

Unsupervised Learning and Continual Learning in Neural Networks



Xu Ji

St Anne's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2021

Acknowledgements

I am very grateful for the support of my supervisor, Andrea Vedaldi. When we first met in 2015, while I was still an undergraduate at Imperial, neither of us could have known how deeply I would end up pursuing this subject. It was after talking to Andrea that I began to consider research seriously as an option, and over the years he has been the source of much inspiration, guidance and wisdom. To say that I was lucky would be an understatement.

I would also like to thank my co-supervisor Joao Henriques, for showing me what dedication looks like. Alex Rogers, for believing in me from the first interview. Wendy Adams, our CDT co-ordinator, without whom the CDT program would collapse. Ashish Thandavan and Jenny Hu, for supporting the crucial infrastructure and running of the lab. And my friends and colleagues who made Oxford the amazing place it was, among them Shuyu Lin, Sam Albanie, James Thewlis, Oliver Groth, Erika Lu, Sophia Koepke, Fatma Guney, Triantafyllos Afouras, Ruth Fong, Gul Varol, Arsha Nagrani, Olivia Wiles, Vicky Kalogeiton, Tom Jakab, Sylvestre Rebuffi, Christian Rupprecht, Yuki Asano, Elliott Wu, Tengda Han, Kai Han, Andrew Brown, Chuhan Zhang, Hala Lamdouar, Max Bain, Mandela Patrick, Sagar Vaze, Sebastien Ehrhardt, Bradley Gram-Hansen, Adam Golinski, Andrea Patane, Fabian Fuchs, Sasha Salter, Edward Wagstaff and Oliver Bent. I am also grateful to Andrew Zisserman and Phil Torr, who were always full of helpful advice.

I would like to thank Yoshua Bengio for the encouragement and stimulation generously shared. Razvan Pascanu, likewise, for opening my eyes. Devon Hjelm for all his insights and the warm welcome to Mila, despite the challenges of the pandemic. Balaji Lakshminarayanan for his supportive guidance. And Alyosha Efros for the fun discussions.

A heartfelt thank you to my close friend Bora Mollamustafaoglu, who has known me longest of all.

Abstract

For decades research has pursued the ambitious goal of designing computer models that learn to solve problems as effectively as humans can. Artificial neural networks – generic, optimizable models that were originally inspired by biological neurons in the brain – appear to provide a promising answer. However, a significant limitation of current models is that they tend to only be reliably proficient at tasks and datasets that they were explicitly trained on. If more than one task or dataset is being trained on, samples need to be appropriately mixed and balanced so that training on successive batches does not induce forgetting of knowledge learned in previous batches, which is an impediment to continual learning. Furthermore, associations need to be made explicit via paired input-target samples for the trained network to achieve its best performance on desired tasks; when the network is trained in an unsupervised manner without explicit targets, in an effort to reduce the cost of data collection, knowledge learned by the network transfers to desired tasks significantly worse compared to supervised training with explicit associations.

Each of these problems relates to the fundamental issue of generalization, which is the ability to perform well despite novelty. In chapter 2, we discuss conditions under which good generalization can be expected to arise, including small model size and similarity between training and test data in supervised, unsupervised and continual learning contexts. Chapter 3 proposes a method for predicting when a model does not generalize to a test sample, deriving generalization bounds that quantify predictive reliability using both model size and similarity with training data. Chapter 4 presents a clustering method that learns how to approximately separate data between semantic concepts with an unsupervised objective that does not use manual labels. Chapter 5 contains a method for performing the task of object localization without specialized training data, by repurposing saliency maps. Chapter 6 contains a continual learning method where the model is forced to reconsider previously held knowledge concurrent with new knowledge, and chapter 7 uses a dynamic architecture to suppress interference from new learning episodes on old knowledge.

Without solutions to these generalization problems, neural networks cannot learn effectively in real time from naturally sequential and un-annotated real world data, which limits their deployment options. Generalization is therefore a problem with immense practical implications, as well as being interesting theoretically and from the perspective of biologically-inspired learning.

Contents

1	Introduction	1
2	Background and Principal Themes	7
2.1	Generalization	8
2.1.1	Incidental Gradient Following	10
2.1.2	Unsupervised Learning as a Generalization Problem	13
2.1.3	Continual Learning as a Generalization Problem	16
2.1.4	Does Model Size Determine Generalization Ability?	18
2.2	Unsupervised Learning	26
2.2.1	Generative Methods	29
2.2.2	Discriminative Methods	35
2.3	Continual Learning	41
2.3.1	Assessing Forgetting	42
2.3.2	Distributed and Non-Distributed Memory	44
2.3.3	Implicit Attention Mechanisms	45
2.3.4	Replay	49
3	Predicting Unreliable Predictions by Shattering a Neural Network	53
4	Invariant Information Clustering for Unsupervised Image Classification and Segmentation	71
5	There and Back Again: Revisiting Backpropagation Saliency Methods	89
6	Automatic Recall Machines: Internal Replay, Continual Learning and the Brain	109
7	On the Role of Neurogenesis in Overcoming Catastrophic Forgetting	125
8	Conclusions	135
	References	139

The promise of the field of AI, spelled out explicitly at its inception in the 1950s and repeated countless times since, is to develop machines that possess intelligence comparable to that of humans. But AI has since been falling short of its ideal: although we are able to engineer systems that perform extremely well on specific tasks, they have still stark limitations, being brittle, data-hungry, unable to make sense of situations that deviate slightly from their training data or the assumptions of their creators...

— François Chollet, 2019

1

Introduction

How does one teach a computer to perform useful tasks? This is the motivation behind neural networks, arguably one of the most fascinating developments in computer science. Modern neural networks originate from the biologically inspired perceptron model (Rosenblatt, 1958), and despite much effort on testing, improving and explaining their performance in the interim, the high-level fundamentals have not changed much.

We begin with a brief overview. This is intended to refresh the memory of the reader rather than as a first instruction; for the latter, see Bishop, 2006. A neural network is a kind of *inference function*: a function f that takes inputs $x \in \mathcal{X}$ and maps them to outputs $y \in \mathcal{Y}$ given parameters $\theta \in \Theta$ which are the variable components of the function¹. Thus $f : \mathcal{X}, \Theta \rightarrow \mathcal{Y}$. More specifically, neural networks are a class of such functions where the computation is structured as a compositional sequence of distributed operations. A standard structure is a chain of layers where each layer involves a linear matrix-vector product followed by a non-linear activation function, which is also called a multi-layer perceptron or MLP (Pal and Mitra, 1992). Notable variations include convolutional layers (Krizhevsky, Sutskever, and Hinton, 2012; Simonyan and Zisserman, 2014; LeCun

¹Typically values of weights and biases, though these could also contain discrete architectural parameters, activation function hyperparameters, and so on.

et al., 1998), recurrent layers (Hochreiter and Schmidhuber, 1997), residual layers (He et al., 2016) and multiplicative attention layers (Devlin et al., 2018). Note that since MLPs are universal function approximators (Hornik, Stinchcombe, and White, 1989), the recent popularity of such variants has arguably not been due to fundamental improvements in expressivity, but likely compatibility with current learning procedures; for example, as beneficial constraints on local minima found (Swirszcz, Czarnecki, and Pascanu, 2016).

In the most general terms, a learning procedure h is a function that takes an instance of model parameters θ and returns a new version with changed behaviour, so $h : \Theta \rightarrow \Theta$. The ideal behaviour of inference function $f(\cdot; \theta)$ is typically specified by a cost or objective function L which is parameterized by data, so $L : \Theta, \Omega \rightarrow \mathcal{R}$, where data $\Omega = \bigcup_{K \in \mathbb{N}^+} (\mathcal{X}, \mathcal{Y})^K$ consists of lists of input-target pairs of any length. Crucially, known training data $S \in \Omega$ is incomplete; typically, it is a finite sample from a data distribution. The full extent of eventual test samples, say $\tilde{S} \in \Omega$, is assumed to be inaccessible, otherwise there would be no point in training a model. Thus the key question is, given known data S and a constraint limiting the deviation of test data² $\tilde{S}$ from S , we would like to obtain the instance of θ that minimizes objective $L(\theta; \tilde{S})$. If S and $\tilde{S}$ are sufficiently similar, for instance drawn from the same distribution, then one can simply design h to output θ that minimizes $L(\theta; S)$ instead, by letting it inspect S and L . This is known as empirical risk minimization.

Neural networks arguably offer the most compelling direction in the search for models capable of being substituted for human inference. Their appeal is threefold:

- **Adaptability.** Neural networks are generic and capable of modelling solutions to many diverse problems. They are highly expressive; bounds on the number of distinct linear regions in input space increase exponentially with depth (Raghu et al., 2017). In fact, modern deep architectures tend to be overparameterized, meaning they have more parameters than necessary for perfect fitting on the training set (Belkin et al., 2019).

²There is no free lunch (Wolpert and Macready, 1997); one cannot optimize for test data if it is entirely unconstrained.

- **Optimizability.** There is a good chance of attaining said solutions via automated optimization, given limited resources: computational power, time, memory and data. Empirically speaking, this seems to be the case despite non-convexity. That is, local minima found by popular optimizers such as stochastic gradient descent appear to be good minima nonetheless.
- **Biological relevance.** The iterative loop of exposure to a few samples, learning from them by adapting connectivity weights, and changing behaviour as a result, bears a strong high-level resemblance to how the brain operates. Thus there is much potential for cross-disciplinary links with neuroscience and psychology. The brain offers both the target cognitive standard and a wealth of inspiration for how we might achieve it, and vice versa, artificial neural networks could help us understand our own cognitive processes.

Given a number of well-publicised successes in the past decade – on image recognition, playing video games and text generation for example (Krizhevsky, Sutskever, and Hinton, 2012; Mnih et al., 2013; Devlin et al., 2018) – it may seem contradictory to suggest that neural networks are clearly still in the early stages of their development. However, as Chollet notes in the opening quote, the field is nowhere near to its ideal. In other words, one could argue that these achievements were landmark demonstrations of the general versatility of computer algorithms, but fall short of being landmarks in the development of human-level artificial intelligence, for several reasons.

First, human-level intelligence is defined more by decent performance on a wide range of diverse tasks than optimal performance on narrow tasks. That is, performance *generalizes* across diverse changes in inputs, or equivalently is *robust* to these changes. Humans also tend to be aware of when they are unlikely to generalize, naturally defaulting to inaction above a threshold of uncertainty. Second, learning is on real-world data that is highly temporally correlated, rather than independently sampled and identically distributed (“i.i.d.”), and also highly imbalanced, where sample count is largely immaterial for retention; one can remember a single glimpse

of a stranger’s face for decades, whilst being oblivious to environmental details encountered on a daily basis. Third, learning is unsupervised: the signal indicating which associations to learn is not provided by an external human oracle, but self-generated most of the time, which is obvious from introspection.

Neural networks, on the other hand, are still mostly trained and tested on narrow tasks, which has likely been exacerbated by the field’s preoccupation with benchmarking (Chollet, 2019). For the best performance, data samples need to be densely annotated – i.e. the desired associations made explicit – and additionally, this is expensive when annotations are manual. For training on multiple tasks, effort has to be made to mix samples so that no task is missing in any small training window, otherwise catastrophic forgetting (McCloskey and Cohen, 1989) of that task occurs. Datasets also need to be balanced to counteract dataset bias (Kotsiantis, Kanellopoulos, and Pintelas, 2006). Even then, in a balanced i.i.d. setting, one cannot expect robust generalization to in-distribution data – networks make wildly erroneous yet confident predictions when presented with pixel corruptions that should be immaterial to image class (Nguyen, Yosinski, and Clune, 2015) – let alone generalization to material novelty, which means data that is novel at an abstract level. Note the latter is a hallmark of human cognition; abstract generalization has long been used as a metric for human intelligence (Chollet, 2019; Binet and Simon, 1904). According to this standard, then, modern neural networks are unintelligent, because they tend to only be proficient at what they were specifically trained for.

The aim of this work is to explore these weaknesses: reliance on supervision, reliance on i.i.d. sampled data, and poor generalization in general. We begin with generalization in section 2.1 because it subsumes the others; one can formulate unsupervised learning and continual learning as generalization problems. We discuss the idea that the solution to good generalization is “smaller” models, given various definitions of model size, and high similarity between training and test data. We examine quantifying generalization ability with error bounds, and in chapter 3 we present a method of error bounding that involves splitting the network into co-dependent linear functions, allowing the predictive uncertainty for test samples

to be quantified, exposing information within the network about what it does not know. The bound happens to capture precisely the same two aforementioned factors for generalization: model size and similarity with training data.

Section 2.2 focuses on unsupervised learning, which is the problem of learning representations without manual annotation that generalize to downstream tasks. We begin with the initial inspirations from biological neurons and Hebbian learning, and discuss the extent to which modern literature on unsupervised learning has deviated from its original motivations. The paper presented in chapter 4 is a clustering method that supported the recent surge in interest in contrastive learning; amongst the method’s strengths is network outputs are directly semantically interpretable without supervised fine-tuning. In chapter 5, we investigate methods for solving the pointing game using saliency maps, a task for which supervisory training signal is not provided.

In section 2.3 we discuss continual learning, which is the problem of generalizing across different tasks given sequential exposure. We begin with the double-edged sword of distributed representations – greater expressivity, beneficial transfer being enabled, but negative transfer or *catastrophic forgetting* also being enabled (French, 1991). We examine natural anti-forgetting and fast-learning mechanisms that neural networks employ during training, which although appear to be insufficient as stand-alone solutions, are nonetheless inadequately acknowledged in the literature. Both presented papers in this area investigate biologically-inspired strategies for mitigating catastrophic forgetting. The first paper in chapter 6 uses learning on internally generated replay data, which plays a major role in human cognition, and the second in chapter 7 reduces representational distributedness by increasing capacity online, where the biological equivalent is neurogenesis.

We conclude with an overview of the arguments presented and an outlook on directions for future work.

This is an integrated thesis based on the following papers, in order of inclusion:

- *Predicting Unreliable Predictions by Shattering a Neural Network.*
[First author.]
Xu Ji, Razvan Pascanu, Devon Hjelm, Andrea Vedaldi, Balaji Lakshminarayanan, Yoshua Bengio. Pre-publication.
- *Invariant Information Clustering for Unsupervised Image Classification and Segmentation.*
[First author.]
Xu Ji, Joao Henriques, Andrea Vedaldi. In ICCV 2019.
- *There and Back Again: Revisiting Backpropagation Saliency Methods.*
[Joint first author.]
Sylvestre Rebuffi*, Ruth Fong*, Xu Ji*, Andrea Vedaldi. In CVPR 2020.
* denotes equal contribution.
- *Automatic Recall Machines: Internal Replay, Continual Learning and the Brain.*
[First author.]
Xu Ji, Joao Henriques, Tinne Tuytelaars, Andrea Vedaldi. In NeurIPS Workshop on Baby Mind 2020.
- *On the Role of Neurogenesis in Overcoming Catastrophic Forgetting.*
[Second author.]
German Parisi, Xu Ji, Stefan Wermter. In NeurIPS Workshop on Continual Learning 2018.

2

Background and Principal Themes

Presumably the child brain is something like a notebook as one buys it from the stationers. Rather little mechanism, and lots of blank sheets. Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed.

— Alan Turing, 1950

2.1 Generalization

Why begin with a work on unsupervised learning and continual learning with generalization? Because these are both specialised generalisation problems. Moreover, generalization is arguably the central problem of machine learning, as well as the key difference between the abilities of human brains and current artificial neural networks. Thus it is a problem of immense theoretical and practical concern.

Generalization can be defined as the ability of a model to perform well on data samples that were unseen in their exact form during training. Different levels of difficulty exist; typically works distinguish between settings where training and test data are considered to be drawn from the same distribution (“in-distribution”), settings where the distributions are materially different (“out-of-distribution”), and various hybrids where the model is exposed to dissimilar and similar training distributions before inference on test samples. This includes a wide range of subdomains including transfer learning (generalizing to test samples from a new distribution given training on that distribution, e.g. finetuning representations learned in an unsupervised manner), few-shot learning (generalizing to a new distribution given a minimal number of training samples from that distribution; Snell, Swersky, and Zemel, 2017), and continual learning (generalizing across different distributions given sequentially ordered exposure; Lopez-Paz and Ranzato, 2017).

A data distribution is a function $p(x)$ that returns a probability density for data x being observed. In general, many different empirically observed datasets can be drawn from the same data distribution. Thus the idea of samples being drawn from different data distributions, which is widely used in work on out-of-distribution generalization, is not a straightforward concept; one can generally consider any two datasets to be draws from the same data distribution – merely unlucky draws,

if they appear dissimilar. In real world scenarios one does not have access to the ground truth data distribution(s) that generate data. Thus although much classical work on generalization (including our work in chapter 3) views datasets through a statistical lens, it can be simpler to take the empirical view and disregard distributions altogether (for example measuring dataset similarity by correlation in their induced gradients, section 2.1.1).

Generalization is not a magic phenomenon; when models generalize well to new samples, prior helpful information (e.g. a good data representation) may not have been intentionally or explicitly specialized to the new data samples, but it happened to be, all the same. Note there is no test of meaningful generalization unless test data is not part of training data, otherwise memorization would suffice – hence the specification of *unseen* test samples – but there is no reasonable expectation of performing well on test data unless it overlaps on some level with training data or priors, which together fully determine the final state of the model after training. The priors, or a-priori assumptions, constitute all of the variables considered in the previous chapter that are fixed prior to training: model f (typically, network architecture), initialization of its parameters θ , and learning procedure h and any objective function(s) L that it uses¹. Training and test data is excluded from priors because it is assumed to be outside the engineer’s control (but use of the training data, as specified by h , is not). Thus the question of generalization can be summarized as: given a search space over priors, which priors are optimal for maximising guarantees on test performance, given some distribution over training environment initializations and constraints on the deviation of test data from the latter? The term environment *initialization* is used because although we want to control for the environment in order to attribute changes in test performance purely to changes in priors, in general one can only fix the initial state of the environment since different priors can induce different training data; for instance, in reinforcement learning, the outputs of models change subsequent training data, and the continual learning method in chapter 6 involves training the model on self-generated data.

¹For stateful learning procedures, i.e. with time-varying parameters, these parameters can be included in θ .

In this light, much of the recent work in machine learning can be viewed as a manual or heuristic meta-optimization for the best priors. When certain priors are shown to perform well across diverse instances of generalization problems – for example, SGD optimization or He initialization (He et al., 2015) – researchers typically take their proficiency for granted, fix them, and iterate on other facets such as structure of learning procedure h and objective L (fig. 2.1).

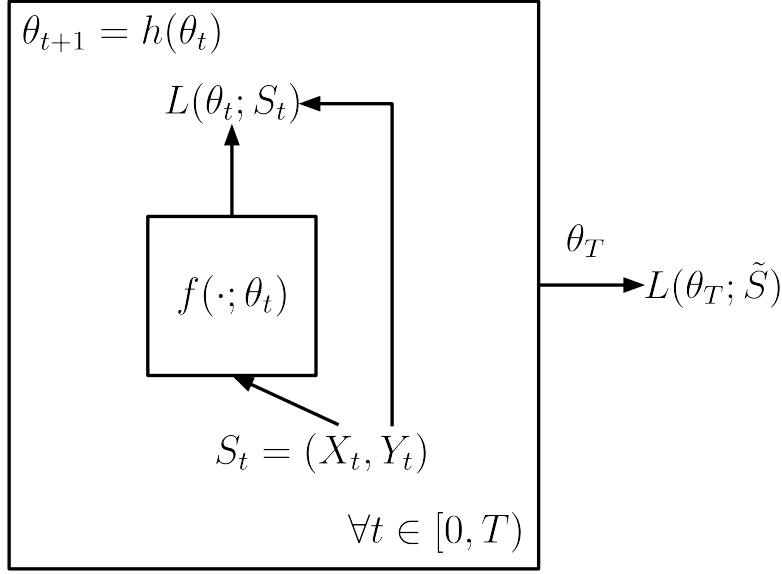


Figure 2.1: Generic computational graph for supervised learning on batched data S_t where t denotes timestep and h has single objective function L . A typical learning procedure h performs stochastic gradient descent on the objective function to update θ .

2.1.1 Incidental Gradient Following

During training we assume that we do not have access to test data $\tilde{S}$, but what if we could fast forward to test time, when performance on $\tilde{S}$ is measured? Under what conditions would a model $f(\cdot; \theta)$ be optimal for an objective $L(\theta; \tilde{S})$? In the previous section we suggested that test performance should be correlated to “overlap” of test data with training data and priors; now we provide a framework for describing more formally what this overlap means. In order to do so, we make use of the Taylor expansion, which is a generic method for function approximation with local gradients. The Taylor expansion often appears in the neural network

literature in the context of continual learning (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018; Yin et al., 2020).

Let us consider the popular case where all training data is fixed (that is, not the general case where the model’s outputs are allowed to change its training data). Call the training data $S \in (\mathcal{X}, \mathcal{Y})^N$ and test data $\tilde{S} \in (\mathcal{X}, \mathcal{Y})^M$, where each has N and M samples respectively. Let the inputs and targets be called X and Y , $(X^j, Y^j) = S^j$ for $1 \leq j \leq N$, and likewise for test data $\tilde{S}$. Let the objective function (or “loss”) L take both dataset-sized and sample-sized inputs, i.e. $L : \Theta, \Omega \rightarrow \mathcal{R}$ with $\Omega = \cup_{K \in \mathcal{N}^+} (\mathcal{X}, \mathcal{Y})^K$.

First note that using the Taylor expansion, we can write inference of the network’s output on a test input $\tilde{X}^i$ as a form of nearest neighbours. Let $\mathcal{O}$ denote an upper bound, $\forall z : \mathcal{O}(f(z)) \leq c f(z)$ for some constant c .

$$\forall i \in [1, M], j \in [1, N] : \quad f(\tilde{X}^i; \theta) = f(X^j; \theta) + \mathcal{O}(\|\tilde{X}^i - X^j\|), \quad (2.1)$$

as well as for the objective function at the level of sampled test pairs $\tilde{S}^i$:

$$\forall i \in [1, M], j \in [1, N] : \quad L(\theta; \tilde{S}^i) = L(\theta; S^j) + \mathcal{O}(\|\tilde{S}^i - S^j\|), \quad (2.2)$$

and the objective function at the level of the dataset, if $M = N$:

$$L(\theta; \tilde{S}) = L(\theta; S) + \mathcal{O}(\|\tilde{S} - S\|), \quad (2.3)$$

because in each case, the error in appropriating a result from training data scales with distance to that data, hence maximally accurate inference using neighbours would mean picking the nearest neighbour. This assumes that L is differentiable and has bounded first order gradients; similar assumptions apply to all Taylor expansions. The first order expansion of eq. (2.3) is:

$$L(\theta; \tilde{S}) = L(\theta; S) + (\tilde{S} - S)^\top \nabla_S L(\theta; S) + \mathcal{O}(\|\tilde{S} - S\|^2). \quad (2.4)$$

This suggests that the further test data $\tilde{S}$ is from training data S , the more important the representation given by θ is for determining generalization, because as $\|\tilde{S} - S\|$ tends to zero, identical behaviour becomes a given, and vice versa, the larger $\|\tilde{S} - S\|$ is, the wider the possible range of $(\tilde{S} - S)^\top \nabla_S L(\theta; S)$, given any bounded range for $\|\nabla_S L(\theta; S)\|$ over possible values for θ .

Incidental gradient following. Consider the training algorithm $h : \Theta \rightarrow \Theta$, where h is a map between instances of f 's parameters. Let h have access to training data S and objective L , and let it take the form of one step of stochastic gradient descent with learning rate η and data batch $S_t \subset S$ on any instance of parameters θ_t :

$$\theta_{t+1} = h(\theta_t) = \theta_t - \eta \nabla_{\theta_t} L(\theta_t; S_t). \quad (2.5)$$

Expanding the loss L on test data $\tilde{S}$ around the parameters, we obtain:

$$L(\theta_{t+1}; \tilde{S}) = L(\theta_t; \tilde{S}) + (\theta_{t+1} - \theta_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\theta_{t+1} - \theta_t\|^2), \quad (2.6)$$

$$= L(\theta_t; \tilde{S}) - \eta \nabla_{\theta_t} L(\theta_t; S_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\eta \nabla_{\theta_t} L(\theta_t; S_t)\|^2), \quad (2.7)$$

which means that training on batch S_t descends the loss on test data $\tilde{S}$ according to how correlated their gradients are. All else equal, namely assuming the same training step magnitude of $\|\eta \nabla_{\theta_t} L(\theta_t; S_t)\|$, this incidental descent on test loss is uniquely maximised when the direction of the gradients $\nabla_{\theta_t} L(\theta_t; S_t)$ and $\nabla_{\theta_t} L(\theta_t; \tilde{S})$ are identical. A special instance of this best case is when the data itself is identical (but in general, the same gradient can be induced by different samples).

Now let us zoom out of this single timestep and allow h to be applied for multiple timesteps before being evaluated. Let t in θ_t correspond to training step index and fix initial parameters θ_0 . Then after T steps, the final parameters are given by:

$$\theta_T = \overbrace{h(\dots h(\theta_0))}^{T \text{ applications}}, \quad (2.8)$$

which incur test loss:

$$L(\theta_T; \tilde{S}) = L(\theta_0; \tilde{S}) + (\theta_T - \theta_0)^\top \nabla_{\theta_0} L(\theta_0; \tilde{S}) + \mathcal{O}(\|\theta_T - \theta_0\|^2), \quad (2.9)$$

which is minimized when:

$$\theta_T - \theta_0 = - \sum_{t=0}^{T-1} \eta \nabla_{\theta_t} L(\theta_t; S_t) = - \nabla_{\theta_0} L(\theta_0; \tilde{S}), \quad (2.10)$$

all else equal, i.e. for any fixed $\|\theta_T - \theta_0\|$. This indicates that the h that yields optimally general θ (using a definition of optimal generalization as minimal $L(\theta; \tilde{S})$) given T repeated applications, is one that *eventually* and *incidentally* follows the

same direction of change in parameters as training on $\tilde{S}$, despite not having access to the latter. Hence we refer to this ideal behaviour as *incidental gradient following*.

Note one can also expand around the parameters at each timestep, rather than just at θ_0 :

$$L(\theta_T; \tilde{S}) \tag{2.11}$$

$$= L(\theta_0; \tilde{S}) + \sum_{t=0}^{T-1} L(\theta_{t+1}; \tilde{S}) - L(\theta_t; \tilde{S}), \tag{2.12}$$

$$= L(\theta_0; \tilde{S}) + \sum_{t=0}^{T-1} (\theta_{t+1} - \theta_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\theta_{t+1} - \theta_t\|^2), \tag{2.13}$$

$$= L(\theta_0; \tilde{S}) + \sum_{t=0}^{T-1} -\eta \nabla_{\theta_t} L(\theta_t; S_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\eta \nabla_{\theta_t} L(\theta_t; S_t)\|^2), \tag{2.14}$$

which is minimized when the training gradient $\nabla_{\theta_t} L(\theta_t; S_t)$ follows the true gradient $\nabla_{\theta_t} L(\theta_t; \tilde{S})$ at every timestep, given any fixed $\|\eta \nabla_{\theta_t} L(\theta_t; S_t)\|$. Note that this finer grained approximation actually has a larger approximation error compared to eq. (2.9) if all else equal is assumed, due to the triangle inequality:

$$\forall k \geq 0 : \quad \sum_{t=0}^{T-1} k \|\theta_{t+1} - \theta_t\|^2 \geq k \|\theta_T - \theta_0\|^2, \tag{2.15}$$

but in practice we would expect all else equal to not hold, i.e. the constants in the upper bound denoted by $\mathcal{O}$ to be lower for eq. (2.14) than eq. (2.9), because the step sizes in the former are generally smaller.

2.1.2 Unsupervised Learning as a Generalization Problem

Collection cost. In unsupervised learning, the goal is to learn a good data representation for a downstream generalization problem using unsupervised data. The distinction between unsupervised and supervised data is often not strict in the literature (Goodfellow, Bengio, and Courville, 2016), but in this work we characterize it as a difference of *collection cost*. Pixels of an image are naturally collected together, as are the different visual frames in a video, the audio track with visual track in a video, or the text accompanying an image on a webpage. Unsupervised data is cheap to obtain from a collection cost perspective – specifically, extra bits are obtainable without needing to expend extra effort – whereas supervised data is not.

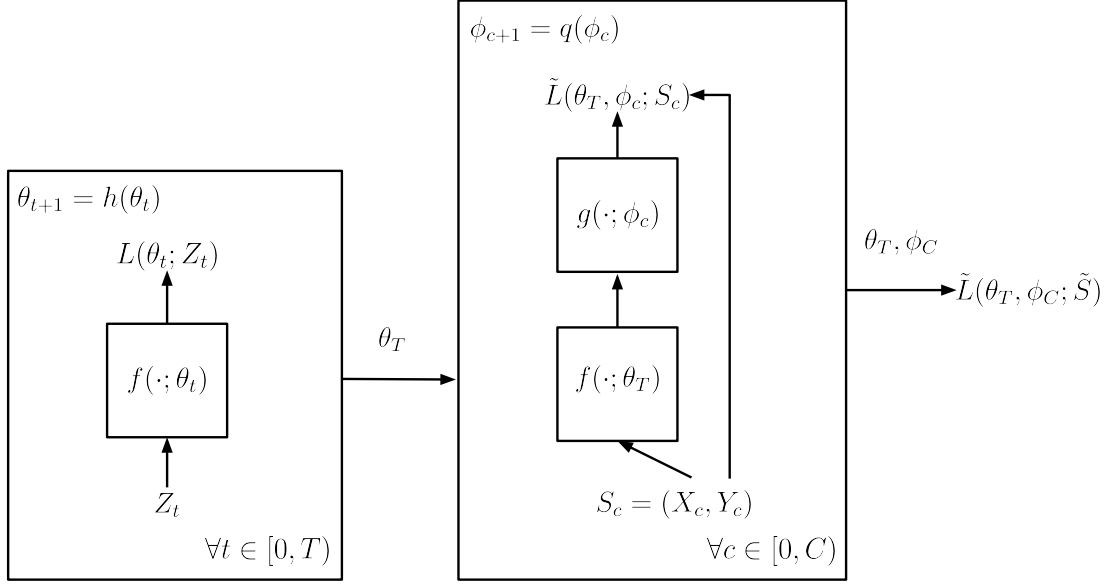


Figure 2.2: Generic computational graph for unsupervised learning on batched data $Z_t \in \Omega$, where h has single objective function $L : \Theta, \Omega \rightarrow \mathcal{R}$ and $\Omega = \bigcup_{K \in \mathcal{N}^+} \mathcal{X}^K$, i.e. L is defined on parameters and the input space only. Parameters θ are optimized in the left box before being reused for the downstream task in the right box. h and q typically perform stochastic gradient descent on their objective function values L and $\tilde{L}$ with respect to parameters θ and ϕ , for T and C timesteps.

We will expand on this idea in section 2.2, but note for now that low collection cost typically implies informational redundancy; due to the causal nature of the world, data that is spatio-temporally clustered is generally influenced by shared factors and thus co-dependent across dimensions. Informational redundancy does not imply low collection cost, however, because the latter is contextualised by the experimental resources of the engineer; images and labels are highly informationally redundant, but not low collection cost when the labels are obtained at additional material cost to the images.

Given unsupervised data Z , the goal is to use it to devise some h (or commonly, simply the objective function L used by h) such that h produces θ_T which is incidentally optimal for a downstream supervised objective $\tilde{L}$ on test set $\tilde{S}$, $\tilde{L}(\theta_T, \phi_T; \tilde{S})$ (fig. 2.2). This description includes a large number of possible settings; Z could be data with any number of modalities, including latent codes, and f could be a generative task or discriminative task for example (section 2.2). Z could even

contain prediction targets, which are copied to the output of f using an identity function; we merely collect all dimensions into one variable Z for simplicity. The point is that Z incurs low collection cost, but the parameters learned from it should transfer well to a downstream objective that does *not*, in general, incur low collection cost, so that the benefits of low collection cost (chiefly, large sample volume at low cost) can also benefit the latter.

The downstream objective $\tilde{L}$ does not need to have its own trainable parameters; ϕ could be empty, which is the case when the output of f is directly evaluated after unsupervised training (for example, in the clustering algorithm presented in chapter 4). But in the more general case, if ϕ is non-empty, it is trained using training set S instead of test set $\tilde{S}$, since $\tilde{S}$ is not accessible for training. Thus fig. 2.2 contains two generalization problems: generalization of θ and ϕ to $\tilde{S}$.

Let h be applied for T timesteps to yield θ_T , as in eq. (2.8). Then assume θ_T is fixed while learning procedure q iterates on ϕ_0 to yield ϕ_C after C timesteps. Call the difference between performance on training and test data δ , which is often referred to as the *generalization gap* (Goodfellow, Bengio, and Courville (2016, fig 5.3; chapter 5)):

$$\delta \triangleq \tilde{L}(\theta_T, \phi_C; \tilde{S}) - \tilde{L}(\theta_T, \phi_C; S). \quad (2.16)$$

Then the final loss on downstream test data can be expanded as:

$$\begin{aligned} & \tilde{L}(\theta_T, \phi_C; \tilde{S}) \\ &= \tilde{L}(\theta_T, \phi_C; S) + \delta \\ &= \tilde{L}(\theta_0, \phi_0; S) + [\theta_T - \theta_0, \phi_T - \phi_0]^\top \nabla_{\theta_0, \phi_0} \tilde{L}(\theta_0, \phi_0; S) \\ & \quad + \mathcal{O}(\|\theta_T - \theta_0, \phi_T - \phi_0\|^2) + \delta \\ &= \tilde{L}(\theta_0, \phi_0; S) + (\theta_T - \theta_0)^\top \nabla_{\theta_0} \tilde{L}(\theta_0, \phi_0; S) + (\phi_T - \phi_0)^\top \nabla_{\phi_0} \tilde{L}(\theta_0, \phi_0; S) \\ & \quad + \mathcal{O}(\|\theta_T - \theta_0, \phi_T - \phi_0\|^2) + \delta, \end{aligned} \quad (2.17)$$

which is minimized by h when 1) $\theta_T - \theta_0 = -\nabla_{\theta_0} \tilde{L}(\theta_0, \phi_0; S)$, and 2) generalization gap δ is minimized, all else equal (i.e. for any fixed $\|\theta_T - \theta_0\|$). Thus to a first approximation, changes in θ_0 produced by an ideal unsupervised learning procedure

h should incidentally follow what they would be if $g \circ f$ were trained directly on downstream training data S , ideally whilst also producing smaller or simpler representations to minimize the generalization gap (this is known as Occam’s Razor and will be discussed further in section 2.1.4). Therefore one can think of the ideal unsupervised learner h as a surrogate for downstream learning objective $\tilde{L}$: it changes θ in similar ways even though θ is wrapped in a different objective. This is a useful intuition to keep in mind when designing h , and in section 2.2 we will consider several unsupervised learning methods that exemplify this idea.

2.1.3 Continual Learning as a Generalization Problem

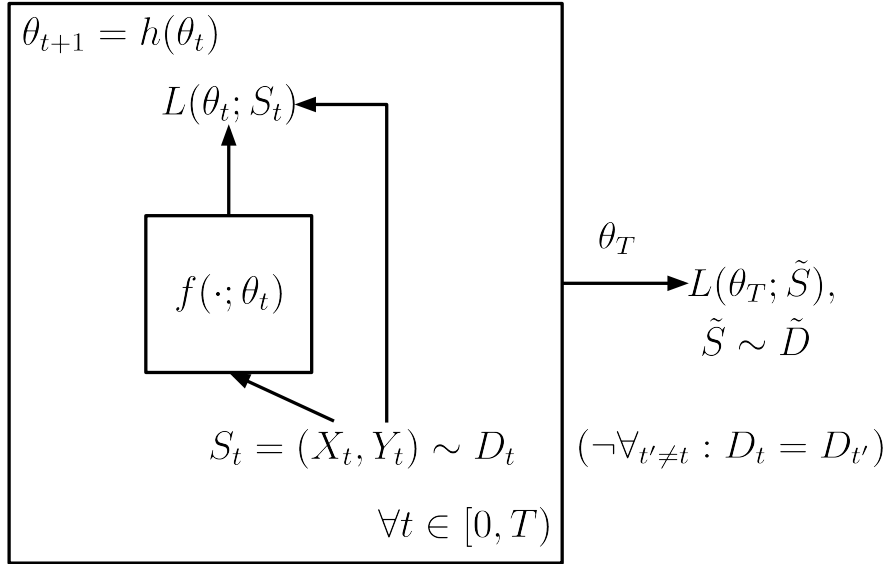


Figure 2.3: Generic computational graph for continual learning on batched data S_t , where h has single objective function L . Unlike in figs. 2.1 and 2.2, h should not perform stochastic gradient descent on $L(\theta_t; S_t)$ because this does not incidentally optimize $L(\theta_T; \tilde{S})$, due to the training batches S_t being materially different to test data $\tilde{S}$.

In continual learning, the goal is the same as the supervised and unsupervised learning problems previously introduced: to obtain parameters that generalize to some held-out test set. Continual learning specifically refers to a subset of these problems where the training data is ordered. More precisely, the material characteristics of the training batch – such as the distribution from which they can be considered to be drawn – vary with the training timestep. Figure 2.3 specifies

the weakest condition that satisfies this, namely $\neg \forall_{t' \neq t} D_{t'} = D_t$, but in general the difference in distributions across timesteps is not only materially present, but can be high. Material difference in two distributions can be defined as training on samples from one distribution compromising performance on samples drawn from the other. In supervised classification for example, ordering training batches such that classes are trained on sequentially, rather than each batch being formed of uniform samples from across classes, is enough to severely compromise performance on a test set containing all the classes. This is known as catastrophic forgetting or interference (McCloskey and Cohen, 1989) and will be discussed more thoroughly in section 2.3. In this section we begin our discussion of the topic by noting that catastrophic forgetting – which being characterized by test loss increasing with training time rather than decreasing, exhibits the exact opposite of improving generalization – can be directly expressed in our incidental gradient following paradigm. Specifically, it is when the implemented gradients do *not* follow the ideal gradients:

$$L(\theta_{t+1}; \tilde{S}) = L(\theta_t; \tilde{S}) + (\theta_{t+1} - \theta_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\theta_{t+1} - \theta_t\|^2), \quad (2.18)$$

$$L(\theta_{t+1}; \tilde{S}) > L(\theta_t; \tilde{S}) \iff (\theta_{t+1} - \theta_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) + \mathcal{O}(\|\theta_{t+1} - \theta_t\|^2) > 0. \quad (2.19)$$

If h simply uses SGD to train on training batch S_t , then we can replace the parameter update with the gradient from S_t :

$$\begin{aligned} & (\theta_{t+1} - \theta_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) > 0 \\ \iff & -\eta \nabla_{\theta_t} L(\theta_t; S_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) > 0 \\ \iff & \nabla_{\theta_t} L(\theta_t; S_t)^\top \nabla_{\theta_t} L(\theta_t; \tilde{S}) < 0, \end{aligned} \quad (2.20)$$

which refers to the case when the training gradient is decorrelated from the ideal gradient, or geometrically the angle between them is greater than 90° . Hence decorrelated gradients (eq. (2.20)) implies the test loss increases (eq. (2.19), ignoring approximation error), and moreover decorrelated gradients are *expected* in continual learning since training batches S_t are drawn from different data distributions and test data $\tilde{S}$ is typically sampled from across all training distributions, meaning it is guaranteed to contain samples from distributions not in S_t , given sufficient sample size.

Thus all continual learning methods replace h with a procedure that is not standard SGD on batch S_t . A simple method, known as replay, is to let h accumulate samples from training batches into a memory, and sample from that memory for synthetic batches to replace S_t . Since each synthetic batch is sampled from all seen data distributions, this increases the correlation of its gradient with $\nabla_{\theta_t} L(\theta_t; \tilde{S})$ such that it becomes positive in eq. (2.20) and the test loss decreases. Other methods such as Chaudhry et al., 2018 and Lopez-Paz and Ranzato, 2017 are derived explicitly from the Taylor expansion and involve projecting $\nabla_{\theta_t} L(\theta_t; S_t)$ onto the closest vector such that it is correlated with gradients from samples in accumulated memory, before applying it to θ . A theme should now be apparent. In all cases the goal is to avoid eq. (2.19): to manufacture an update $\theta_{t+1} - \theta_t$ that is incidentally correlated with the gradient from test data, so that the test loss decreases with training, and thus generalization improves.

2.1.4 Does Model Size Determine Generalization Ability?

The idea that smaller or simpler theories are superior, all else equal, has a long history. It is commonly referred to as Occam’s Razor, or “plurality should not be posited without necessity”, and is attributed to William of Ockham, a religious philosopher born in 1285 in the village of Ockham, England who was also a student at Oxford (Spade, 1999). “Razor” refers to the shaving away of extraneous assumptions. Ockham’s own usage of the notion included justifying the attribution of divine miracles to God; the principle by itself is not an irrefutable logical argument.

However, the idea that a model’s performance on training data should be offset by model size, when selection is performed on trained models, is a recurring theme in machine learning. Importantly, in certain cases it is no longer heuristic, as one can prove that smaller models induce lower probabilistic upper bounds on generalization error. Recall that a model’s final state at evaluation, including details such as size, is fully determined by priors and training data, and training data (or more generally in the case of active models, environment initialization) is typically

fixed. So this principle amounts to a criterion on priors: model architecture, initialization and training procedure.

