R-50-FPN	60.1	64.8	65.0	65.8	65.5	**17.1**	14.8	20.4	22.3	20.4
Cosine	R-50-FPN	59.7	66.1	66.9	67.1	67.0	16.9	**21.6**	**21.6**	**27.5**	**25.5**

Table 7.2: AP50 scores on 20-way VOC test proposed by FewShot-YOLOv2 [83]. All models are trained on the base classes and not fine-tuned to the novel class exemplars. The same exemplars obtained from [83] were used to evaluate all models. Best novel class results among the same backbone are in bold.

Inference

For all splits, we use the same exemplars provided by [83] to evaluate the base and novel class metrics using the model obtained at the end of the training schedule. Like §7.4.1, evaluation on shots > 1 uses per-channel averaging of the exemplar features at the comparison layer.

Baselines

We also trained FewShot-YOLOv2 [83], Meta R-CNN [85], and RepMet [88] using code provided by each respective authors and their original schedules, but without fine-tuning to few-shot exemplars as we want to evaluate their suitability under an online setting. To better control for differences in model architecture, we also trained a ResNet-50 variant of RepMet without deformable convolutions [167], OHEM [168], or Soft-NMS [59] using our training schedule to isolate the difference between both methods to just their metric learning formulations. Finally, we also show results where we substituted the cosine similarity function with a Euclidean distance (L2) function like in Prototypical Networks [73].

Results

We report our results and baselines on base and novel classes in Table 7.2. While FS-YOLOv2 and Meta R-CNN were meta-learned, they were not designed to do online detection, which is reflected in their poor performance on the novel unseen classes. In contrast, the original RepMet implementation on R-101-DCN is comparable or better than any of the R-50-FPN variants shown in the table. However, this edge may be attributed to a deeper backbone and/or bells-and-whistles, as we see that on average, the simpler cosine formulation outperforms the vanilla R-50-FPN variant of RepMet on the novel classes by a large margin.

We also notice that the base class performance of cosine is very slightly worse compared to other methods. However, the primary objective of an online few-shot

method is to perform well on the novel classes, and we interpret this as a reasonable trade-off to make for pushing novel class performance.

We also show results of using the Euclidean distance as the comparison metric, and we see that it is slightly worse on average than using cosine, highlighting the importance of using cosine.

7.4.3 Ablations

Unless otherwise stated, we detail ablations on Cos R-CNN by comparing results on 5-way ImageNet detection. For these ablations, we use a slightly different episodic testing format: we evaluate all images by forming N episodes of 1 query image each, where N is the number of images in the dataset. This allows for quicker evaluation without losing comprehensiveness. Some ablations are based on few-shot VOC, and for these ablations we report novel class results on the first split.

Cosine Form

Here we examine the necessity of applying a parametrized scalar affine transformation (in the form of scale γ and bias β) on the cosine similarity output. Doing this frees the constraint that logits have to lie between $[-1, 1]$ into $[-\infty, \infty]$, while still keeping the advantages of a cosine-based formulation. Results on Table 7.3a confirm this intuition. We observe that the scale factor γ is the most influential component and should almost always be used in conjunction with cosine similarity. In the presence of γ, the bias factor β seems to have less of an effect, and in the case above, slightly degrades detection performance.

Loss Objective

In Table 7.3b, we further compare learning the model through a cross-entropy (softmax) vs. a binary cross-entropy (sigmoid) objective, which is a common alternative formulation used in other detectors such as RetinaNet [33]. For binary cross-entropy,

we do not directly compute background logits, but instead rely on the class detection scores being low for background RoIs. While in standard non-few-shot detection pipelines either formulation works, we find that for cosine-similarity based few-shot detection, a softmax loss gives superior results. Observing the detection results, we find there is a tendency to have predictions of all exemplar classes for the same box, which brings down the detection performance by reducing the classification rate and suppression of background classes.

Form	Base	Novel
No scale or bias	49.2	35.3
Scale	**85.9**	**65.2**
Scale and bias	85.5	64.5

(a) Cosine form

Loss	Base	Novel
Softmax	**85.5**	**64.5**
Sigmoid	45.4	48.3

(b) Loss objective

Table 7.3: Ablations on the cosine box head reported as AP_{50}.