Classical paradigm. Why should we penalize model size at all? Why do we need an evaluation metric other than training error? In general, many model instances can attain the same training error but different test errors, and since generalization ability is defined by test error, it is the latter that we actually want to optimize for. The incidental gradient following approximation introduced previously is useful for motivation and intuition, but not for predicting generalization in practice, because it is post-hoc: it assumes access to realized test data $\tilde{S}$, which we do not have. Obviously if the test data is available, quantifying generalization is trivial; one can simply compute the test error. Thus a useful metric for generalization ability, as a function of a prior – for instance, architecture – does not depend on realized test data but its constraint relative to training data. Recall that due to the no free lunch theorem (Wolpert and Macready, 1997), this is the best we can do: exact test data is unknown but needs to be constrained, and the constraints should be loose enough that they have a good chance of reflecting test data with fidelity, otherwise the metric is irrelevant in the eventual test setting.

The Akaike information criterion (AIC, Akaike, 1974) is one metric for predicting the generalization ability of models. It specifies that models should be selected by minimizing:

$$\text{AIC}(\theta, S) = 2|\theta| + \log(L(\theta; S)), \quad (2.21)$$

where $L(\theta; S) = -p(\theta; S)$, i.e. the negative likelihood of the model given training data. This captures training performance offset by model size $|\theta|$ as the number of parameters, which is a general theme in classical quantifiers for generalization ability.

Another example of this is error bounds that use Vapnik-Chervonenkis (VC) dimension (Vapnik, 1968). VC dimension is defined on a hypothesis set or *model class* and is the size of the largest dataset that can be arbitrarily labelled if one is allowed to pick any model in the class. It is a measure of model size that is not

exactly the number of parameters, but scales with it, since the larger the number of parameters, the more expressive the class of all models with that architecture. In short, it proxies complexity of a model as the adaptability of the space within which it was searched. One can write the following probabilistic upper bound on test error using VC dimension (see Mohri, Rostamizadeh, and Talwalkar, 2018 for a review). Let $\hat{R}_S(f)$ denote empirical error of f , i.e. error on training dataset S with N samples, and let $R(f)$ denote true error of f , i.e. expected error over test datasets, also of size N , where samples are drawn i.i.d. from the same distribution as those in S . Then with probability $> 1 - \delta$, for all models f in class $\mathcal{F}$:

$$R(f) \leq \hat{R}_S(f) + \sqrt{\frac{2 \text{VCDim}(\mathcal{F}) \log \frac{eN}{\text{VCDim}(\mathcal{F})}}{N}} + \sqrt{\frac{\log \frac{1}{\delta}}{2N}}, \quad (2.22)$$

$$= \hat{R}_S(f) + \mathcal{O}\left(\sqrt{\frac{\log(N/\text{VCDim}(\mathcal{F}))}{N/\text{VCDim}(\mathcal{F})}}\right) + \sqrt{\frac{\log \frac{1}{\delta}}{2N}}, \quad (2.23)$$

and $\sqrt{\log(x)/x}$ is monotonically decreasing for $x > e$. This suggests that minimizing the upper bound on true error corresponds to maximizing the training dataset size and minimizing model size, all else equal.

The idea that large model size is bad for generalization, all else equal, is backed by experimental evidence. It has long been accepted that test error empirically follows a “U” shaped curve with respect to model size, with the initial descent occurring due to insufficient expressivity being remedied, followed by an ascent in test error due to the model *overfitting* or overspecializing to training data (fig. 2.4 and Goodfellow, Bengio, and Courville, 2016). Under this paradigm, the optimal model size is at the base of the curve, which is approximately where the smallest model size for which training data can be perfectly fitted is (recall that test data cannot be used for model selection or optimization). This was the standard view until at least 2019, when the double descent phenomenon, which we will discuss next, was documented.

Double descent paradigm. Recently, Belkin et al., 2019 and Nakkiran et al., 2019 empirically observed that for deep neural networks, when the horizontal axis in fig. 2.4 is extended into millions of parameters, the increasing trend in test

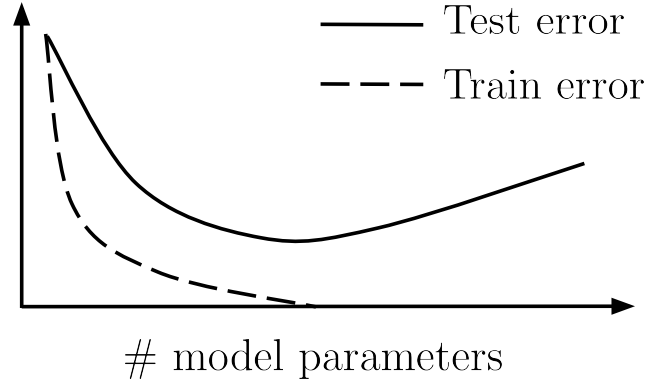


Figure 2.4: Classical relationship between model size and training and test errors.

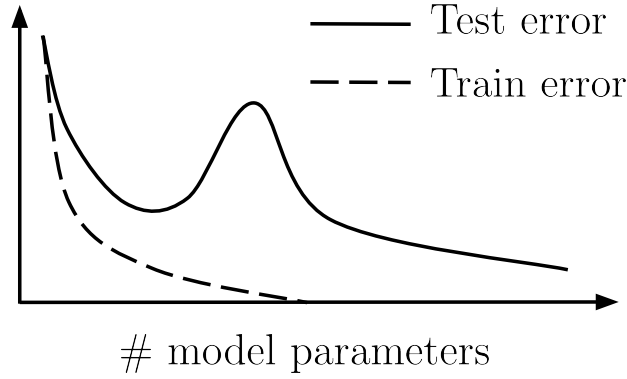


Figure 2.5: In deep neural networks, beyond a threshold in capacity, test error *decreases* with number of parameters, rather than increasing as the classical paradigm predicts.

error is reversed, and test error starts to *decrease* with model size again. This is known as double descent (fig. 2.5). A consensus for its explanation has not yet been reached, but one idea is that the number of parameters is not a good metric for model capacity in the context of generalization, that it does not necessarily scale monotonically with realized model expressivity – which is certainly true, since expressivity in neural networks can be reduced by setting weight and bias values to 0 – and thus bounds that increase with number of parameters are excessively loose and conceal the decrease observed in the right side of fig. 2.5.

Subfunction error bounds (chapter 3) suggest using an alternative measure of size for neural networks: the number of activation regions in input space, which is the same as the number of unique feasible activation patterns. A novel error bound is derived using subfunctions which scales directly with the number of

activation regions. Other notable works on network activation regions include Hanin and Rolnick, 2019, which argued that the number of realized activation regions in deep networks is far smaller than theoretical upper bounds based on the number of parameters, and Gamba et al., 2020, which described the regularity of hyperplane arrangements in convolutional neural networks. Meanwhile Belkin himself suggests the salient measure of capacity as the norm of network weights (Belkin et al., 2019). Both measures resolve a key weakness of using the number of parameters: it is independent of realized weight configuration, i.e. the data and training procedure, whereas activation patterns and weight norms are not. In addition, a mutually supportive relationship appears to exist between number of activation regions and weight norms, at least at one extreme: as weight norms tend to 0, all hyperplanes in the network collapse to the same decision for all inputs and thus expressivity collapses. L2 penalization of weight norms is an established regularization method (Bishop, 2006), which empirically supports the notion of smaller weight norms being conducive for generalization. Further, note that dropout (Srivastava et al., 2014), another established regularization method, encourages redundancy across nodes in each layer (Goodfellow et al., 2013), which one would also expect to suppress expressivity. Whereas L2 regularization and dropout are simple to introduce as priors (and in fact Goodfellow et al., 2013 employs dropout for continual learning, where non-stationarity in the training data means stronger generalization priors are called for compared to i.i.d. training), minimizing the number of activation regions explicitly is not as obvious. However, one strategy is to maximize sparsity of activations. This reduces the cardinality of the co-domain of the map between inputs and binarized activation patterns, which also reduces the number of activation regions. Sparsity has in fact been suggested as a prior for good generalization (Goyal and Bengio, 2020).

On human-level generalization. The human brain has approximately 86 billion neurons (Azevedo et al., 2009), and additionally the number of stimuli that can be represented increases exponentially with the number of the neurons (Rolls,

Treves, and Tovee, 1997), which is reminiscent of how the upper bound on the number of activation patterns in ReLU networks also increases exponentially with number of nodes, given the same network width (specifically the bound is $\mathcal{O}(\text{width}^{\text{depth} \times \text{input dims}})$, Raghu et al., 2017). From this, we conjecture that any “small solutions” embodied by the brain are likely not obtained due to restrictions in the number of parameters. Likewise in artificial networks, double descent has shown that generalization can actually improve with number of parameters, all else equal. However, other size metrics we have considered are not entirely satisfactory quantifiers of generalization ability either, or we would be much closer to human-level generalization than we are. Considering model complexity alone, no matter by which metric, seems to be missing something.

We highlight another aspect that requires explicit consideration if we are to progress towards human-level generalization: how well a model generalizes to test data should be quantified by the *difficulty* of the test data – size or complexity, and difference from training data – as well as its performance. If one defines generalization ability as merely test error, then one cannot differentiate between the abilities of models that do well on a narrowly defined test task that is semantically identical to the training data, and models that do equally well on a diverse test task that is semantically different to the training task, because realized test error is the same, and error bounds merely attempt to approximate the latter. In other words, the nature of the data itself is material for quantifying generalization ability, because some train-test combinations should be considered harder than others. The extent to which gradients from training on one correlate with gradients from training on the other (section 2.1.1) is one possible definition of easiness. Note that one interpretation of Occam’s Razor is that smaller models generalize better to more test tasks, but this is not the common interpretation of error bounds, which purport that smaller models generalize better given the same test task, or more specifically the same test data distribution.

Thus metrics for generalization ability should measure sample efficiency. Models that generalize well should be characterized by learning from a relatively small

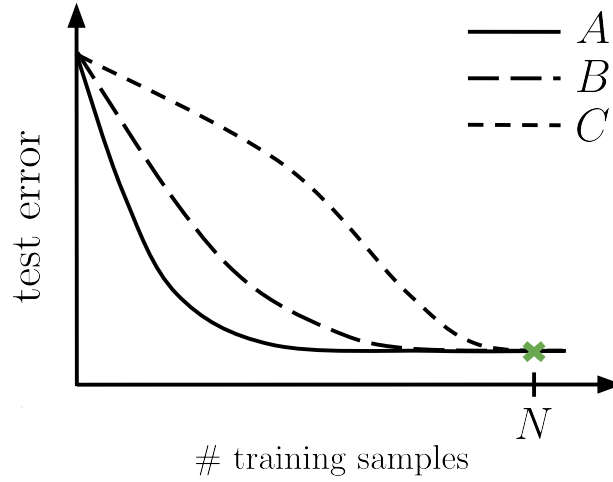


Figure 2.6: Given a training dataset of size N , priors A, B, C perform equally in the limit of training data volume, but according to the arguments presented, A produces models that generalize best.

number of samples – for instance, a single exposure – and becoming spontaneously proficient at a large number of different samples. To borrow the checkerboard analogy from Goodfellow, Bengio, and Courville, 2016, good generalization is the difference between an agent that can only accurately fill neighbouring pixels, given the specification of a few labelled pixels, versus one that can complete the full checkerboard image. Thus one can think of good generalization as the efficient discovery of meaningful structure from redundant samples. This is not captured by classical error bounds in statistical machine learning, where the only explicitly stated relationship between generalization ability and volume of training data is the former scaling with the latter, and hence the emphasis is on maximising training data to maximise generalization ability, whereas we would like to consider maximising the ratio of test data to training data.

Obviously, even a sample efficient learner benefits from more training samples in general, but can be differentiated from an inefficient learner by measuring test performance when training samples are restricted. In other words, reducing the number of training samples is one way of increasing the relative difficulty of the test set. The example in fig. 2.6 illustrates how the relative goodness of different priors can be exposed in this manner. In the limit of volume of training data, test

data from the same distribution is guaranteed to be subsumed under training data, hence typically it becomes trivial to fit test data perfectly and all priors converge to the same perfect performance, as they do on the right side of fig. 2.6. A sample efficient learner can only be distinguished from a sample inefficient learner given a test set that is sufficiently difficult relative to the training set. Thus the pursuit of models that generalize well should not only involve seeking priors that produce the best test set performance, but also seeking relatively difficult test sets.

If the inputs to a system cause the same pattern of activity to occur repeatedly, the set of active elements constituting that pattern will become increasingly strongly interassociated. That is, each element will tend to turn on every other element and (with negative weights) to turn off the elements that do not form part of the pattern. To put it another way, the pattern as a whole will become “auto-associated”. We may call a learned (auto-associated) pattern an engram.

— D. Alan Allport, 1985

2.2 Unsupervised Learning

Unsupervised learning is not a strictly defined term in machine learning (Goodfellow, Bengio, and Courville, 2016)². It is commonly considered to be the acquisition of meaningful information from data without explicit sample-level human supervision (obviously, human supervision is still present in the priors of the model). We maintain that the fundamental difference between unsupervised and supervised learning lies in the level of *collection cost* of data (section 2.1.2), which could be measured by economic cost, for example. Collecting images by crawling the internet is a cheap way of gathering data. Labelling the collected images manually substantially increases cost, and above some threshold it is no longer considered unsupervised. There is no strict definition for this threshold in monetary terms, thus methods naturally sit on a spectrum of supervision rather than in binary categories, but it is common practice to consider *any* sample-level human effort as raising the cost above the threshold for “unsupervised”.

The notion of collection cost is important, however, because the spirit of unsupervised learning relates to the general effort of data collection as a whole. One could argue that the level of participation of humans is only material in so far as it increases the cost of the training procedure, because the fundamental motivation of unsupervised learning, for both artificial and biological neural networks, is arguably to minimize the cost of learning, which corresponds to maximising data volume for any given cost budget and quality of data³. At one extreme, if the cost of learning

²Some authors prefer the term self-supervised learning; there is no widely accepted difference.

³Data quality, for example with respect to a learning objective, must be fixed because volume can be trivially maximised by reducing quality.

is too high, it would kill the biological agent or stop the machine experiment. But the benefit of cheaply acquired data is also evident before this extreme is reached: exposure to more data is beneficial for generalization in both humans and machines. In humans, for example, test error decreases with the amount of practice. Gladwell, 2008 suggested that 10,000 hours is enough to make someone an expert at any skill⁴. For artificial neural networks, generalization bounds show that upper bounds on true error shrink with increasing training sample size (Vapnik, 1998; Mohri, Rostamizadeh, and Talwalkar, 2018), which holds empirically in general despite its invalidation under certain conditions (Belkin et al., 2019). According to such bounds, being limited to supervised datasets is a significant restriction on training data size and thus imposes a limit on generalization ability. The biological parallel is that humans would learn very little if learning strictly required the direction of a teacher. Thus, acquiring larger volumes of data to improve generalization is a fundamental motivation for unsupervised learning. Whilst in the previous chapter we argued that devising learning procedures for good generalization is about maximising the efficiency of learning from a finite sample of data, unsupervised learning is about maximising the sample of data from which one learns.

The origins of unsupervised learning with artificial neural networks coincide with the origins of artificial neural networks themselves, which is unsurprising since unsupervised learning is motivated by what the brain can do, and so are artificial neural networks. Concurrent with the development of the perceptron was the development of Hebbian learning theory, which centres on a biologically-inspired unsupervised learning rule that can be summarized as “neurons that fire together, wire together”, attributed to Canadian psychologist Donald Hebb (Hebb, 1949). Hebbian learning involves adjusting weights to better reflect an optimal relationship between nodes, and in this sense, many learning methods can be considered Hebbian in nature. Perhaps the most important connection to modern neural network training regimes is the observation that backpropagation is equivalent

⁴Generalization in this case would be measured on a test distribution focused on the skill. Generalization ability is not *necessarily* measured on general tasks, though we argued in section 2.1.4 that it should be, for the purpose of advancing machines towards human-level intelligence.

to contrastive Hebbian learning (where contrast or error is propagated from the output layer and fed to a Hebbian update rule) under the assumption that the architecture is feedforward (Xie and Seung, 2003). The connection is easy to see, given that backpropagation weight updates in ReLU networks have the form:

$$\Delta w_{i,j} = -\eta a_i \delta_j \quad (2.24)$$

where $w_{i,j}$ is the weight from node i to node j , η is the learning rate, a_i is the activation of node i in the forward pass, and $\delta_j = \nabla_{x_j} L(\cdot)$ where L is the objective function and x_j is the pre-activation value of node j . Since δ_j is also computed by activation propagation in neural networks, it can be framed as node activation, making eq. (2.24) a Hebbian update rule: neurons that fire strongly in the forward pass, and neurons that fire strongly in the backwards pass, wire together.

Evidently, it is not the update rule that makes a learning procedure unsupervised, since supervised learning in neural networks overwhelmingly uses backpropagation. It is the data: Hebb’s rule was devised with the notion of *self-organizing* in mind, or the idea that the model learns from uncured (i.e. low collection cost) datasets in a manner the engineer neither explicitly controls nor predicts. It is the assumed lack of data collection cost that allows Hebbian learning to be considered as unsupervised. As we shall see in the next section, this quality of not quite knowing what behaviour the trained model will acquire, which can be a downside of minimal curation and generic, application-agnostic objective functions when one intends to use the model for a specific purpose, is a recurring theme for unsupervised learning methods.

Common learning objectives. In this section we will focus on computer vision, i.e. learning algorithms that operate on visual data, and images in particular. Computer vision is arguably the domain that has seen the greatest growth with the advent of deep learning, and has a relatively established literature on unsupervised learning with neural networks.

The literature is often split into two groups based on one facet: whether the method includes density estimation or sampling (data generation), or not. However,

these two groups are not as different as they superficially seem. All learning objectives, including unsupervised ones, revolve around the extraction of meaningful structure from data, which is to say the discovery of its latent, low-dimensional factors. For example a 256×256 pixel image of a dog on a lawn can be described for most intents and purposes by a single sentence with a few words. Seen in a unifying high-level light, learned data generation can be viewed as the discovery and assignment of structure to latent codes, learned density estimation as the discovery of structure and the discrimination of likelihood based on said structure, and other objectives – contrastive learning, in-painting, clustering – are, similarly, discriminative problems based on discovered structure in the data. Again, we assert that the “unsupervised” qualifier depends on the cost of data, not the structure of the algorithm: modelling density of the data distribution is sometimes considered the canonical unsupervised learning problem (Goodfellow, Bengio, and Courville, 2016), but is not unsupervised if one is modelling $p(x, y)$ and y has been obtained from manual labels. It just so happens that cheap data, typically less structured compared to curated data, elicits learning algorithms that often share similar high-level structure.

2.2.1 Generative Methods

In this section we provide a summary of several well-known methods that model or sample from the distribution of training data.

Many of these methods are motivated by learning a *maximum likelihood* estimate of the true data distribution, $p^*(x)$. It is assumed that this is not directly accessible, but one has access to a dataset containing N samples from it. Let $\hat{p}(x)$ be the probability of x being obtained by sampling this dataset (e.g. $\frac{1}{N}$ for its elements). We use single variable $x \in \mathcal{X}$ for generality, but it could contain any number of modalities. The goal of maximum likelihood estimation is to search a space of distributions $\mathcal{P}$ for the best $p \in \mathcal{P}$ according to:

$$\min_{p \in \mathcal{P}} D(\hat{p}(x) \parallel p(x)) = \min_{p \in \mathcal{P}} \mathbb{E}_{x \sim \hat{p}(x)} \left[\log \frac{\hat{p}(x)}{p(x)} \right] = \max_{p \in \mathcal{P}} \mathbb{E}_{x \sim \hat{p}(x)} [\log p(x)], \quad (2.25)$$

where $\log$ refers to the natural logarithm and D denotes the Kullback-Leibler divergence, with abuse of notation for x . From eq. (2.25) we see that p obtained in this way maximises the log likelihood of empirically observed data, hence the name maximum likelihood estimation.

Since this is a work on neural networks, in this section we discuss generative methods that are based on neural networks, i.e. model data density using learned neural network features, rather than methods that use fixed data features such as k-means and Gaussian mixture models (GMMs), which also optimize for a generative model that maximizes the likelihood of observed data. Note that k-means and GMMs are also generalization problems, despite not typically being framed as such; each has a loss function on samples based on distance from cluster mean, and the ideal model induces the lowest loss on both training and test data.

One method for obtaining $p(x)$ from eq. (2.25) with a neural network is normalizing flows (Rezende and Mohamed, 2015). The idea is to model $p(x)$ using learnable data transform $f_\theta(x)$:

$$p_\theta(x) = p(f_\theta(x)) |\det \nabla_x f_\theta(x)| \quad (2.26)$$

$$\log p_\theta(x) = \log p(f_\theta(x)) + \log |\det \nabla_x f_\theta(x)|, \quad (2.27)$$

where f_θ is an invertible bijector and crucially, unlike $p_\theta(x)$, the distribution for $p(f_\theta(x))$ is fixed to a known, evaluable distribution such as the standard normal $\mathcal{N}(\cdot; 0, \mathcal{I})$, hence it is written with no intrinsic parameters. This technique is known as a change of variables. The term $|\det \nabla_x f_\theta(x)|$ is to account for the change in support by applying f , i.e. the difference in ranges between x and $f(x)$, so that total probability mass of 1 is preserved. f can be composed of a stream of M bijectors $f_\theta = f_\theta^M \circ \dots \circ f_\theta^1$ (hence the name “flow”), increasing its expressivity. Let the intermediate outputs be $z_j = f_\theta^j(z_{j-1})$ for $j > 1$ and $z_1 = f_\theta^1(x)$. Then:

$$\log p_\theta(x) = \log p(z_1) + \sum_{j=2}^M \log |\det \nabla_{z_{j-1}} z_j|. \quad (2.28)$$

Learning the data density consists of optimizing θ for the objective in eq. (2.25) given this definition for $\log p_\theta(x)$. Sampling from the learned distribution $p_\theta(x)$ can be

performed by first sampling $z = f_\theta(x)$ according to its fixed distribution, typically the standard normal as mentioned, and generating x by applying the inverse of f_θ to z .

The invertible function f_θ can take many forms, including neural network layers. Recently, Behrmann et al., 2019 used residual convolutional neural network layers of the form $f_\theta^j(z_{j-1}) = z_{j-1} + g_\theta^j(z_{j-1})$ where g_θ^j is a convolution, enforcing invertibility by constraining the Lipschitz constant for all g_θ^j to be < 1 , using spectral normalization (Miyato et al., 2018). However, inverting each f_θ^j required the sum of an infinite series, which the authors truncated as an approximation, introducing bias; Chen et al., 2019 improved on this by suggesting an unbiased approximator. The appeal of using residual neural network layers is that in theory one could reuse a single network to perform, for example, accurate image classification, density estimation and data generation simultaneously. As an added bonus, probability density of a sample could be used to detect outliers at test time, for example to identify new classes for the open set problem (Zhang et al., 2020). Residual flows density was tested as an unreliability metric in chapter 3, but was found to underperform subfunctions.

Autoregressive methods, such as Oord et al., 2016a; Oord et al., 2016b, are similar to normalizing flows in that they also transform simple distributions such as the standard normal $\mathcal{N}(\cdot; 0, \mathcal{I})$ into complex distributions that can be both estimated for observed samples and sampled from for data generation. In this case, the complex density $p(x)$ of N sequential variables $x = (x_i)_{1 \leq i \leq N}$ is modelled as the product of simple conditional densities, where the parameters of the latter are a learned function of the value of all variables seen so far in the sequence. For example, assuming the conditional probability distributions are normal distributions:

$$p_\theta(x) = \prod_{1 \leq i \leq N} p_\theta(x_i; x_{1:i-1}), \quad (2.29)$$

$$\text{where } p_\theta(x_i; x_{1:i-1}) = \mathcal{N}(x_i; \mu_i, \rho_i) \quad (2.30)$$

$$\text{and } \mu_i = a_\theta^i(x_{1:i-1}) \text{ and } \rho_i = (e^{b_\theta^i(x_{1:i-1})})^2, \quad (2.31)$$

where a_θ^i and b_θ^i are neural networks. Learning involves optimizing θ via maximum likelihood estimation (eq. (2.25)) as before, and data generation involves successively

sampling x_i according to the dynamically computed mean and variance variables conditioned on all $x_{1:i-1}$. Somewhat surprisingly, enforcing the unnatural assumption that pixels should be dependent only on predecessors, which is clearly not true in general, does not seem to prevent autoregressive models from generating realistic natural images (Salimans et al., 2017).

Finally, there are a number of important data generation methods where sample density is *not* directly estimatable, chief among them generative adversarial networks (GANs, Goodfellow et al., 2014) and variational autoencoders (VAEs, Kingma and Welling, 2013). In VAEs, one maximises a lower bound on the log likelihood of the data, rather than the log likelihood of the data itself. Continuing with the theme of transforming simple distributions into complex ones, auxiliary variable z with (simple) conditional probability distribution $q(z; x)$ is introduced, yielding the following lower bound:

$$\log p(x) = \log \int \frac{q(z; x)}{q(z; x)} p(x, z) dz \geq \mathbb{E}_{z \sim q} \left[\log \frac{p(z) p(x; z)}{q(z; x)} \right]. \quad (2.32)$$

Turning function inference into an optimization problem such as maximising its lower bound is known as the variational principle. Assuming that $p_\theta(x; z)$ and $q_\phi(z; x)$ are multivariate normal distributions with vector of means μ and vector of variances ρ computed by neural network inference using parameters θ and ϕ , then the lower bound in eq. (2.32) can be maximized using the following gradient:

$$\nabla_{\theta, \phi} \mathbb{E}_{z \sim q_\phi} \left[\log \frac{p(z) p_\theta(x; z)}{q_\phi(z; x)} \right] = \nabla_{\theta, \phi} \mathbb{E}_{\epsilon \sim \mathcal{N}(\cdot; 0, I)} \left[\log \frac{p(\alpha) p_\theta(x; \alpha)}{q_\phi(\alpha; x)} \right], \quad (2.33)$$

$$\text{where } \alpha = \mu_\phi(x) + \epsilon \cdot \rho_\phi(x), \quad (2.34)$$

which is known as the reparameterization trick. The problem this solves is that the gradient operator in the first term of eq. (2.33) cannot be pushed through the expectation, as the expectation depends on its variables; by changing the sampling to be over ϵ , this is resolved. For an intuitive interpretation of the objective, note that the bound can be rewritten as:

$$\mathbb{E}_{z \sim q_\phi} \left[\log \frac{p(z) p_\theta(x; z)}{q_\phi(z; x)} \right] = \mathbb{E}_{z \sim q_\phi} [\log p_\theta(x; z)] - D(q_\phi(z; x) \parallel p(z)), \quad (2.35)$$

with abuse of notation for inputs of Kullback-Leibler divergence D . Thus maximising it involves simultaneously maximising the conditional likelihood of data, whilst minimizing the difference between the conditional likelihood of codes and the fixed prior on codes. These objectives are not complimentary with respect to expressivity of the codes; the former is a data reconstruction objective that motivates increasing expressivity whilst the latter constitutes regularization that decreases expressivity. This means training with the VAE objective naturally entails a balance between fitting and not overfitting training data, which is conducive to generalization (section 2.1). Further, note that q_ϕ is a distribution that maps input x to latent codes z , and p_θ is a distribution that maps latent codes z to inputs x , and thus conceptually they can be stacked into a mapping going from the input space to latent code back to the input space, which explains the term “autoencoder”. In the next section we will encounter another type of autoencoder that does not attempt to model a bound on the data density distribution.

GANs (Goodfellow et al., 2014) are generative models that also do not directly model the density of data samples, and thus like VAEs one cannot easily use them to compute density. However, they have demonstrated remarkable success in data generation, despite the common perception that they are difficult to train. Again, the high level idea is to learn a transformation from a simple distribution to the complex true data distribution. Specifically, one trains a generator $G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ that maps the space of latent codes $\mathcal{Z}$ (which follow a simple fixed distribution, such as uniform) to input data space $\mathcal{X}$, and discriminator $D_\phi : \mathcal{X} \rightarrow [0, 1]$ that maps input data space $\mathcal{X}$ to the likelihood that the input data is real as opposed to generated. Let the distribution of codes be p_z and the distribution of real data samples be p_x . Then the training objective is to find G_θ and D_ϕ that satisfy:

$$\min_{G_\theta} \max_{D_\phi} \mathbb{E}_{x \sim p_x} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D_\phi(G_\theta(z)))]. \quad (2.36)$$

This is an example of the minimax strategy, used in game theory (Cho and Kreps, 1987) as well as in other domains of artificial intelligence (notably in playing chess, Shannon, 1993), where competing agents assume that the other agent always opposes

their interest. However, one could argue that in the case of GANs, training is more of a collaboration than a true competition between G_θ and D_ϕ ; improvements in the discriminator’s ability induce improvements in the generator’s ability, and vice versa. For instance, if the discriminator happens to be very suboptimal and outputs $D_\phi(G_\theta(z)) = 1$ for all codes z , then a global optimum for the generator has been achieved and there is no motivation for its parameters to change, even if its outputs do not resemble samples from the true data distribution. Likewise, if the generator is very suboptimal and only produces images that the discriminator easily identifies as fake, i.e. $D_\phi(G_\theta(z)) = 0$ for all codes z , a global optimum for the second half of eq. (2.36) has been achieved and the learning signal from eq. (2.36) is weaker for the discriminator (in terms of maximal gradient magnitude, for example), all else equal.

GANs have been extended extensively and have been used for style transfer (Karras, Laine, and Aila, 2019), with cycle consistency constraints (Zhu et al., 2017), image editing (Zhu et al., 2016), and also representation learning (Radford, Metz, and Chintala, 2015), which is the focus of the next section. They have also been studied in the context of network saliency (Bau et al., 2018), which can be considered one unsupervised optimization problem stacked on top of another: the network is first trained unsupervised, then one infers the salient features for any given inference also without supervision. In chapter 5 we conduct a study of saliency methods and use saliency maps as an unsupervised method for solving the pointing game problem.

There are a large number of variations of the methods in this section, for which we do not go into detail because this is not a work specifically about distribution matching. However, we would like to draw the reader’s attention again to the importance of generalization, which is particularly obvious in the context of data generation. In all methods, one only has access to a finite number of samples from the true data distribution: the gaps in input space between empirically observed inputs are not explicitly accounted for, so the behaviour of the model in these empty areas is entirely dictated by priors. The link between implicit constraints from priors and the behaviour of the trained model is still not well understood

and in practice priors are usually selected for empirically, with a trial-and-error approach. This is true across much of deep learning, but is especially important in the generation of high-dimensional data such as images, where the space of possible optima is typically extremely large if one were merely to consider the empirical dataset. This is because the objectives of such models require them to *preserve expressivity*, which one can think of as granularity (measured in neural networks for example by the number of activation patterns or regions, chapter 3). Unlike objectives where the expressivity of the output space is much lower than the input space, such as classification or clustering, a model that generates natural images is severely under-constrained by the training data, because the space of natural images is so large that training data necessarily occupies only a small fraction of the space one is interested in mapping to. Thus there exists numerous global optima for model parameters that appear equally as good if one were just to consider the training data, but these optima actually vary widely in quality when assessed according to other samples, due to high expressivity of the model. This relates to the fundamental principle of generalization discussed in section 2.1 that smaller or less expressive models generalize better, all else equal. In short, a likely reason for the difficulty of training image generation models that generalize well is that training data constrains a relatively small proportion of the network’s behaviour because the latter is highly expressive, increasing the importance of choosing the right priors.

2.2.2 Discriminative Methods

First we discuss the difference between generative methods and discriminative methods. Assume that there are two variables of interest, x and y . Whereas generative methods would learn a model for $p(x, y)$ (whether the probability can be explicitly evaluated or not), discriminative methods learn a model for $p(y; x)$ (Ng and Jordan, 2002). Thus discriminative methods solve conditional prediction problems. When the model is not stochastic, for example if one uses a deterministic neural network, there is only one possible output y per input x and thus training the network becomes the problem of learning a one-to-one mapping.

Generative modelling is always an option for any dataset, because associating an instance of y with each x is not essential as one can always simply model $p(x)$, whereas discriminative modelling requires the separation of variables. For this reason it is traditionally associated with supervised learning, because obtaining a dataset consisting of paired samples, for example, often entails a human constructing the pairing on a per-sample basis. However, this is not necessarily the case; y does not need to be a manually annotated label. It could be for example a code representing the input (He et al., 2020; Chen et al., 2020; Caron et al., 2020), a cluster identity, which is just a form of sparse discrete code (Asano, Rupprecht, and Vedaldi, 2019; Caron et al., 2018 and our work in chapter 4), the nature of the relationship with a transformed or spatio-temporally proximal sample (Doersch, Gupta, and Efros, 2015; Isola et al., 2015), or a transformation of the input (such as denoising autoencoders (Vincent et al., 2008), or colourisation (Zhang, Isola, and Efros, 2016)). These are all examples of unsupervised learning as the data incurs low collection cost (section 2.1.2).

For the task of predicting codes, the objective cannot be maximum likelihood estimation with $D(\hat{p}(y; x) \parallel p_\theta(y; x))$ for example, since y is not a fixed absolute target. Instead, the engineer typically specifies an abstract constraint on $p_\theta(y; x)$ and optimizes θ such that this constraint is achieved. The constraint is abstract by necessity, since in unsupervised learning we assume no effort per sample is taken. A typical consequence of this abstractness is that multiple global optima for θ exist, and thus priors (including said constraints) should be selected in order to steer the optimization towards optima that generalize well. For example, some methods both constrain expressivity using an output bottleneck (recall that smaller models aid generalization), whilst supporting expressivity to prevent it collapsing (Hadsell, Chopra, and LeCun, 2006 and chapter 4).

Many modern unsupervised discriminative methods focus on *representation learning* (Bengio, Courville, and Vincent, 2013), which is the optimization of data features that generalize well to later (or “downstream”) tasks⁵. This is

⁵Obviously, we would expect any network successfully optimized for an objective (including the generative objectives in section 2.2.1) to have learned meaningful features for that objective.

directly connected to the idea of incidental gradient following we discussed in section 2.1.1; a sign of a method that produces representations that generalize to other tasks is incidentally following the gradients that would be obtained by directly training on those tasks. Learned representations often require further learning, for example linear probing (learning a new linear function on top of the representation function), when being adapted to downstream tasks, but if the output space of the representation is shared with downstream tasks, appending new parameters or even undertaking further training is not necessarily essential. This is an important bonus that motivates training models that output text or cluster assignments, which are both human-interpretable co-domains, that does not apply to models that learn to map to continuous codes.

However, the community has recently devoted a large amount of effort to investigating representation learning via predicting codes. An important class of such methods is *contrastive learning*. Let $f : \mathcal{X} \rightarrow \mathcal{R}^C$ be a deterministic neural network. Contrastive learning imposes the constraint that that dissimilarity in $f(x)$ and $f(x')$ for inputs x, x' should shrink as x tends to x' according to some distance metric, and grow as distance between x and x' grows, which is implemented by optimizing an objective that pulls predictions for similar inputs together whilst pushing predictions for dissimilar inputs apart (Hadsell, Chopra, and LeCun, 2006). A popular technique for contrastive learning is generating similar pairs using image transformations that preserve their semantic content (chapter 4 and He et al., 2020; Chen et al., 2020; Caron et al., 2020; Zbontar et al., 2021), which is an idea that dates back to Becker and Hinton, 1992. Our work in chapter 4 achieves the simultaneous pulling and pushing using the conditional entropy and entropy components of mutual information.

Thus for non-triviality, representation learning often refers specifically to methods that produce representations intended for downstream tasks for which they were not explicitly trained, for example a different dataset or objective.

Should one use unsupervised pre-training? If representation learning yields an optimal representation for a downstream task, then using it to initialize the downstream network decreases distance to the downstream network’s optimum parameters compared to random initialization (by definition of Euclidian norm on parameters), allowing in theory for the optimum downstream network to be reached in fewer training steps than from random initialization, all else equal. However this is only an advantage if directly training on the downstream task would not find the same optimum in fewer total steps (where the total includes any pre-training), and the incidental gradient following paradigm (section 2.1.1) suggests that this is unlikely as one would expect task mismatch to decrease gradient correlation. Furthermore, no specialization is generally better than poor specialization, because for any parameter initialization and optimum in parameter space, staying at the initialization is preferable to moving in the opposite direction, with respect to distance from the optimum. This issue of mismatch was anticipated by Vapnik, 1998, who believed that “one should solve the [classification] problem directly and never solve a more general problem as an intermediate step”. As argued from the beginning of this chapter, the main argument for unsupervised learning is primarily that it enables access to larger amounts of training data, which benefits generalization; if the downstream task does not suffer from lack of training data, then this advantage is suppressed.

The issue of mismatch with downstream tasks means it is unsurprising that for a long time unsupervised pre-training for supervised classification tasks – i.e. pre-training on a task that is not supervised classification – underperformed the standard practice of supervised pre-training, which is typically with ImageNet (Deng et al., 2009). Recently, some unsupervised learning methods claim to have broken this ceiling in certain settings (He et al., 2020), but the degree of engineering that goes into modern training pipelines makes performance gains difficult to attribute.

According to a recent study on unsupervised contrastive learning methods (Cole et al., 2021), the best dataset for contrastive pre-training is the downstream dataset, followed by the most semantically similar dataset, which corroborates our intuition

on incidental gradient matching. Further, it found that performance with contrastive pre-training lags far behind purely supervised training when the downstream task is fine-grained, which is also not particularly surprising: methods that predict codes employ information loss that is not explicitly optimized for the evaluated task (unlike supervised classification) but depends on priors from the engineer, making models vulnerable to low expressivity. Within contrastive pre-training methods, no single method appears to be universally superior (Kotar et al., 2021). Different methods perform better in different settings, which when combined with the large amount of investigative effort expended, suggests that performance gains from using contrastive learning itself have been saturated, with residual gains coming from changes in other priors such as choice of image transformations used.

Note on unsupervised learning in humans. While using unsupervised learning to improve performance on specific downstream tasks is an inherently useful and important problem, the original inspiration for the domain was to find learning algorithms that could cope with the same standard of learning environment as the human brain. The current focus on representation learning, and even the strict separation of “unsupervised” and “supervised” regimes, has arguably somewhat lost sight of this. Humans do learn in environments devoid of explicit manual direction from other humans (notably in infancy, Smith and Yu, 2008) but far from exclusively. On the contrary, adult intelligence generally depends on explicit supervision in education systems. Just as important as the ability to learn in a supervised or unsupervised fashion is being able to do *both* seamlessly; the brain switches from unsupervised and supervised learning without fundamental changes in learning procedure or objective, and furthermore this is not even necessarily a conundrum for artificial learning algorithms, because as argued earlier in this chapter, the level of supervision is an economic concern that specifically refers to the cost of training data, rather than some fundamental difference in the nature of training data, learning procedure or training objective. The brain does not learn in a pre-training vacuum before training for “material” outputs; all outputs of the brain

at all times are material, because all have real consequences. Moreover, a significant proportion of unsupervised learning in the brain occurs on data that is self-generated or recalled whilst data presented by the environment is ignored (chapter 6), an aspect which is disregarded by the literature. In short, we are still a long way from human-style unsupervised learning, and ironically the current focus on *specifically* unsupervised training procedures is arguably not conducive to achieving it.

If after its original training is finished a network is exposed to the learning of new information, then the originally learned information will typically be greatly disrupted or lost.

— Anthony Robins, 1995

2.3 Continual Learning

Continual learning is the problem of learning from a sequence of non-stationary distributions whilst generalizing to all seen distributions. This sequentiality breaks the i.i.d. (“independently sampled and identically distributed”) assumption on training samples that is adopted by many popular experimental settings and theoretical works in machine learning, and this is a problem for two reasons.

The first is that neural network learning methods that work reasonably well in i.i.d. settings – in particular, stochastic gradient descent on the current training batch – tend to fail in non-stationary settings. The severity of failure is so pronounced that it is literally referred to as “catastrophic forgetting” or “catastrophic interference”⁶ (French, 1991; Goodfellow et al., 2013). Empirically, this is observed as very low performance on test sets containing a mix of samples from all training distributions. Specifically, the network overfits to its current training distribution relative to other seen training distributions, so performance on test samples from the current distribution increases whilst those from other distributions decrease, imposing a low ceiling on total test performance since the model is only proficient at one distribution at any time. In chapter 6, for example, we see that accuracy on previously seen classes drops to 0 when classes are presented sequentially during training for image classification.

The second is that non-stationarity is the natural condition of real world data. Climate changes, landscape is developed or destroyed, objects move, and internet content is created. Thus the data distribution changes with time; samples within each distribution are co-dependent, and the sequential distributions themselves are also co-dependent. Changes in distribution are not generally sharp either, but phase in and out in a continuous manner, making them difficult to characterise

⁶There is no widely accepted difference: catastrophic interference is catastrophic forgetting.

precisely. Furthermore, compounding this objective non-stationarity are subjective sources of non-stationarity local to the sensor or agent: the agent moves, and the agent’s representation changes, both of which materially change learning conditions even if the environment can be considered stationary from an objective viewpoint (though representation change also occurs during stationary training with SGD, which arguably makes this a less important consideration). Finally, it is often unacknowledged that the agent has constrained resources and typically only makes small observations that capture a fraction of all data available, and smaller size means batches of samples are more biased, effectively increasing non-stationarity of observed data *even if the environment is stationary from an objective viewpoint*⁷.

Non-stationarity is a complex issue and our understanding of the topic is still immature. Some believe that the difficulty of characterising it with conventional statistics suggests the latter is fundamentally lacking, which is one of the motivations behind work on causality (Pearl, 2009). In any case, it is clear that the i.i.d. assumption under which many popular machine learning methods have been developed is highly unrepresentative of real world data when it is learned from online. This introduces a disconnect that must be bridged with effort – most commonly, storing observed samples and randomly resampling to mimic the i.i.d. setting (Mnih et al., 2013) – which is unsatisfactory, likely suboptimal (due to underoptimized selection), and not possible in all applications. The non-stationary nature of real-world data means that continual learning is not just a theoretical problem, but a highly practical and widely relevant one.

2.3.1 Assessing Forgetting

There are two metrics commonly used to assess the performance of continual learning methods (Chaudhry et al., 2019). Let $a_{i,j}$ denote accuracy on the test set of task j

⁷There is currently no widely accepted quantitative metric for the level of non-stationarity, but intuitively one can think of it as increasing with dissimilarity between consecutively presented distributions and decreasing with distribution change speed. This is trivially minimized if there is only a single data distribution. However one generally only has access to the drawn samples rather than the true distribution, and biased samples from a single distribution *appear* to be drawn from different distributions.

after training on task i . The average accuracy after task T is:

$$A_T = \frac{1}{T} \sum_{j=1}^T a_{T,j}. \quad (2.37)$$

Average forgetting after task T is:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} f_j^T \quad (2.38)$$

$$\text{where } f_j^i = \max_{l \in \{1, \dots, i-1\}} a_{l,j} - a_{i,j}. \quad (2.39)$$

Note that the objective of continual learning is not *eliminating* forgetting, for two reasons. The first is that this has a trivial solution according to eq. (2.38): simply set the learning rate to 0 and never change the network’s parameters. Thus the objective is still to obtain the model with best test set performance. Forgetting is an auxiliary metric that is subsumed by this, because high forgetting means low performance on seen tasks, all else equal.

Secondly, if one is using human learning as the standard for continual learning, then note the brain also suffers from forgetting. For example, the loss of one’s native language due to prioritisation of another language is known as *attrition* and is well-documented in humans who migrate (Schmid, 2004). From introspection, it is clear that humans forget previously held knowledge on a regular basis. However, forgetting in humans is generally not as catastrophic or severe as in neural networks. In fact, learning in the biological brain appears to *depend* on non-stationarity to a degree, which is unsurprising since it evolved in the context of a non-stationary environment. Wood, 2016 showed with experiments on chicks that non-stationary environments impair the development of object recognition abilities. Thus despite modern artificial neural networks being inspired by biological neurons, it appears that the latter are naturally suited to learning on non-stationarity data, whilst the opposite is true for the former.

One notable theory for how the brain avoids catastrophic forgetting is that it contains two memory systems: a slow learning, parametric and memory-efficient system in the neocortex, which is trained via replaying memories from a fast learning, non-parametric and memory-inefficient system in the hippocampus (Hassabis et al.,

2017). The idea is that the latter does not catastrophically forget because non-distributedness prevents interference (French, 1991), while the former does not catastrophically forget because learning is slow and replay mixes previous memories. Both of these ideas have been used by methods for alleviating catastrophic forgetting in artificial neural networks.

2.3.2 Distributed and Non-Distributed Memory

Neural networks generally implement distributed representations (Hinton, 1986), meaning inputs are projected onto overlapping patterns. This means the store of knowledge (or memory) in the projection function is also distributed, i.e. implicated parameters overlap across samples. In chapter 3 we saw one way of formalizing this overlap, using subfunctions. Distributedness allows for high expressivity; n bits can express up to 2^n patterns if overlap is allowed and only n if not. Overlap also means representations are shared between and thus optimized across training distributions, minimising redundancy from duplication, compared to if an independent representation was optimized per training data distribution. Inference in distributed representations naturally generalises to patterns not present in training using an implicitly learned distance metric, whereas inference in completely non-distributed representations such as non-parameteric memory (e.g. lookup tables) does not generalize to patterns not seen in training (French, 1991), unless an explicit external distance metric is provided (an example of this is the kernel in Gaussian Processes (Bishop, 2006)).

However, lack of feature sharing across samples, which inhibits interference, has advantages. Attempting to increase the independence or modularity of node representations to prevent forgetting is an idea that dates back to the node activation sharpening algorithm of French, 1991. More recently, this idea has been seen in dynamic architectures, for example Rusu et al., 2016 which appends network modules with new tasks, decreasing distributedness of the neuronal representation. Obviously, if the representations of two nodes do not share any updateable parameters, either can be updated atomically without interference of the other. But as we have

argued, this is a double edged sword, as non-distributedness sacrifices on memory efficiency, transfer and generalisation (French, 1991; Bengio, Courville, and Vincent, 2013). Note that transfer being inhibited is particularly problematic; inhibiting any interference with a subset of previously learned parameters means both positive *and* negative interference is prevented. Furthermore, rather than allowing the degree of distributedness to be naturally determined from optimization for an objective, non-distributedness that is enforced a priori increases dependency on the engineer’s skill. For these reasons, we maintain that enforcing non-distributedness is not an ideal solution to inhibiting catastrophic forgetting, going against some of the author’s early work in chapter 7.

2.3.3 Implicit Attention Mechanisms

Optimization refers to the part of the learning procedure that concerns how to descend objective functions, rather than the definition of the objectives themselves. Within continual learning, some works have asserted the opinion that the optimization algorithm should be adapted to inhibit catastrophic forgetting. These methods generally explicitly estimate a parameter’s importance for seen tasks and correspondingly anchor its value from changing (Zenke, Poole, and Ganguli, 2017; Aljundi et al., 2018), for example by penalizing changes according to Fisher information, i.e. the sensitivity of behaviour to parameter changes (Kirkpatrick et al., 2017).

One could argue that SGD is inherently reckless, in that the only obvious constraint for a training step to not forget previous knowledge is that it can only move a small distance from the previous instance of parameters, which contain the previous knowledge. This illustrates why the speed of distribution change is important for characterizing level of non-stationarity, because the greater the number of learning steps, the higher the upper bound on distance moved. However, we argue that this is *not* the only in-built constraint. Backpropagation contains attention mechanisms that already offer some protection against forgetting, and the

degree of synergy with the aforementioned methods is not obvious partly because this is not explicitly acknowledged by authors.

Implicit attention mechanisms. Contrary to what some might surmise from reading popular recent work (Devlin et al., 2018; Graves, Wayne, and Danihelka, 2014; Pritzel et al., 2017), attention in neural networks is not a recently invented phenomenon. In this section we list some of the simple implicit attention mechanisms that exist in basic ReLU-activated MLPs. Some are rather obvious, but they are important and rarely acknowledged, which undermines understanding the contributions of additional explicit attention mechanisms.

Let x_i be the vector of pre-activation values in layer i , a_i be the corresponding post-activation values (also simply called “activations”), $w_{i-1 \rightarrow i}$ be the incoming weights and b_{i-1} be the bias term. Inference in layer i consists of:

$$x_i = w_{i-1 \rightarrow i} a_{i-1} + b_{i-1}, \quad (2.40)$$

$$a_i = \text{ReLU}(x_i). \quad (2.41)$$

For simplicity we will ignore the bias term; note that it can be incorporated into the weights by using unit input activation values. Thus we consider layers of the form:

$$x_i = w_{i-1 \rightarrow i} a_{i-1}, \quad (2.42)$$

$$a_i = \text{ReLU}(x_i). \quad (2.43)$$

First, note that negative weights serve a comparable purpose to inhibitory neurons (Jonas and Buzsaki, 2007) in biological brains: they allow the activations of some nodes to turn off the activations of other nodes, in the next layer (fig. 2.7).

Second, the activation of a node behaves as a pseudo-probability for the likelihood of the current input pattern matching an importance weighted sum of the node’s previously seen training patterns. This is because the weights themselves are an importance weighted sum of previously seen training patterns. Let $w_{i-1 \rightarrow i,j}$ be the input weights of node j in layer i (fig. 2.8) and let $w_{i-1 \rightarrow i,j}^0$ be its value at

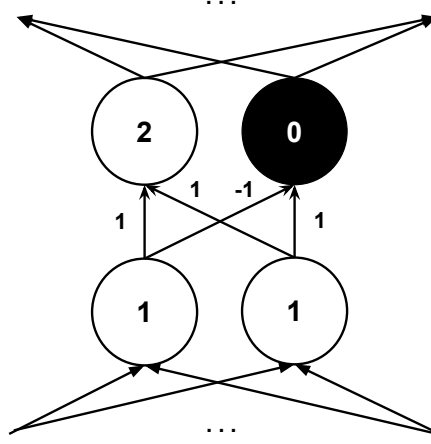


Figure 2.7: Nodes are labelled with activations. Negative weight values implement activation inhibition.

initialization. Assume that the network is trained with SGD and each batch has size 1, for notational simplicity. Weights after T updates are given by:

$$w_{i-1 \rightarrow i,j}^T = w_{i-1 \rightarrow i,j}^0 - \eta \sum_{t=1}^{T-1} a_{i-1}^t \nabla_{x_{i,j}^t} L^t, \quad (2.44)$$

where $\nabla_{x_{i,j}^t} L^t$ is the scalar first derivative of the pre-activation with respect to objective function L in timestep t . Set $a_{i-1}^0 = 1$. Then eq. (2.44) can be simplified as:

$$w_{i-1 \rightarrow i,j}^T = \sum_{t=0}^{T-1} c^t a_{i-1}^t, \quad (2.45)$$

for scalars c^t . This means weight values are a weighted sum of seen training patterns, where weight c^t corresponds to importance of the node for each timestep's error, i.e. the degree to which the node's activation contributed to the error, to a first

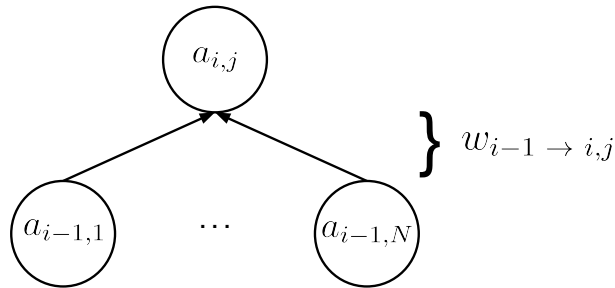


Figure 2.8: Single node activation $a_{i,j}$ as a function of input activation pattern a_{i-1} .

approximation. Furthermore note that in view of eq. (2.45), weight initialization (which is typically drawn from a uniform distribution, He et al., 2015) can simply be viewed as a noisy preliminary training pattern.

Equation (2.45) means that the instance of input pattern a_{i-1} that activates $a_{i,j}$ the most, all else equal, is precisely this importance weighted sum of seen patterns. Given any fixed norm k for vector a_{i-1}^T , then:

$$\operatorname{argmax}_{a_{i-1}^T} a_{i,j}^T \quad \text{s.t. } \|a_{i-1}^T\| = k, \quad (2.46)$$

$$= \operatorname{argmax}_{a_{i-1}^T} w_{i-1 \rightarrow i,j}^T a_{i-1}^T \quad \text{s.t. } \|a_{i-1}^T\| = k, \quad (2.47)$$

$$= \frac{k}{\|\sum_{t=0}^{T-1} c^t a_{i-1}^t\|} \sum_{t=0}^{T-1} c^t a_{i-1}^t, \quad (2.48)$$

because the dot product of any two vectors with fixed Euclidian norm is maximised when their directions are identical. More generally, activation correlates with input pattern-training pattern correlation, even if the optimum in eq. (2.48) is not reached, which follows trivially from the formula for the forward pass (eq. (2.42)):

$$a_{i,j}^T \propto \left(\sum_{t=0}^{T-1} c^t a_{i-1}^t \right) a_{i-1}^T, \quad \text{assuming } a_{i,j}^T > 0. \quad (2.49)$$

In other words, activation patterns in a MLP can be interpreted as attention maps that measure the goodness-of-fit between each node's current input activation pattern and its historical training samples, all else equal i.e. given equal input pattern norms⁸. This variation in goodness-of-fit induces natural variations in learning speed across timesteps: due to the weight update rule, the weights of a node change more, all else equal, when its input nodes activate strongly.

Finally, note that non-linearities act as gates that restrict learning so that relevant representations are refined and irrelevant representations remain untouched, in any given timestep. A node's local representation, which is its immediate incoming weights, is protected from adaptation when it does not activate, which according to the goodness-of-fit argument is when it is deemed irrelevant for the current input (fig. 2.9).

⁸In practice, the regularization of pattern norms is commonly explicitly implemented by measures such as input whitening or batch normalization.

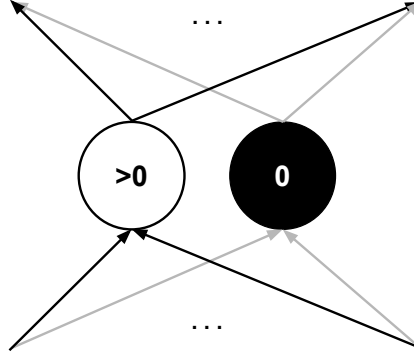


Figure 2.9: If a node does not activate, the update rule means that outgoing and incoming weights are prevented from changing. The latter is due to the first order derivative of the loss with respect to the node’s pre-activation value being 0. The weights prevented from changing are shown in grey.

However, it is evident from empirical experiments that these implicit attention mechanisms are insufficient to inhibit catastrophic forgetting to satisfactory levels, at least in common experimental settings (chapter 6; Chaudhry et al., 2018; Chaudhry et al., 2019; Aljundi et al., 2019). Assuming that these experimental settings are not unreasonable or unrepresentative of practical continual learning applications, then additional measures clearly need to be taken. In the next section, we discuss the most common approach undertaken in the current literature.

2.3.4 Replay

Replay is the idea that training on the current real environment batch should be supplemented by considering past samples. It is a simple idea with many possible interpretations. The basic method is storing observed training samples in a buffer and randomly sampling it to produce training batches that contain a mix of samples from all seen data distributions (Chaudhry et al., 2019; Rolnick et al., 2019; Mnih et al., 2013). The mixed batches can replace each real environmental batch entirely, or be trained with in conjunction.

One can select samples from the buffer uniformly, or use more sophisticated importance sampling strategies. Note there is a connection to sample efficiency,

which we discussed in section 2.1.4 as being important for characterizing generalization. Since effective training samples can deviate from what the environment provides, the model is able to prioritise areas in input space (for example, previously observed samples) that may be more challenging than the current environmental batch. Methods such as those in chapter 6 and Aljundi et al., 2019 implement this idea by selecting replay samples based on high loss values.

A data generator can be trained to store seen samples (Shin et al., 2017; Kemker and Kanan, 2017), thus using distributed neural network memory rather than a non-distributed buffer, which in theory increases capacity, as discussed. Since a generator that uses distributed memory is itself vulnerable to forgetting, it is trained with a data distribution matching objective that combines newly observed samples with samples from the generator’s previous distribution in a recursive manner. Such approaches are more biologically plausible than using buffers; even though hippocampal encodings are *relatively* independent (Hassabis et al., 2017; Kumaran, Hassabis, and McClelland, 2016), they are still represented by distributed neurons, and the brain is not known to contain strictly non-distributed representations such as tables. Methods that share the parameters of the generative model with the main inference model (chapter 6, Ven and Tolias, 2018) further increase biological plausibility, since the brain also appears to share memory for inference and replay. Namely the same neurons activated during experiences are re-activated during their replay (Gelbard-Sagiv et al., 2008; Carr, Jadhav, and Frank, 2011).

Instead of storing previous samples, one can store instances of previous models, which as discussed contain a compressed store of previous samples (eq. (2.45)). In order to extract training samples from previous models, one can use old outputs as alternative targets for inputs in the current batch, which is known as distillation (Li and Hoiem, 2017; Hinton, Vinyals, and Dean, 2015), or use backpropagation to generate input samples that maximise divergence between old and new models, to determine and reinforce what the model has forgotten (chapter 6, Yin et al., 2019).

Fundamentally, continual learning concerns generalization: the goal is to learn a model that generalizes across all data distributions when they are presented

sequentially during training. Note that it is a special case of the imbalanced dataset generalization problem: continual learning can be viewed as an extreme variant of imbalanced sampling where environmental batches contain zero samples from other data distributions. Furthermore, note that training a representation for one task that generalizes well to other tasks, which was discussed in the context of unsupervised learning (section 2.2), is also the goal of continual learning.

One well-known continual learning approach, GEM (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018), connects directly to the incidental gradient matching paradigm for generalization discussed in section 2.1.1. Assume the current environment batch is S^t and one has access to a buffer of previous samples $\tilde{S}$. Let the loss function on previous samples be $L(\theta^t; \tilde{S})$ for parameters θ^t at the beginning of timestep t . The loss function on previous samples at the beginning of timestep $t + 1$ can be expressed using the Taylor expansion as:

$$L(\theta^{t+1}; \tilde{S}) = L(\theta^t; \tilde{S}) + (\theta^{t+1} - \theta^t)^\top \nabla_{\theta^t} L(\theta^t; \tilde{S}) + \mathcal{O}(\|\theta^{t+1} - \theta^t\|^2), \quad (2.50)$$

thus its increase as a result of training on S^t , or negative interference, is given by:

$$L(\theta^{t+1}; \tilde{S}) - L(\theta^t; \tilde{S}) \quad (2.51)$$

$$= (\theta^{t+1} - \theta^t)^\top \nabla_{\theta^t} L(\theta^t; \tilde{S}) + \mathcal{O}(\|\theta^{t+1} - \theta^t\|^2) \quad (2.52)$$

$$= -\eta(\nabla_{\theta^t} L(\theta^t; S^t))^\top \nabla_{\theta^t} L(\theta^t; \tilde{S}) + \mathcal{O}(\|\eta \nabla_{\theta^t} L(\theta^t; S^t)\|^2). \quad (2.53)$$

In other words, the loss on buffered samples increases if its gradients are decorrelated with the gradient from training on the current environment batch, and this is a special case of incidental gradient following (section 2.1.1). To prevent this loss increase, one can transform the training gradient $g = \nabla_{\theta^t} L(\theta^t; S^t)$ into the closest alternative, $\tilde{g}^*$, such that it is correlated with the gradient from buffered samples $\hat{g} = \nabla_{\theta^t} L(\theta^t; \tilde{S})$:

$$\tilde{g}^* = \operatorname{argmin}_{\tilde{g}} \frac{1}{2} \|g - \tilde{g}\|^2 \quad \text{s.t.} \quad \tilde{g}^\top \hat{g} \geq 0, \quad (2.54)$$

which can be solved analytically to show:

$$\tilde{g}^* = g - \frac{g^\top \hat{g}}{\hat{g}^\top \hat{g}} \hat{g}. \quad (2.55)$$

Training the model consists of applying $\tilde{g}^*$ instead of g , i.e. $\theta^{t+1} = \theta^t - \eta \tilde{g}^*$. This constitutes replay as previously seen samples are used to constrain learning on the current environmental batch, even though implementing the constraint takes the form of gradient transformation rather than simply training on the seen samples. Its conceptual complexity relative to the latter, and also the fact that eq. (2.54) is a hard constraint that forbids loss increases on stored samples $\tilde{S}$ and thus de-prioritises learning new samples, are two reasons why it has not been adopted as widely as basic replay.

3

Predicting Unreliable Predictions by Shattering a Neural Network

This paper was released on arXiv in June 2021 and is under review.

Predicting Unreliable Predictions by Shattering a Neural Network

Xu Ji
Oxford University

Razvan Pascanu
DeepMind

Devon Hjelm
Mila

Andrea Vedaldi
Oxford University

Balaji Lakshminarayanan
Google Brain

Yoshua Bengio
Mila

Abstract

Piecewise linear neural networks can be split into subfunctions, each with its own activation pattern, domain, and empirical error. Empirical error for the full network can be written as an expectation over empirical error of subfunctions. Constructing a generalization bound on subfunction empirical error indicates that the more densely a subfunction is surrounded by training samples in representation space, the more reliable its predictions are. Further, it suggests that models with fewer activation regions generalize better, and models that abstract knowledge to a greater degree generalize better, all else equal. We propose not only a theoretical framework to reason about subfunction error bounds but also a pragmatic way of approximately evaluating it, which we apply to predicting which samples the network will not successfully generalize to. We test our method on detection of misclassification and out-of-distribution samples, finding that it performs competitively in both cases. In short, some network activation patterns are associated with higher reliability than others, and these can be identified using subfunction error bounds.

1 Introduction

ReLU-based neural networks represent piecewise linear functions, a property previously exploited to characterize the expressivity of these models (Montúfar et al., 2014; Raghu et al., 2017; Hanin and Rolnick, 2019). In this work, however, we exploit it to characterize generalization, though as we will see, the two properties are linked.

Specifically, ReLU networks provide a straightforward decomposition of the neural network into linear subfunctions corresponding to different binary activation patterns or activation regions, which are the non-overlapping polytopes in input space within which the same activation units are active or positive-valued. These linear subfunctions rely on different subsets of network parameters, where the overlap in parameters correlates with how related or tied the subfunctions are to each other. Based on this, we derive an empirical error metric for subfunctions that captures dependencies between subfunctions and yields an error bound that is non-trivial for subfunctions without training samples, which is crucial since test samples overwhelmingly belong to such subfunctions in deep networks. Intuitively, the bound scales inversely with how densely the subfunction’s activation region is surrounded by training samples in subfunction kernel space, and directly with how poorly the network performs on those samples. Thus the better the network’s performance on close samples and the more there are of them, the more reliable the subfunction can be considered to be, as one would expect. The idea that reliability should scale with density was proposed as early as Bishop (1994). Our contribution is using generalization bounds that incorporate density to measure reliability, where density is computed using subfunctions. In other words, we exploit the implicit modularity

and representation reuse present within neural networks to expose which behaviours are strongly or weakly supported by training samples.

Predicting which test samples lead to network predictions that are unreliable and rejecting those above a threshold is particularly important in mission-critical applications like healthcare, where deferring to humans would be preferable to the model making a mistake (Kompa et al., 2021). Traditional full-model empirical error, or error bounds, are unusable since they do not vary with samples. We propose to construct bounds per subfunction and use the bound of the corresponding subfunction for a given sample as its unreliability metric. Figure 1 depicts our approach, where Figure 1(c) shows how a simple MLP shatters the input space into subfunctions, and Figure 1(d) and (e) show unreliability as predicted by our approach. We test a computationally tractable approximation of the bound on misclassification detection (predicting on which in-distribution test inputs the model will make mistakes) and out-of-distribution detection (predicting which test inputs come from a dataset materially different to the training one).

The terms unreliability and uncertainty are sometimes used interchangeably. In this work we use unreliability to mean a quantity that scales with predicted true error and uncertainty in that prediction. Unreliability subsumes uncertainty but not vice versa; one can predict with certainty that a model is always wrong, i.e. unreliable. Reliability corresponds to predicted generalization ability, since a model generalizes to a sample perfectly when there is no true error. Note the unreliability predictor is assumed not to have access to ground truth targets, otherwise true error is trivially known.

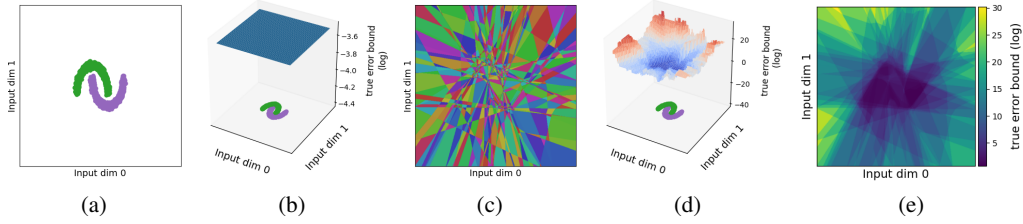


Figure 1: Shattering a neural network trained on the halfmoons dataset into co-dependent linear subfunctions to obtain a heatmap of unreliability across the input space. The model is a MLP with two hidden ReLU layers of 32 units each. (a) Training data. (b) Traditional full-model error bound: shown here is the single model bound from Mohri et al. (2018, eq. 2.17). (c) Linear activation regions. (d) Ours: subfunction error bound as unreliability. (e) Unreliability heatmap in 2D.

2 Subfunction error bounds

In this section, we introduce the required notation and main technical contribution of this work, namely our approach of estimating unreliability using subfunction error bounds.

2.1 Subfunctions

Let $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ be a piecewise linear neural network whose computational graph is comprised of linear functions and ReLU activation functions. Let the set of non-linear nodes in f_θ , corresponding to ReLUs, be $V = (v_j)_{1 \leq j \leq M}$, ordered according to any fixed ordering. Define $\text{subfunction}(p; f_\theta, V)$ to be a computational graph constructor that takes network f_θ and binary activation pattern $p \in \{0, 1\}^M$ and returns f_θ with nodes v_j where $p^j = 0$ replaced with a constant zero activation function ($g(x) = 0$), and nodes v_j where $p^j = 1$ replaced with the identity activation function ($g(x) = x$), therefore transforming the non-linearities of V into linearities.

Fix f_θ . Let $P \subset \{0, 1\}^M$, $P = (p_i)_{1 \leq i \leq C}$ be the ordered set of all feasible activation patterns¹, meaning all possible patterns induced by traversing the input space $\mathcal{X}$. The set of all subfunctions is:

$$H = (h_i)_{1 \leq i \leq C}, \quad \text{where } h_i \triangleq \text{subfunction}(p_i; f_\theta, V). \quad (1)$$

¹In general this is not equal to the full bit space $\{0, 1\}^M$, and computing it exactly is an open problem (Montúfar et al., 2014).

Recall that activation regions are the non-overlapping polytopes in input space corresponding to activation patterns. Assume an ordering where activation region a_i corresponds to subfunction h_i :

$$a_i \triangleq \{x \in \mathcal{X} \mid \forall j \in [1, M] : p_i^j = \mathbb{1}_{v_j(x) > 0}\}, \quad (2)$$

where $v_j(x)$ denotes the output of node v_j when $f_\theta(x)$ is computed. Note that subfunctions are identified by the pattern and have a different domain, so subfunctions with different patterns are considered distinct even if behaviorally identical. We overload the previous definition of H so the subfunction induced for any input sample $x \in \mathcal{X}$ is:

$$H(x) \triangleq h_i \text{ where } x \in a_i. \quad (3)$$

2.2 Error bounds

Fix a known training dataset S containing N sampled pairs, $S \in (\mathcal{X} \times \mathcal{Y})^N$, where samples are drawn iid from distribution D . Define the empirical error or risk of the full network on this dataset:

$$\widehat{R}_S(f_\theta) \triangleq \frac{1}{|S|} \sum_{n=1}^{|S|} r(S^n; f_\theta), \quad (4)$$

where r is a bounded error function on samples: $0 \leq r((x, y); f_\theta) \leq 1$ for all $x, y \in (\mathcal{X} \times \mathcal{Y})$ and S^n is the n -th element of S . Note empirical error, e.g. proportion of incorrect classifications, is not necessarily the objective function for training; it is fine for the latter to not have finite bounds.

Consider quantifying the empirical error for a subfunction h_i using traditional empirical error (eq. (4)). Let its activation region dataset be $S_i \triangleq S \cap a_i$ with $N_i \triangleq |S_i|$ samples drawn iid from its data distribution D_i , defined as $\mathbb{P}_{D_i}(x, y) = \mathbb{P}_D(x, y | x \in a_i)$. Then:

$$\widehat{R}_{S_i}(h_i) = \frac{1}{N_i} \sum_{n=1}^{N_i} r(S_i^n; f_\theta), \quad (5)$$

$$\text{and for any } \delta \in (0, 1], \text{ with probability } > 1 - \delta: \quad \mathbb{E}_{S_i}[\widehat{R}_{S_i}(h_i)] \leq \widehat{R}_{S_i}(h_i) + \sqrt{\frac{\log \frac{2}{\delta}}{2N_i}}. \quad (6)$$

The bound comes from Hoeffding's inequality². There are several issues with this formulation. First, it treats the empirical error of different subfunctions as independent when in general, they are not. Since different activation regions are bounded by shared hyperplanes, and hence the subfunctions share parameters, there exists useful evidence for a subfunction's performance outside its own activation region. Second, test samples overwhelmingly induce unseen activation patterns in deep networks, and the bound in eq. (6) is infinite if $N_i = 0$, making this quantity uninformative for the purposes of comparing subfunctions.

This motivates the following empirical risk metric for subfunctions. Fix non-negative kernel or closeness distance function between subfunctions, $k : H \times H \rightarrow \mathcal{R}^{\geq 0}$. For any subfunction $h_i \in H$, define its probability mass:

$$\mathbb{P}(h_i) \triangleq \frac{\sum_{j \in [1, C]} N_j k(h_i, h_j)}{\sum_{l \in [1, C]} \sum_{j \in [1, C]} N_j k(h_l, h_j)}, \quad \text{so by construction } \sum_{i \in [1, C]} \mathbb{P}(h_i) = 1, \quad (7)$$

which quantifies how densely h_i is locally populated by training samples, local meaning in subfunction kernel space, not Euclidean closeness in input space; though note these are related as Euclidean

²Note that whilst such bounds are loose in general, these subfunctions are linear functions. Under the assumption of equivalent complexity, using bounds based on complexity, which might be tighter, would not make a material difference as the quantity of interest is the relative difference between bounds.

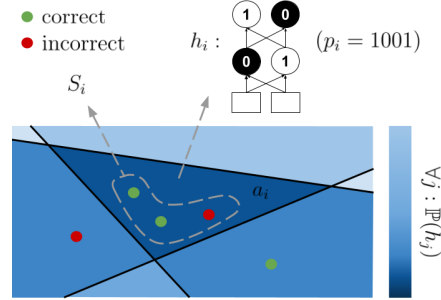


Figure 2: Notation summary. Subfunction h_i , pattern p_i , smooth density $\mathbb{P}(h_i)$, activation region a_i and its dataset S_i . If data falls on a boundary it is considered to be on the “0” side (eq. (2)).

distance of 0 between samples implies the same induced subfunction. Rewrite the empirical error of the network as an expectation over subfunction empirical error:

$$\hat{R}_S(f_\theta) = \frac{1}{N} \sum_{(x,y) \in S} r((x,y); f_\theta) \quad (8)$$

$$= \frac{1}{N} \sum_{(x,y) \in S} \frac{1}{\sum_{j \in [1,C]} \mathbb{P}(h_j) k(H(x), h_j)} \sum_{i \in [1,C]} \mathbb{P}(h_i) k(H(x), h_i) r((x,y); f_\theta) \quad (9)$$

$$= \frac{1}{N} \sum_{i \in [1,C]} \mathbb{P}(h_i) \sum_{(x,y) \in S} \frac{k(H(x), h_i) r((x,y); f_\theta)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) k(H(x), h_j)} \quad (10)$$

$$= \mathbb{E}_{h_i} \left[\frac{1}{N} \sum_{(x,y) \in S} \frac{k(H(x), h_i) r((x,y); f_\theta)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) k(H(x), h_j)} \right] \quad (11)$$

$$= \mathbb{E}_{h_i} \left[\frac{1}{N} \sum_{l \in [1,C]} \frac{k(h_l, h_i) N_l \hat{R}_{S_l}(h_l)}{\sum_{j \in [1,C]} \mathbb{P}(h_j) k(h_l, h_j)} \right] \triangleq \mathbb{E}_{h_i} [\hat{R}_S^*(h_i)]. \quad (12)$$

Theorem 2.1 (Subfunction true error bound). Assume that $\tilde{D}_S$ is a distribution over size- N datasets that have the same activation region data distributions and dataset sizes (D_i and N_i , $\forall i \in [1, C]$) as S . Consider S as a dataset drawn iid from $\tilde{D}_S$. Then $\forall i \in [1, C]$, for any given $\delta \in (0, 1]$, with probability $> 1 - \delta$:

$$\underbrace{R_{\tilde{D}_S}^*(h_i) \triangleq \mathbb{E}_S[\hat{R}_S^*(h_i)]}_{\text{true error}} \leq \underbrace{\hat{R}_S^*(h_i)}_{\text{empirical error}} + \underbrace{\frac{1}{\mathbb{P}(h_i)} \sqrt{\log \frac{2}{\delta}}}_{\text{generalization gap bound}} \sqrt{\frac{2}{2N}}. \quad (13)$$

Proof in appendix A. Intuitively this implies that the more a subfunction is surrounded by samples for which the model makes accurate predictions - both from its own domain and domains of other related subfunctions, weighted by the kernel - the lower its empirical error and bound on generalization gap, and thus the lower its bound on true error, given any fixed δ . This improves upon eq. (6) by being finite even for subfunctions without training samples, given positive-valued kernels. Although $\frac{1}{\mathbb{P}(h_i)}$ can be very large in general, it is tempered by a factor of $\frac{1}{\sqrt{N}}$, and again note that the relative difference in bounds is the quantity of interest when assessing subfunctions. Taking an expectation over subfunctions yields:

$$\mathbb{E}_{h_i} [R_{\tilde{D}_S}^*(h_i)] \leq \mathbb{E}_{h_i} \left[\hat{R}_S^*(h_i) + \frac{1}{\mathbb{P}(h_i)} \sqrt{\log \frac{2}{\delta}} \right] = \underbrace{\hat{R}_S(f_\theta)}_{\text{reconstructed true error bound (RTEB)}} + C \sqrt{\log \frac{2}{\delta}} \sqrt{\frac{2}{2N}}. \quad (14)$$

Ideal subfunction kernel. Note that for now the derivation assumes the subfunction kernel is given, which raises the question of what a meaningful subfunction kernel might be. Consider the case of a network with two independent subfunctions that do not share any parameters. In this case, samples in the activation region for one are entirely relevant for its own empirical error whilst entirely irrelevant for the other. In contrast, if all parameters were shared between subfunctions, all samples would be equally relevant. We propose to reason about the relationship between subfunctions via Taylor expansions. Recall that S is the full dataset with $|S| = N$ samples. Call the inputs and targets of the dataset X_S and Y_S respectively, $\forall n \in [1, |S|] : (X_S^n, Y_S^n) \triangleq S^n$. Consider the scenario where instead of being limited to N_i iid samples $S_i \sim D_i$, we had access to N iid samples $\tilde{S}_i \sim D_i$. Assume the data distribution is well conditioned in that each target is in fact predictable from each input. Then $\exists q, g$ s.t. $\forall (x, y) \in S \cup \tilde{S}_i : r((x, y); f_\theta) = q(g(x))$, and $\forall n$:

$$r((X_S^n, Y_S^n); f_\theta) = q(g(X_S^n)) = q(g(X_{\tilde{S}_i}^n)) + \mathcal{O}(\|g(X_S^n) - g(X_{\tilde{S}_i}^n)\|), \quad (15)$$

using ideal samples $\tilde{S}_i$ as the anchor in the Taylor expansion and assuming bounded first order derivatives for q as standard. Then the ideal kernel k for computing the error on given $\tilde{S}_i$ using

samples from S is given by:

$$\operatorname{argmin}_k \frac{1}{N} \sum_{n=1}^N \|r((X_{\tilde{S}_i}^n, Y_{\tilde{S}_i}^n); f_\theta) - r((X_S^n, Y_S^n); f_\theta) k(g(X_{\tilde{S}_i}^n), g(X_S^n))\|, \quad (16)$$

$$\text{which is satisfied when } k(g(X_{\tilde{S}_i}^n), g(X_S^n)) = \frac{r((X_{\tilde{S}_i}^n, Y_{\tilde{S}_i}^n); f_\theta)}{r((X_S^n, Y_S^n); f_\theta)} = 1 - \frac{\mathcal{O}(\|g(X_S^n) - g(X_{\tilde{S}_i}^n)\|)}{r((X_S^n, Y_S^n); f_\theta)} \quad (17)$$

via eq. (15), which supports the intuition that the ideal kernel value should scale inversely with representational distance, for non-zero $r((X_S^n, Y_S^n); f_\theta)$.

Fewer activation regions improves reconstructed true error bound. Equation (14) indicates models with a smaller number of patterns or activation regions $C = |P| = |H|$ have lower RTEB. We can think of the number of patterns as being correlated to *abstraction strength*, hence simpler or more abstract models generalize better over the domain, a widely accepted view in the community also known as Occam’s razor (Friedman et al., 2001). Specifically, for a neural network split at any layer into two networks g and f , if one defines abstraction strength of $f \circ g$ as the number of patterns of g minus that of f , then any alternative $\tilde{f}$ providing stronger abstraction due to fewer patterns has a lower RTEB than f , all else equal. This also emphasizes, similar to previous work (Montúfar et al., 2014; Hanin and Rolnick, 2019), that number of activation regions might be a meaningful metric of capacity, though by no means perfect or accurate. We conjecture that there is a potentially interesting avenue to the recent *double descent* phenomenon (Belkin et al., 2019; Nakkiran et al., 2019) where it was observed, counter to accepted wisdom, that generalization in deep networks can be improved by increasing the number of parameters and damaged by increasing training data, in certain regimes. The RTEB formula suggests that the number of activation regions is a more meaningful model size metric for generalization than the number of parameters, and also that increasing dataset size N is only beneficial as long as the square root of the rate of change in N is larger than the rate of change in C , all else equal. We leave further investigation of this potential connection to future work.

Coarse partitioning case. Note that our technical derivation allows for an arbitrary partitioning of the model. When the partitioning is coarser than linear activation functions, i.e. M is fewer than the total number of activation nodes and subfunctions H are piecewise linear, eq. (13) becomes a bound on piecewise linear subfunction error and still holds. Though eq. (14) also still holds, using it to compare different models (as opposed to using eq. (13) to compare subfunctions, as we do in the experiments) is ill-founded if the granularity of partitioning is allowed to vary, since model and granularity constitute two factors of variation. For example, one could choose the vacuous partitioning where V is the empty set, so $P = \{\emptyset\}$ contains the zero length bit pattern, $C = |P| = 1$ and the subfunction is the full model, which is a minimization of the generalization gap bound in RTEB over partitionings. This recovers the original model error bound (Mohri et al. (2018, eq. 2.17)) and amounts to not using our method. Thus for model comparison, the granularity of partitioning should be constant and non-vacuous, for example by utilizing the *maximum* RTEB generalization gap bound over partitionings, which implies all models under consideration are partitioned into linear subfunctions. Furthermore, linear subfunctions provides the maximum resolution at which one can measure reliability using STEB, and is the regime where assuming equal subfunction complexity is most valid (since piecewise linear subfunctions constructed by considering a subset of nodes will each subsume a different number of linear subfunctions, in general), which is relevant because STEB does not consider variations in subfunction complexity.

Limitations. There are a few limitations and assumptions made within our work. First and foremost, reliability is measured by relating probabilistic upper error bounds of different functions. However, superiority of a probabilistic upper bound of a quantity does not imply the superiority of the quantity itself. This is a drawback of all probabilistic generalization bounds of this form, including the classical results for model selection (Mohri et al., 2018). Thus in assuming that changes in bound value are representative of changes in true error, we are making a leap of faith, supported by empirical evidence via experiments. Further, in the interest of making assumptions explicit, note the bound assumes subfunction activation region dataset sizes are constant, just as the traditional full model bound (Mohri et al. (2018, eq. 2.17)) assumes constant dataset size. Finally, our derivation relies on a decomposition of the neural network into subfunctions and defining a subfunction kernel, which

incurs a choice. The correctness of kernel choice depends on its ultimate objective, which in eq. (16) we assumed to be augmenting the samples of each subfunction. The ideal kernel in this case is intractable, hence in practice we use an approximation motivated by eq. (17), detailed in due course.

3 Related work

Subfunction error brings together three areas of machine learning research that are not typically combined. The first is work on activation regions and linear regions, motivated by characterising neural network expressivity or capacity (Montúfar et al., 2014; Raghu et al., 2017; Hanin and Rolnick, 2019). The second is characterizing model-level generalization with generalization bounds, intended for model selection (Mohri et al., 2018; Friedman et al., 2001). The third is sample-level generalization with unreliability metrics intended for out-of-distribution or unreliable in-distribution sample selection (Ovadia et al., 2019).

For the latter, a well known method is simply using the maximum response of the final softmax prediction layer as a proxy for sample reliability (Hendrycks and Gimpel, 2017; Geifman and El-Yaniv, 2017; Cordella et al., 1995; Chow, 1957). Related metrics include prediction entropy (Shannon, 1948), and margin (Scheffer et al., 2001). ODIN (Liang et al., 2017) combines max response with temperature scaling and input perturbations. Lee et al. (2018) trains a logistic regressor on layer-wise distances of a sample’s features to its nearest class, with the idea that distance to features of the nearest class should scale with unreliability. This also uses input perturbations and was shown to outperform ODIN. Jiang et al. (2018) combines the idea of the margin method (where reliability is likelihood of top class relative to second class) with class distance. Non-parametric kernel based methods such as Gaussian processes provide measures of uncertainty that scale with distance from samples (an idea that is similar at a high level to ours) and can be appended to a neural network base (Liu et al., 2020). Zhang et al. (2020) assume density in latent space is correlated to reliability, using residual flows (Chen et al., 2019) for the density model. If multiple models trained on the same dataset are available (which we do not assume), one could use ensemble model metrics such as variance, max response or entropy (Lakshminarayanan et al., 2016); an ensemble can also be simulated in a single model with dropout (Geifman and El-Yaniv, 2017; Gal and Ghahramani, 2016).

To set this work in context, we cast generalization bounds as a sample-level unreliability metric, by dividing the network into activation regions whose functions are co-dependent according to a kernel. Note that we do not deal with training neural networks to minimize uncertainty, but analyze trained models. Further, by uncertainty we mean epistemic uncertainty (reducible by increasing sample size) as opposed to aleatoric (e.g. measurement error).