Cos RPN

One possible design decision is to also apply our cosine similarity formulation to the RPN, modifying the standard RPN objectness classifier by taking the maximum detection score at any point across all exemplars as the objectness score.

In Table 7.4, we show the results of a standard RPN against a Cos RPN. The Cos RPN is a slight modification to the exemplar embedding function within the RPN. We use a single fully-connected layer as the embedder function.

Method/Shot	1	2	3	5	10
Standard	27.9	**33.0**	**32.1**	**36.2**	33.6
Cos (Linear)	**30.6**	29.1	29.9	33.6	**34.2**

Table 7.4: Cos RPN embedder ablations reported as AP_{50}.

From the overall trends, Cos RPN appears to be less optimal compared to standard RPN for few-shot detection. A possible explanation is that object appearances are too

Momentum	Queue size	Base	Novel
0.0	0	**65.4**	26.0
0.9	10	64.9	29.8
0.99	10	64.4	27.7
0.99	100	62.9	28.9
0.999	100	62.4	**30.6**

(a) Across hyperparameters

RPN	Method/Shot	1	2	3	5	10
Standard	Standard	27.7	30.0	26.7	32.2	31.3
	Su-MoCo	**27.9**	**33.0**	**32.1**	**36.2**	**33.6**
Cos (Linear)	Standard	26.0	27.5	27.4	31.9	31.6
	Su-MoCo	**30.6**	**29.1**	**29.9**	**33.6**	**34.2**

(b) Across n-shots

Table 7.5: Ablations on Su-MoCo AP_{50} performance on Split 1 of few-shot VOC.

varied to reliably cluster around a single centre, and so a cosine similarity formulation would be less able to capture all possible modes of variation. Alternatively, it is possible that the diverse variety of object appearances in the base training data is sufficient to enable the standard RPN propose good bounding boxes of unseen objects. As such, we adopt standard RPN in our main experiments.

However, this does not mean that the cosine similarity in the box head is superfluous. It is also possible that while standard RPN in a standard R-CNN system predicts objectness, standard RPN in Cos RPN could also be benefitting from gradients back-propagated by the cosine box head which allows it to generate good proposals of novel class objects. Additionally, without the cosine formulation, online detections of novel classes will not be possible.

Su-MoCo

We first ablate the momentum and queue size hyperparameters for Su-MoCo as shown in Table 7.5a. Momentum controls how quickly the exemplar pathway weights evolve during training, and the queue size determines the effective size of exemplars eval-

uated in a training iteration. We find performance on novel classes improves as we increase queue size, confirming the intuition that the model learns better feature representations if it can consider more exemplars during a training iteration. However, selecting an appropriate momentum value for the selected queue size is also important. As queue sizes increase, momentum also needs to increase as the exemplar pathway weights need to evolve more slowly in order to ensure old exemplars in the queue still remain informative [139]. We find that a good rule is to set the momentum value to $1 - q^{-1}$ where q is the queue size. As we find $q = 100$ to work best for novel classes, we use this value and thus a momentum of 0.999 for our experiments in §7.4.2.

These findings on queue size corroborate [139], but we find that for supervised learning (our setting), the queue sizes do not have to be extremely large in order to observe concrete improvements in performance.

Finally, in Table 7.5b, we demonstrate the effectiveness of adopting Su-MoCo, where we obtain improvement of 2-4% across the board on novel classes compared with not using Su-MoCo.

7.5 Conclusion

In this chapter, we have proposed the Cos R-CNN online few-shot object detector, and validated its design through ablation studies. The performance of Cos R-CNN was evaluated on a recently introduced ImageNet 5-way few-shot detection benchmark, beating the state-of-the-art in 1/5/10-shot setups by 8/4/1% whilst also being a simpler formulation. This is also reflected in the few-shot PASCAL VOC benchmark, beating contemporary methods on the novel classes by up to 20%.

Chapter 8

Conclusion

[redacted]

[redacted]

[redacted]

Bibliography

[1] D. Baulcombe, I. Crute, B. Davies, J. Dunwell, M. Gale, J. Jones, J. Pretty, W. Sutherland, and C. Toulmin, *Reaping the Benefits: Science and the Sustainable Intensification of Global Agriculture*. The Royal Society, 2009.