4 Practical considerations

We would like to compute subfunction true error bound for real neural networks, which faces several challenges. First, the intractability of computing the set of all feasible activation patterns given inputs, which remains an open problem, so in experiments we use the set of all conceivable patterns, $P = \{0, 1\}^M$. This means the error bound is computed in an altered subfunction space where some infeasible subfunctions are assigned a non-zero density. Second, the sums in the normalization constants of eqs. (7) and (12) are thus exponential in pattern length: $C = |H| = |P| = 2^M$. We resolve this with computationally tractable exact solutions for kernels that use the Hamming distance, i.e. $k(h_i, h_j) = d(\text{Hamming}(p_i, p_j))$ for any $d : \mathcal{N} \rightarrow \mathcal{R}^{\geq 0}$. For normalization in eq. (7), we reduce computational complexity from exponential $\mathcal{O}(N2^M)$ to linear $\mathcal{O}(M)$ (derivations in appendix B):

$$\sum_{l \in [1, C]} \sum_{\substack{j \in [1, C] \\ \text{s.t. } N_j > 0}} N_j k(h_l, h_j) = N \sum_{b=0}^M d(b) \binom{M}{b} \triangleq u, \quad (18)$$

and for normalization in eq. (12), from exponential $\mathcal{O}(N^2 2^M)$ to polynomial $\mathcal{O}(N^2 + M^3)$:

$$\forall_{\substack{l \in [1, C] \\ \text{s.t. } N_l > 0}} : \sum_{\substack{j \in [1, C] \\ \text{s.t. } N_j > 0}} \mathbb{P}(h_j) k(h_l, h_j) = \frac{1}{u} \sum_{\substack{i \in [1, C] \\ \text{s.t. } N_i > 0}} N_i z(\text{Hamming}(p_l, p_i)), \quad (19)$$

$$\text{where } \forall a \in [0, M] : z(a) \triangleq \sum_{b=0}^M d(b) \sum_{c=0}^b \binom{a}{c} \binom{M-a}{b-c} d(a + (b-c) - c). \quad (20)$$

For tractability on deep networks, we also use a coarse partitioning by taking activations from the last ReLU layer, so subfunctions are piecewise linear; as discussed in section 2.2, eq. (13) still holds. Finally, instead of using the ideal kernel (which we define in section 2.2 but do not solve for, specifically because $r((X_{\tilde{S}_i}^n, Y_{\tilde{S}_i}^n); f_\theta)$ is unknown), we use validation data to empirically fit a Gaussian kernel $d(b) \triangleq \exp(-\frac{b^2}{2\rho^2})$ with variance parameter ρ , motivated by the upper bound in eq. (17).

5 Experiments

We took pre-trained classification networks for CIFAR100 and CIFAR10 (Krizhevsky, 2009) and tested how well several single-model unreliability metrics predicted which test samples were incorrectly classified by the classification network (in-distribution setting), or came from CIFAR10/CIFAR100 or SVHN instead of CIFAR100/CIFAR10 (out-of-distribution or OOD setting). Both are cast as binary classification and the definitions for positive/negative were: predictor output: accept/reject, in-distribution ground truth: classified correct/incorrect, out-of-distribution ground truth: old distribution/new distribution. Evaluation metrics were area under the graphs of coverage vs. effective accuracy (AUCEA) and false positive rate vs. true positive rate (AUROC), with the latter as standard for OOD experiments.

$$\text{coverage} = \frac{TP + FP}{TP + FP + TN + FN} \quad \text{effective accuracy} = \text{precision} = \frac{TP}{TP + FP} \quad (21)$$

$$\text{false positive rate} = \frac{FP}{FP + TN} \quad \text{true positive rate} = \frac{TP}{TP + FN}. \quad (22)$$

All methods operate by producing a metric per sample that is assumed to scale with unreliability, so 1K thresholds for discretizing into accept/reject decisions were uniformly sampled across the maximum test set range. Model architectures are standard, except the addition of a linear layer to ResNet50 to enable collection of penultimate ReLU features. The in-distribution validation set was used to select all hyperparameters, including for OOD, i.e. we assume that OOD data is truly unseen for all parameters. The two hyperparameters of our method are δ and ρ , and log was taken to suppress large magnitudes of the bound. Pseudocode is given in Algorithm 1.

Algorithm 1: Tractable subfunction error bounds for prediction unreliability

- 1 **Require:** pre-trained model f_θ and training/validation/test datasets.
 - 2 Get activation patterns ($\sim$ subfunctions) for training and validation data from last ReLU layer;
 - 3 **for** hyperparameter values δ, ρ **do**
 - 4 Compute global normalization constant using ρ (eq. (18));
 - 5 For training data subfunctions, compute normalization constants using ρ (eq. (19));
 - 6 For validation data subfunctions, compute log STEB using δ, ρ and normalization constants (eq. (13));
 - 7 Compute validation metric AUCEA using log STEB of sample’s subfunction as unreliability;
 - 8 **if** highest AUCEA **then**
 - 9 Store δ, ρ as best with normalization constants;
 - 10 Get activation patterns for test data from last ReLU layer;
 - 11 For test data subfunctions, compute log STEB using best δ, ρ and normalization constants (eq. (13));
 - 12 Compute test metrics using log STEB of sample’s subfunction as unreliability.
-

Table 1: Model trained on CIFAR100. Out-of-distribution detection (AUROC) vs CIFAR10/SVHN.

	→ CIFAR10		→ SVHN	
	VGG16	ResNet50	VGG16	ResNet50
Residual flows density (Chen et al., 2019)	0.495 ± 2E-04	0.495 ± 2E-04	0.090 ± 2E-04	0.090 ± 2E-04
GP (Liu et al. (2020) w/ fixed features)	0.708 ± 6E-03	0.404 ± 3E-02	0.830 ± 3E-02	0.396 ± 8E-02
Class distance (Lee et al., 2018)	0.627 ± 4E-02	0.513 ± 4E-02	0.833 ± 3E-02	0.579 ± 1E-01
Margin (Scheffer et al., 2001)	0.716 ± 2E-03	0.736 ± 3E-03	0.771 ± 5E-03	0.790 ± 3E-02
Entropy (Shannon, 1948)	0.725 ± 3E-03	0.745 ± 3E-03	0.786 ± 6E-03	0.814 ± 4E-02
Max response (Cordella et al., 1995)	0.719 ± 3E-03	0.740 ± 3E-03	0.776 ± 6E-03	0.799 ± 3E-02
Subfunctions (ours)	0.738 ± 2E-03	0.750 ± 7E-03	0.797 ± 8E-03	0.807 ± 3E-02

Table 2: Model trained on CIFAR10. Out-of-distribution detection (AUROC) vs CIFAR100/SVHN.

	→ CIFAR100		→ SVHN	
	VGG16	ResNet50	VGG16	ResNet50
Residual flows density	$0.513 \pm 2\text{E-}04$	$0.513 \pm 1\text{E-}04$	$0.084 \pm 1\text{E-}04$	$0.084 \pm 1\text{E-}04$
GP	$0.810 \pm 1\text{E-}02$	$0.575 \pm 2\text{E-}02$	$0.844 \pm 2\text{E-}02$	$0.473 \pm 9\text{E-}02$
Class distance	$0.673 \pm 7\text{E-}02$	$0.468 \pm 3\text{E-}02$	$0.806 \pm 6\text{E-}02$	$0.462 \pm 1\text{E-}01$
Margin	$0.829 \pm 2\text{E-}03$	$0.825 \pm 5\text{E-}03$	$0.854 \pm 4\text{E-}02$	$0.856 \pm 2\text{E-}02$
Entropy	$0.853 \pm 2\text{E-}03$	$0.822 \pm 6\text{E-}03$	$0.869 \pm 3\text{E-}02$	$0.858 \pm 3\text{E-}02$
Max response	$0.829 \pm 3\text{E-}03$	$0.827 \pm 5\text{E-}03$	$0.850 \pm 4\text{E-}02$	$0.858 \pm 2\text{E-}02$
Subfunctions (ours)	$0.858 \pm 4\text{E-}03$	$0.862 \pm 2\text{E-}03$	$0.886 \pm 2\text{E-}02$	$0.915 \pm 2\text{E-}02$

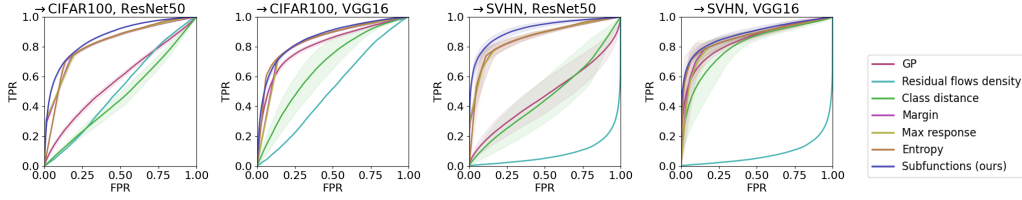


Figure 3: ROC curves with standard deviation for OOD experiments on CIFAR10.

Table 3: Model trained on CIFAR100. Predicting mis-classification on in-distribution test.

	VGG16		ResNet50	
	AUCEA	AUROC	AUCEA	AUROC
Residual flows density	$0.293 \pm 1\text{E-}02$	$0.622 \pm 1\text{E-}02$	$0.293 \pm 2\text{E-}02$	$0.630 \pm 1\text{E-}02$
GP	$0.882 \pm 1\text{E-}03$	$0.803 \pm 2\text{E-}03$	$0.670 \pm 2\text{E-}02$	$0.384 \pm 2\text{E-}02$
Class distance	$0.838 \pm 3\text{E-}02$	$0.739 \pm 6\text{E-}02$	$0.627 \pm 7\text{E-}02$	$0.476 \pm 3\text{E-}02$
Margin	$0.899 \pm 2\text{E-}03$	$0.852 \pm 4\text{E-}03$	$0.870 \pm 6\text{E-}03$	$0.855 \pm 4\text{E-}03$
Entropy	$0.895 \pm 2\text{E-}03$	$0.856 \pm 5\text{E-}03$	$0.916 \pm 4\text{E-}03$	$0.859 \pm 4\text{E-}03$
Max response	$0.899 \pm 2\text{E-}03$	$0.853 \pm 4\text{E-}03$	$0.864 \pm 7\text{E-}03$	$0.857 \pm 4\text{E-}03$
Subfunctions (ours)	$0.904 \pm 1\text{E-}03$	$0.864 \pm 3\text{E-}03$	$0.902 \pm 4\text{E-}03$	$0.827 \pm 4\text{E-}03$

Table 4: Model trained on CIFAR10. Predicting mis-classification on in-distribution test.

	VGG16		ResNet50	
	AUCEA	AUROC	AUCEA	AUROC
Residual flows density	$0.442 \pm 5\text{E-}02$	$0.520 \pm 1\text{E-}02$	$0.492 \pm 3\text{E-}02$	$0.577 \pm 1\text{E-}02$
GP	$0.983 \pm 1\text{E-}03$	$0.865 \pm 8\text{E-}03$	$0.943 \pm 5\text{E-}03$	$0.625 \pm 2\text{E-}02$
Class distance	$0.948 \pm 2\text{E-}02$	$0.669 \pm 8\text{E-}02$	$0.900 \pm 3\text{E-}02$	$0.471 \pm 6\text{E-}02$
Margin	$0.982 \pm 2\text{E-}03$	$0.900 \pm 4\text{E-}03$	$0.848 \pm 1\text{E-}02$	$0.894 \pm 4\text{E-}03$
Entropy	$0.989 \pm 1\text{E-}04$	$0.914 \pm 3\text{E-}03$	$0.984 \pm 1\text{E-}03$	$0.890 \pm 5\text{E-}03$
Max response	$0.980 \pm 3\text{E-}03$	$0.898 \pm 4\text{E-}03$	$0.832 \pm 1\text{E-}02$	$0.895 \pm 5\text{E-}03$
Subfunctions (ours)	$0.988 \pm 3\text{E-}04$	$0.907 \pm 3\text{E-}03$	$0.983 \pm 2\text{E-}03$	$0.889 \pm 6\text{E-}03$

The use of subfunctions was the best method on hard OOD settings (from CIFAR10 to CIFAR100 and vice versa, tables 1 and 2 and fig. 3). For the SVHN experiments, it was either best or second best. For the in-distribution setting, where margins were smaller, the top two performers in most experiments were found to be entropy and subfunctions (tables 3 and 4). Note these also dominated the best OOD performances (tables 1 and 2), though here subfunctions had the clear advantage. We conclude that these methods were good predictors of unreliability for both in-distribution and out-of-distribution scenarios. The AUCEA results for the in-distribution setting mean that using either

Table 5: Test accuracy of original models.

Dataset	Model	Accuracy
CIFAR100	VGG16	$0.704 \pm 1\text{E-}03$
CIFAR100	ResNet50	$0.729 \pm 8\text{E-}03$
CIFAR10	VGG16	$0.920 \pm 2\text{E-}03$
CIFAR10	ResNet50	$0.908 \pm 5\text{E-}03$

subfunctions or entropy would have raised effective model accuracy (i.e. accuracy of the accepted samples) to $90 \sim 92\%$ from $70 \sim 73\%$ for the original CIFAR100 models, and to $98 \sim 99\%$ from $91 \sim 92\%$ for the original CIFAR10 models (tables 3 to 5), on average over accept/reject thresholds.

Figure 4: Sample confusion matrix, OOD for CIFAR10 $\rightarrow$ CIFAR100 on ResNet50.

is clear from the correlation between green and lower unreliability ranking that subfunction error bounds rate OOD classes semantically closer to the training classes as more reliable. Note that even the exceptions towards the right hand side are justified, because the inclusion of e.g. bicycles, lawn mowers and rockets in “vehicles” is questionable; certainly these objects do not correspond visually to the vehicle classes in CIFAR10. In addition, we ran the same plot for the other methods and found that in every case the correlation between unreliability and semantic novelty was less strong than for subfunctions (appendix D).

Figure 5: OOD for CIFAR10 $\rightarrow$ CIFAR100 on ResNet50. 10k CIFAR100 test samples were ranked by unreliability (log STEB). Boxplots summarize rankings per class (lower = less unreliable). Green denotes superclasses similar to CIFAR10: carnivores, omnivores, herbivores, mammals and vehicles.

Subfunction error is a form of empirical error that shares training samples across activation patterns according to their degree of representational overlap, defined by a kernel. We showed empirically that its bound, which combines empirical error with density in kernel space, is a meaningful measure of unreliability: subfunctions densely supported by training samples tend to be more reliable. Subfunction error encourages a novel way of thinking about generalization; specifically, that one has access to rich information contextualizing the risk of any given input sample that is wasted by treating the network as a black box. If it were computationally tractable to use the full activation pattern, compute the feasible regions or ideal kernel, we might expect a further performance advantage. In the meantime, the proposed approach presents a competitive alternative to the simpler entropy and max response methods, with higher computational cost justified to an extent by a strong advantage on out-of-distribution detection.

References

- M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019.
- C. M. Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.
- R. T. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.
- C.-K. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, pages 247–254, 1957.
- L. P. Cordella, C. De Stefano, F. Tortorella, and M. Vento. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147, 1995.
- J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*. Springer series in statistics New York, 2001.
- Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. *arXiv preprint arXiv:1705.08500*, 2017.
- B. Hanin and D. Rolnick. Deep ReLU networks have surprisingly few activation patterns. *Advances in Neural Information Processing Systems*, 32:361–370, 2019.
- D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.
- H. Jiang, B. Kim, M. Y. Guan, and M. R. Gupta. To trust or not to trust a classifier. In *NeurIPS*, pages 5546–5557, 2018.
- B. Kompa, J. Snoek, and A. L. Beam. Second opinion needed: communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, 4(1):1–6, 2021.
- A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.
- K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *arXiv preprint arXiv:1807.03888*, 2018.
- S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.
- J. Z. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax-Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.
- M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.
- G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. *arXiv preprint arXiv:1402.1869*, 2014.
- P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.
- Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

- Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.
- M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017.
- T. Scheffer, C. Decomain, and S. Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.
- C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423, 1948.
- H. Zhang, A. Li, J. Guo, and Y. Guo. Hybrid models for open set recognition. In *European Conference on Computer Vision*, pages 102–117. Springer, 2020.

Predicting Unreliable Predictions by Shattering a Neural Network: Supplementary Material

A Proof for Theorem 2.1

This follows the general Chernoff/Hoeffding bounding technique. Fix model f_θ , patterns P and thus subfunctions set H . Let the true data distribution for each activation region a_i be D_i . Assume that $\tilde{D}_S$ is a distribution over size- N datasets that have the same activation region data distributions and dataset sizes (D_i and N_i , $\forall i \in [1, C]$) as S . Consider S as a dataset drawn iid from $\tilde{D}_S$. Without loss of generality, since dataset order is immaterial for the error metrics, assume the index n of sample S^n determines its activation region and distribution from which it is drawn independently of other samples. That is, $\forall n \in [1, N] : S^n \sim D_{J^n}$ iid for some fixed $J \in [1, C]^N$. Recall that D_i is constructed as $\mathbb{P}_{D_i}(x, y) = \mathbb{P}_D(x, y | x \in a_i)$. Call the inputs and targets X_S and Y_S , $\forall n \in [1, N] : (X_S^n, Y_S^n) \triangleq S^n$. Recall that empirical error function for samples is bounded: $\forall x, y \in (\mathcal{X} \times \mathcal{Y}) : 0 \leq r((x, y); f_\theta) \leq 1$ and kernel k is non-negative valued.

Choose any subfunction $h_i \in H$, $i \in [1, C]$. Then $\forall \epsilon > 0, t > 0, \tilde{S} \sim \tilde{D}_S$ iid:

$$\mathbb{P}(\hat{R}_S^*(h_i) - R_{\tilde{D}_S}^*(h_i) \geq \epsilon) = \mathbb{P}(\hat{R}_S^*(h_i) - \mathbb{E}_{\tilde{S}}[\hat{R}_S^*(h_i)] \geq \epsilon) \quad (23)$$

$$= \mathbb{P}(e^{t(\hat{R}_S^*(h_i) - \mathbb{E}_{\tilde{S}}[\hat{R}_S^*(h_i)])} \geq e^{t\epsilon}) \quad (24)$$

$$\leq e^{-t\epsilon} \mathbb{E}_{\tilde{S}}[e^{t(\hat{R}_S^*(h_i) - \mathbb{E}_{\tilde{S}}[\hat{R}_S^*(h_i)])}] \quad (25)$$

$$= e^{-t\epsilon} \mathbb{E}_{\tilde{S}}[e^{t \sum_{n=1}^N \left(\frac{k(H(X_S^n), h_i) r(S^n; f_\theta)}{N \sum_{j \in [1, C]} \mathbb{P}(h_j) k(H(X_S^n), h_j)} - \mathbb{E}_{\tilde{S}^n} \left[\frac{k(H(X_S^n), h_i) r(\tilde{S}^n; f_\theta)}{N \sum_{j \in [1, C]} \mathbb{P}(h_j) k(H(X_S^n), h_j)} \right] \right)}] \quad (26)$$

$$= e^{-t\epsilon} \prod_{n=1}^N \mathbb{E}_{S^n}[e^{t \left(\frac{k(H(X_S^n), h_i) r(S^n; f_\theta)}{N \sum_{j \in [1, C]} \mathbb{P}(h_j) k(H(X_S^n), h_j)} - \mathbb{E}_{\tilde{S}^n} \left[\frac{k(H(X_S^n), h_i) r(\tilde{S}^n; f_\theta)}{N \sum_{j \in [1, C]} \mathbb{P}(h_j) k(H(X_S^n), h_j)} \right] \right)}] \quad (27)$$

$$\leq e^{-t\epsilon} \prod_{n=1}^N e^{\frac{t^2 \left(\frac{1}{N \mathbb{P}(h_i)} - 0 \right)^2}{8}} \quad (28)$$

$$= e^{\frac{t^2}{8N \mathbb{P}(h_i)^2} - t\epsilon}, \quad (29)$$

by making use of the following:

- eq. (24) $\rightarrow$ eq. (25) : Markov's inequality. For random variable $Z \geq 0$ and constant $a > 0$, $\mathbb{P}(Z \geq a) \leq \frac{\mathbb{E}[Z]}{a}$.
- eq. (25) $\rightarrow$ eq. (26) : definition of $\hat{R}_S^*(h_i)$ and linearity of expectation.
- eq. (26) $\rightarrow$ eq. (27) : $\hat{R}_S^*(h_i)$ is a sum over independently drawn samples. $\forall h_j \in H : \mathbb{P}(h_j)$ is constant because activation region dataset sizes are constant.
- eq. (27) $\rightarrow$ eq. (28) : first bound the length of the range of the exponent. $\forall i \in [1, C], n \in [1, N] : 0 \leq \frac{k(H(X_S^n), h_i) r(S^n; f_\theta)}{N \sum_{j \in [1, C]} \mathbb{P}(h_j) k(H(X_S^n), h_j)} \leq \frac{k(H(X_S^n), h_i)}{N \mathbb{P}(h_i) k(H(X_S^n), h_i)} = \frac{1}{N \mathbb{P}(h_i)}$. Subtracting a constant does not change the length of the range of a random variable. Then apply Hoeffding's lemma: for random variable Z where $a \leq Z \leq b$ and $\mathbb{E}[Z] = 0$, then $\forall t > 0 : \mathbb{E}[e^{tZ}] \leq e^{\frac{t^2(b-a)^2}{8}}$ holds.

Find the optimal t as the one that yields the tightest bound:

$$\nabla_t e^{\frac{t^2}{8N \mathbb{P}(h_i)^2} - t\epsilon} = 0, \quad t = 4N\epsilon \mathbb{P}(h_i)^2, \quad (30)$$

which is a minimum because the second derivative is positive. Substitute into eq. (29) :

$$\mathbb{P}(\hat{R}_S^*(h_i) - R_{\tilde{D}_S}^*(h_i) \geq \epsilon) \leq e^{-2N\epsilon^2 \mathbb{P}(h_i)^2}. \quad (31)$$

The symmetric case can be proved in the same way:

$$\mathbb{P}(\widehat{R}_S^*(h_i) - R_{\widehat{D}_S}^*(h_i) \leq -\epsilon) = \mathbb{P}(R_{\widehat{D}_S}^*(h_i) - \widehat{R}_S^*(h_i) \geq \epsilon) \leq e^{-2N\epsilon^2 \mathbb{P}(h_i)^2}, \quad (32)$$

specifically because swapping the order of the subtraction in eq. (27) does not change the value of the squared range used in eq. (28). Combining eq. (31) and eq. (32):

$$\mathbb{P}_S(|\widehat{R}_S^*(h_i) - R_{\widehat{D}_S}^*(h_i)| \geq \epsilon) \leq 2e^{-2N\epsilon^2 \mathbb{P}(h_i)^2}. \quad (33)$$

Setting the right hand side to δ and solving for ϵ completes the derivation. Namely for any $\delta > 0$, the following hold with probability $\leq \delta$:

$$|\widehat{R}_S^*(h_i) - R_{\widehat{D}_S}^*(h_i)| \geq \frac{1}{\mathbb{P}(h_i)} \sqrt{\frac{\log \frac{2}{\delta}}{2N}} \quad (34)$$

$$R_{\widehat{D}_S}^*(h_i) - \widehat{R}_S^*(h_i) \geq \frac{1}{\mathbb{P}(h_i)} \sqrt{\frac{\log \frac{2}{\delta}}{2N}}. \quad (35)$$

Hence the following hold with probability $> 1 - \delta$:

$$R_{\widehat{D}_S}^*(h_i) < \widehat{R}_S^*(h_i) + \frac{1}{\mathbb{P}(h_i)} \sqrt{\frac{\log \frac{2}{\delta}}{2N}} \quad (36)$$

$$R_{\widehat{D}_S}^*(h_i) \leq \widehat{R}_S^*(h_i) + \frac{1}{\mathbb{P}(h_i)} \sqrt{\frac{\log \frac{2}{\delta}}{2N}}. \quad (37)$$

□.

B Derivations for Section 4

Normalization term in eq. (7). Computed once for the network.

$$\sum_{l \in [1, C]} \sum_{\substack{j \in [1, C] \\ \text{s.t. } N_j > 0}} N_j k(h_l, h_j) = \sum_{\substack{j \in [1, C] \\ \text{s.t. } N_j > 0}} N_j \sum_{l \in [1, C]} k(h_l, h_j) = N \sum_{b=0}^M d(b) \binom{M}{b} \triangleq u, \quad (38)$$

where the trick is that $\sum_{l \in [1, C]} k(h_l, h_j) = \sum_{l \in [1, C]} d(\text{Hamming}(p_l, p_j))$ is the same for all h_j due to completeness of the bit space P , and the specific value can be found by looping through all possible Hamming distances b . This reduces the computational complexity from $\mathcal{O}(N2^M)$ to $\mathcal{O}(M)$, because the number of populated activation regions, i.e. length of loop over j , is upper bounded by the number of samples N , and $\binom{M}{b}$ can be iteratively updated in $\mathcal{O}(1)$ in the loop over b :

$$\binom{M}{b} = \binom{M}{b-1} \frac{M - (b-1)}{b}. \quad (39)$$

Normalization term in eq. (12). Computed $\forall l \in [1, C]$ s.t. $N_l > 0$:

$$\sum_{j \in [1, C]} \mathbb{P}(h_j) k(h_l, h_j) \quad (40)$$

$$= \frac{1}{u} \sum_{j \in [1, C]} \sum_{\substack{i \in [1, C] \\ \text{s.t. } N_i > 0}} k(h_l, h_j) k(h_j, h_i) N_i \quad (41)$$

$$= \frac{1}{u} \sum_{\substack{i \in [1, C] \\ \text{s.t. } N_i > 0}} N_i \sum_{j \in [1, C]} k(h_l, h_j) k(h_j, h_i) \quad (42)$$

$$= \frac{1}{u} \sum_{\substack{i \in [1, C] \\ \text{s.t. } N_i > 0}} N_i \sum_{b=0}^M \overbrace{\sum_{c=0}^b \binom{\text{Hamming}(p_l, p_i)}{c} \binom{M - \text{Hamming}(p_l, p_i)}{b-c}}^{\text{from } k(h_l, h_j)} \overbrace{d(\text{Hamming}(p_l, p_i) + (b-c) - c)}^{\text{from } k(h_j, h_i)}. \quad (43)$$

Note the loops over i and l only need to range over populated activation regions due to the inclusion of their sample counts (N_i and N_l) as multiplicative factors in their loop contents, which means the iterations are bounded by $\mathcal{O}(N)$. The problematic loop is over j , i.e. all subfunctions/activation regions whether populated by samples or not, which is $\mathcal{O}(2^M)$ when P is the full bit space $\{0, 1\}^M$ (section 4). However, similar to eq. (38) above, it is also this fullness that we exploit.

Now we explain eq. (43) in detail. The value $z(\text{Hamming}(p_l, p_i))$ corresponds to the sum over j in eq. (42): loop over every bit pattern p_j in P , multiply its closeness to p_l with its closeness to p_i (both fixed outside this loop), and sum. Now group p_j by bit distance to p_l , b . Consider one instance of b (fig. 6). Within this group for b , the closeness of every p_j to p_l is constant, namely equal to $d(b)$. But they have different distances to p_i . However we know the number of bits that are different between p_l and p_i : this is $\text{Hamming}(p_l, p_i)$. Consider a specific instance of p_j . Visualize p_l turning into p_j by flipping bit one at a time; each will either bring the pattern closer to or further away from p_i . There is a budget of b different bits to flip to reach p_j . Say c of those flips brought it closer (i.e. turned the bit value at that position into the same as p_i 's). Then we know $b - c$ brought it further. So the number of different bits between p_j and p_i is exactly $\text{Hamming}(p_l, p_i) + (b - c) - c$. And the number of p_j that fit this description is the number of ways of choosing c from the different bits between p_l and p_i multiplied by the number of ways of choosing $b - c$ from the shared bits between p_l and p_i : $\binom{\text{Hamming}(p_l, p_i)}{c} \binom{M - \text{Hamming}(p_l, p_i)}{b-c}$.

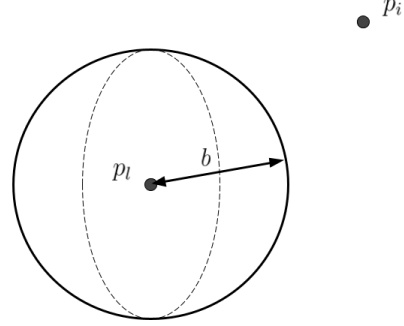


Figure 6: Sphere of p_j that are b bits away from p_l . For illustration only.

Conveniently, $\binom{q}{g} = 0$ for $g > q$, so there is no need to add special cases to exclude the cases where the number we seek to choose is greater than the number available.

Equation (43) enables computational complexity reduction of eq. (41), including the outer loop over l , from exponential to polynomial time: $\mathcal{O}(N^2 2^M)$ to $\mathcal{O}(N^2 + M^3)$ (excluding computation of u). The M^3 comes from constructing lookup table z , with the cubed power coming from looping over all possible values for $\text{Hamming}(p_l, p_i)$, b and c . This subsumes M^2 to compute $\forall q \in [0, M], \forall g \in [0, M] : \binom{q}{g}$ in the same iterative manner as eq. (39).

C Experimental details

Experiments were averaged over 5 random seeds. The classification models were trained with SGD optimization with learning rate 0.1, momentum 0.9, weight decay 5e-4 and standard schedules: 100 epochs with learning rate *0.1 every 30 epochs (CIFAR10) and 200 epochs with learning rate *0.2 every 60 epochs (CIFAR100). For each unreliability metric, each threshold yielded one set

of accept/reject decisions for the test set which yielded one point in each evaluation graph, and area under graph was computed using the trapezoidal rule. For the in-distribution setting, AUCEA corresponds to effective model accuracy (i.e. of accepted samples) averaged over different thresholds, and thus can be compared against original model accuracy.

All code and experimental commands are provided.

C.1 Subfunction error bound hyperparameters

The searched values for likelihood parameter δ were [0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] and for kernel width parameter ρ were [32, 48, 64, 98, 128, 192, 256, 384, 512] (VGG16) and [4, 6, 8, 12, 16, 24, 32, 48, 64] (ResNet50). Validation set AUCEA was used for selection (algorithm 1). The selected values are in table 6.

Dataset	Model	ρ	δ
CIFAR100	VGG16	128.0 (#: 5)	0.001 (#: 5)
CIFAR100	ResNet50	16.0 (#: 5)	0.001 (#: 4), 0.1 (#: 1)
CIFAR10	VGG16	48.0 (#: 3), 98.0 (#: 1), 64.0 (#: 1)	0.001 (#: 4), 0.9 (#: 1)
CIFAR10	ResNet50	16.0 (#: 3), 24.0 (#: 2)	0.001 (#: 4), 0.3 (#: 1)

Table 6: Hyperparameters used for subfunction error. Brackets denote number of seeds.

C.2 Dataset statistics

All results infer unreliability of the test sets. The datasets are publicly available.

Dataset	Train	Validation	Test
CIFAR10 (Krizhevsky, 2009)	42500	7500	10000
CIFAR100 (Krizhevsky, 2009)	42500	7500	10000
SVHN (Netzer et al., 2011)	62269	10988	26032

C.3 Computational resources

Experiments were run given a shared cluster of machines with approximately 140 GPUs. Jobs required less than 24GB GPU memory. For subfunction error, normalization constants (section 4) were computed first in a pre-computation phase. This took up most of the runtime and was parallelized by splitting jobs by architecture, dataset, seed and ρ ; each job took approximately 20 minutes. Subsequent inference on the test sets (i.e. computing unreliability of unseen samples) took approximately 2 - 10 minutes per combination of architecture, dataset and seed.

D Additional results: Boxplots for other methods

Settings apart from the method are the same as fig. 5.

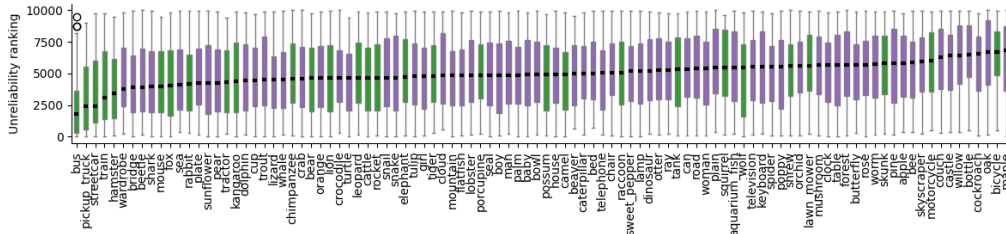
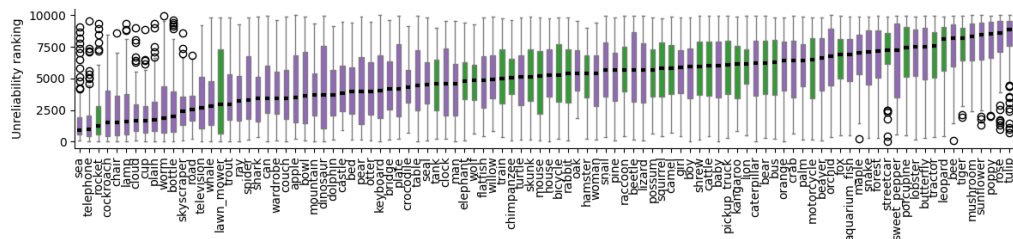
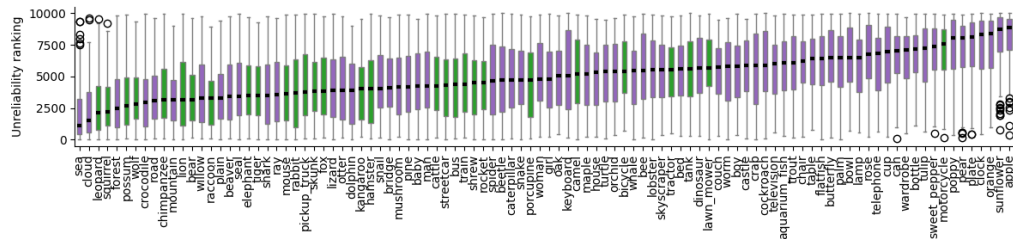
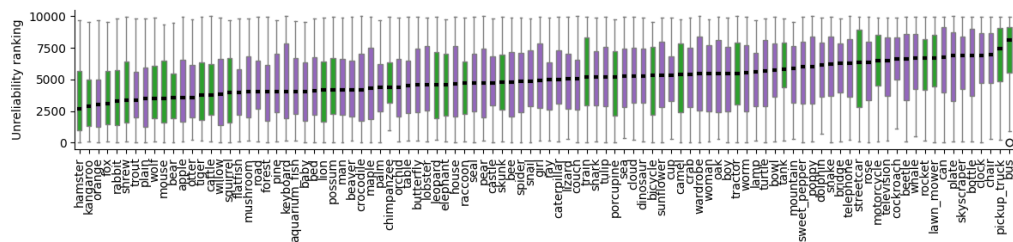
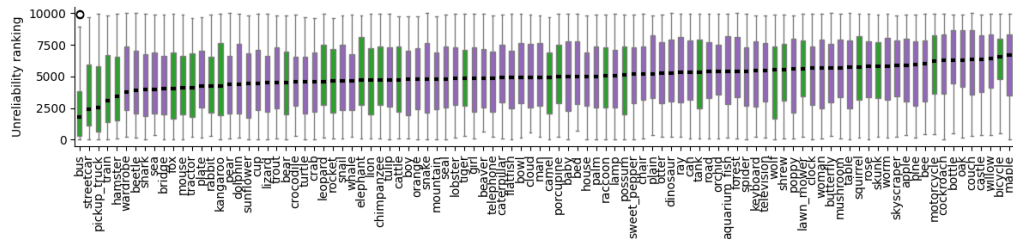
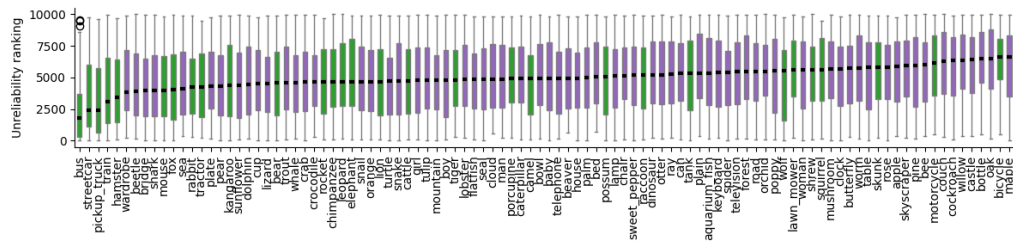


Figure 7: Entropy.



Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

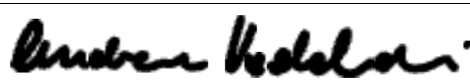
Title of Paper	Predicting Unreliable Predictions by Shattering a Neural Network
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Xu Ji, Razvan Pascanu, Devon Hjelm, Andrea Vedaldi, Balaji Lakshminarayanan, Yoshua Bengio (2021). Predicting Unreliable Predictions by Shattering a Neural Network. Under review.

Student Confirmation

Student Name:	Xu Ji		
Contribution to the Paper	The initial idea is mine, most of the technical contribution is mine, including all formal derivations. I performed all experiments. I wrote the first draft of the paper and then worked on it in conjunction with mostly Razvan Pascanu; Devon Hjelm, Andrea Vedaldi, Balaji Lakshminarayanan and Yoshua Bengio all helped with giving feedback and direction. In particular, Razvan Pascanu, Andrea Vedaldi and Balaji Lakshminarayanan helped with in-depth feedback on theory, experiments, and writing.		
Signature		Date	24/05/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Andrea Vedaldi			
Supervisor comments The description is accurate.			
Signature		Date	25/05/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

4

Invariant Information Clustering for Unsupervised Image Classification and Segmentation

This paper was published in ICCV 2019.

Invariant Information Clustering for Unsupervised Image Classification and Segmentation

Xu Ji
University of Oxford
xuji@robots.ox.ac.uk

João F. Henriques
University of Oxford
joao@robots.ox.ac.uk

Andrea Vedaldi
University of Oxford
vedaldi@robots.ox.ac.uk

Abstract

We present a novel clustering objective that learns a neural network classifier from scratch, given only unlabelled data samples. The model discovers clusters that accurately match semantic classes, achieving state-of-the-art results in eight unsupervised clustering benchmarks spanning image classification and segmentation. These include STL10, an unsupervised variant of ImageNet, and CIFAR10, where we significantly beat the accuracy of our closest competitors by 6.6 and 9.5 absolute percentage points respectively. The method is not specialised to computer vision and operates on any paired dataset samples; in our experiments we use random transforms to obtain a pair from each image. The trained network directly outputs semantic labels, rather than high dimensional representations that need external processing to be usable for semantic clustering. The objective is simply to maximise mutual information between the class assignments of each pair. It is easy to implement and rigorously grounded in information theory, meaning we effortlessly avoid degenerate solutions that other clustering methods are susceptible to. In addition to the fully unsupervised mode, we also test two semi-supervised settings. The first achieves 88.8% accuracy on STL10 classification, setting a new global state-of-the-art over all existing methods (whether supervised, semi-supervised or unsupervised). The second shows robustness to 90% reductions in label coverage, of relevance to applications that wish to make use of small amounts of labels. github.com/xu-ji/IIC

1. Introduction

Most supervised deep learning methods require large quantities of manually labelled data, limiting their applicability in many scenarios. This is true for large-scale image classification and even more for segmentation (pixel-wise classification) where the annotation cost per image is very high [38, 21]. Unsupervised clustering, on the other hand, aims to group data points into classes entirely

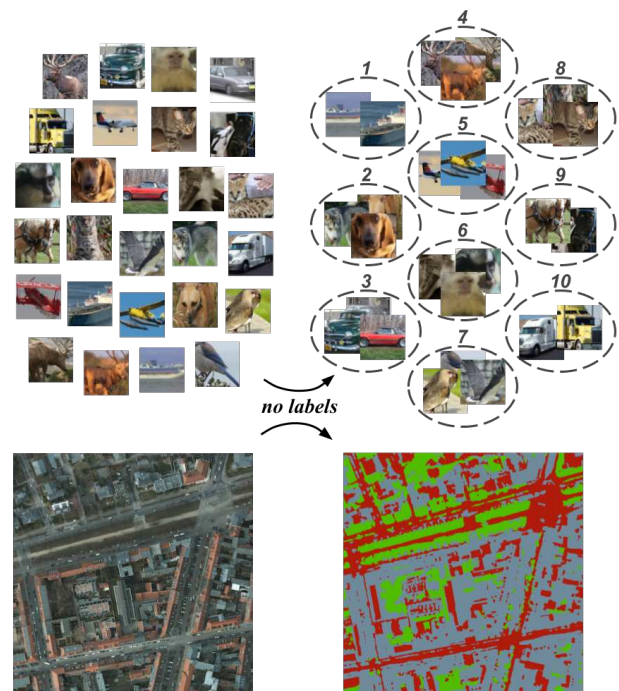


Figure 1: Models trained with IIC on entirely unlabelled data learn to cluster images (top, STL10) and patches (bottom, Potsdam-3). The raw clusters found directly correspond to semantic classes (dogs, cats, trucks, roads, vegetation etc.) with state-of-the-art accuracy. Training is end-to-end and randomly initialised, with no heuristics used at any stage.

without labels [25]. Many authors have sought to combine mature clustering algorithms with deep learning, for example by bootstrapping network training with k-means style objectives [51, 24, 7]. However, trivially combining clustering and representation learning methods often leads to degenerate solutions [7, 51]. It is precisely to prevent such degeneracy that cumbersome pipelines — involving pre-training, feature post-processing (whitening or PCA), clustering mechanisms external to the network — have evolved [7, 17, 18, 51].

In this paper, we introduce Invariant Information Clustering (IIC), a method that addresses this issue in a more principled manner. IIC is a generic clustering algorithm that

directly trains a randomly initialised neural network into a classification function, end-to-end and without any labels. It involves a simple objective function, which is the mutual information between the function’s classifications for paired data samples. The input data can be of any modality and, since the clustering space is discrete, mutual information can be computed exactly.

Despite its simplicity, IIC is intrinsically robust to two issues that affect other methods. The first is clustering degeneracy, which is the tendency for a single cluster to dominate the predictions or for clusters to disappear (which can be observed with k-means, especially when combined with representation learning [7]). Due to the entropy maximisation component within mutual information, the loss is not minimised if all images are assigned to the same class. At the same time, it is optimal for the model to predict for each image a single class with certainty (i.e. one-hot) due to the conditional entropy minimisation (fig. 3). The second issue is noisy data with unknown or distractor classes (present in STL10 [10] for example). IIC addresses this issue by employing an auxiliary output layer that is parallel to the main output layer, trained to produce an overclustering (i.e. same loss function but greater number of clusters than the ground truth) that is ignored at test time. Auxiliary overclustering is a general technique that could be useful for other algorithms. These two features of IIC contribute to making it the only method amongst our unsupervised baselines that is robust enough to make use of the noisy unlabelled subset of STL10, a version of ImageNet [14] specifically designed as a benchmark for unsupervised clustering.

In the rest of the paper, we begin by explaining the difference between semantic clustering and intermediate representation learning (section 2), which separates our method from the majority of work in unsupervised deep learning. We then describe the theoretical foundations of IIC in statistical learning (section 3), demonstrating that maximising mutual information between pairs of samples under a bottleneck is a principled clustering objective which is equivalent to distilling their shared abstract content (co-clustering). We propose that for static images, an easy way to generate pairs with shared abstract content from unlabelled data is to take each image and its random transformation, or each patch and a neighbour. We show that maximising MI automatically avoids degenerate solutions and can be written as a convolution in the case of segmentation, allowing for efficient implementation with any deep learning library.

We perform experiments on a large number of datasets (section 4) including STL, CIFAR, MNIST, COCO-Stuff and Potsdam, setting a new state-of-the-art on unsupervised clustering and segmentation in all cases, with results of 59.6%, 61.7% and 72.3% on STL10, CIFAR10 and COCO-Stuff-3 beating the closest competitors (53.0%, 52.2%, 54.0%) with significant margins. Note that train-

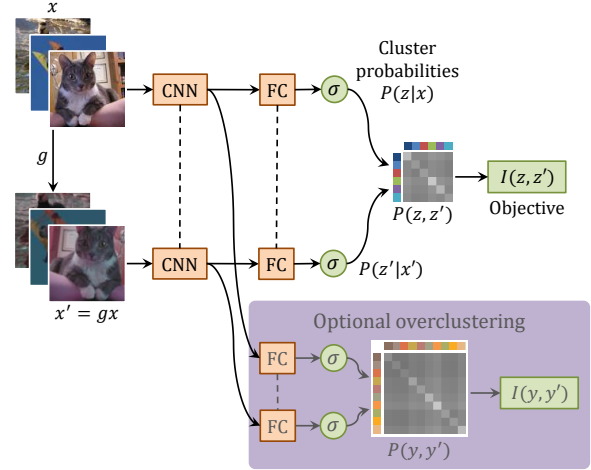


Figure 2: IIC for image clustering. Dashed line denotes shared parameters, g is a random transformation, and I denotes mutual information (eq. (3)).

ing deep neural networks to perform large scale, real-world segmentations from scratch, without labels or heuristics, is a highly challenging task with negligible precedent. We also perform an ablation study and additionally test two semi-supervised modes, setting a new global state-of-the-art of 88.8% on STL10 over all supervised, semi-supervised and unsupervised methods, and demonstrating the robustness in semi-supervised accuracy when 90% of labels are removed.

2. Related work

Co-clustering and mutual information. The use of information as a criterion to learn representations is not new. One of the earliest works to do so is by Becker and Hinton [3]. More generally, learning from paired data has been explored in co-clustering [25, 16] and in other works [50] that build on the information bottleneck principle [20].

Several recent papers have used information as a tool to train deep networks in particular. IMSAT [28] maximises mutual information between data and its representation and DeepINFOMAX [27] maximizes information between spatially-preserved features and compact features. However, IMSAT and DeepINFOMAX combine information with other criteria, whereas in our method information is the only criterion used. Furthermore, both IMSAT and DeepINFOMAX compute mutual information over continuous random variables, which requires complex estimators [4], whereas IIC does so for discrete variables with simple and exact computations. Finally, DeepINFOMAX considers the information $I(\mathbf{x}, f(\mathbf{x}))$ between the features $\mathbf{x}$ and a deterministic function $f(\mathbf{x})$ of it, which is in principle the same as the entropy $H(\mathbf{x})$; in contrast, in IIC information does not trivially reduce to entropy.

Semantic clustering versus intermediate representation learning. In semantic clustering, the learned function directly outputs discrete assignments for high level (i.e.

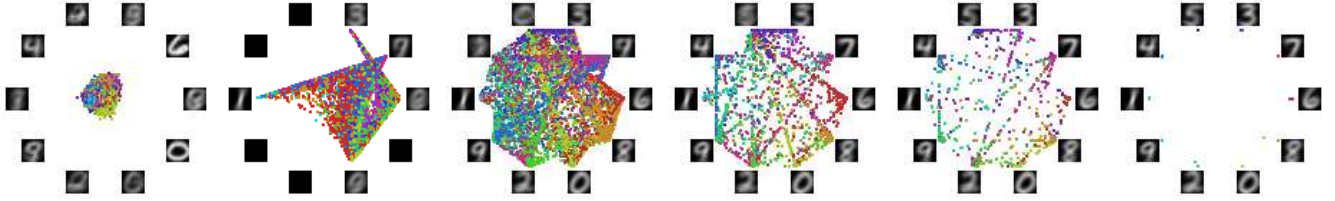


Figure 3: Training with IIC on unlabelled MNIST in successive epochs from random initialisation (left). The network directly outputs cluster assignment probabilities for input images, and each is rendered as a coordinate by convex combination of 10 cluster vertices. There is no cherry-picking as the entire dataset is shown in every snapshot. Ground truth labelling (unseen by model) is given by colour. At each cluster the average image of its assignees is shown. With neither labels nor heuristics, the clusters discovered by IIC correspond perfectly to unique digits, with one-hot certain prediction (right).

semantic) clusters. Intermediate representation learners, on the other hand, produce continuous, distributed, high-dimensional representations that must be post-processed, for example by k-means, to obtain the discrete low-cardinality assignments required for unsupervised semantic clustering. The latter includes objectives such as generative autoencoder image reconstruction [48], triplets [46] and spatial-temporal order or context prediction [37, 12, 17], for example predicting patch proximity [30], solving jigsaw puzzles [41] and inpainting [43]. Note it also includes a number of clustering methods (DeepCluster [7], exemplars [18]) where the clustering is only auxiliary; a clustering-style objective is used but does not produce groups with semantic correspondence. For example, DeepCluster [7] is a state-of-the-art method for learning highly-transferable intermediate features using overclustering as a proxy task, but does not automatically find semantically meaningful clusters. As these methods use auxiliary objectives divorced from the semantic clustering objective, it is unsurprising that they perform worse than IIC (section 4), which directly optimises for it, training the network end-to-end with the final clusterer implicitly wrapped inside.

Optimising image-to-image distance. Many approaches to deep clustering, whether semantic or auxiliary, utilise a distance function between input images that approximates a given grouping criterion. Agglomerative clustering [2] and partially ordered sets [1] of HOG features [13] have been used to group images, and exemplars [18] define a group as a set of random transformations applied to a single image. Note the latter does not scale easily, in particular to image segmentation where a single 200×200 image would call for 40k classes. DAC [8], JULE [52], DeepCluster [7], ADC [24] and DEC [51] rely on the inherent visual consistency and disentangling properties [23] of CNNs to produce cluster assignments, which are processed and reinforced in each iteration. The latter three are based on k-means style mechanisms to refine feature centroids, which is prone to degenerate solutions [7] and thus needs explicit prevention mechanisms such as pre-training, cluster-reassignment or feature cleaning via PCA and whitening [51, 7].

Invariance as a training objective. Optimising for function outputs to be persistent through spatio-temporal or

non-material distortion is an idea shared by IIC with several works, including exemplars [18], IMSAT [28], proximity prediction [30], the denoising objective of Tagger [22], temporal slowness constraints [55], and optimising for features to be invariant to local image transformations [47, 29]. More broadly, the problem of modelling data transformation has received significant attention in deep learning, one example being the transforming autoencoder [26].

3. Method

First we introduce a generic objective, Invariant Information Clustering, which can be used to cluster any kind of unlabelled paired data by training a network to predict cluster identities (section 3.1). We then apply it to image clustering (section 3.2, fig. 2 and fig. 3) and segmentation (section 3.3), by generating the required paired data using random transformations and spatial proximity.

3.1. Invariant Information Clustering

Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ be a paired data sample from a joint probability distribution $P(\mathbf{x}, \mathbf{x}')$. For example, $\mathbf{x}$ and $\mathbf{x}'$ could be different images containing the same object. The goal of Invariant Information Clustering (IIC) is to learn a representation $\Phi : \mathcal{X} \rightarrow \mathcal{Y}$ that preserves what is in common between $\mathbf{x}$ and $\mathbf{x}'$ while discarding instance-specific details. The former can be achieved by maximizing the mutual information between encoded variables:

$$\max_{\Phi} I(\Phi(\mathbf{x}), \Phi(\mathbf{x}')), \quad (1)$$

which is equivalent to maximising the predictability of $\Phi(\mathbf{x})$ from $\Phi(\mathbf{x}')$ and vice versa.

An effect of equation eq. (1), in general, is to make representations of paired samples the same. However, it is not the same as merely minimising representation distance, as done for example in methods based on k-means [7, 24]: the presence of entropy within I allows us to avoid degeneracy, as discussed in detail below.

If Φ is a neural network with a small output capacity (often called a “bottleneck”), eq. (1) also has the effect of discarding instance-specific details from the data. Clustering imposes a natural bottleneck, since the representation

space is $\mathcal{Y} = \{1, \dots, C\}$, a finite set of class indices (as opposed to an infinite vector space). Without a bottleneck, i.e. assuming unbounded capacity, eq. (1) is trivially solved by setting Φ to the identity function because of the data processing inequality [11], i.e. $I(\mathbf{x}, \mathbf{x}') \geq I(\Phi(\mathbf{x}), \Phi(\mathbf{x}'))$.

Since our goal is to learn the representation with a deep neural network, we consider soft rather than hard clustering, meaning the neural network Φ is terminated by a (differentiable) softmax layer. Then the output $\Phi(\mathbf{x}) \in [0, 1]^C$ can be interpreted as the distribution of a discrete random variable z over C classes, formally given by $P(z = c | \mathbf{x}) = \Phi_c(\mathbf{x})$. Making the output probabilistic amounts to allowing for uncertainty in the cluster assigned to an input.

Consider now a pair of such cluster assignment variables z and z' for two inputs $\mathbf{x}$ and $\mathbf{x}'$ respectively. Their conditional joint distribution is given by $P(z = c, z' = c' | \mathbf{x}, \mathbf{x}') = \Phi_c(\mathbf{x}) \cdot \Phi_{c'}(\mathbf{x}')$. This equation states that z and z' are independent when conditioned on specific inputs $\mathbf{x}$ and $\mathbf{x}'$; however, in general they are *not* independent after marginalization over a dataset of input pairs $(\mathbf{x}_i, \mathbf{x}'_i)$, $i = 1, \dots, n$. For example, for a trained classification network Φ and a dataset of image pairs where each image contains the same object of its pair but in a randomly different position, the random variable constituted by the class of the first of each pair, z , will have a strong statistical relationship with the random variable for the class of the second of each pair, z' ; one is predictive of the other (in fact identical to it, in this case) so they are highly dependent. After marginalization over the dataset (or batch, in practice), the joint probability distribution is given by the $C \times C$ matrix $\mathbf{P}$, where each element at row c and column c' constitutes $\mathbf{P}_{cc'} = P(z = c, z' = c')$:

$$\mathbf{P} = \frac{1}{n} \sum_{i=1}^n \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}'_i)^\top. \quad (2)$$

The marginals $\mathbf{P}_c = P(z = c)$ and $\mathbf{P}_{c'} = P(z' = c')$ can be obtained by summing over the rows and columns of this matrix. As we generally consider symmetric problems, where for each $(\mathbf{x}_i, \mathbf{x}'_i)$ we also have $(\mathbf{x}'_i, \mathbf{x}_i)$, $\mathbf{P}$ is symmetrized using $(\mathbf{P} + \mathbf{P}^\top)/2$.

Now the objective function eq. (1) can be computed by plugging the matrix $\mathbf{P}$ into the expression for mutual information [36], which results in the formula:

$$I(z, z') = I(\mathbf{P}) = \sum_{c=1}^C \sum_{c'=1}^C \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_c \cdot \mathbf{P}_{c'}}. \quad (3)$$

Why degenerate solutions are avoided. Mutual information (3) expands to $I(z, z') = H(z) - H(z|z')$. Hence, maximizing this quantity trades-off minimizing the conditional cluster assignment entropy $H(z|z')$ and maximising individual cluster assignments entropy $H(z)$. The smallest value of $H(z|z')$ is 0, obtained when the cluster assignments are exactly predictable from each other. The

largest value of $H(z)$ is $\ln C$, obtained when all clusters are equally likely to be picked. This occurs when the data is assigned evenly between the clusters, equalizing their mass. Therefore the loss is not minimised if all samples are assigned to a single cluster (i.e. output class is identical for all samples). Thus as maximising mutual information naturally balances reinforcement of predictions with mass equalization, it avoids the tendency for degenerate solutions that algorithms which combine k-means with representation learning are susceptible to [7]. For further discussion of entropy maximisation, and optionally how to prioritise it with an entropy coefficient, see supplementary material.

Meaning of mutual information. The reader may now wonder what are the benefits of maximising mutual information, as opposed to merely maximising entropy. Firstly, due to the soft clustering, entropy alone could be maximised trivially by setting all prediction vectors $\Phi(\mathbf{x})$ to uniform distributions, resulting in no clustering. This is corrected by the conditional entropy component, which encourages deterministic one-hot predictions. For example, even for the degenerate case of identical pairs $\mathbf{x} = \mathbf{x}'$, the IIC objective encourages a deterministic clustering function (i.e. $\Phi(\mathbf{x})$ is a one-hot vector) as this results in null conditional entropy $H(z|z') = 0$. Secondly, the objective of IIC is to find what is common between two data points that share redundancy, such as different images of the same object, explicitly encouraging distillation of the common part while ignoring the rest, i.e. instance details specific to one of the samples. This would not be possible without pairing samples.

3.2. Image clustering

IIC requires a source of paired samples $(\mathbf{x}, \mathbf{x}')$, which are often unavailable in unsupervised image clustering applications. In this case, we propose to use *generated* image pairs, consisting of image $\mathbf{x}$ and its *randomly perturbed* version $\mathbf{x}' = g\mathbf{x}$. The objective eq. (1) can thus be written as:

$$\max_{\Phi} I(\Phi(\mathbf{x}), \Phi(g\mathbf{x})), \quad (4)$$

where both image $\mathbf{x}$ and transformation g are random variables. Useful g could include scaling, skewing, rotation or flipping (geometric), changing contrast and colour saturation (photometric), or any other perturbation that is likely to leave the content of the image intact. IIC can then be used to recover the factor which is *invariant* to which of the pair is picked. The effect is to learn a function that partitions the data such that clusters are closed to the perturbations, without dropping clusters. The objective is simple enough to be written in six lines of PyTorch code (fig. 4).

Auxiliary overclustering. For certain datasets (e.g. STL10), training data comes in two types: one known to contain only relevant classes and the other known to contain irrelevant or distractor classes. It is desirable to train a

```

def IIC(z, zt, C=10):
    P = (z.unsqueeze(2) * zt.unsqueeze(1)).sum(dim=0)
    P = ((P + P.t()) / 2) / P.sum()
    P[(P < EPS).data] = EPS
    Pi = P.sum(dim=1).view(C, 1).expand(C, C)
    Pj = P.sum(dim=0).view(1, C).expand(C, C)
    return (P * (log(Pi) + log(Pj) - log(P))).sum()

```

Figure 4: IIC objective in PyTorch. Inputs z and zt are $n \times C$ matrices, with C predicted cluster probabilities for n sampled pairs (i.e. CNN softmax predictions). For example, the prediction for each image in a dataset and its transformed version (e.g. using standard data augmentation).

clusterer specialised for the relevant classes, that still benefits from the context provided by the distractor classes, since the latter is often much larger (for example 100K compared to 13K for STL10). Our solution is to add an auxiliary overclustering head to the network (fig. 2) that is trained with the full dataset, whilst the main output head is trained with the subset containing only relevant classes. This allows us to make use of the noisy unlabelled subset despite being an unsupervised clustering method. Other methods are generally not robust enough to do so and thus avoid the 100k-samples unlabelled subset of STL10 when training for unsupervised clustering ([8, 24, 51]). Since the auxiliary overclustering head outputs predictions over a larger number of clusters than the ground truth, whilst still maintaining a predictor that is matched to ground truth number of clusters (the main head), it can be useful in general for increasing expressivity in the learned feature representation, even for datasets where there are no distractor classes [7].

3.3. Image segmentation

IIC can be applied to image segmentation identically to image clustering, except for two modifications. Firstly, since predictions are made for each pixel densely, clustering is applied to image patches (defined by the receptive field of the neural network for each output pixel) rather than whole images. Secondly, unlike with whole images, one has access to the spatial relationships between patches. Thus, we can add *local spatial invariance* to the list of geometric and photometric invariances in section 3.2, meaning we form pairs of patches not only via synthetic perturbations, but also by extracting pairs of adjacent patches in the image.

In detail, let the RGB image $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$ be a tensor, $u \in \Omega = \{1, \dots, H\} \times \{1, \dots, W\}$ a pixel location, and $\mathbf{x}_u$ a patch centered at u . We can form a pair of patches $(\mathbf{x}_u, \mathbf{x}_{u+t})$ by looking at location u and its neighbour $u+t$ at some small displacement $t \in T \subset \mathbb{Z}^2$. The cluster probability vectors for all patches $\mathbf{x}_u$ can be read off as the column vectors $\Phi(\mathbf{x}_u) = \Phi_u(\mathbf{x}) \in [0, 1]^C$ of the tensor $\Phi(\mathbf{x}) \in [0, 1]^{C \times H \times W}$, computed by a single application of the convolutional network Φ . Then, to apply IIC, one simply substitutes pairs $(\Phi_u(\mathbf{x}), \Phi_{u+t}(\mathbf{x}))$ in the calculation of the joint probability matrix (2).

The geometric and photometric perturbations used be-

fore for whole image clustering can be applied to individual patches too. Rather than transforming patches individually, however, it is much more efficient to transform all of them in parallel by perturbing the entire image. Any number or combination of these invariances can be chained and learned simultaneously; the only detail is to ensure indices of the original image and transformed image class probability tensors line up, meaning that predictions from patches which are intended to be paired together do so.

Formally, if the image transformation g is a geometric transformation, the vector of cluster probabilities $\Phi_u(\mathbf{x})$ will not correspond to $\Phi_u(g\mathbf{x})$; rather, it will correspond to $\Phi_{g(u)}(g\mathbf{x})$ because patch $\mathbf{x}_u$ is sent to patch $\mathbf{x}_{g(u)}$ by the transformation. All vectors can be paired at once by applying the reverse transformation g^{-1} to the tensor $\Phi(g\mathbf{x})$, as $[g^{-1}\Phi(g\mathbf{x})]_u = \Phi_{g(u)}(g\mathbf{x})$. For example, flipping the input image will require flipping the resulting probability tensor back. In general, the perturbation g can incorporate geometric and photometric transformations, and g^{-1} only needs to undo geometric ones. The segmentation objective is thus:

$$\max_{\Phi} \frac{1}{|T|} \sum_{t \in T} I(\mathbf{P}_t), \quad (5)$$

$$\mathbf{P}_t = \frac{1}{n|G||\Omega|} \sum_{i=1}^n \sum_{g \in G} \sum_{u \in \Omega} \overbrace{\Phi_u(\mathbf{x}_i) \cdot [g^{-1}\Phi(g\mathbf{x}_i)]_{u+t}^\top}^{\text{Convolution}}.$$

Hence the goal is to maximize the information between each patch label $\Phi_u(\mathbf{x}_i)$ and the patch label $[g^{-1}\Phi(g\mathbf{x}_i)]_{u+t}$ of its transformed neighbour patch, in expectation over images $i = 1, \dots, n$, patches $u \in \Omega$ within each image, and perturbations $g \in G$. Information is in turn averaged over all neighbour displacements $t \in T$ (which was found to perform slightly better than averaging over t before computing information; see supplementary material).

Implementation. The joint distribution of eq. (5) for all displacements $t \in T$ can be computed in a simple and highly efficient way. Given two network outputs for one batch of image pairs $\mathbf{y} = \Phi(\mathbf{x}), \mathbf{y}' = \Phi(g\mathbf{x})$ where $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^{n \times C \times H \times W}$, we first bring $\mathbf{y}'$ back into the coordinate-space of $\mathbf{y}$ by using a bilinear resampler¹ [32], which inverts any geometrical transforms in g , $\mathbf{y}' \leftarrow g^{-1}\mathbf{y}'$. Then, the inner summation in eq. (5) reduces to the convolution of the two tensors. Using any standard deep learning framework, this can be achieved by swapping the first two dimensions of each of y and y' , computing $\mathbf{P} = y * y'$ (a 2D convolution with padding d in both dimensions), and normalising the result to produce $\mathbf{P} \in [0, 1]^{C \times C \times (2d+1) \times (2d+1)}$.

4. Experiments

We apply IIC to fully unsupervised image clustering and segmentation, as well as two semi-supervised settings. Ex-

¹The core differentiable operator in spatial transformer networks [32].

	STL10	CIFAR10	CFR100-20	MNIST
Random network	13.5	13.1	5.93	26.1
K-means [53]†	19.2	22.9	13.0	57.2
Spectral clustering [49]	15.9	24.7	13.6	69.6
Triplets [46]‡	24.4	20.5	9.94	52.5
AE [5]‡	30.3	31.4	16.5	81.2
Sparse AE [40]‡	32.0	29.7	15.7	82.7
Denoising AE [48]‡	30.2	29.7	15.1	83.2
Variational Bayes AE [34]‡	28.2	29.1	15.2	83.2
SWWAE 2015 [54]‡	27.0	28.4	14.7	82.5
GAN 2015 [45]‡	29.8	31.5	15.1	82.8
JULE 2016 [52]	27.7	27.2	13.7	96.4
DEC 2016 [51]†	35.9	30.1	18.5	84.3
DAC 2017 [8]	47.0	52.2	23.8	97.8
DeepCluster 2018 [7]† ‡	33.4*	37.4*	18.9*	65.6 *
ADC 2018 [24]	53.0	32.5	16.0*	99.2
IIC (lowest loss sub-head)	59.6	61.7	25.7	99.2
IIC (avg sub-head $\pm$ STD)	59.8 ± 0.844	57.6 ± 5.01	25.5 ± 0.462	98.4 ± 0.652

Table 1: **Unsupervised image clustering.** Legend: †Method based on k-means. ‡Method that does not directly learn a clustering function and requires further application of k-means to be used for image clustering. *Results obtained using our experiments with authors’ original code.

	STL10
No auxiliary overclustering	43.8
Single sub-head ($h = 1$)	57.6
No sample repeats ($r = 1$)	47.0
Unlabelled data segment ignored	49.9
Full setting	59.6

Table 2: **Ablations of IIC (unsupervised setting).** Each row shows a single change from the full setting. The full setting has auxiliary overclustering, 5 initialisation heads, 5 sample repeats, and uses the unlabelled data subset of STL10.

isting baselines are outperformed in all cases. We also conduct an analysis of our method via ablation studies. For minor details see supplementary material.

4.1. Image clustering

Datasets. We test on STL10, which is ImageNet adapted for unsupervised classification, as well as CIFAR10, CIFAR100-20 and MNIST. The main setting is pure unsupervised clustering (IIC) but we also test two semi-supervised settings: *finetuning* and *overclustering*. For unsupervised clustering, following previous work [8, 51, 52], we train on the full dataset and test on the labelled part; for the semi-supervised settings, train and test sets are separate.

As for DeepCluster [7], we found Sobel filtering to be beneficial, as it discourages clustering based on trivial cues such as colour and encourages using more meaningful cues such as shape. Additionally, for data augmentation, we repeat images within each batch r times; this means that multiple image pairs within a batch contain the same original image, each paired with a different transformation, which encourages greater distillation since there are more examples of which visual details to ignore (section 3.1). We set $r \in [1, 5]$ for all experiments. Images are rescaled and cropped for training (prior to applying trans-

forms g , consisting of random additive and multiplicative colour transformations and horizontal flipping) and a single center crop is used at test time for all experiments except semi-supervised finetuning, where 10 crops are used.

Architecture. All networks are randomly initialised and consist of a ResNet or VGG11-like base b (see sup. mat.), followed by one or more heads (linear predictors). Let the number of ground truth clusters be k_{gt} and the output channels of a head be k . For IIC, there is a main output head with $k = k_{gt}$ and an auxiliary overclustering head (fig. 2) with $k > k_{gt}$. For semi-supervised overclustering there is one output head with $k > k_{gt}$. For increased robustness, each head is duplicated $h = 5$ times with a different random initialisation, and we call these concrete instantiations sub-heads. Each sub-head takes features from b and outputs a probability distribution for each batch element over the relevant number of clusters. For semi-supervised finetuning (table 3), the base is copied from a semi-supervised overclustering network and combined with a single randomly initialised linear layer where $k = k_{gt}$.

Training. We use the Adam optimiser [33] with learning rate 10^{-4} . For IIC, the main and auxiliary heads are trained by maximising eq. (3) in alternate epochs. For semi-supervised overclustering, the single head is trained by maximising eq. (3). Semi-supervised finetuning uses a standard logistic loss.

Evaluation. We evaluate based on accuracy (true positives divided by sample size). For IIC we follow the standard protocol of finding the best one-to-one permutation mapping between learned and ground-truth clusters (from the main output head; auxiliary overclustering head is ignored) using linear assignment [35]. While this step uses labels, it does not constitute learning as it merely makes the metric invariant to the order of the clusters. For semi-supervised overclustering, each ground-truth cluster may correspond to the union of several predicted clusters. Evaluation thus requires a many-to-one discrete map from k to k_{gt} , since $k > k_{gt}$. This extracts some information from the labels and thus requires separated training and test set. Note this mapping is found using the training set (accuracy is computed on the test set) and does not affect the network parameters as it is used for evaluation only. For semi-supervised finetuning, output channel order matches ground truth so no mapping is required. Each sub-head is assessed independently; we report average and best sub-head (as chosen by lowest IIC loss) performance.

Unsupervised learning analysis. IIC is highly capable of discovering clusters in unlabelled data that accurately correspond to the underlying semantic classes, and outperforms all competing baselines at this task (table 1), with significant margins of 6.6% and 9.5% in the case of STL10 and CI-

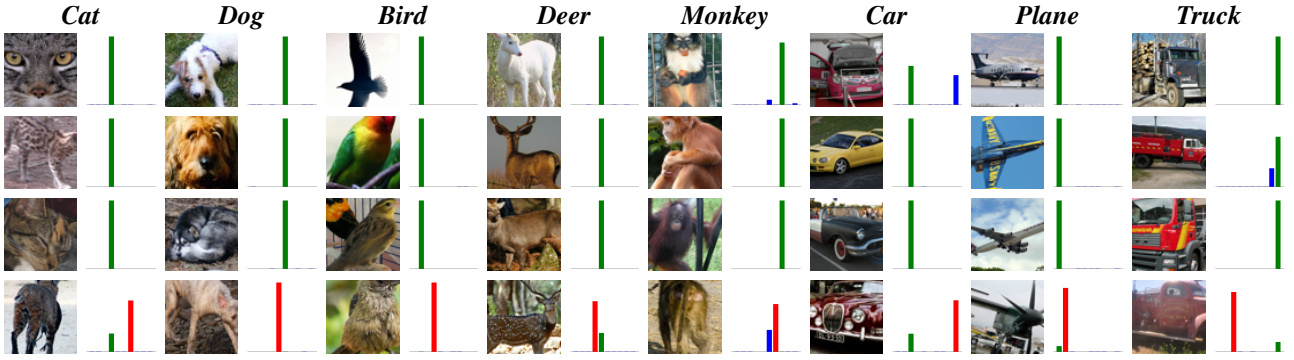


Figure 5: **Unsupervised image clustering (IIC) results on STL10.** Predicted cluster probabilities from the best performing head are shown as bars. Prediction corresponds to tallest, ground truth is green, incorrectly predicted classes are red, and all others are blue. The bottom row shows failure cases.

	STL10
Dosovitskiy 2015 [18]†	74.2
SWWAE 2015 [54]†	74.3
Dundar 2015 [19]	74.1
Cutout* 2017 [15]	87.3
Oyallon* 2017 [42]†	76.0
Oyallon* 2017 [42]	87.6
DeepCluster 2018 [7]	73.4*
ADC 2018 [24]	56.7*
DeepINFOMAX 2018 [27]	77.0
IIC plus finetune†	79.2
IIC plus finetune	88.8

Table 3: **Fully and semi-supervised classification.** Legend: *Fully supervised method. †Our experiments with authors’ code. ‡Multi-fold evaluation.

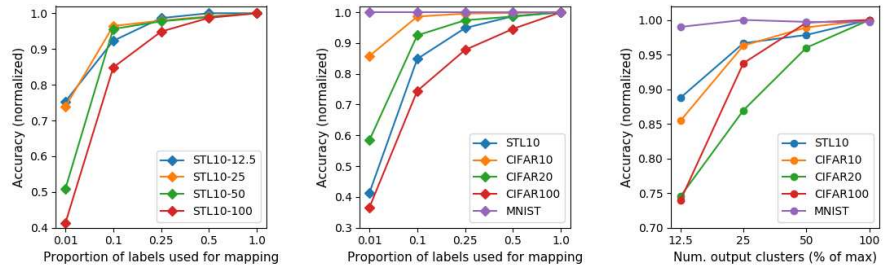


Figure 6: **Semi-supervised overclustering.** Training with IIC loss to overcluster ($k > k_{gt}$) and using labels for evaluation mapping only. Performance is robust even with 90%-75% of labels discarded (left and center). STL10- r denotes networks with output $k = \lceil 1.4r \rceil$. Overall accuracy improves with the number of output clusters k (right). For further details see supplementary material.

FAR10. As mentioned in section 2, this underlines the advantages of end-to-end optimisation instead of using a fixed external procedure like k-means as with many baselines. The clusters found by IIC are highly discriminative (fig. 5), although note some failure cases; as IIC distills purely visual correspondences within images, it can be confused by instances that combine classes, such as a deer with the coat pattern of a cat. Our ablations (table 2) illustrate the contributions of various implementation details, and in particular the accuracy gain from using auxiliary overclustering.

Semi-supervised learning analysis. For semi-supervised learning, we establish a new state-of-the-art on STL10 out of all reported methods by finetuning a network trained in an entirely unsupervised fashion with the IIC objective (recall labels in semi-supervised overclustering are used for evaluation and do not influence the network parameters). This explicitly validates the quality of our unsupervised learning method, as we beat even the supervised state-of-the-art (table 3). Given that the bulk of parameters within semi-supervised overclustering are trained unsupervised (i.e. all network parameters), it is unsurprising that Figure 6 shows a 90% drop in the number of available labels for STL10 (decreasing the amount of labelled data available from 5000 to 500 over 10 classes) barely impacts performance, costing just $\sim 10\%$ drop in accuracy. This setting has lower label requirements than finetuning because whereas the latter learns all network parameters, the former

only needs to learn a discrete map between k and k_{gt} , making it an important practical setting for applications with small amounts of labelled data.

4.2. Segmentation

Datasets. Large scale segmentation on real-world data using deep neural networks is extremely difficult without labels or heuristics, and has negligible precedent. We establish new baselines on scene and satellite images to highlight performance on textural classes, where the assumption of spatially proximal invariance (section 3.3) is most valid. COCO-Stuff [6] is a challenging and diverse segmentation dataset containing “stuff” classes ranging from buildings to bodies of water. We use the 15 coarse labels and 164k images variant, reduced to 52k by taking only images with at least 75% stuff pixels. COCO-Stuff-3 is a subset of COCO-Stuff with only sky, ground and plants labelled. For both COCO datasets, input images are shrunk by two thirds and cropped to 128×128 pixels, Sobel pre-processing is applied for data augmentation, and predictions for non-stuff pixels are ignored. Potsdam [31] is divided into 8550 RGBIR 200×200 px satellite images, of which 3150 are unlabelled. We test both the 6-label variant (roads and cars, vegetation and trees, buildings and clutter) and a 3-label variant (Potsdam-3) formed by merging each of the 3 pairs. All segmentation training and testing sets have been released with our code.

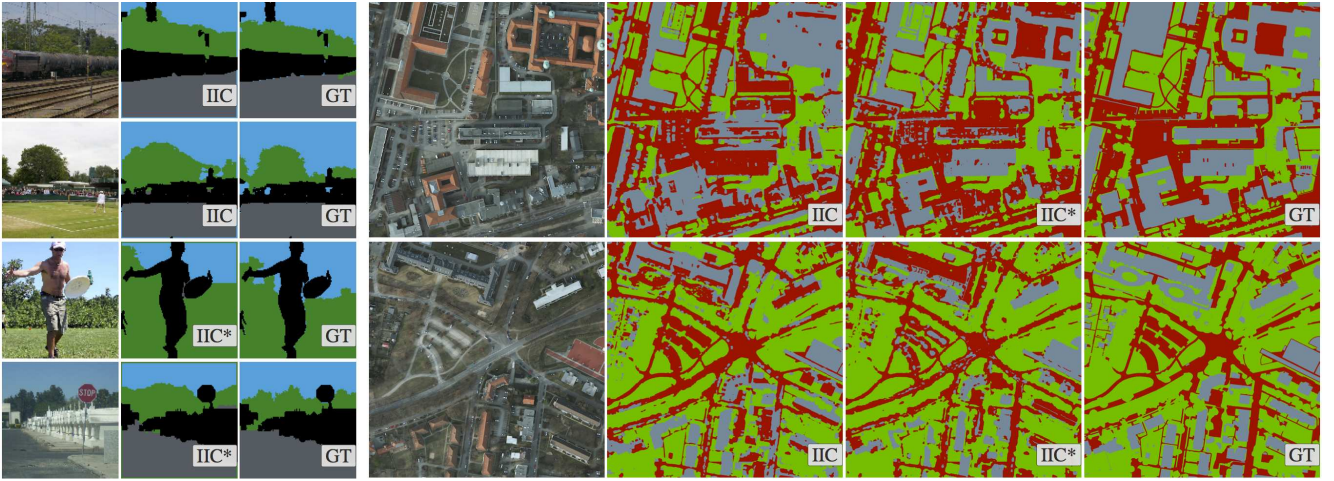


Figure 7: **Example segmentation results (un- and semi-supervised).** Left: COCO-Stuff-3 (non-stuff pixels in black), right: Potsdam-3. Input images, IIC (fully unsupervised segmentation) and IIC* (semi-supervised overclustering) results are shown, together with the ground truth segmentation (GT).

	COCO-Stuff-3	COCO-Stuff	Potsdam-3	Potsdam
Random CNN	37.3	19.4	38.2	28.3
K-means [44] [†]	52.2	14.1	45.7	35.3
SIFT [39] [‡]	38.1	20.2	38.2	28.5
Doersch 2015 [17] [‡]	47.5	23.1	49.6	37.2
Isola 2016 [30] [‡]	54.0	24.3	63.9	44.9
DeepCluster 2018 [7] ^{† ‡}	41.6	19.9	41.7	29.2
IIC	72.3	27.7	65.1	45.4

Table 4: **Unsupervised segmentation.** IIC experiments use a single sub-head. Legend: [†]Method based on k-means. [‡]Method that does not directly learn a clustering function and requires further application of k-means to be used for image clustering.

Architecture. All networks are randomly initialised and consist of a base CNN b (see sup. mat.) followed by head(s), which are 1×1 convolution layers. Similar to section 4.1, overclustering uses k 3-5 times higher than k_{gt} . Since segmentation is much more expensive than image clustering (e.g. a single 200×200 Potsdam image contains 40,000 predictions), all segmentation experiments were run with $h = 1$ and $r = 1$ (sec. 4.1).

Training. The convolutional implementation of IIC (eq. (5)) was used with $d = 10$. For Potsdam-3 and COCO-Stuff-3, the optional entropy coefficient (section 3.1 and sup. mat.) was used and set to 1.5. Using the coefficient made slight improvements of 1.2%-3.2% on performance. These two datasets are balanced in nature with very large sample volume (e.g. $40,000 \times 75$ predictions per batch for Potsdam-3) resulting in stable and balanced batches, justifying prioritisation of equalisation. Other training details are the same as section 4.1.

Evaluation. Evaluation uses accuracy as in section 4.1, computed per-pixel. For the baselines, the original authors' code was adapted from image clustering where available, and the architectures are shared with IIC for fairness. For baselines that required application of k-means to produce per-pixel predictions (table 4), k-means was trained with randomly sampled pixel features from the training set (10M

for Potsdam, Potsdam-3; 50M for COCO-Stuff, COCO-Stuff-3) and tested on the full test set to obtain accuracy.

Analysis. Without labels or heuristics to learn from, and given just the cluster cardinality (3), IIC automatically partitions COCO-Stuff-3 into clusters that are recognisable as sky, vegetation and ground, and learns to classify vegetation, roads and buildings for Potsdam-3 (fig. 7). The segmentations are notably intricate, capturing fine detail, but are at the same time locally consistent and coherent across all images. Since spatial smoothness is built into the loss (section 3.3), all our results are able to use raw network outputs without post-processing (avoiding e.g. CRF smoothing [9]). Quantitatively, we outperform all baselines (table 4), notably by 18.3% in the case of COCO-Stuff-3. The efficient convolutional formulation of the loss (eq. (5)) allows us to optimise over all pixels in all batch images in parallel, converging in fewer epochs (passes of the dataset) without paying the price of reduced computational speed for dense sampling. This is in contrast to our baselines which, being not natively adapted for segmentation, required sampling a subset of pixels within each batch, resulting in increased loss volatility and training speeds that were up to $3.3\times$ slower than IIC.

5. Conclusions

We have shown that it is possible to train neural networks into semantic clusterers without using labels or heuristics. The novel objective presented relies on statistical learning, by optimising mutual information between related pairs - a relationship that can be generated by random transforms - and naturally avoids degenerate solutions. The resulting models classify and segment images with state-of-the-art levels of semantic accuracy. Being not specific to vision, the method opens up many interesting research directions, including optimising information in datastreams over time.

Acknowledgments. We are grateful to ERC StG IDIU-638009 and EPSRC AIMS CDT for support.

References

- [1] Miguel A Bautista, Artsiom Sanakoyeu, and Bjorn Ommer. Deep unsupervised similarity learning using partially ordered sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7130–7139, 2017. [3](#)
- [2] Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Tikhoncheva, and Bjorn Ommer. Cliquesnn: Deep unsupervised exemplar learning. In *Advances in Neural Information Processing Systems*, pages 3846–3854, 2016. [3](#)
- [3] Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161, 1992. [2](#)
- [4] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R Devon Hjelm, and Aaron Courville. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018. [2](#)
- [5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007. [6](#)
- [6] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. *arXiv preprint arXiv:1612.03716*, 2016. [7](#)
- [7] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *arXiv preprint arXiv:1807.05520*, 2018. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#), [8](#)
- [8] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5879–5887, 2017. [3](#), [5](#), [6](#)
- [9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. [8](#)
- [10] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011. [2](#)
- [11] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. [4](#)
- [12] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. Deeppermnet: Visual permutation learning. *arXiv preprint arXiv:1704.02729*, 2017. [3](#)
- [13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. [3](#)
- [14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. [2](#)
- [15] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. [7](#)
- [16] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003. [2](#)
- [17] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. [1](#), [3](#), [8](#)
- [18] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015. [1](#), [3](#), [7](#)
- [19] Aysegul Dundar, Jonghoon Jin, and Eugenio Culurciello. Convolutional clustering for unsupervised learning. *arXiv preprint arXiv:1511.06241*, 2015. [7](#)
- [20] Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 152–161. Morgan Kaufmann Publishers Inc., 2001. [2](#)
- [21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. [1](#)
- [22] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Juergen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, pages 4484–4492, 2016. [3](#)
- [23] Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. Binding via reconstruction clustering. *arXiv preprint arXiv:1511.06418*, 2015. [3](#)
- [24] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*, pages 18–32. Springer, 2018. [1](#), [3](#), [5](#), [6](#), [7](#)
- [25] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972. [1](#), [2](#)
- [26] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011. [3](#)
- [27] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018. [2](#), [7](#)
- [28] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. *arXiv preprint arXiv:1702.08720*, 2017. [2](#), [3](#)

- [29] Ka Yu Hui. Direct modeling of complex invariances for visual object features. In *International Conference on Machine Learning*, pages 352–360, 2013. 3
- [30] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015. 3, 8
- [31] ISPRS. ISPRS 2D Semantic Labeling Contest. <http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html>. [Online; accessed 10-October-2018]. 7
- [32] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 5
- [33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6
- [34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 6
- [35] Harold W Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008*, pages 29–47. Springer, 2010. 6
- [36] Erik G Learned-Miller. Entropy and mutual information. 4
- [37] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 667–676. IEEE, 2017. 3
- [38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755. Springer, 2014. 1
- [39] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 8
- [40] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, pages 1–19, 2011. 6
- [41] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 3
- [42] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5618–5627, 2017. 7
- [43] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016. 3
- [44] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011. 8
- [45] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 6
- [46] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48, 2004. 3, 6
- [47] Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. *arXiv preprint arXiv:1206.6418*, 2012. 3
- [48] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010. 3, 6
- [49] Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian k-means. *IEEE Transactions on Knowledge & Data Engineering*, (1):1–1, 2015. 6
- [50] Pu Wang, Carlotta Domeniconi, and Kathryn Blackmond Laskey. Information bottleneck co-clustering. Citeseer. 2
- [51] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016. 1, 3, 5, 6
- [52] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016. 3, 6
- [53] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005. 6
- [54] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 6, 7
- [55] Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y Ng. Deep learning of invariant features via simulated fixations in video. In *Advances in neural information processing systems*, pages 3203–3211, 2012. 3

Invariant Information Clustering for Unsupervised Image Classification and Segmentation: Supplementary Material

Xu Ji
University of Oxford
xuji@robots.ox.ac.uk

João F. Henriques
University of Oxford
joao@robots.ox.ac.uk

Andrea Vedaldi
University of Oxford
vedaldi@robots.ox.ac.uk

1. Release

We implemented IIC in PyTorch [7]. The code, datasets and trained models have been released. github.com/xu-ji/IIC

2. Further experimental details

We used three generic CNN bases b across our experiments: A (ResNet34 [5]), B (4 convolutional layers) and C (6 convolutional layers). For details see table 1. See table 2 for per-experiment details including b , batch size, input channels, input size, and number of clusters used in overclustering denoted by k . Recall the latter refers to the sole output head for semi-supervised overclustering but to the auxiliary head for unsupervised IIC, where the main head produces output with dimensionality k_{gt} . For segmentation, bilinear resampling is used to resize the network output back to input size for implementational simplicity. Since there is one pooling layer in network C which halves spatial size, this is by a factor of 2.