[2] H. C. J. Godfray and T. Garnett, "Food security and sustainable intensification," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 369, no. 1639, p. 20120273, 2014.

[3] R. S. Satake, *Development and designing of a spectro-separator for rice applications*. Japanese Society of Agricultural Machinery, 2001.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[10] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[13] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[14] N. Kundu, G. Rani, and V. S. Dhaka, "Seeds classification and quality testing using deep learning and yolo v5," in *Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence*, pp. 153–160, 2021.

[15] T. Luo, J. Zhao, Y. Gu, S. Zhang, X. Qiao, W. Tian, and Y. Han, "Classification of weed seeds based on visual images and deep learning," *Information Processing in Agriculture*, 2021.

[16] H. Gao, T. Zhen, and Z. Li, "Detection of wheat unsound kernels based on improved resnet," *IEEE Access*, vol. 10, pp. 20092–20101, 2022.

[17] C. Pinto, J. Furukawa, H. Fukai, and S. Tamura, "Classification of green coffee bean images basec on defect types using convolutional neural network (cnn)," in *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*, pp. 1–5, IEEE, 2017.

[18] R. Janandi and T. W. Cenggoro, "An implementation of convolutional neural network for coffee beans quality classification in a mobile information system," in *2020 International Conference on Information Management and Technology (ICIMTech)*, pp. 218–222, IEEE, 2020.

[19] Y. Unal, Y. S. Taspinar, I. Cinar, R. Kursun, and M. Koklu, "Application of pre-trained deep convolutional neural networks for coffee beans species detection," *Food Analytical Methods*, vol. 15, no. 12, pp. 3232–3243, 2022.

[20] E. O. Olaniyi, O. K. Oyedotun, and K. Adnan, "Intelligent grading system for banana fruit using neural network arbitration," *Journal of Food Process Engineering*, vol. 40, no. 1, p. e12335, 2017.

[21] A. Helwan, M. K. Sallam Ma'aitah, R. H. Abiyev, S. Uzelaltinbulat, and B. Sonyel, "Deep learning based on residual networks for automatic sorting of bananas," *Journal of Food Quality*, vol. 2021, pp. 1–11, 2021.

[22] "How a japanese cucumber farmer is using deep learning and tensorflow." `https://cloud.google.com/blog/products/ai-machine-learning/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow`, 2016. Accessed: 2022-10-22.

[23] Y. J. Heo, S. J. Kim, D. Kim, K. Lee, and W. K. Chung, "Super-high-purity seed sorter using low-latency image-recognition based on deep learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3035–3042, 2018.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[25] G. W. P. Data, K. Ngu, D. W. Murray, and V. A. Prisacariu, "Interpolating convolutional neural networks using batch normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 574–588, 2018.

[26] G. W. P. Data, H. Howard-Jenkins, D. Murray, and V. Prisacariu, "Cos r-cnn for online few-shot object detection," *arXiv preprint arXiv:2307.13485*, 2023.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[28] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, Ieee, 2005.

[30] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 524–531, IEEE, 2005.

[31] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their location in images," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, pp. 370–377, IEEE, 2005.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision*, pp. 21–37, Springer, 2016.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017.

[34] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019.

[35] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 213–229, Springer, 2020.

[36] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[37] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7310–7311, 2017.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645, Springer, 2016.

[40] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 390–391, 2020.

[41] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[43] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[44] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[46] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

[47] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

[48] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12974, 2020.

[49] X. Dai, A. Wan, P. Zhang, B. Wu, Z. He, Z. Wei, K. Chen, Y. Tian, M. Yu, P. Vajda, *et al.*, "Fbnetv3: Joint architecture-recipe search using predictor pretraining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16276–16285, 2021.

[50] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1458–1465, IEEE, 2005.

[51] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE com-*

puter society conference on computer vision and pattern recognition (CVPR'06), vol. 2, pp. 2169–2178, IEEE, 2006.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[53] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

[54] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018.

[55] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[56] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[57] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

[58] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[59] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS–improving object detection with one line of code," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5561–5569, 2017.

[60] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750, 2018.

[61] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6569–6578, 2019.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[63] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[64] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.

[65] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[66] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[67] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[68] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*, pp. 1126–1135, PMLR, 2017.