A	B	C
1×Conv@64	1× Conv@64	1× Conv@64
3×BasicBlock@64	1× MaxPool	1× Conv@128
4×BasicBlock@128	1× Conv@128	1× MaxPool
6×BasicBlock@256	1× MaxPool	2× Conv@256
3×BasicBlock@512	1× Conv@256	2× Conv@512
1×AvgPool	1× MaxPool	
	1× Conv@512	

Table 1: Architecture bases b , showing layer type and output channels. Pooling layers do not change channel size. Convolutional layers have filter size 3 or 5 and stride 1 or 2. The models used are standard ResNet and VGG-style networks. Implementations are given in the code.

3. Semi-supervised overclustering study

Paper fig. 6 contains accuracies normalised by dividing by the maximum accuracy for each series. The absolute accuracies are given in table 3 and table 4.

		b	n	h	r	k_{in}	k_{gt}	k	crop size(s)	input size
IIC	STL10	A	700	5	5	2	10	70	64	64
	CIFAR10	A	660	5	3	2	10	70	20	32
	CIFAR100-20	A	1000	5	5	2	20	140	20	32
	MNIST	B	700	5	5	1	10	50	16, 20, 24	24
	COCO-Stuff-3	C	120	1	1	5	3	15	128	128
	COCO-Stuff	C	60	1	1	5	15	45	128	128
	Potsdam-3	C	75	1	1	4	3	24	200	200
	Potsdam	C	60	1	1	4	6	36	200	200
	STL10	A	1400	5	5	2	10	140	64	64
	CIFAR10	A	1320	5	3	2	10	140	20	32
IIC*	CIFAR100-20	B	2800	5	5	5	20	280	20	24
	MNIST	B	350	5	5	1	10	25	16, 20, 24	24
	COCO-Stuff-3	C	180	1	1	5	3	15	128	128
	COCO-Stuff	C	90	1	1	5	15	45	128	128
	Potsdam-3	C	75	1	1	4	3	9	200	200
	Potsdam	C	60	1	1	4	6	24	200	200

Table 2: IIC denotes unsupervised clustering, IIC* denotes semi-supervised overclustering. n denotes batch size, h and r denote number of sub-heads and sample repeats (see paper section 4.1), k_{in} denotes input channels (1 for greyscale, 2 for Sobel filtered, 4 for RGBIR, 5 for Sobel filtered with RGB), k_{gt} denotes number of ground truth clusters, k denotes number of output channels for overclustering. COCO-Stuff and COCO-Stuff-3 are scaled by 0.33 prior to cropping; cropped images are scaled to final input size with bilinear resampling.

4. Rendering predictions

To generate the visualisation in paper fig. 3, the entire MNIST dataset was run through each network snapshot. The prediction for each image x , say $z = \Phi(x) \in [0, 1]^C$ for C classes (see paper section 3.1), was rendered as a point with coordinate position p :

$$p = \left[\sum_{c=1}^C z_c \cdot \sin\left(\frac{2\pi c}{C}\right), \sum_{c=1}^C z_c \cdot \cos\left(\frac{2\pi c}{C}\right) \right].$$

	STL10		CIFAR10		CIFAR100-20		CIFAR100		MNIST	
% of max k	k	ACC	k	ACC	k	ACC	k	ACC	k	ACC
100	140	63.1	140	65.0	280	34.7	1000	20.3	100	98.6
50	70	61.4	70	62.2	140	33.1	500	20.3	50	98.6
25	35	59.7	35	60.5	70	30.0	250	19.1	25	98.7
12.5	18	54.8	18	53.7	35	25.7	125	15.0	13	97.9

Table 3: Absolute accuracy for semi-supervised overclustering experiments in paper fig. 6-right.

STL10	1.0		0.5		0.25		0.1		0.01	
% of max k	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC
100	5000	63.1	2500	61.0	1250	58.6	500	52.4	50	25.5
50	5000	61.4	2500	59.8	1250	59.1	500	57.8	50	30.7
25	5000	59.7	2500	59.2	1250	58.5	500	57.6	50	44.1
12.5	5000	54.8	2500	54.8	1250	54.1	500	50.6	50	41.3

	1.0		0.5		0.25		0.1		0.01	
	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC	n_a	ACC
STL10	5000	63.1	2500	61.0	1250	58.6	500	52.4	50	25.5
CIFAR10	50000	62.9	25000	62.7	12500	62.6	5000	62.0	500	53.9
CIFAR100-20	50000	34.5	25000	34.0	12500	33.6	5000	31.9	500	20.1
CIFAR100	50000	20.3	25000	19.2	12500	17.9	5000	15.1	500	7.43
MNIST-25	60000	98.9	30000	98.9	15000	98.9	6000	98.9	600	98.9

Table 4: Absolute accuracy for semi-supervised overclustering experiments in paper fig. 6-left (top) and fig. 6-center (bottom). n_a denotes number of labels used to find mapping from output k to k_{gt} for evaluation.

5. Optional entropy coefficient

Consider inserting a coefficient, $\lambda \geq 1$, into the definition of mutual information (eq. 3, paper section 3.1):

$$I_\lambda(z, z') = \sum_{c=1}^C \sum_{c'=1}^C \mathbf{P}_{cc'} \cdot \ln \frac{\mathbf{P}_{cc'}}{\mathbf{P}_c^\lambda \cdot \mathbf{P}_{c'}^\lambda} \quad (1)$$

$$= I_1(z, z') + (\lambda - 1) \cdot (H(z) + H(z')). \quad (2)$$

For $\lambda = 1$, this reduces to the standard mutual information definition. However, inserting an exponent of $\lambda > 1$ into the denominator of (1) translates into prioritising the maximisation of prediction entropy (2).

6. Expectation over all shifts $t \in T$

Recall that IIC for segmentation involves maximising mutual information between a patch and all its neighbours within local box given by T (paper section 3.3). An alternative formulation of paper eq. (5) would involve bringing the expectation over T within the computation for information as follows:

$$\max_{\Phi} I(\mathbf{P}),$$

$$\mathbf{P} = \frac{1}{n|T||G||\Omega|} \sum_{i=1}^n \sum_{t \in T} \sum_{g \in G} \sum_{u \in \Omega} \overbrace{\Phi_u(\mathbf{x}_i) \cdot [g^{-1}\Phi(g\mathbf{x}_i)]_{u+t}^\top}^{\text{Convolution}}.$$

We found paper eq. (5) to work marginally but consistently better, for example by 0.1% for COCO-Stuff-3 and 0.02% for Potsdam-3. This is likely because closer neighbours are more informative than farther ones, and an external expectation avoids entangling the signal between close and far neighbours prior to computing mutual information.

7. Random transformations g

Horizontal flipping, random crops and random colour changes in hue, saturation and brightness constitute the g used in most of our experiments. We also tried random affine transforms but found our models performed better

without them, as the presence of skew and scaling materially affected the network’s ability to distill visual correspondences between pairs of images.

8. Dataset sizes

For the sizes of the training and testing sets used in our experiments, see table 5 and table 6.

	STL10		CIFAR10		CIFAR100-20		MNIST	
	Train	Test	Train	Test	Train	Test	Train	Test
IIC	113k	13k	60k	60k	60k	60k	70k	70k
Semi-supervised	105k	8k	50k	10k	50k	10k	60k	10k

Table 5: Datasets for image clustering.

	COCO-Stuff-15		COCO-Stuff-3		Potsdam-6		Potsdam-3	
	Train	Test	Train	Test	Train	Test	Train	Test
IIC	51804	51804	36660	36660	8550	5400	8550	5400
Semi-supervised	49629	2175	35228	1432	7695	855	7695	855

Table 6: Datasets for segmentation.

9. Baseline experiments

DeepCluster [1], also originally implemented in PyTorch, was adapted from the released image clustering code for both purely unsupervised image clustering and segmentation. Since this is not the intended task for the method, DeepCluster was used as a feature learner, with k-means performed on learned feature representations in order to obtain cluster assignments for evaluation. Data augmentation transforms are used as with IIC, the same b as IIC is used for each model’s feature representation, and the number of output clusters is set to $10 \times k_{gt}$ as suggested by the paper. The feature descriptor lengths range from 4096 (image clustering) to 512 (segmentation). For image clustering, the k-means procedures at training and test time are both trained and evaluated on the full training and test sets respectively. For segmentation, since all descriptors for the training set cannot fit in RAM (needed not only for the implementation of k-means, but also for the PCA dimensionality reduction) it was necessary to use sampling for k-means both during computation of the pseudolabels for training, and evaluation. This was done with 10M and 50M samples for Potsdam* and COCO-Stuff* datasets respectively. Once the k-means centroids were obtained, training still occurred over

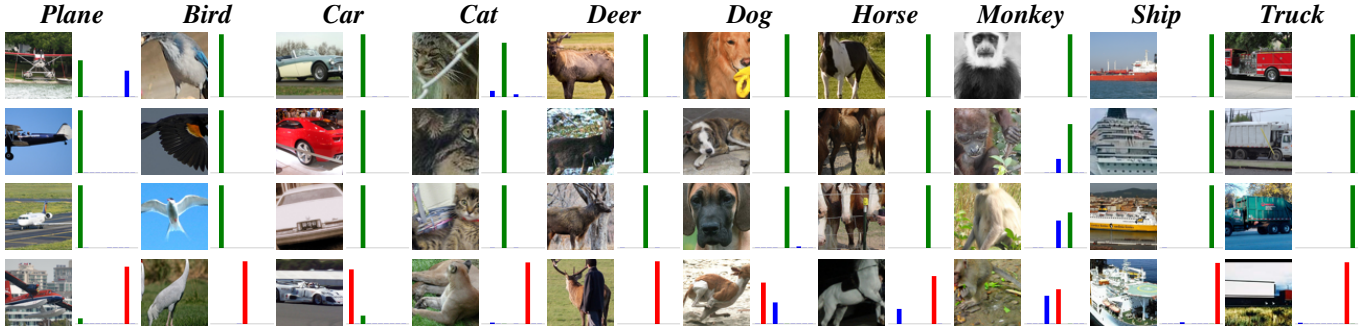


Figure 1: Additional unsupervised clustering (IIC) results on STL10. Predicted cluster probabilities shown as bars. Prediction corresponds to tallest, ground truth is green, incorrectly predicted classes are red, and all others are blue. The bottom row shows failure cases.

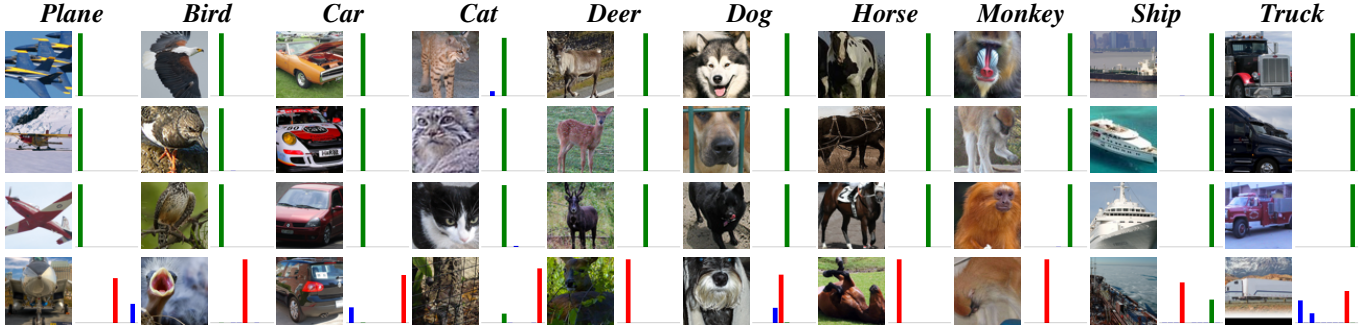


Figure 2: Semi-supervised overclustering results on STL10. Predicted cluster probabilities shown as bars. Prediction corresponds to tallest, ground truth is green, incorrectly predicted classes are red, and all others are blue. The bottom row shows failure cases.



Figure 3: Additional unsupervised segmentation (IIC) results on COCO-Stuff-3 (non-stuff pixels in black). Left to right for each triplet: image, prediction, ground truth.



Figure 4: Additional semi-supervised clustering for segmentation results on COCO-Stuff-3 (non-stuff pixels in black). Left to right for each triplet: image, prediction, ground truth.

the entire training set with accuracy computed over the entire test set. For the semi-supervised experiment, finetuning of the learned representation was used, as with IIC.

ADC [4], originally implemented in TensorFlow, was adapted from the released code for image clustering only. For the fully unsupervised CIFAR100-20 experiment (paper table 1), since ADC was already implemented for CIFAR100, we adopted the existing architecture and training settings for CIFAR100 when training CIFAR100-20.

Similarly, we adopted the existing architecture and settings included for STL10 for the semi-supervised experiment, training an SVM on top of fixed features as this is the semi-supervised implementation provided in their code.

Triplets [8] was implemented as a representation learner by setting the positive example for each image to be its random transform, using the same transformations as the IIC experiments for fairness. The negative example for each image was set to a randomly selected image. K-means was run



Figure 5: Additional segmentation results for unsupervised IIC and semi-supervised overclustering on Potsdam-3. Left to right for each quadruplet: image, IIC prediction, semi-supervised overclustering prediction, ground truth.

on the learned embeddings to obtain cluster assignments.

For segmentation baselines Isola [6] and Doersch [3], which are unsupervised feature learning methods without segmentation code, we use our own implementation. Since both operate by predicting the spatial relationship between pairs of patches (spatial proximity and exact relative position respectively), we adapted them to segmentation by randomly sampling pairs from the dense features produced by b (which are either close or far for Isola, for example), using additional linear layers to predict the spatial relationship, minimising the distance between this prediction and known ground truth, and backpropagating gradients end-to-end.

10. Further discussion of evaluation

For image clustering we use $h = 5$ sub-heads, which are different randomly initialised instantiations of the same final layer, for increased robustness (paper section 4.1). We evaluate the accuracy of the model by identifying the sub-head with the lowest training loss on the main output head and reporting its test set accuracy. This evaluation procedure makes no use of labels at all as the sub-head is selected using the unsupervised loss. In table 7, we additionally report the test set accuracy of the sub-head with the highest training set accuracy, which illustrates the maximum achieved score by the sub-heads. For IIC, since the training and test sets are the same (unlabelled data notwithstanding), the latter is equivalent to running an identical training procedure h times, selecting a different sub-head each time, and reporting best performance out of these runs.

	STL10	CIFAR10	CFR100-20	MNIST
IIC (best sub-head)	61.0	61.7	25.7	99.3

Table 7: Image clustering accuracy of best sub-head.

11. Further discussion of avoiding degeneracy

Here we continue the discussion on how IIC avoids undesirable solutions (paper section 3.1). Recall that maximising mutual information entails maximising entropy and minimising conditional entropy. Consider the malevolent

case of all images being assigned the ambivalent prediction $[\frac{1}{C}, \dots, \frac{1}{C}]$ for C clusters. For example, for $C = 2$, if for both datapoints in all pairs the model predicts $[0.5, 0.5]$, then $\mathbf{P} = [[0.25, 0.25], [0.25, 0.25]]^T$. Conditional entropy $H(z|z')$ would not be minimised to 0 in this case, as the predictions are not deterministic (one-hot) [2]. To achieve $H(z|z') = 0$, predictions would need to be $[0, 1]$ or $[1, 0]$. Thus as IIC minimises $H(z|z')$, it avoids ambivalent solutions.

On the other hand, if all images are assigned to the same cluster i , the joint distribution would be all-zero except at $\mathbf{P}_{ii} = 1$. Likewise, the marginals would be all-zero except at $\mathbf{P}_i = 1$. Entropy $H(z)$ would not be maximised, and indeed would be minimised to $H(z) = 0$ as:

$$H(z) = - \sum_{c=1}^C \mathbf{P}_c \cdot \ln \mathbf{P}_c. \quad (3)$$

Thus as IIC maximises $H(z)$, it also avoids this degenerate solution.

12. Additional examples

For more examples of image clustering and segmentation results for both unsupervised IIC and semi-supervised overclustering, see fig. 1, fig. 2, fig. 3, fig. 4 and fig. 5.

References

- [1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *arXiv preprint arXiv:1807.05520*, 2018. 2
- [2] Thomas M Cover and Joy A Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991. 4
- [3] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 4
- [4] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*, pages 18–32. Springer, 2018. 3
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1
- [6] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015. 4
- [7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 1
- [8] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48, 2004. 3

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	Invariant Information Clustering for Unsupervised Image Classification and Segmentation
Publication Status	☒ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Xu Ji, Joao Henriques, Andrea Vedaldi. Invariant Information Clustering for Unsupervised Image Classification and Segmentation. Published in Proceedings of the International Conference on Computer Vision (ICCV) 2019.

Student Confirmation

Student Name:	Xu Ji		
Contribution to the Paper	The initial idea was developed by all three authors together. I performed all experiments. I wrote the first draft of the paper and then all three authors edited it together. I have been mostly responsible for follow-up, including reader questions and the online code repository.		
Signature		Date	24/05/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Andrea Vedaldi			
Supervisor comments The description is accurate.			
Signature		Date	25/05/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

5

There and Back Again: Revisiting Backpropagation Saliency Methods

This paper was published in CVPR 2020.

There and Back Again: Revisiting Backpropagation Saliency Methods

Sylvestre-Alvise Rebuffi* Ruth Fong* Xu Ji* Andrea Vedaldi

Visual Geometry Group, University of Oxford

{srebuffi, ruthfong, xuji, vedaldi}@robots.ox.ac.uk

Abstract

Saliency methods seek to explain the predictions of a model by producing an importance map across each input sample. A popular class of such methods is based on backpropagating a signal and analyzing the resulting gradient. Despite much research on such methods, relatively little work has been done to clarify the differences between such methods as well as the desiderata of these techniques. Thus, there is a need for rigorously understanding the relationships between different methods as well as their failure modes. In this work, we conduct a thorough analysis of backpropagation-based saliency methods and propose a single framework under which several such methods can be unified. As a result of our study, we make three additional contributions. First, we use our framework to propose NormGrad, a novel saliency method based on the spatial contribution of gradients of convolutional weights. Second, we combine saliency maps at different layers to test the ability of saliency methods to extract complementary information at different network levels (e.g. trading off spatial resolution and distinctiveness) and we explain why some methods fail at specific layers (e.g., Grad-CAM anywhere besides the last convolutional layer). Third, we introduce a class-sensitivity metric and a meta-learning inspired paradigm applicable to any saliency method for improving sensitivity to the output class being explained.

1. Introduction

The adoption of deep learning methods by high-risk applications, such as healthcare and automated driving, gives rise to a need for tools that help machine learning practitioners understand model behavior. Given the highly-parameterized, opaque nature of deep neural networks, developing such methods is non-trivial, and there are many possible approaches. In the basic case, even the predictions of the model itself, either unaltered or after being distilled into a simpler function [11, 3], can be used to shed light on

its behaviour.

Saliency is the specific branch of interpretability concerned with determining not what the behaviour of a model is for whole input samples, but which parts of samples contribute the most to that behaviour. Thus by definition, determining saliency - or attribution - necessarily involves reversing the model's inference process in some manner [20]. Propagating a signal from the output layer of a neural network model back to the input layer is one way of explicitly achieving this.

The number of diverse works based on using signal backpropagation for interpretability in computer vision [37, 27, 4, 38] is testimony to the power of this simple principle. Typically, these techniques produce a heatmap for any given input image that ranks its pixels according to some metric of importance for the model's decision. Inspired by the work of [1], we propose to delve deeper into such methods by discussing some of the similarities, differences and potential improvements that can be illustrated.

We begin with presenting a framework that unifies several backpropagation-based saliency methods by dividing the process of generating a saliency map into two phases: *extraction* of the contribution of the gradient of network parameters at each spatial location in a particular network layer, and *aggregation* of such spatial information into a 2D heatmap. GradCAM [27], linear approximation [17] and gradient [29] can all be cast as such processes. Noting that no appropriate technique has yet been proposed for properly aggregating contributions from convolutional layers, we introduce NormGrad, which uses the Frobenius norm for aggregation. We introduce identity layers to allow for the generation of saliency heatmaps at all spatially-grounded layers in the network (i.e. even after ReLU), since NormGrad computes saliency given a parameterised network layer.

We conduct a thorough analysis of backpropagation-based saliency methods in general, with evaluation based on utilising saliency heatmaps for weak localisation. Notably, we conduct an investigation into simple techniques for combining saliency maps taken from different network layers - in contrast to the popular practice of computing maps at the input layer [29] - and find that using a weighted aver-

*indicates equal contribution

age of maps from all layers consistently improves performance for several saliency methods, compared to taking the single best layer. However, not all layers are equally important, as we discover that models optimised on datasets such as ImageNet [26] and PASCAL VOC [7] learn features that become increasingly *class insensitive* closer towards the input layer. This provides an explanation for why Grad-CAM [27] produces unmeaningful saliency heatmaps at certain layers earlier than the last convolutional layer, as the sensitivity of the gradient to class across spatial locations is eliminated by Grad-CAM’s spatial gradient averaging, meaning such layers are devoid of class sensitive signals from which to form saliency heatmaps.

Finally, building off [20, 1], we introduce a novel metric for quantifying the class sensitivity of a saliency method, and present a meta-learning inspired paradigm that increases the class sensitivity of any method by adding an inner SGD step into the computation of the saliency heatmap.

2. Related work

Saliency methods. Our work focuses on backpropagation-based saliency methods; these techniques are computationally efficient as they only require one forward and backward pass through a network. One of the earliest methods was [29] which visualised the gradient at the input with respect to an output class being explained. Several authors have since proposed adaptations in order to improve the heatmap’s visual quality. These include modifying the ReLU derivatives (Deconvnet [37], guided backprop [32]) and averaging over randomly perturbed inputs (SmoothGrad [31]) to produce masks with reduced noise. Several works have explored visualizing saliency at intermediate layers by combining information from activations and gradients, notably CAM [39], Grad-CAM [27], and linear approximation (a.k.a. gradient $\times$ input) [17]. Conservation-preserving methods (Excitation Backprop [38], LRP [4], and DeepLIFT [28]) modify the backward functions of network layers in order to “preserve” an attribution signal such that it sums to one at any point in the network. Reference-based methods average over attributions from multiple interpolations [33] between the input and a non-informative (e.g. black) reference input or use a single reference input with which to compare a backpropagated attribution signal [28].

Although we focus on backpropagation-based methods, another class of methods studies the effects that perturbations on the input induce on the output. [37] and [24] generate saliency maps by weighting input occlusion patterns by the induced changes in model output. [10, 9, 15] learn for saliency maps that maximally impact the model, and [5] trains a model to predict effective maps. LIME [25] learns linear weights that correspond to the effect of including or excluding (via perturbation) different image patches in an

image. Perturbation-based approaches have also been used to perform object localisation [30, 35, 34].

Assessing and unifying saliency methods. A few works have studied if saliency methods have certain desired sensitivities (e.g., to specific model weights [1] or the output class being explained [20]) and if they are invariant to unmeaningful factors (e.g., constant shift in input intensity [16]). [20] showed that gradient [29], deconvnet [37], and guided backprop [32] tend to produce class insensitive heatmaps. [1] introduced sanity check metrics that measure how sensitive a saliency method is to model weights by reporting the correlation between a saliency map on a trained model vs. a partially randomized model.

Other works quantify the utility of saliency maps for weak localization [38, 10] and for impacting model predictions. [38] introduced Pointing Game, which measures the correlation between the maximal point extracted from a saliency map with pixel-level semantic labels. [10] extracts bounding boxes from saliency maps and measures their agreement with ground truth bounding boxes. [36] evaluates attribution methods using relative feature importance. [22] proposes a dataset designed for measuring visual explanation accuracy. [4, 24, 15] present variants of a perturbation-based evaluation metric that measures the impact of perturbing (or unperturbing for [15]) image patches in order of importance as given by a saliency map. However, these perturbed images are outside the training domain; [13] mitigates this by measuring the performance of classifiers re-trained on perturbed images (i.e., with 20% of pixels perturbed).

To our knowledge, the only work that has been done to unify saliency methods focuses primarily on “invasive” techniques that change backpropagation rules. The α -LRP variant [4] and Excitation Backprop [38] share the backpropagation rule, and DeepLIFT [28] is equivalent to LRP when 0 is used as the reference activation throughout a network. [19] unifies a few methods (e.g., LIME [25], LRP [4], DeepLIFT [28]) under the framework of additive feature attribution.

3. Method

Preliminaries. Consider a training set $\mathcal{D}$ of pairs $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$ are (color) images and $\mathbf{y} \in \{1, \dots, C\}$ their labels. Furthermore, let $\mathbf{y} = \Phi_{\theta}(\mathbf{x})$ be the output of a neural network model whose parameters θ are optimized using the cross-entropy loss ℓ to predict labels from images.

3.1. Extract & Aggregate framework

In most methods, the saliency map is obtained as a function of the network activations, computed in a forward pass,

and information propagated from the output of the network back to its input using the backpropagation algorithm. While some methods modify backpropagation in some way, here we are interested in those, such as gradient, linear approximation, and all variants of our proposed NormGrad saliency method, that do not.

In order to explain these “non-invasive” methods, we suggest that their saliency maps can be interpreted as a measure of how much the corresponding pixels contribute to changing the model parameters during a training step. We then propose a principled two-phase framework capturing this idea. In the extraction phase, a method isolates the contribution to the gradient from each spatial location. We use the fact that the gradient of spatially shared weights can be written as the sum over spatial locations of a function of the activation gradients and input features. In the aggregation phase, each spatial summand is converted into a scalar using an aggregation function, thus resulting in a saliency map.

Algorithm 1 Extract & Aggregate

1. **Extract.** Compute spatial contributions to the summation of the gradient of the weights.
 - Choose a layer whose parameters are shared spatially (layers from table 1).
 - Alternatively, insert an identity layer (section 3.1.2) at the targeted location.
 2. **Aggregate.** Transform these spatial contributions into saliency maps using an aggregation function.
 - Norm: NormGrad (ours)
 - Voting/Summing: linear approximation [17]
 - Filtering: GradCAM [27], selective NormGrad (ours)
 - Max: Gradient backpropagation [29]
-

3.1.1 Phase 1: Extract

We first choose a target layer in the network at which we plan to compute a saliency map. Assuming that the network is a simple chain¹, we can write $L = \ell \circ \Phi = h \circ k_w \circ q$, where k_w is the target layer parameterised by w , h is the composition of all layers that follow it (including loss ℓ), and q is the composition of layers that precede it. Then, $\mathbf{x}^{in} = q(\mathbf{x}) \in \mathbb{R}^{K \times H \times W}$ denotes the input to the target layer, and its output is given by $\mathbf{x}^{out} = k_w(\mathbf{x}^{in}) \in \mathbb{R}^{K' \times H' \times W'}$.

¹Other topologies are treated in the same manner, but the notation is more complex.

Layer	Spatial contribution	Size at each location
Bias	$\mathbf{g}_u^{out}$	vector: K'
Scaling	$\mathbf{g}_u^{out} \odot \mathbf{x}_u^{in}$	vector: K'
Conv $N \times N$	$\mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{in \top}$	matrix: $K' \times N^2 K$

Table 1. Formulae and sizes of the spatial contributions to the gradient of the weights for layers with spatially shared parameters. $\odot$ denotes the elementwise product and $\mathbf{x}_{u,N \times N}^{in}$ is the $N^2 K$ vector obtained by unfolding the $N \times N$ patch around the target location.

Convolutional layers with general filter shapes. For convolutions with an $N \times N$ kernel size, we can re-write the convolution using the matrix form:

$$\mathbf{X}^{out} = \mathbf{W} \mathbf{X}_{N \times N}^{in} \quad (1)$$

where $\mathbf{X}^{out} \in \mathbb{R}^{K' \times HW}$ and $\mathbf{W} \in \mathbb{R}^{K' \times N^2 K}$ are the output and filter tensors reshaped as matrices and $\mathbf{X}_{N \times N}^{in} \in \mathbb{R}^{N^2 K \times HW}$ is a matrix whose column $\mathbf{x}_{u,N \times N}^{in} \in \mathbb{R}^{N^2 K}$ contains the unfolded patches at location u of the input tensor to which filters are applied.² Then, the gradient w.r.t. the filter weights W is given by

$$\frac{dL}{d\mathbf{W}} = \sum_{u \in \Omega} \frac{d}{d\mathbf{W}} \langle \mathbf{g}_u^{out}, \mathbf{W} \mathbf{x}_{u,N \times N}^{in} \rangle = \sum_{u \in \Omega} \mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{in \top}, \quad (2)$$

where $\mathbf{g}_u^{out} = dh/d\mathbf{x}_u^{out}$ is the gradient of the “head” of the network. Thus, for the convolutional layer case, the spatial summand is an outer product of two vectors; thus, the spatial contribution at each location to the gradient of the weights is a matrix of size $K' \times N^2 K$.

Other layer types. Besides convolutional layers, bias and scaling layers also share their parameters spatially. In modern architectures, these are typically used in batch normalization layers [14]. We denote $\mathbf{b}, \alpha \in \mathbb{R}^K$ as the parameters for the bias and scaling layers respectively. They are defined respectively as follows:

$$x_{ku}^{out} = x_{ku}^{in} + b_k, \quad x_{ku}^{out} = \alpha_k x_{ku}^{in}.$$

Then, the gradients w.r.t. parameters are given by

$$\frac{dL}{d\mathbf{b}} = \sum_{u \in \Omega} \mathbf{g}_u^{out}, \quad \frac{dL}{d\alpha} = \sum_{u \in \Omega} \mathbf{g}_u^{out} \odot \mathbf{x}_u^{in}. \quad (3)$$

where $\odot$ is the elementwise product. For these two types of layer, the spatial summand is a vector of size K , the number of channels. Table 1 summarizes the spatial contributions to the gradients for the different layers.

²This operation is called `im2row` in MATLAB or `unfold` in PyTorch.

3.1.2 Virtual identity layer

So far, we have only extracted spatial gradient contributions for layers that share parameters spatially. We will now extend our summand extraction technique to any location within a CNN by allowing the insertion of a virtual identity layer at the target location. This layer is a conceptual construction that we introduce to derive in a rigorous and unified manner the equations employed by various methods to compute saliency.

We are motivated by the following question: if we were to add an identity operator at a target location, how should this operator’s parameters be changed? A virtual identity layer is a layer which shares its parameters spatially and is set to the identity. Hence, it could be any of the layer from table 1, like a bias or a scaling layer; then, $b_k = 0$ or $c_k = 1$ respectively for $k \in \{1, \dots, K\}$. It could also be a $N \times N$ convolutional layer with filter bank $\mathbf{W} \in \mathbb{R}^{K \times N^2 K}$ such that $w_{kk'ij} = \delta_{k,k'} \delta_{i,0} \delta_{j,0}$, where $\delta_{a,b}$ is the Kronecker delta function³, for $(i, j) \in \{-\frac{N-1}{2}, \dots, \frac{N-1}{2}\}$.

Because of the nature of the identity, this layer does not change the information propagated in either the forward or backward direction. Conceptually, it is attached to the part of the model that one wishes to inspect as shown in fig. 1. This layer is never “physically” added to the model (i.e., the model is not modified); its “inclusion” or “exclusion” simply denotes whether we are using input or output activations ($\mathbf{x}^{in}$ or $\mathbf{x}^{out}$). Indeed the backpropagated gradient $\mathbf{g}^{out}$ at the output of the identity as shown in fig. 1 is the gradient that would have been at the output of the layer preceding the identity. This construction allows to examine activations and gradients at the same location (e.g., $\mathbf{x}^{out}$ and $\mathbf{g}^{out}$) as most existing saliency methods do. We now can use the formulae defined in section 3.1.1 when analyzing the gradient of the weights of this identity layer. For example, if we consider an identity scaling layer, the spatial contribution would be $\mathbf{g}_u^{out} \odot \mathbf{x}_u^{out}$. Or, for a $N \times N$ identity convolutional layer, it would be $\mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{out \top}$.

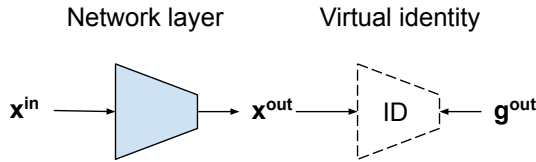


Figure 1. **Virtual identity.** Here, we visualize inserting an identity after a specific layer in the network for saliency computation purposes. The gradient coming from later stages of the network that is the gradient at output of the network layer, is now also the gradient at the output of the identity, $\mathbf{g}^{out}$.

³Kronecker delta function: $\delta_{a,b} = 1$ if $a = b$; otherwise, $\delta_{a,b} = 0$.

3.1.3 Phase 2: Aggregate

Following the extraction phase (section 3.1.1), we have the local contribution at each spatial location to the gradients of either an existing layer or a virtual identity layer. Each spatial location is associated with a vector or matrix (table 1). In this section, we describe different aggregation functions to map these vectors or matrices to a single scalar per spatial position. We drop the spatial location u in the notations.

Understanding existing saliency methods. One possible aggregation function is the sum of the elements in a vector. When combined with a virtual scaling identity layer (section 3.1.2), we obtain the linear approximation method [17]: $\sum_k g_k^{out} x_k^{out}$. The contributions from the scaling identity encode the result of channels changing (after the gradient is applied) at each location; thus, the sum aggregation function acts as a voting mechanism. The resulting saliency map highlights the locations that would be most impacted if following through with the channel updates.

Aggregation functions can also be combined with element-wise filtering functions (e.g., the absolute value function). Another aggregation function takes the maximum absolute value of the vector: $\text{maxabs}(\mathbf{x}) = \max_k |x_k|$. If we combine a virtual bias identity layer in phase 1, which gives $\mathbf{g}^{out}$ as the spatial contribution, with the maxabs function for aggregation, we obtain at each spatial location the gradient [29] method: $\max_k |g_k^{out}|$.

As for CAM [39] and Grad-CAM [27], we cannot directly use the spatial contributions extracted at each location because they spatially average $\mathbf{g}^{out}$.⁴ However, for architectures that use global average pooling followed after their convolutional layers (e.g., ResNet architectures), $\mathbf{g}_u^{out} = \bar{\mathbf{g}}^{out}$. Then, CAM and Grad-CAM and CAM can be viewed as combining a virtual scaling identity layer from phase 1 with summing and positive filtering (i.e., $\text{filter}_+(x) = (x)_+$) functions for aggregation.

NormGrad. The sum and max functions have clear interpretations when using bias or scaling identity layers; however, they cannot be easily transported to convolutional identity layers as interactions between input channels can vary depending on the output channel and are represented as a matrix, not as a vector as are the case for scaling and bias layers. Thus, we would like to have an aggregation function that could be used to aggregate any type of spatial contribution, regardless of its shape.

Using the norm function satisfies this criterion, for example the L^2 norm when dealing with vectors and the Frobenius norm for matrices. We note that the matrices obtained at each location for convolutional layers are the outer

⁴Because CAM global average pooling + one fully connected layer, $\mathbf{g}^{out}$ is equal to the fully connected weights.

Phase 1	Phase 2	Saliency map
Bias IDL	Max	$\max_k g_k^{out}$
Scaling IDL	Sum	$\sum_k g_k^{out} x_k^{out}$
Scaling IDL	F + N	$\ (g^{out} \odot x^{out})_+\ $
Conv 1×1 IDL	Norm	$\ g^{out}\ \ x^{out}\ $
Real Conv 3×3	Norm	$\ g^{out}\ \ x_{3 \times 3}^{in}\ $

Table 2. **Combining layers and aggregation functions for saliency.** g_k^{out} and x_k^{out} are tensor slices for channel k and contain only spatial information. IDL denotes an identity layer. F + N is positive filtering + norm. From top to bottom, the rows correspond to the following saliency methods: 1., gradient, 2., linear approximation, 3., NormGrad selective, 4., NormGrad, and 5., NormGrad without the virtual identity trick.

products of two vectors. For such matrices, the Frobenius norm is equal to the product of the norms of the two vectors. For example, for an existing 1×1 convolutional layer, we consider the saliency map given by $\|g^{out}\| \|x^{in}\|$. We call this class of saliency methods derived by using norm as an aggregation function, “NormGrad”.

Saliency maps from the NormGrad outlined above are not as class selective as other methods because they highlight the spatial locations that contribute the most to gradient of the weights, both positively and negatively. One way to introduce class selectivity is to use positive filtering before applying the norm. If we apply norm and positive filtering aggregation to a scaling identity layer, the resulting saliency map is given by $\|(g^{out} \odot x^{out})_+\|$. Throughout the rest of the paper, we call this variant “selective NormGrad”.

4. Experiments

In this section we quantitatively and qualitatively evaluate the performance of a large number of backpropagation-based saliency methods. Code for our framework is released at http://github.com/srebuffi/revisiting_saliency. Additional experiments such as image captioning visualizations are included in the appendix.

Experimental set-up. Unless otherwise stated, saliency maps are generated on images from either the PASCAL VOC [7] 2007 test set or the ImageNet [6] 2012 validation set for either VGG16 [29] or Resnet50 [12]. For PASCAL VOC, we use a model pre-trained on ImageNet with fine-tuned fully connected layers on PASCAL VOC. We use [9]’s TorchRay interpretability package to generate saliency methods for all other saliency methods besides our NormGrad and meta-saliency methods as well as to evaluate saliency maps on [38]’s Pointing Game (see [38] for more details). For all correlation analysis, we compute the Spearman’s correlation coefficient [21] between saliency maps that are upsampled to the input resolution: 224×224 .

4.1. Justifying the virtual identity trick.

In order to justify our use of the virtual identity trick, we compare NormGrad saliency maps generated at 1×1 and 3×3 convolutional layers both with and without the virtual identity trick (4th and 5th rows respectively in table 2) for VGG16 and ResNet50. We first computed the correlations between saliency maps generating with and without the virtual identity trick. We found that the mean correlation across $N = 50k$ ImageNet validation images was over 95% across all layers we tested. We also evaluated their performance on the Pointing Game [38] and found that the mean absolute difference in pointing game accuracy was $0.53\% \pm 0.62\%$ across all layers we tested (see supp. for more details and full results). This empirically demonstrates that using the virtual identity trick closely approximates the behavior of calculating the spatial contributions for the original convolutional layers.

4.2. Combining saliency maps across layers.

Training linear classifiers on top of intermediate representations is a well-known method for evaluating the learned features of a network at different layers [2]. This suggests that saliency maps, too, may have varying levels of meaningfulness at different layers.

We explore this question by imposing several weighting methods for combining the layer-wise saliency maps of ResNet50 and VGG16 and measuring the resulting performance on PASCAL VOC Pointing Game on both the “difficult” and “all” image sets. To determine the weight γ_j for a given layer j out of J layers in a network we use:

1. **Feature spread.** Given the set of feature activations at layer j , x^i for $i \in M$ input images sampled uniformly across classes, compute the spatial mean $\bar{x}^i = \sum_{u \in \Omega} x_u^i$. $\gamma_j = \frac{1}{M} \sum_{i=1}^M |\bar{x}^i - \bar{x}^\mu|$ where $\bar{x}^\mu = \frac{1}{M} \sum_{i=1}^M |\bar{x}^i|$.
2. **Classification accuracy.** Given the set of feature activations at layer j , x^i for $i \in M$ input images sampled uniformly across classes, train a linear layer Ψ using image labels y^i . $\gamma_j = \frac{1}{M} \sum_{i=1}^M \delta_{\Psi(x^i), y^i}$ where $\delta_{\Psi(x^i), y^i} = 1$ if $\Psi(x^i) = y^i$, 0 otherwise.
3. **Linear interpolation.** $\gamma_j = \frac{j}{J}$.
4. **Uniform.** $\gamma_j = \frac{1}{J}$.

To obtain a combined saliency map M from maps m_j from each layer j , the weights are normalised and applied either additively, $M = \sum_{j=0}^J \gamma_j \cdot m_j$, or with a product, $M = \prod_{j=0}^J m_j^{\gamma_j}$ (see fig. 2 for visualization).

Table 3 shows that weighted saliency maps produce the best overall performance in all four key test cases, which is surprising as there were only four weight schemes tested

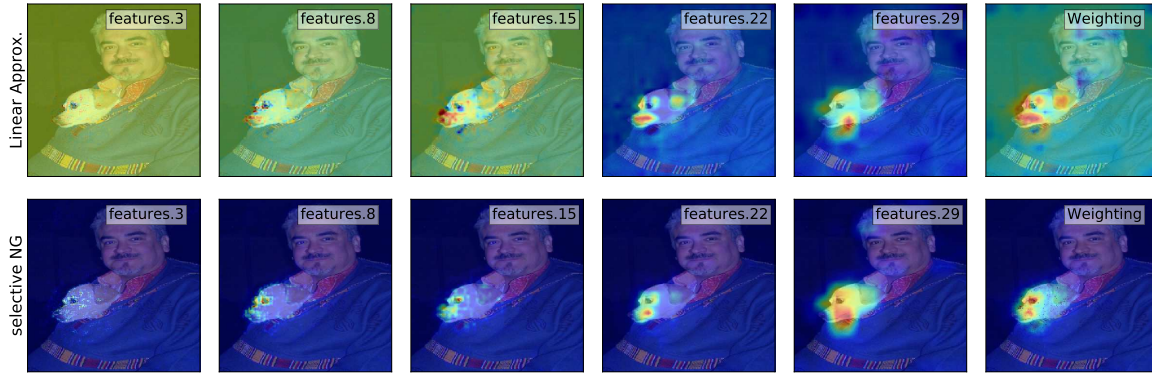


Figure 2. **Saliency maps at different network depths and as a weighted combination.** Linear approximation vs. selective NormGrad saliency maps of VGG16 on VOC2007. The first 5 images of each row correspond to different depths whereas the last one is a weighted product combination (using classification accuracy weights) of the first saliency maps. We observe that the weighted version produces more fine grained maps for both methods.

	All				Difficult			
	Resnet50		VGG16		Resnet50		VGG16	
	b.s.	b.w.	b.s.	b.w.	b.s.	b.w.	b.s.	b.w.
CEB	90.7	88.6	82.1	78.2	82.2	82.2	67.0	65.2
EB	84.5	83.1	77.5	75.7	71.5	71.3	57.8	56.1
GC	90.3	90.5	86.6	80.6	82.3	82.6	74.0	67.8
Gd	83.9	83.3	86.6	82.7	70.3	69.4	66.4	67.4
Gds	80.0	77.4	76.8	77.2	62.9	59.5	57.9	59.4
Gui	82.3	81.0	75.8	74.4	67.9	63.4	53.0	51.6
LA	90.2	91.2	86.4	86.9	81.9	83.8	74.5	77.4
NG	84.6	83.5	81.9	81.8	72.2	70.2	64.8	64.6
sNG	87.4	88.7	86.0	86.8	77.0	79.1	72.6	74.5

Table 3. **Pointing game results on VOC07.** b.s. and b.w. stand for best single layer and best weighted combination. (C)EB: (Contrastive) Excitation Backprop, GC: GradCAM, Gd(s): Gradient (sum), Gui: guided backprop, LA: linear approximation, (s)NG: (selective) NormGrad.

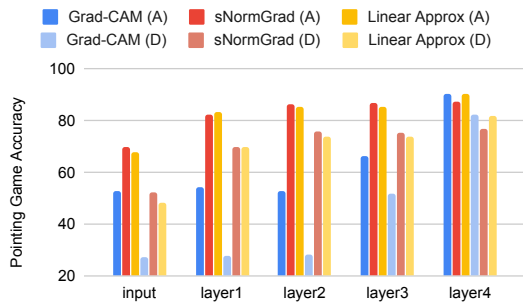


Figure 3. **Select Pointing Game results.** Results for ResNet50 on VOC07 at different network depths (A: all images; D: difficult subset). Grad-CAM performs worse at every layer except the last conv layer and lower than pointing at the center (all: 69.6%; diff: 42.4%) at most layers.

(in addition to best single layer), none of which were explicitly optimised for use with saliency maps. Our results strongly indicate that linear approximation in particular benefits from combining maps from different layers, and linear approximation with layer combination consistently produces the best performance overall and beats far more complex methods at weak localisation using a single forward-backward pass (see supp. for full results).

Note that the feature spread and classification accuracy metrics can both be used as indicators of class sensitivity (section 4.3). This is because if feature activations are uniform for images sampled across classes, it is not possible for them to be sensitive to - or predictive of - class, and the classification accuracy metric is an explicit quantisation of how easily features can be separated into classes. We observe from the computed weights that both metrics generally increase with layer depth (see supp.).

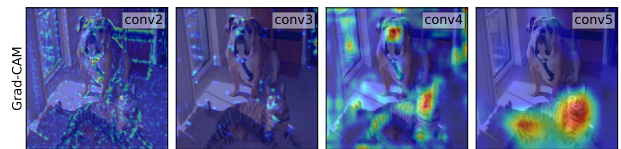


Figure 4. **Grad-CAM failure mode.** Grad-CAM saliency maps w.r.t. “tiger cat” at different depths of VGG16. Grad-CAM only works at the last conv layer (rightmost col).

Explanation of Grad-CAM failure mode. Figure 4 showed qualitatively that Grad-CAM does not produce meaningful saliency maps at any layer except the last convolutional layer, which is confirmed by Grad-CAM’s Pointing Game results at earlier layers. Class sensitivity - as measured by our weighting metrics - increasing with layer depth offers an explanation for this drop in performance. Since

Grad-CAM spatially averages the backpropagated gradient before taking a product with activations, each pixel location in the heatmap receives the same gradient vector (across channels) *irrespective* of the image content contained within its receptive field. Thus, if the activation map used in the ensuing product is also not class selective - firing on both dogs and cats for example, fig. 4 - the saliency map cannot be. On the other hand, methods that do not spatially average gradients such as NormGrad (fig. 3) can rely on gradients that are free to vary across the heatmap with underlying class, increasing the class sensitivity of the resulting saliency map.

4.3. An explicit metric for class sensitivity

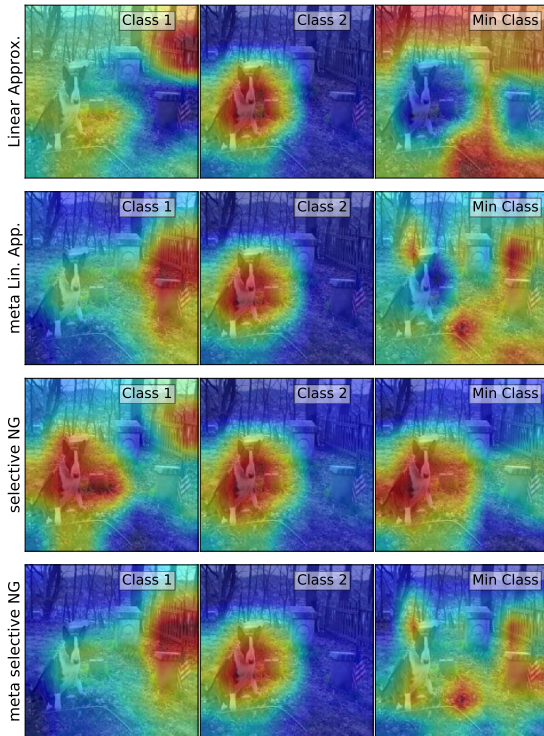


Figure 5. **Class sensitivity with and without meta-saliency.** Min class saliency maps that use meta-saliency (row 2 and 4, right col) are less informative than those that don't use meta-saliency (rows 1 and 3, right col). Class 1 is the ground truth class (fence), class 2 is the maximally predicted class (Cardigan Welsh corgi), min class is the minimally predicted class (black widow spider).

[20] qualitatively shows that early backprop-based methods (e.g., gradient, deconvnet, and guided backprop) are not sensitive to the output class being explained by showing that saliency maps generated w.r.t. different output classes and gradient signals appear visually indistinguishable. Thus, similar to [1], we introduce a sanity check to measure a saliency method's output class sensitivity. We compute the correlation between saliency maps w.r.t. to output class pre-

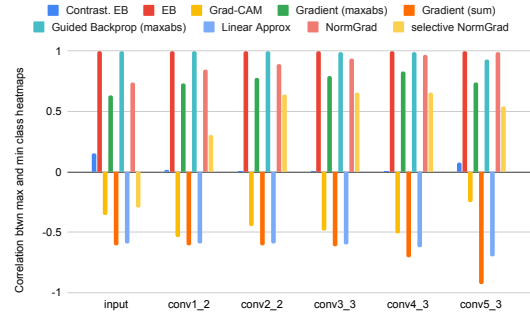


Figure 6. **Class sensitivity of saliency methods.** This plot shows the correlation between VGG16 saliency maps computed w.r.t. to the maximally and minimally predicted class (closer to zero is better).

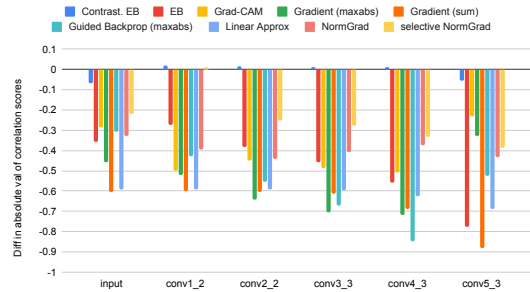


Figure 7. **Meta-saliency improves class sensitivity for all saliency methods.** Using meta-saliency yields weaker correlations between the saliency maps w.r.t. the maximally and minimally predicted output class compared to not using meta-saliency (lower is better).

dicted with highest confidence (max class) and that predicted with lowest confidence (max and min class respectively) for $N = 1000$ ImageNet val. images (1 per class).

We would expect saliency maps w.r.t. the max class to be visually salient while those w.r.t. to the min class to be uninformative (because the min class is not in the image). Thus, we desire the correlation scores to be close to zero.

Figure 6 shows results for various saliency methods. We observe that excitation backprop and guided backprop yield correlation scores close to 1 for all layers, while contrastive excitation backprop yields scores closest to 0. Furthermore, methods using sum aggregation (e.g., gradient [sum], linear approx, and Grad-CAM) have negative scores (i.e., their max-min-class saliency maps are anti-correlated). This is because sum aggregation acts as a voting mechanism; thus, these methods reflect the fact that the network has learned anti-correlated relationships between max and min classes.

4.4. Meta-saliency analysis

As a general method for improving the sensitivity of saliency heatmaps to the output class used to generate the

gradient, we propose to perform an inner SGD step before computing the gradients with respect to the loss. This way we can extend any saliency method to second order gradients. This is partly inspired by the inner step used in, for example, few shot learning [8] and architecture search [18]. We want to minimize:

$$L(\theta, x) = \ell(\theta - \epsilon \nabla_{\theta} \ell(\theta, x), x). \quad (4)$$

We take $\epsilon \ll 1$ to use a Taylor expansion of this loss at θ and we now have the resulting approximated loss:

$$L(\theta, x) \approx \ell(\theta, x) - \epsilon \|\nabla_{\theta} \ell(\theta, x)\|^2. \quad (5)$$

As done in the previous section, we can now take the gradient of the loss with respect to the parameters θ :

$$\nabla_{\theta} L(\theta, x) \approx \nabla_{\theta} \ell(\theta, x) - 2\epsilon \nabla_{\theta}^2 \ell(\theta, x) \nabla_{\theta} \ell(\theta, x). \quad (6)$$

Using a finite difference scheme of step h as in [23], we can approximate the hessian-vector product by:

$$\nabla_{\theta}^2 \ell(\theta, x) \nabla_{\theta} \ell(\theta, x) = \frac{\nabla_{\theta} \ell(\theta, x) - \nabla_{\theta} \ell(\theta - h \nabla_{\theta} \ell(\theta, x), x)}{h} + O(h).$$

where $\theta^- = \theta - h \nabla_{\theta} \ell(\theta, x)$. We chose on purpose a backward finite difference such that two terms cancel each other when taking $h = 2\epsilon$ and we get:

$$\nabla_{\theta} L(\theta, x) \approx \nabla_{\theta'} \ell(\theta', x).$$

where $\theta' = \theta - 2\epsilon \nabla_{\theta} \ell(\theta, x)$ corresponds to one step of SGD of learning rate 2ϵ . We notice that if we take $\epsilon \rightarrow 0$, this formula boils back down to the original gradient of the weights without meta step. We further note that this meta saliency approach only requires one more forward-backward pass compared to usual saliency backpropagation methods.

Conversely, if we would like to get an importance map that highlights the degradation of the model's performance, we should add an inner step with gradient ascent within the loss. Hence by minimizing the resulting loss $-\ell(\theta + \epsilon \nabla_{\theta} \ell(\theta, x), x)$, we get the same formula for the gradients of the weights but with $\theta' = \theta + 2\epsilon \nabla_{\theta} \ell(\theta, x)$.

We hypothesize that applying meta-saliency to a saliency method should decrease correlation strength because allowing the network to update one SGD step in the direction of the min class should “destroy” the informativeness of the resulting saliency map. We use a learning rate $\epsilon = 0.001$ for the class sensitivity quantitative analysis. Figure 5 shows qualitatively that this appears to be the case: without meta-saliency, selective NormGrad and linear approximation yield max (class 2) and min class heatmaps that are highly positively and negatively correlated respectively. However, when meta-saliency is applied, the min class saliency map appears more random. Figure 7 shows results comparing the max-min class correlation scores with and

without meta-saliency. These results demonstrate that meta-saliency decreases max-min class correlation strength for nearly all saliency methods and suggest that meta-saliency can increase the class sensitivity for any saliency method.

4.5. Model weights sensitivity

[1] shows that some saliency methods (e.g., Guided Backprop in particular) are not sensitive to model weights as they are randomized in a cascading fashion from the end to the beginning of the network. Figure 8 shows qualitatively that, by the late conv layers, saliency maps for linear approximation and selective NormGrad are effectively scrambled (top two rows). It also highlights that, because meta-saliency increases class selectivity and is allowed to take one SGD in the direction of the target class, it takes relatively longer (i.e., more network depth) to randomize a meta-saliency heatmap (bottom row and see appendix).

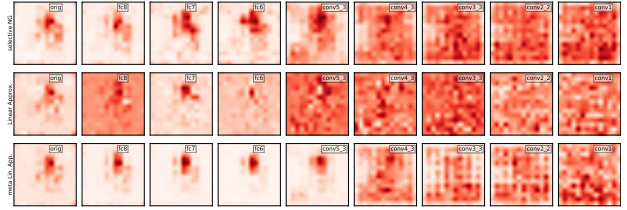


Figure 8. **Model weights sensitivity.** Sanity check by randomizing VGG16 model weights in a cascading fashion for the “Irish terrier” image from [1]. Top row: selective NormGrad, middle row: linear approximation, bottom row: linear approximation with meta-saliency (lower correlation with orig heatmap [leftmost col] is better). All methods look random after conv4_3. By comparing the last two rows at conv5_3, we see that meta-saliency enforces more class sensitivity than the non-meta variant.

5. Conclusions

We introduced a principled framework based on the contribution of each spatial location to the weights’ gradient. This framework unifies several existing backpropagation-based methods and allowed us to systematically explore the space of possible saliency methods. We use it for example to formulate NormGrad, a novel saliency method. We also studied how to combine saliency maps from different layers, discovering that it can consistently improve weak localization performance and produce high resolution maps. Finally, we introduced a class-sensitivity metric and proposed meta-saliency, a novel paradigm applicable to any existing method to improve sensitivity to the target class.

6. Acknowledgments

This work is supported by Mathworks/DTA, the Rhodes Trust (M.P.), EPSRC AIMS CDT and ERC 638009-IDIU.

References

- [1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 1, 2, 7, 8
- [2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv*, 2016. 5
- [3] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proc. NIPS*, 2014. 1
- [4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015. 1, 2
- [5] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. NIPS*, 2017. 2
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5
- [7] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 5
- [8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017. 8
- [9] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, 2019. 2, 5
- [10] Ruth Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. CVPR*, 2017. 2
- [11] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv*, 2017. 1
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5
- [13] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv*, 2018. 2
- [14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015. 3
- [15] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Xrai: Better attributions through regions. In *Proc. ICCV*, 2019. 2
- [16] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 2019. 2
- [17] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv*, 2016. 1, 2, 3, 4
- [18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, 2018. 8
- [19] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. NIPS*, 2017. 2
- [20] Aravindh Mahendran and Andrea Vedaldi. Salient deconvolutional networks. In *Proc. ECCV*, 2016. 1, 2, 7
- [21] Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences*, 12, 2004. 5
- [22] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks. In *Proc. ICLR*, 2019. 2
- [23] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 1994. 8
- [24] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proc. BMVC*, 2018. 2
- [25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. KDD*, 2016. 2
- [26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2
- [27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 1, 2, 3, 4
- [28] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proc. ICML*, 2017. 2
- [29] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*, 2014. 1, 2, 3, 4, 5
- [30] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *Proc. ICCV*, 2017. 2
- [31] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv*, 2017. 2
- [32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv*, 2014. 2
- [33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proc. ICML*, 2017. 2
- [34] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *Proc. CVPR*, 2017. 2
- [35] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proc. CVPR*, 2017. 2
- [36] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv*, 2019. 2
- [37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, 2014. 1, 2

- [38] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *Proc. ECCV*, 2016. 1, 2, 5
- [39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. CVPR*, 2016. 2, 4

There and Back Again: Revisiting Backpropagation Saliency Methods

Supplementary Material

A. Pointing Game details

We use [6]’s Pointing Game for quantitative evaluation of saliency methods. Saliency maps are computed with respect to every object class present in each image. If the maximally salient point for each class lands on the ground truth annotation for that object (within a threshold of 15 pixels), then a “hit” is recorded; otherwise, a “miss” is recorded. Pointing game accuracy is computed as the mean over per-class accuracies given by the following:

$$\frac{|\text{hits}|}{|\text{hits} + \text{misses}|}.$$

We evaluate VGG16 and ResNet50 networks that have been trained on ImageNet and fine-tuned on PASCAL VOC and COCO. We evaluate on the PASCAL VOC 2007 test split ($N = 4952$ images) and COCO 2014 val split ($N \approx 50k$). We also show performance on the difficult subsets of the data provided by [6]; these are images for which the total area of the annotations (bounding boxes for PASCAL VOC and segmentation masks for COCO) for the given object class is less than 25% of the image size and for which there is at least one other object class present. We use [2]’s TorchRay library for evaluation; see [6] for more details.

B. Virtual identity trick correlations

We computed the correlation between saliency maps generated both with and without the virtual identity trick (paper section 4.1). The high correlation shown in Table 1, as well as the minimal difference in pointing game performance ($0.53\% \pm 0.62\%$) demonstrates that the identity trick closely approximates the behaviour of calculating the spatial contributions for the original convolutional layers.

Architecture	Layer	Correlation
ResNet50	layer1.0.conv1	99.48 ± 0.55
ResNet50	layer2.0.conv1	99.32 ± 0.35
ResNet50	layer3.0.conv1	99.16 ± 0.42
ResNet50	layer4.0.conv1	98.35 ± 1.05
VGG16	features.2	97.77 ± 1.34
VGG16	features.7	99.15 ± 0.42
VGG16	features.14	98.99 ± 0.57
VGG16	features.21	97.79 ± 1.42
VGG16	features.28	94.33 ± 3.09

Table 1. **Correlations between saliency maps generated with and without the virtual identity trick.** For both VGG16 and ResNet50, the high values of correlation across the network (min of 94.33) justify the use of the identity trick.

	ResNet50		VGG16	
	All	Difficult	All	Difficult
CEB	layer3	layer4	feat.29	feat.29
EB	layer4	layer4	feat.29	feat.29
GC	layer4	layer4	feat.29	feat.29
Gd	layer2	layer2	feat.29	feat.22
Gds	layer2	layer2	feat.8	feat.8
Gui	layer3	layer3	input	input
LA	layer4	layer4	feat.29	feat.29
NG	layer3	layer3	feat.22	feat.22
sNG	layer4	layer4	feat.29	feat.29

Table 2. **Best individual layer for the Pointing game on VOC07.** (C)EB: (Contrastive) Excitation Backprop, GC: GradCAM, Gd(s): Gradient (sum), Gui: guided backprop, LA: linear approximation, (s)NG: (selective) NormGrad.

C. Performance of combining saliency maps

As noted in paper section 4.2, feature spread and classification accuracy are both interpretable as measures of feature sensitivity with respect to the class of the input image. Figure 1 shows that both increase with network depth, with the exception of feature spread for VGG16, which decreases in the last two layers despite simultaneously increasing classification accuracy. We note that classification accuracy is a more reliable metric than feature spread as it directly codes for the separability of features with respect to class, whereas feature spread is susceptible to non-material differences in the absolute scale of activation values across layers. In the case of VGG16, this means the features in the last two layers differ less across images (and hence, classes) in absolute terms, but are nonetheless highly discriminative of class.

The performance gains of our weighting schemes over using the best individual layer are given in table 3. The best individual layers for computing maps are given in table 2, which is notable as in no case does the practice of computing saliency at the earliest layer produce the best performance.

D. Meta-saliency analysis

We hypothesized that meta vs. non-meta saliency maps should be less correlated for validation images than for training images. This is because the inner gradient step of meta-saliency should not be as impactful for a seen training image as for an unseen validation image.

For both NormGrad and selective NormGrad, we evaluated over time the average correlation between the impor-

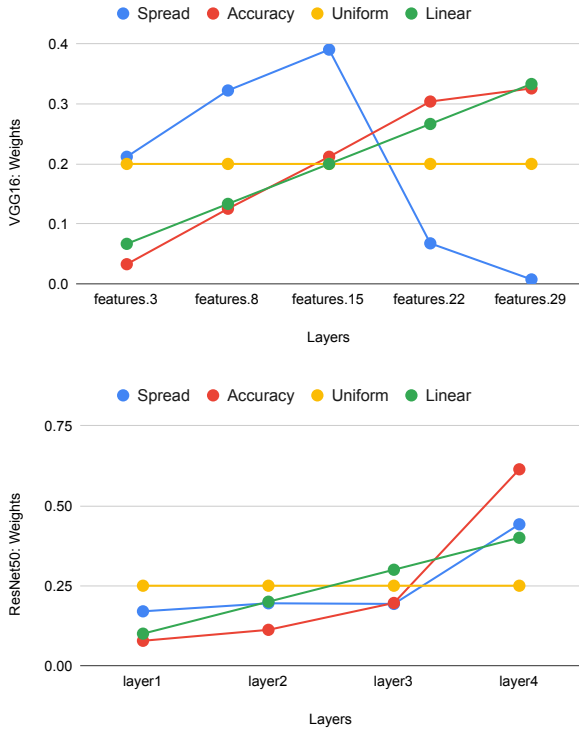


Figure 1. **Weights for the different weighting methods.** VGG16 on top and ResNet50 on the second row.

tance maps with and without meta-saliency on the training and testing sets respectively. Figure 2 demonstrates that the correlation scores are indeed decreased for validation images.

E. Model weights sensitivity

In fig. 5, we show the effects of cascading randomization on linear approximation and selective NormGrad methods by using the same images as [1] (“junco”, “corn” and “Irish terrier”) to illustrate the model weights sensitivity.

Qualitatively, both methods - with and without meta-saliency - demonstrate the desired sensitivity to model weights as the saliency maps progressively lose focus from the object target as layer depth decreases. We also observe that using meta-saliency on top of the chosen base saliency method (shown on every other line of fig. 5) delays the degradation in saliency maps to earlier layers.

F. Image captioning

We show in this section a few examples of our proposed selective NormGrad method applied to the image captioning setting. We use a variant [4] of the original neuraltalk2 [3] with a ResNet101 as backbone network followed by the LSTM caption model. As in [5] we do a back-

			ResNet50		VGG16	
			All	Difficult	All	Difficult
LA	+	accuracy	0.57	1.47	0.22	2.09
LA	+	spread	0.81	1.66	-3.01	-7.18
LA	+	linear	0.68	0.74	0.36	2.22
LA	+	uniform	0.58	1.29	-0.61	-0.20
LA	×	accuracy	0.87	1.84	0.42	2.91
LA	×	spread	1.01	1.78	-2.92	-6.80
LA	×	linear	0.52	0.72	0.49	2.87
LA	×	uniform	0.32	0.83	-0.25	0.52
sNG	+	accuracy	1.08	1.37	0.48	0.85
sNG	+	spread	0.94	1.40	-3.66	-6.43
sNG	+	linear	0.94	1.67	0.62	0.85
sNG	+	uniform	0.25	0.54	-0.93	-2.55
sNG	×	accuracy	1.26	1.65	0.83	1.88
sNG	×	spread	1.12	2.09	-3.13	-6.18
sNG	×	linear	0.63	0.95	0.72	0.62
sNG	×	uniform	0.89	1.79	-0.27	-1.76

Table 3. **Score gains compared to best individual layer performance for weighting methods with Linear Approximation (LA) and selective NormGrad (sNG) on the Pointing game on VOC07.** Paper showed that LA and sNG benefit from weighting methods. Here we can observe that using the weights as exponents in a product (lines with $\times$) is the most effective solution for both LA and sNG on ResNet50 and VGG16. Both the weightings using the features spread and the layer accuracy perform the best on ResNet50 but only the layer accuracy perform consistently across datasets and saliency methods.

ward pass using the log probability of the generated caption as objective function. We apply selective NormGrad before the final global average pooling and at the ReLU layers just after the downsampling shortcuts of the third and fourth macro blocks. We also compare these saliency maps with the product combination map of these layers using a linear weighting scheme (see section 4.2 in the paper). We notice in fig. 6 that using a combination of layers produces a sharper saliency map than for individual layers.

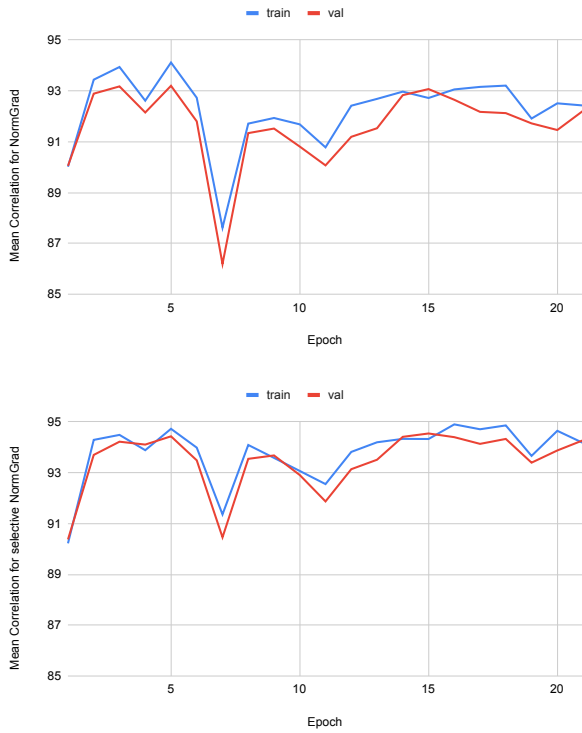


Figure 2. **Correlation between meta and non-meta saliency maps over time for train and val splits.** For both NormGrad (top) and selective NormGrad (bottom), the correlation is lower on the validation split.

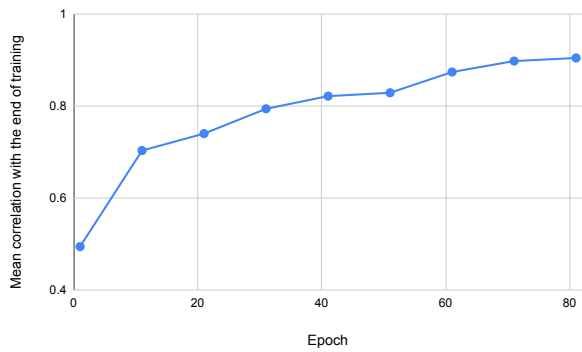


Figure 3. **Mean correlation on the val split between the meta-saliency maps of selective NormGrad at epoch t and the end of training.** The saliency maps stabilize at the end of the training.

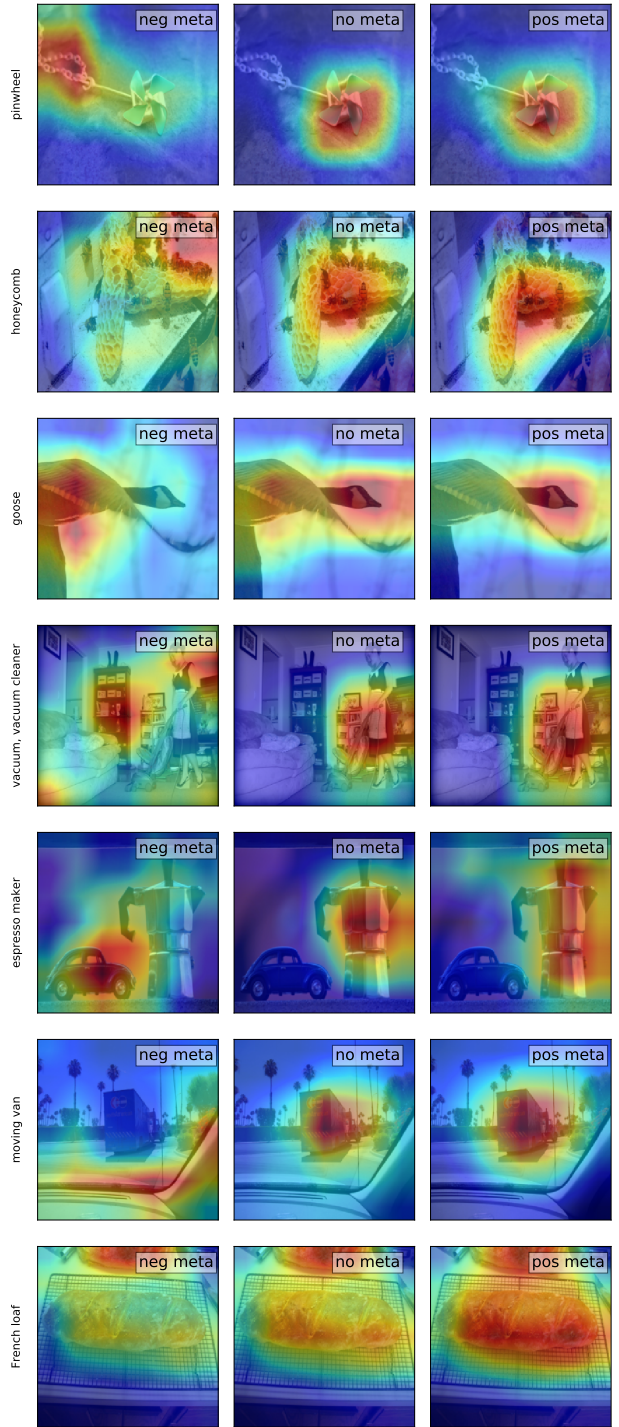


Figure 4. **Examples of positive and negative meta-saliency for selective NormGrad.** Negative meta-saliency (left images) corresponds to a gradient ascent inner step whereas a gradient descent step is used for the positive meta-saliency (images on the right). The center images do not use meta-saliency.



Figure 5. Model weights sensitivity on VGG16. Linear Approximation and selective NormGrad with and without meta-saliency.

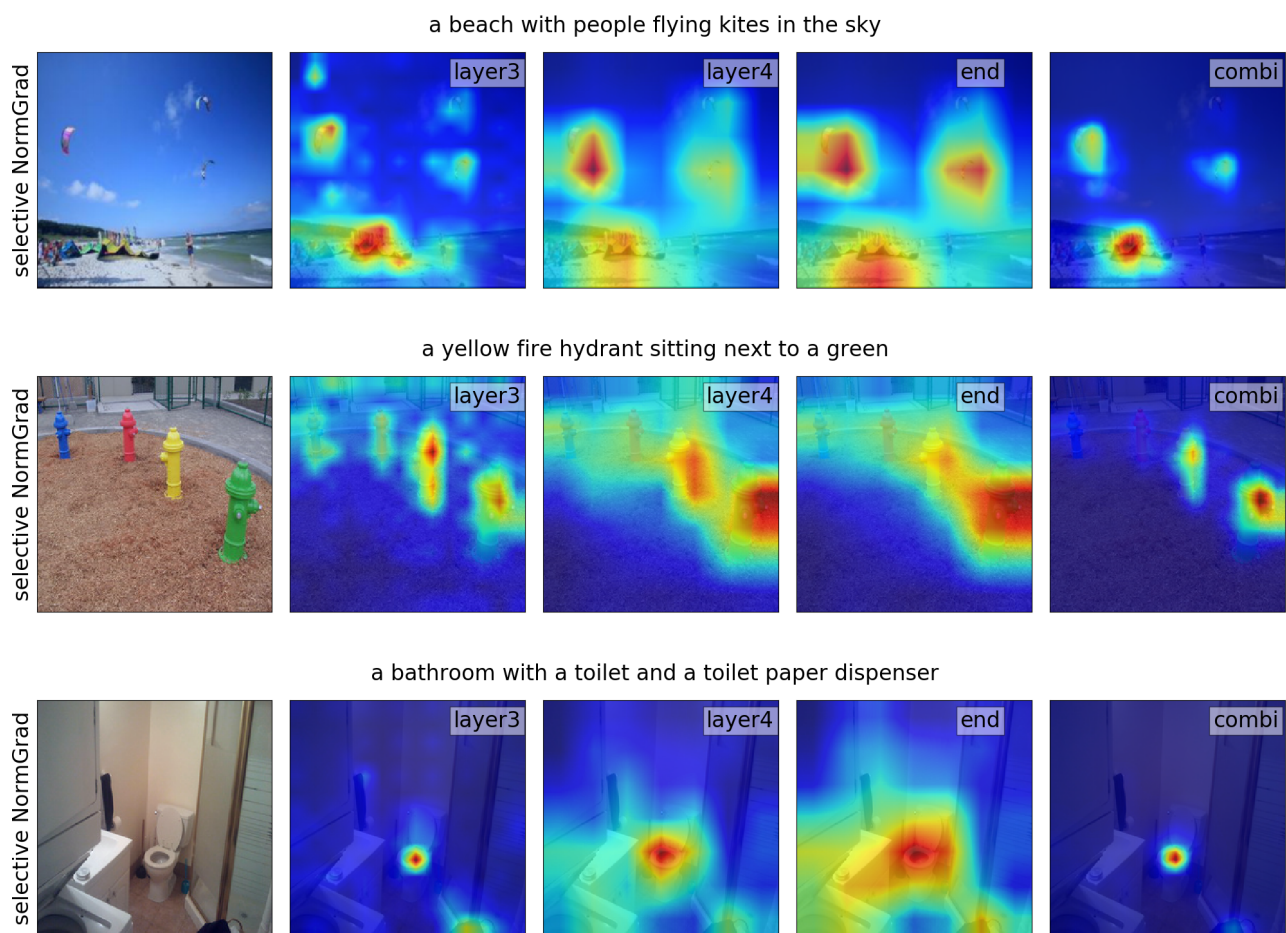


Figure 6. **Image captioning explanations.** Selective NormGrad is applied at different layers or combination of layers (last column) of a ResNet101. We observe that saliency maps at individual layers highlight a big part of the images. Using a combination of layers allows a clearer focus on the important parts of the images.

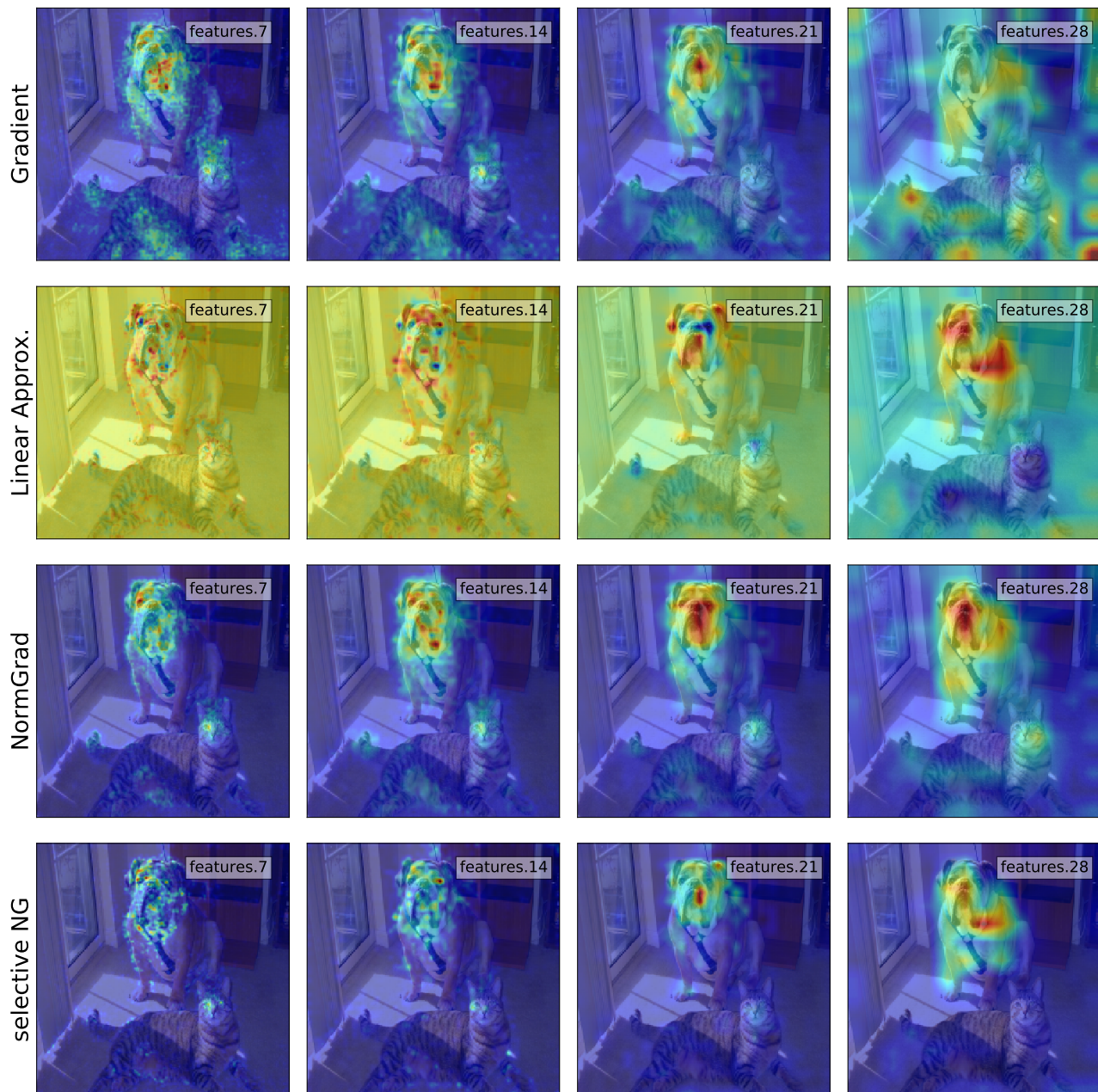


Figure 7. **Visualizations for some methods of the Extract & Aggregate framework at different layers of VGG16.** The gradient backproagation method (first row) works well at all stages except at the end of the network. Selective NormGrad (last row), NormGrad (third row) and Linear Approximation (second row) perform well across the network. Finally we can observe that selective NormGrad and Linear Approximation are more class selective than NormGrad as the non targeted "tiger cat" appears more in the third row.

References

- [1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 2
- [2] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, 2019. 1
- [3] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proc. CVPR*, 2015. 2
- [4] Ruotian Luo, Brian Price, Scott Cohen, and Gregory Shakhnarovich. Discriminability objective for training descriptive captions. *arXiv*, 2018. 2
- [5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 2
- [6] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *Proc. ECCV*, 2016. 1

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	There and Back Again: Revisiting Backpropagation Saliency Methods
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Sylvestre Rebuffi*, Ruth Fong*, Xu Ji*, Andrea Vedaldi. * denotes equal contribution. There and Back Again: Revisiting Backpropagation Saliency Methods. Published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

Student Confirmation

Student Name:	Xu Ji		
Contribution to the Paper	Sylvestre Rebuffi, Ruth Fong and I jointly designed research, ran experiments, and generated figures. Sylvestre Rebuffi came up with the initial idea for the Extract & Aggregate framework and the NormGrad formulation. Ruth Fong came up with the initial idea for the class sensitivity analysis and metric. I came up with the idea for combining saliency maps at different layers. Sylvestre Rebuffi, Ruth Fong, myself and Andrea Vedaldi jointly wrote the paper text.		
Signature 	Date	24/05/2021	

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Andrea Vedaldi			
Supervisor comments The description is accurate.			
Signature 	Date	25/05/2021	

This completed form should be included in the thesis, at the end of the relevant chapter.

6

Automatic Recall Machines: Internal Replay, Continual Learning and the Brain

This paper was published in NeurIPS Workshop on Baby Mind 2020.