[69] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017.

[70] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[71] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, p. 0, Lille, 2015.

[72] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[73] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[74] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

[75] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, pp. 1842–1850, PMLR, 2016.

[76] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[77] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[78] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3018–3027, 2017.

[79] H. Bilen and A. Vedaldi, "Universal representations: The missing link between faces, text, planktons, and cat breeds," *arXiv preprint arXiv:1701.07275*, 2017.

[80] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[81] H. Chen, Y. Wang, G. Wang, and Y. Qiao, "LSTD: A low-shot transfer detector for object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[82] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[83] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8420–8429, 2019.

[84] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[85] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin, "Meta R-CNN: Towards general solver for instance-level low-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9577–9586, 2019.

[86] Y.-X. Wang, D. Ramanan, and M. Hebert, "Meta-learning to detect rare objects," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9925–9934, 2019.

[87] X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez, and F. Yu, "Frustratingly simple few-shot object detection," in *International Conference on Machine Learning*, pp. 9919–9928, PMLR, 2020.

[88] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein, "RepMet: Representative-based metric learning for classification and few-shot object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5197–5206, 2019.

[89] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018.

[90] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *International conference on learning representations*, 2018.

[91] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 403–412, 2019.

[92] C. Doersch, A. Gupta, and A. Zisserman, "Crosstransformers: spatially-aware few-shot transfer," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21981–21993, 2020.

[93] Y. Hu, V. Gripon, and S. Pateux, "Leveraging the feature distribution in transfer-based few-shot learning," in *International Conference on Artificial Neural Networks*, pp. 487–499, Springer, 2021.

[94] D. Wertheimer, L. Tang, and B. Hariharan, "Few-shot classification with feature map reconstruction networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8012–8021, 2021.

[95] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv preprint arXiv:1803.00676*, 2018.

[96] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," *arXiv preprint arXiv:1903.03096*, 2019.

[97] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," *arXiv preprint arXiv:1909.02729*, 2019.

[98] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[99] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[100] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[101] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai, "Few-shot object detection with attention-rpn and multi-relation detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4013–4022, 2020.

[102] J.-M. Perez-Rua, X. Zhu, T. M. Hospedales, and T. Xiang, "Incremental few-shot object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 13846–13855, 2020.

[103] G. Han, Y. He, S. Huang, J. Ma, and S.-F. Chang, "Query adaptive few-shot object detection with heterogeneous graph convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3263–3272, 2021.

[104] A. Wu, Y. Han, L. Zhu, and Y. Yang, "Universal-prototype enhancing for few-shot object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9567–9576, 2021.

[105] G. Zhang, Z. Luo, K. Cui, and S. Lu, "Meta-detr: Few-shot object detection via unified image-level meta-learning," *arXiv preprint arXiv:2103.11731*, vol. 2, no. 6, 2021.

[106] B. Sun, B. Li, S. Cai, Y. Yuan, and C. Zhang, "Fsce: Few-shot object detection via contrastive proposal encoding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7352–7362, 2021.

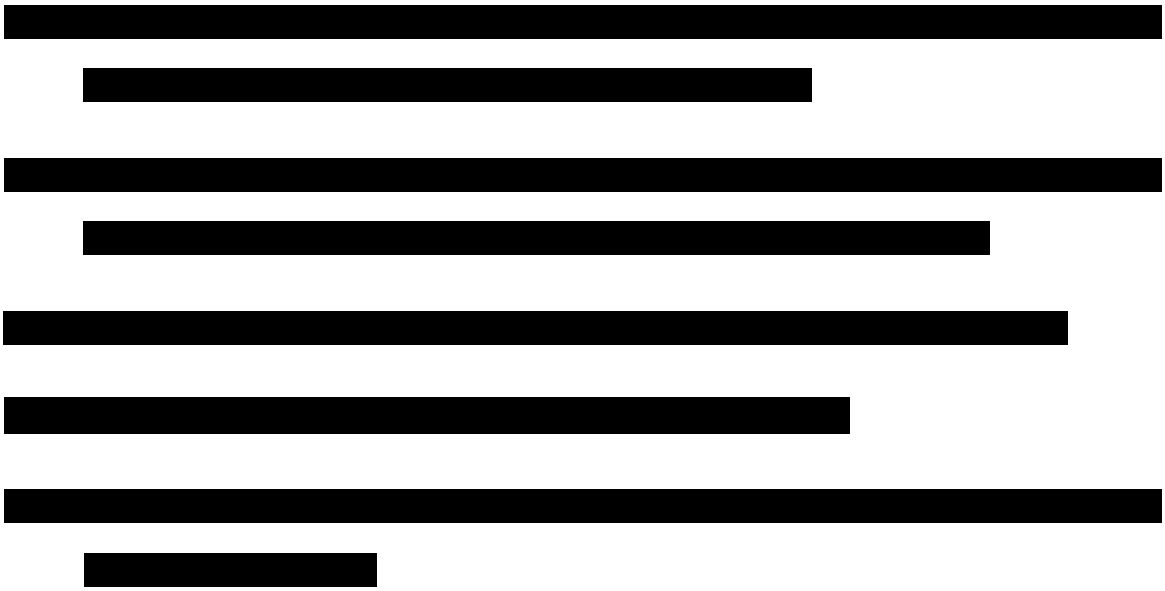
[107] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang, "Defrcn: Decoupled faster r-cnn for few-shot object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8681–8690, 2021.

[108] P. Kaul, W. Xie, and A. Zisserman, "Label, verify, correct: A simple few shot object detection method," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14237–14247, 2022.

[109] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[110] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[111] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.



[118] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1301–1310, 2017.

[119] N. Dvornik, J. Mairal, and C. Schmid, "Modeling visual context is key to augmenting object detection datasets," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 364–380, 2018.

[120] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2918–2928, 2021.

[121] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[122] S. Zagoruyko and N. Komodakis, "Diracnets: Training very deep neural networks without skip-connections," *arXiv preprint arXiv:1706.00388*, 2017.

[123] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13733–13742, 2021.

[124] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[125] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "An improved one millisecond mobile backbone," *arXiv preprint arXiv:2206.04040*, 2022.

[126] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[127] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of freebies for training object detection neural networks," *arXiv preprint arXiv:1902.04103*, 2019.

[128] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[129] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[130] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

[131] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[132] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[133] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "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*, pp. 658–666, 2019.

[134] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[135] "COCO detection eval." `https://cocodataset.org/#detection-eval`. Accessed: 2022-08-22.

[136] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[137] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[138] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[139] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

[140] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21271–21284, 2020.

[141] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[142] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4918–4927, 2019.

[143] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

[144] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," *Advances in neural information processing systems*, vol. 33, pp. 3833–3845, 2020.

[145] T. Mensink, J. Uijlings, A. Kuznetsova, M. Gygli, and V. Ferrari, "Factors of influence for transfer learning across diverse appearance domains and task types," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9298–9314, 2021.

[146] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[147] J. Sontag, *Rice Processing: The Comprehensive Guide to Global Technology and Innovative Products*. Erling, 2014.

[148] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *International Conference on Learning Representations*, 2017.

[149] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[150] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[151] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, pp. 2285–2294, PMLR, 2015.

[152] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "NDDR-CNN: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3205–3214, 2019.

[153] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[154] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, vol. 5, pp. 2443–2446, IEEE, 1999.

[155] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[156] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei, "Label efficient learning of transferable representations across domains and tasks," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[157] N. Lawrence and A. Hyvärinen, "Probabilistic non-linear principal component analysis with gaussian process latent variable models.," *Journal of Machine Learning Research*, vol. 6, no. 11, 2005.

[158] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[159] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[160] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2018.

[161] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[162] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7229–7238, 2018.

[163] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, 2017.

[164] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[165] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

[166] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[167] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 764–773, 2017.

[168] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016.

[169] "Embedded systems developer kits & modules — nvidia jetson." `https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems`. Accessed: 2022-12-09.

[170] S. Abnar, M. Dehghani, B. Neyshabur, and H. Sedghi, "Exploring the limits of large scale pre-training," *arXiv preprint arXiv:2110.02095*, 2021.

[171] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, *et al.*, "Scaling vision transformers to 22 billion parameters," *arXiv preprint arXiv:2302.05442*, 2023.

[172] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10925–10934, 2022.