Automatic Recall Machines: Internal Replay, Continual Learning and the Brain

Xu Ji¹, João Henriques¹, Tinne Tuytelaars², Andrea Vedaldi¹

¹VGG, University of Oxford

²KU Leuven

Abstract

Replay in neural networks involves training on sequential data with memorized samples, which counteracts forgetting of previous behavior caused by non-stationarity. We present a method where these auxiliary samples are generated on the fly, given only the model that is being trained for the assessed objective, without extraneous buffers or generator networks. Instead the implicit memory of learned samples within the assessed model itself is exploited. Furthermore, whereas existing work focuses on reinforcing the full seen data distribution, we show that optimizing for not forgetting calls for the generation of samples that are specialized to each real training batch, which is more efficient and scalable. We consider high-level parallels with the brain, notably the use of a single model for inference and recall, the dependency of recalled samples on the current environment batch, top-down modulation of activations and learning, abstract recall, and the dependency between the degree to which a task is learned and the degree to which it is recalled. These characteristics emerge naturally from the method without being controlled for.

1 Introduction

The ability to learn effectively from sequential or non-stationary data remains a fundamental difference between human brains and artificial neural networks. The development of biological brains is impaired when the environment changes too fast [Wood, 2016]. On the other hand, it has long been observed that artificial neural networks catastrophically forget previously learned behavior when data is presented in a sequential manner [McCloskey and Cohen, 1989]. In other words, while biological learning favors datastreams that focus on one task at a time, the effectiveness of stochastic gradient descent depends on the opposite quality of exposure to all tasks within a small temporal window.

The retention of learned behavior in neural networks can be increased with experience replay, enforced non-distributedness, or parameter-level regularization. Of these, replay is perhaps the most promising approach. Parameter-level regularization has been shown to underperform [Chaudhry et al., 2019, van de Ven and Tolias, 2019], likely due to restricting change with excessive stringency [Ramapuram et al., 2020]. Non-distributed representations limit parameter sharing a priori, preventing interference but sacrificing on memory efficiency and transfer [French, 1991, Bengio et al., 2013].

Replay, where training is augmented with auxiliary batches intended to capture previously seen data [Robins, 1995], avoids both these issues as it allows for constraints to be imposed at functional level over a distributed representation. However, existing work generally uses buffers or generator networks to memorize the seen data [Chaudhry et al., 2019, Shin et al., 2017]. Buffers are memory inefficient, and generator networks have proved difficult to train for natural images given only sequential data [Lesort et al., 2019a, Aljundi et al., 2019a]. In both cases it is also inconvenient to require additional memory for data memorization, especially as the model being trained for directly pertinent tasks already has an implicit memory of past training samples.

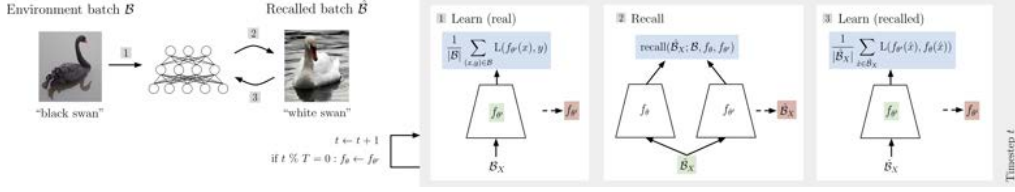


Figure 1: Automatic Recall Machines. In each stage, the model or recalled samples (green) are optimized for objective (blue) and updated (red). Reinforcing the resulting maximally interfered $\hat{\mathcal{B}}_X$ with old outputs from f_θ optimally minimizes forgetting of $f_{\theta'}$, because in theory if the maximal divergence between old and new non-local behavior is 0, then all such divergences are 0. Thus training on $\mathcal{B}$ updates local behavior while training on $\hat{\mathcal{B}}$ protects behavior in the rest of input space.

The goal of this work, Automatic Recall Machines (ARM), is to optimally exploit the implicit memory in the tasks model for not forgetting, by using its parameters for both inference and generation (fig. 1). Instead of attempting to reproduce the full seen data distribution, we generate specifically the optimal samples conditioned on the current real batch, which is more efficient and scalable. Our derivation shows that these samples are the points in input space whose outputs are maximally changed given training on the real batch, providing an explanation for why training with the most dissonant related sample is optimal for not forgetting, an intuition that was used in Aljundi et al. [2019a]. Memory reconsolidation in humans also appears to favor conflicting experiences [Sinclair and Barense, 2018].

We use *recall* to refer to internally generated replay. *Automatic* refers to recall being a direct consequence of learning on the environment. Without an initial learning step, there is no change in knowledge held by the network to compensate for and thus no recall; vice-versa, environmental batches that cause a significant change in network knowledge induce more learning from recall (eq. (6)). In the brain, experiences associated with higher surprise and thus representational change also increase the intensity of replay during and after the experience [Cheng and Frank, 2008, O’Neill et al., 2008].

2 Automatic Recall Machines

Given the space of outputs, $\mathcal{Y} = \{1, \dots, C\}$, and space of inputs, for example images $\mathcal{X} = [0, 1]^{3 \times H \times W}$, define the knowledge of the categorical neural network f_θ as its graph:

$$\text{graph}(f_\theta) = \{(x, y) \in \mathcal{X} \times \mathcal{Y} : f_\theta(x) = y\}. \quad (1)$$

When training on real batch $\mathcal{B} \subset \mathcal{D}$ from non-stationary datastream $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ to obtain updated parameters θ' , we would like the difference between old θ and new θ' to amount to a *local* change in knowledge. θ could be the snapshot of parameters immediately preceding θ' or, more generally, preceding θ' by $T \geq 1$ timesteps, since we make no assumption that $T = 1$. Let $\mathcal{B}_X$ and $\mathcal{B}_Y$ denote the inputs and output targets of $\mathcal{B}$. Define knowledge preservation or not forgetting as:

$$\forall \hat{x} \in \mathcal{X}. \neg \text{local}(\hat{x}; \mathcal{B}, f_\theta, f_{\theta'}) : f_\theta(\hat{x}) = f_{\theta'}(\hat{x}) \quad (2)$$

$$\text{where } \text{local}(\hat{x}; \mathcal{B}, f_\theta, f_{\theta'}) = f_\theta(\hat{x}) \in \mathcal{B}_Y \vee f_{\theta'}(\hat{x}) \in \mathcal{B}_Y. \quad (3)$$

This states that predictions for the input space should be the same for f_θ and $f_{\theta'}$, except inputs mapped to the classes in $\mathcal{B}$ i.e. classes currently being trained. Equation (2) is satisfied iff:

$$\max_{\substack{\hat{x} \in \mathcal{X} \\ \neg \text{local}(\hat{x}; \mathcal{B}, f_\theta, f_{\theta'})}} D(f_{\theta'}(\hat{x}), f_\theta(\hat{x})) = 0, \quad (4)$$

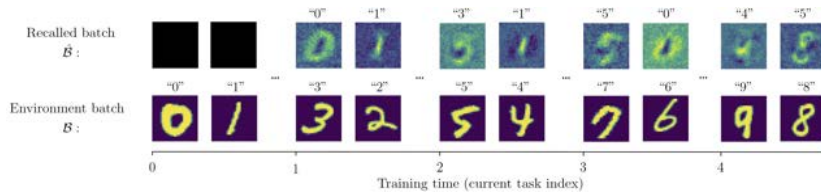


Figure 2: Real $\mathcal{B}$ and generated $\hat{\mathcal{B}}$ for MNIST. Images in the top row were initialized from the image below before being optimized by recall (eq. (8)).

where D indicates divergence; we use the symmetric Jensen-Shannon divergence. From eq. (4), we see that in order to suppress the forgetting of non-local knowledge, the maximally violating input should result in no violation. Hence the best knowledge preserving parameters in each training iteration on original batch $\mathcal{B}$ can be given by:

$$\theta^* = \arg \min_{\theta'} \left[\left(\frac{1}{N} \sum_{(x,y) \in \mathcal{B}} L(f_{\theta'}(x), y) \right) + \sup_{\substack{\hat{x} \in \mathcal{X} \\ \neg \text{local}(\hat{x}; \mathcal{B}, f_{\theta}, f_{\theta'})}} D(f_{\theta'}(\hat{x}), f_{\theta}(\hat{x})) \right], \quad (5)$$

where L denotes the original tasks loss, which is cross-entropy in our experiments. This can be approximated by:

$$\theta^* = \arg \min_{\theta'} \left[\left(\frac{1}{N} \sum_{(x,y) \in \mathcal{B}} L(f_{\theta'}(x), y) \right) + \lambda_0 L(f_{\theta'}(\hat{x}^*), f_{\theta}(\hat{x}^*)) \right] \quad (6)$$

$$\text{where } \hat{x}^* = \arg \max_{\substack{\hat{x} \in \mathcal{X} \\ \neg \text{local}(\hat{x}; \mathcal{B}, f_{\theta}, f_{\theta'})}} D(f_{\theta'}(\hat{x}), f_{\theta}(\hat{x})), \quad (7)$$

with $\hat{x}^*$ being the maximally violating input. The key idea is replaying maximally violating inputs with their old targets minimizes the maximal divergence between old model f_{θ} and new model $f_{\theta'}$, thus optimizing for not forgetting via eq. (4). Note no gradient exists for $\hat{x}^*$ until $\theta' \neq \theta$. This is what is meant by *automatic* (section 1); the optimization signal during recall (eq. (7)) and from recall (eq. (6)) scales with the change between θ and θ' . In practice, we fill a full replay batch of $M > 1$ images, $\hat{\mathcal{B}}_X$, by maximising:

$$\text{recall}(\hat{\mathcal{B}}_X; \mathcal{B}, f_{\theta}, f_{\theta'}) = \left[\frac{1}{M} \sum_{\hat{x} \in \hat{\mathcal{B}}_X} \left(D(f_{\theta'}(\hat{x}), f_{\theta}(\hat{x})) + \frac{\lambda_1}{C} \sum_{y \in \text{set}(\mathcal{B}_Y)} L(f_{\theta}(\hat{x}), y) + \frac{\lambda_2}{C} \sum_{y \in \text{set}(\mathcal{B}_Y)} L(f_{\theta'}(\hat{x}), y) \right) \right] + \text{reg}(\hat{\mathcal{B}}_X; f_{\theta}). \quad (8)$$

This is eq. (7) applied to each element of the batch with the constraint on local converted into regularization terms. $\hat{\mathcal{B}}_Y = [f_{\theta}(\hat{x}) : \hat{x} \in \hat{\mathcal{B}}_X]$, $\text{set}(\mathcal{B}_Y) = \{y : y \in \mathcal{B}_Y\}$ and $C = |\text{set}(\mathcal{B}_Y)|$. Regularization across the batch is provided by reg :

$$\text{reg}(\hat{\mathcal{B}}_X; f_{\theta}) = \lambda_3 \mathcal{H}(\hat{\mathcal{B}}_Y) - \left(\frac{\lambda_4}{M} \sum_{\hat{y} \in \hat{\mathcal{B}}_Y} L(\hat{y}, \arg \max(\hat{y})) \right) - \lambda_5 L2(\hat{\mathcal{B}}_X) - \lambda_6 TV(\hat{\mathcal{B}}_X), \quad (9)$$

comprising of entropy maximisation with sharpening to prevent duplication, and L2 norm and total variation minimization to smooth inputs [Mahendran and Vedaldi, 2015, Yin et al., 2019]. $\mathcal{H}$ denotes entropy. The full training procedure is given in algorithm 1 in Appendix.

3 Experiments

3.1 Quantitative results

We use standard ‘‘class-incremental’’ training and evaluation protocol on sequential CIFAR10-5, MiniImageNet-20 and MNIST-5k-5 [Aljundi et al., 2019a], where suffix indicates number of tasks. On CIFAR10 and MiniImageNet, replaying 100 recalled images with ARM outperforms experience replay (ER) with 100 real stored images, regardless of whether MIR is used for the latter (tables 1 and 5). On MiniImageNet, replaying 100 recalled images with ARM achieves comparable performance to ER with 500 stored real images, whilst more than halving the use of additional memory.

Unit lag is perhaps the most interesting variant of ARM from the perspective of biological plausibility, as the old network is the current network at the start of each training iteration, hence additional memory cost is 0. Performance drops, as using an extremely proximal version of parameters for distillation allows for more drift. However, ARM still outperforms ADI and standard distillation (table 2).

3.2 Self-reinforcing loop between task performance and recall

ARM reduces forgetting, observable in table 1 and fig. 4 where accuracies remain elevated beyond the training window of the task. In particular, we observed that the first task was simultaneously strongly recalled (fig. 3) and performed (fig. 4) throughout training, far beyond its own training window.

Method	+Sample mem. #images	MB	+Model mem. #params	MB	M	Accuracy	Forgetting
Naive SGD†	0	0	0	0	0	55.2 ± 5.0	-
ER	200	0.61	0	0	10	27.5 ± 1.2	50.5 ± 2.4
ER-MIR	200	0.61	0	0	10	29.8 ± 1.1	50.2 ± 2.0
iCarl (5 iter)	200	0.61	0	0	-	28.6 ± 1.2	49.0 ± 2.4
GEM	200	0.61	0	0	-	16.8 ± 1.1	73.5 ± 1.7
ER	100	0.31	0	0	10	22.4 ± 1.1	66.2 ± 6.2
ER-MIR	100	0.31	0	0	10	23.6 ± 0.9	61.4 ± 1.8
Naive SGD	0	0	0	0	0	15.0 ± 3.1	69.4 ± 4.7
GEN	0	0	8.63M	34.5	100	15.3 ± 0.5	61.3 ± 5.1
GEN-MIR	0	0	9.50M	38.0	40	15.3 ± 1.2	61.0 ± 1.2
Distill (LwF)	0	0	1.09M	4.38	100	19.2 ± 0.3	60.9 ± 3.9
ADI	0	0	1.09M	4.38	100	24.8 ± 0.9	12.0 ± 4.5
ARM	0	0	1.09M	4.38	100	26.4 ± 1.2	8.07 ± 5.2

Table 1: Sequential CIFAR10. † denotes stationary. Chance accuracy is 10.0.

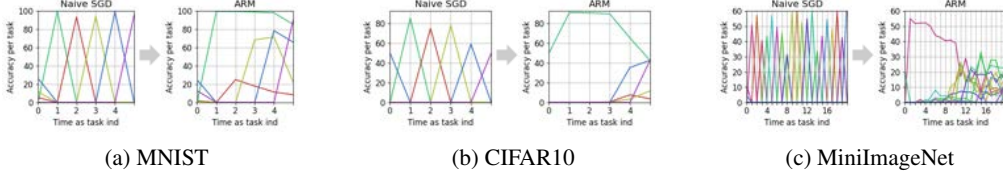


Figure 4: Recall causes characteristic changes to learning behavior. Tasks are retained beyond their training window and positive backwards transfer [Lopez-Paz and Ranzato, 2017] is observed.

The model tended to fixate on this strongly recalled task to the detriment of performance on other tasks, which were also not as strongly recalled. The interplay between performance and recall is expected as recall in ARM depends entirely on the implicit memory of the tasks model; if a class is not retained by the tasks model, optimization cannot recover meaningful samples for that class from the weights. This forms a parallel with the biological brain, where increased hippocampal replay leads to increased task performance [Schuck and Niv, 2019] and vice versa [Mattar and Daw, 2018].

3.2.1 Abstract nature of recall

Images from real $\mathcal{B}_X$ provide a favorable initialization in the optimization for maximally interfered $\hat{\mathcal{B}}_X$, since interference scales with the amount of feature overlap between them. For MNIST, the recalled images are clearly reminiscent of the class of their associated target (fig. 2); for CIFAR10 and MiniImageNet, this is not the case, as the recalled images resemble their initialized images with added noise. However, we found that training on recalled samples produced gradients that were consistently more correlated with those produced from training on real images from the target class than real images from the originator class, with strongest correlations in the highest layers (fig. 7). Thus we found that it was not necessary for input samples to empirically resemble a particular class for the representation of that class to be reinforced, and it was the upper or abstract layers of the network where the most meaningful representational changes were induced by recall. In the brain, mental imagery is thought not to propagate to the retina [Pearson et al., 2015], which has led to the idea of replay from intermediate levels in neural networks [van de Ven et al., 2020]. However, this requires fixing parameters unaffected by replay, due to lack of protection from forgetting. ARM demonstrates that replaying samples at input-output level, and thus protecting the trained network end-to-end, has effects that can nonetheless be characterized as abstract.

3.2.2 Natural avoidance of unseen classes

ARM does not include any constraints on avoiding unseen classes. Remarkably, we discover that it naturally does so, across diversity weights (empty lower triangles, fig. 3). This is likely because unseen classes have relatively indiscriminative representations, having thus far been trained not to activate, and are therefore de-prioritized in the optimization for samples with diverging outputs.

4 Conclusion

Avoiding catastrophic forgetting in artificial neural networks naturally gives rise to recall mechanisms reminiscent of human cognition. Internally generated, conditional replay is a promising approach that reduces catastrophic forgetting, memory complexity, and outperforms other forms of generative replay on natural images.

	Distill (LwF)	ADI	ARM
Baseline (CIFAR10)	19.2 ± 0.3	24.8 ± 0.9	26.9 ± 1.1
Unit lag ($T = 1$)	15.8 ± 2.6	14.4 ± 1.6	16.3 ± 1.8
No distill	-	20.0 ± 0.8	24.4 ± 1.7

Table 2: Testing distillation based methods: unit lag and removing standard distillation.

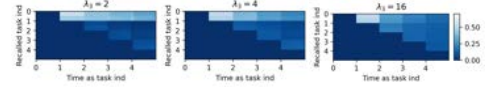


Figure 3: History of recalled classes during training for different diversity λ_3 , CIFAR10.

Broader Impact

This work deals with artificial neural networks internally generating imagined data. Research at the intersection of machine and biological intelligence has the potential to give back to neuroscience and further the understanding of our own cognition, which is a deeply compelling objective. Removing the need to explicitly store buffered data also helps algorithms overcome data protection requirements.

References

- Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.
- Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pages 11849–11860, 2019a.
- Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pages 11816–11825, 2019b.
- Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-recursal: Solving the catastrophic forgetting problem in deep neural networks. *arXiv preprint arXiv:1802.03875*, 2018.
- Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- Margaret F Carr, Shantanu P Jadhav, and Loren M Frank. Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature neuroscience*, 14(2):147, 2011.
- Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.
- Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc’Aurelio Ranzato. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486*, 2019.
- Sen Cheng and Loren M Frank. New experiences enhance coordinated neural activity in the hippocampus. *Neuron*, 57(2):303–313, 2008.
- Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.
- Robert M French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. 1991.
- Hagar Gelbard-Sagiv, Roy Mukamel, Michal Harel, Rafael Malach, and Itzhak Fried. Internally generated reactivation of single neurons in human hippocampus during free recall. *Science*, 322(5898):96–101, 2008.
- Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.
- Mattias P Karlsson and Loren M Frank. Awake replay of remote experiences in the hippocampus. *Nature neuroscience*, 12(7):913, 2009.

- Ronald Kemker and Christopher Kanan. Fearnnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- Hemant S Kudrimoti, Carol A Barnes, and Bruce L McNaughton. Reactivation of hippocampal cell assemblies: effects of behavioral state, experience, and eeg dynamics. *Journal of Neuroscience*, 19(10):4090–4101, 1999.
- Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7): 512–534, 2016.
- Frantzeska Lavda, Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Continual classification learning using generative models. *arXiv preprint arXiv:1810.10612*, 2018.
- Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017.
- Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019a.
- Timothée Lesort, Alexander Gepperth, Andrei Stoian, and David Filliat. Marginal replay vs conditional replay for continual learning. In *International Conference on Artificial Neural Networks*, pages 466–480. Springer, 2019b.
- Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.
- Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.
- Marcelo G Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617, 2018.
- Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
- Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- Joseph O’Neill, Timothy J Senior, Kevin Allen, John R Huxter, and Jozsef Csicsvari. Reactivation of experience-dependent cell assembly patterns in the hippocampus. *Nature neuroscience*, 11(2): 209–215, 2008.
- Joel Pearson, Thomas Naselaris, Emily A Holmes, and Stephen M Kosslyn. Mental imagery: functional mechanisms and clinical applications. *Trends in cognitive sciences*, 19(10):590–602, 2015.
- Jason Ramapuram, Magda Gregorova, and Alexandros Kalousis. Lifelong generative modeling. *Neurocomputing*, 2020.

- Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.
- Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesaro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.
- Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.
- Edmund T Rolls, Alessandro Treves, and Martin J Tovee. The representational capacity of the distributed encoding of information provided by populations of neurons in primate temporal visual cortex. *Experimental Brain Research*, 114(1):149–162, 1997.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pages 348–358, 2019.
- Jacob Russin, Randall C O’Reilly, and Yoshua Bengio. Deep learning needs a prefrontal cortex. 2020.
- Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- Nicolas W Schuck and Yael Niv. Sequential replay of nonspatial task states in the human hippocampus. *Science*, 364(6447):eaaw5181, 2019.
- Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017.
- Alyssa H Sinclair and Morgan D Barense. Surprise and destabilize: prediction error influences episodic memory reconsolidation. *Learning & Memory*, 25(8):369–381, 2018.
- Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.
- Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.
- Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-like replay for continual learning with artificial neural networks. 2020.
- Risto Vuorio, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim. Meta continual learning. *arXiv preprint arXiv:1806.06928*, 2018.
- Robert C Williamson and Uwe Helmke. Existence and uniqueness results for neural network approximations. *IEEE Transactions on Neural Networks*, 6(1):2–13, 1995.
- Justin N Wood. A smoothness constraint on the development of object recognition. *Cognition*, 153: 140–145, 2016.
- Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M Alvarez, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. *arXiv preprint arXiv:1912.08795*, 2019.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

A Appendix

A.1 Background

In this section we discuss four underlying principles of our method.

A.1.1 Distributed representations

ARM trains a single neural network for all tasks, avoiding tables and task-specific parameters, as these factors introduce enforced non-distributedness. Neural networks generally implement distributed representations, meaning inputs are represented by overlapping patterns. Distributedness allows for high memory efficiency [Bengio et al., 2013]; for example, n bits stores 2^n patterns if overlap is allowed and n if not. Overlap also means representations are shared between and thus optimized across tasks, minimising redundancy from duplication. Behaviour generalises to patterns not present in training, which is not true of non-distributed representations such as lookup tables [French, 1991].

Tables and buffers [Graves et al., 2014, Mnih et al., 2013, Chaudhry et al., 2019] are examples of non-distributed representations, having separated cells that allow for atomic updates without interference, but sacrificing on memory efficiency, transfer and generalisation [French, 1991, Bengio et al., 2013]. Rather than allowing the degree of distributedness to be naturally determined from optimization for an objective, as seen in neural network training, non-distributedness is enforced a priori. Dynamic architectures [Rusu et al., 2016] also exploit separation, as previous parameters can be isolated as tasks change, resulting in high memory complexity with respect to the number of tasks.

Data representation in the brain is distributed, where the number of stimuli that can be represented increases exponentially with the number of neurons [Rolls et al., 1997]. Distributedness varies, with for example regions implicated in fast-learning episodic memory exhibiting greater pattern separation [Kumaran et al., 2016], but on a continuous spectrum rather than with the hard difference seen between neural network and tabular representations.

A.1.2 One model for inference and recall

In ARM the invertibility of the tasks model, with its implicit memory of past training samples, is used to generate recalled samples. Buffers or models trained for sample generation are not used.

In contrast, replay methods typically require extra memory explicitly dedicated to the auxiliary task of sample memorisation, either in the form of buffers [Chaudhry et al., 2019, Aljundi et al., 2019a,b, Riemer et al., 2018, Rebuffi et al., 2017, Rolnick et al., 2019] or dedicated generators [Shin et al., 2017, Kamra et al., 2017, Lavda et al., 2018, Lesort et al., 2019b, Atkinson et al., 2018, Kemker and Kanan, 2017]. Some parameter-level constraints (i.e. non-replay) also make use of buffers [Lopez-Paz and Ranzato, 2017, Chaudhry et al., 2018]. Buffers being non-distributed are relatively memory inefficient. Generator networks have so far proved difficult to train given only non-stationary data in class-incremental settings [Lesort et al., 2019a, Aljundi et al., 2019a]. Notable exceptions with internal generation, with key differences to our method highlighted, include Adaptive DeepInversion [Yin et al., 2019] (generates full seen data distribution, so classes are arbitrarily pre-selected and representativeness of batch statistics is enforced), and Replay-through-Feedback [van de Ven et al., 2020, van de Ven and Tolias, 2018] (tasks model is overlaid with encoder of VAE, thus shares decoder parameters in symmetric case; again, sampling is intended to represent full distribution).

The brain also appears to share memory for inference and recall; namely the same neurons activated during experiences are re-activated during their recall [Gelbard-Sagiv et al., 2008, Carr et al., 2011].

A.1.3 Functional rather than parameter-level constraints

A neural network can have multiple instantiations with different parameter values that execute the same overall behavior [Williamson and Helmke, 1995]. Thus explicit constraints aimed at preventing changes in function should be specified at the functional level, i.e. input-output, rather than at parameter level, as noted in Ramapuram et al. [2020]. Note the two are related as functional-level constraints are implemented as parameter-level constraints in parametric models, and parameter-level constraints also constrain function; the difference lies in that a functional-level constraint can be implemented as multiple parameter-level constraints but not vice-versa. The stringency of parameter

level regularization may explain its underperformance compared to replay [van de Ven and Tolias, 2019, Chaudhry et al., 2019]. ARM offers the benefit of being data-free, like some parameter level regularizers, while being a replay method and therefore constraining at functional level.

Like ARM, LwF [Li and Hoiem, 2017] is an example of imposing functional constraints without needing to explicitly store the seen data distribution. Unlike ARM, the replay samples are taken from the current batch without optimization. Similarly, ARM is a form of distillation [Hinton et al., 2015].

Parameter-level methods include EWC [Kirkpatrick et al., 2017], SI [Zenke et al., 2017], GEM [Lopez-Paz and Ranzato, 2017], A-GEM [Chaudhry et al., 2018], Meta Continual Learning [Vuorio et al., 2018], IMM [Lee et al., 2017], Memory Aware Synapses [Aljundi et al., 2018]. These methods generally explicitly anchor either the values or gradients of parameters from excessively changing.

From an evolutionary perspective, it is overall behavior that is directly implicated for survival, as opposed to the local behaviour of specific neurons. For example, a major influence on representations in the brain is optimization for rewards from tasks [Miller and Cohen, 2001].

A.1.4 Conditionality of recall and top-down modulation

In ARM, recalled samples are optimized given model snapshots separated by training on the current real batch, and are thus conditioned on the latter. Thus we avoid the replay-all-seen-tasks paradigm that is inefficient and unsustainable for large numbers of tasks. Furthermore, by generating subsequent training inputs by backpropagation - analogous to a forward pass in the opposing direction - ARM exhibits literal top-down modulation of subsequent activations and learning within the tasks network.

Few generative replay methods (appendix A.1.2) consider conditionality on the temporally local environment. Rather, most aim to reproduce the full seen data distribution, with the idea that training a network with replay simulates stationary training when replay is representative [Shin et al., 2017]. Thus randomization is used to select from buffers or choose codes or classes for decoding samples. In contrast, Aljundi et al. [2019a] is a replay method that selectively replays based on prediction change caused by training on the current real batch. ARM provides a theoretical argument for why this is reasonable, as well as using internal generation instead of external buffers or generators.

Human recall is also a response conditioned on the environment as opposed to uniformly sampled across all knowledge. Memory retrieval is conditioned on current sensory input [Carr et al., 2011]. Top-down modulation is a form of conditionality where activations representing higher-level or downstream concepts influence the activations of lower-level or upstream representations, reversing the bottom-up processing direction [Miller and Cohen, 2001]. It is believed that many of the cognitive capabilities associated with the prefrontal cortex, namely various forms of reasoning, depend on its top-down attentional modulation of other areas, conditioned on the current goal [Miller and Cohen, 2001, Russin et al., 2020]. Note that the backpropagation algorithm itself can be seen as a form of attentional modulation of lower-level representations based on higher-level activations. However, unlike the brain, which is highly recurrent [Kumaran et al., 2016], this top-down modulation is truncated in that it does not yield subsequent forward pass activations nor additional learning steps. In ARM, this is corrected as an initial learning step produces recalled inputs, yielding subsequent activations and an additional learning step for reinforcing these activations. This allows for a chain of inference induced by training on the environment that is longer than 1 pass over memory.

Algorithm 1: Automatic Recall Machines with distillation

```

1 Require: randomly initialized  $f_\theta$ , data  $D$ , lag  $T$ , batch sizes  $N$  and  $M$ , steps  $S$ , rates  $\eta_{0,1}$  and  $\lambda_{0...6}$ .
2  $\theta' = \theta$ 
3 for  $t \in [0, |D|)$  do
4    $\mathcal{B} = D_t$ 
5    $\theta' = \theta' - \eta_0 \nabla_{\theta'} \frac{1}{N} \sum_{(x,y) \in \mathcal{B}} L(f_{\theta'}(x), y)$  ▷ inference and learning on  $D$ 
6    $\hat{\mathcal{B}}_X = \mathcal{B}_X[i_{0..M-1}]$ ,  $i \sim U(0, N-1)$ 
7   for  $s \in [0, S)$  do
8      $\hat{\mathcal{B}}_X = \hat{\mathcal{B}}_X + \eta_1 \nabla_{\hat{\mathcal{B}}_X} \text{recall}(\hat{\mathcal{B}}_X; \mathcal{B}, f_\theta, f_{\theta'})$  ▷ infer recalled samples
9      $\theta' = \theta' - \eta_0 \nabla_{\theta'} \frac{1}{M+N} \sum_{\hat{x} \in \hat{\mathcal{B}}_X \cup \mathcal{B}_X} \lambda_0 L(f_{\theta'}(\hat{x}), f_\theta(\hat{x}))$  ▷ learn on recalled samples
10    if  $(t+1) \% T = 0$  then
11       $\theta = \theta'$ 

```

A.2 Datasets

Dataset	Tasks	Classes	Classes per task	#Train	#Test	#Val	Batch size $ \mathcal{B} $
CIFAR10	5	10	2	47.5K	10K	2.5K	10
MiniImageNet	20	100	5	45.6K	12K	2.4K	10
MNIST	5	10	2	5K	10K	3K	10

Table 3: Dataset statistics.

Datasets are summarized in table 3 and follow Aljundi et al. [2019a]. Pre-processing was not used except resizing MiniImageNet images to 84x84 as standard.

A.3 Evaluation

Average accuracy and forgetting (average drop in accuracy) were measured on a held-out test set at the end of training on all tasks, as defined in Chaudhry et al. [2019].

Average accuracy. Let $a_{i,j}$ denote performance on the test set of task j after training on task i . The average accuracy after task T is:

$$A_T = \frac{1}{T} \sum_{j=1}^T a_{T,j}. \quad (10)$$

Forgetting. Average forgetting after task T is:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} f_j^T \quad (11)$$

$$\text{where } f_j^i = \max_{l \in \{1, \dots, i-1\}} a_{l,j} - a_{i,j}. \quad (12)$$

A.4 Architectures and memory computations

Following Aljundi et al. [2019a], ResNet18 was used for CIFAR10 and MiniImageNet, and a multi-layer perceptron with two hidden layers was used for MNIST. The number of parameters for each is given in tables 1, 5 and 6. To compute auxiliary memory usage, we assumed 3 bytes (RGB) or 1 byte (grayscale) per pixel for sample memory, and 4 bytes (single precision floating point) per parameter for model memory.

A.5 Hyperparameters

Dataset	η_0	η_1	$\lambda_0 : \tilde{\mathcal{B}}_X$	$\lambda_0 : \mathcal{B}_X$	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	S
MNIST	0.05	25.0	1.0	1.0	1.0	0.1	16.0	0.1	1.0	1.0	10
CIFAR10	0.01	10.0	1.0	1.0	1.0	0.1	16.0	0.1	1.0	1.0	10
MiniImageNet	0.01	10.0	1.0	2.0	1.0	0.1	1.0	1.0	1.0	1.0	10

Table 4: Hyperparameters.

Hyperparameter values were selected based on performance on a held-out validation set. A grid search was conducted on CIFAR10 for $\lambda_{1...4}$ and other values were finalized by ablation. Hyperparameters for ARM experiments in tables 1, 5 and 6 are given in table 4. Experiments in table 2 use the same values except $\lambda_4 = 1.0$ for no distill, and an additional weight of 8.0 on D (eq. (8)) for unit lag, as we found it was beneficial to increase the emphasis on maximal divergence in this case. Hyperparameter values for ADI and LwF were determined in the same manner. The implementation of Aljundi et al. [2019a] was used for ER and GEN experiments.

A.6 Code

The implementation can be found at www.github.com/xu-ji/ARM.

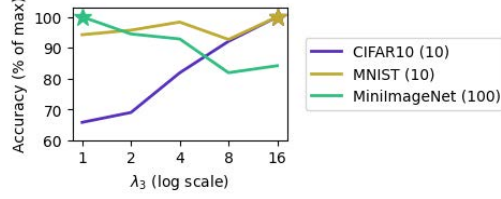


Figure 5: Optimal diversity weight ($\star$) is negatively correlated with the number of classes.

A.7 Experiments

A.7.1 Protocol

Tasks are sequential and formed by partitioning training classes equally. Output classes are unconstrained throughout training and evaluation (single head, Farquhar and Gal [2018]). Batch size is kept constant as it affects effective task training length. As in Aljundi et al. [2019a], replay begins from the end of the 1st task. The update lag T is set as the number of training iterations per task (equivalently, updating when a batch contains novel classes) [Aljundi et al., 2019a] but we also test unit lag. Hyperparameters are selected using validation set, evaluation uses test set, and our experiments are repeated 5 times. Hyperparameters and ablations are given in the Appendix.

Few generative methods target class-incremental learning on natural images, which is very difficult in our online single-pass setting. Other works generally test online generation on digits (MIR [Aljundi et al., 2019a], GEN [Shin et al., 2017]), pre-processed features [Kemker and Kanan, 2017], or avoid class-incremental learning by viewing entire datasets as a single task (ADI [Yin et al., 2019]).

A.7.2 Tunable diversity of recall

We use almost identical hyperparameters across all three datasets, as in most cases the same values work well. Varying diversity via λ_3 biases recall towards sparse or dense (fig. 3). High $\lambda_3 = 16$ is optimal for MNIST and CIFAR10 while low $\lambda_3 = 1$ is optimal for MiniImageNet (fig. 5), which benefits more from selectivity as the space of possible choices is much larger. With a low diversity weight on CIFAR10, we observed a natural prioritization of recent classes (fig. 3, left). This is likely due to reduced but present forgetting causing a decrease in the discriminativeness of older class representations relative to recent classes, resulting in the latter being highlighted in the optimization for samples with diverging outputs. In the brain, replay is also most prevalent immediately following an experience, and decays with time [Kudrimoti et al., 1999, Karlsson and Frank, 2009].

A.7.3 Benefits of optimized selection and distributed memory

We hypothesized that the performance boost from allowing classes to be selective (ARM, *-MIR) as opposed to random (ADI, ER, GEN) would be most obvious with a large number of classes. The benefit of distributed (ARM, ADI, GEN-*) over non-distributed (ER-*) data stores should also be more obvious due to the lower memory complexity of distributed representations (appendix A.1.1).

Method	+Sample mem. #images	+Model mem. MB	M	Accuracy	Forgetting
Naive SGD†	0	0	0	25.2 ± 4.7	-
ER	500	10.6	0	5.60 ± 0.4	55.2 ± 1.2
ER-MIR	500	10.6	0	5.40 ± 0.4	55.8 ± 0.9
ER	100	2.12	0	4.20 ± 0.4	58.6 ± 0.9
ER-MIR	100	2.12	0	4.00 ± 0.0	59.2 ± 0.9
Naive SGD	0	0	0	3.80 ± 0.2	51.2 ± 2.0
GEN	0	0	27.1M	2.33 ± 0.3	34.3 ± 0.7
GEN-MIR	0	0	27.1M	2.00 ± 0.0	35.3 ± 0.5
Distill (LwF)	0	0	1.16M	4.63 ± 0.2	41.6 ± 1.6
ADI	0	0	1.16M	2.99 ± 0.6	8.11 ± 2.1
ARM	0	0	1.16M	5.53 ± 0.8	11.9 ± 2.7

Table 5: Sequential MiniImageNet. † denotes stationary. 3 task iterations made, as with Aljundi et al. [2019a]; stationary uses 3 epochs. Chance accuracy is 1.0.

Method	+Sample mem. #images	+Model mem. MB	M	Accuracy	Forgetting
Naive SGD†	0	0	0	87.1 ± 0.6	-
ER	50	0.04	0	62.8 ± 3.1	42.0 ± 3.7
ER-MIR	50	0.04	0	63.8 ± 4.6	40.6 ± 5.9
ER	25	0.02	0	51.6 ± 2.7	57.0 ± 3.3
ER-MIR	25	0.02	0	51.6 ± 2.6	56.4 ± 3.3
Naive SGD	0	0	0	18.8 ± 0.5	95.6 ± 2.7
GEN	0	0	1.14M	4.58 ± 0.6	19.5 ± 0.8
GEN-MIR	0	0	1.08M	4.31 ± 0.3	17.0 ± 0.4
Distill (LwF)	0	0	478K	1.91 ± 2.5	58.0 ± 1.7
ADI	0	0	478K	1.91 ± 2.6	11.5 ± 5.0
ARM	0	0	478K	1.91 ± 2.6	21.8 ± 1.6

Table 6: Sequential MNIST. † denotes stationary. Chance accuracy is 10.0.

Unfortunately, datasets with more classes are also more difficult in general, with lower baseline or chance performance. However, we were able to verify both these hypotheses on the 100 classes of MiniImageNet (table 5). The 2.5% accuracy boost from ARM compared to ADI equates to the learning of 2.5 additional classes or 50% of a task. In contrast, the boost on MNIST equates to only a tenth of a class or 5% of a task (table 6). On MiniImageNet, storing the old model costs 4.6MB but achieves comparable performance to storing 500 images at a cost of 10.6MB.

A.7.4 Lower memory complexity than external generative replay

External generative replay methods GEN and GEN-MIR perform worse than ARM on natural images (tables 1 and 5), despite incurring greater memory costs. This is because 3 additional networks must be stored: old and new versions of the generator network and an old version of the tasks network, whereas ARM uses only the latter. In particular, GEN and GEN-MIR struggled on MiniImageNet and underperformed the naive SGD baseline. These results support the hypothesis that for online continual learning on datasets of moderate to high complexity, obtaining meaningful samples from the existing tasks model is easier than training a separate generator.

A.7.5 Ablation

	Accuracy	Forgetting
Baseline (CIFAR10)	26.4 $\pm$ 1.2	8.07 $\pm$ 5.2
$\lambda_1 = 0, \lambda_2 = 0$	22.8 $\pm$ 1.8	29.1 $\pm$ 2.0
$\lambda_3 = 0$	15.3 $\pm$ 1.3	4.17 $\pm$ 0.9
$\lambda_4 = 0$	25.7 $\pm$ 2.0	6.69 $\pm$ 5.4
$\lambda_5 = 0$	25.8 $\pm$ 1.2	6.95 $\pm$ 5.1
$\lambda_6 = 0$	26.4 $\pm$ 1.7	6.36 $\pm$ 3.7
$M = 150$ (+50)	26.3 $\pm$ 0.9	16.5 $\pm$ 5.5
$M = 50$ (-50)	24.4 $\pm$ 1.2	5.71 $\pm$ 4.5
$S = 20$ (doubled)	25.1 $\pm$ 1.1	8.11 $\pm$ 7.2
$S = 5$ (halved)	24.6 $\pm$ 1.5	13.5 $\pm$ 8.4
Cross-entropy as D	24.6 $\pm$ 1.2	24.9 $\pm$ 5.2
Random noise init $\tilde{\mathcal{B}}_X$	16.0 $\pm$ 1.4	4.00 $\pm$ 1.1
Recall 2x per t	17.5 $\pm$ 0.6	0.57 $\pm$ 0.5
Recall 4x per t	17.4 $\pm$ 1.0	0.44 $\pm$ 0.9

Table 7: Ablation study on CIFAR10. Each result makes a change from the baseline.

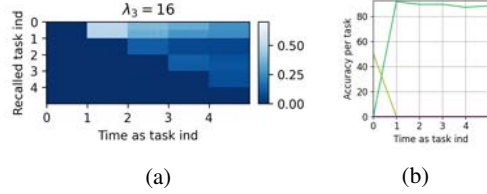


Figure 6: CIFAR10 with 2 recalls per timestep t . Contrast with fig. 3 right, and fig. 4b.

The contributions of different ARM implementation details were measured with ablation (table 7). Among the least important was image regularization; L2 norm (λ_5) and TV minimization (λ_6) were added mainly for fairness with ADI, which also uses them. ARM still outperformed ADI without them. The most important factors were initializing recalled samples from the real batch instead of random noise, and using entropy weight (λ_3) to minimize duplication.

M is used to denote auxiliary replay batch size. Note the material “size” for replay is not M , but additional persistent memory size. Deliberate subsampling of the buffer (M less than size of buffer) is used in Aljundi et al. [2019a], which prevents identical replay batches across training iterations. This does not apply to ARM as replay images are not sampled from a fixed buffer. On CIFAR10, we found an increase in performance up to $M = 100$ and diminishing returns thereafter.

Our implementation does not provide a formal guarantee of not forgetting as we only recall once per training iteration, taking a single learning step towards minimizing divergence (algorithm 1, line 9). We tested increasing the number of training steps from recall, looping from line 9 to line 6 in algorithm 1. This was found to lead to very strong recall of the first task throughout the training sequence, despite high λ_3 (fig. 6), and decreased overall performance (table 7). ARM typically demonstrates a warm-up phase with fixation on the first task gradually reducing to accommodate other tasks (fig. 4), which was suppressed in this case (appendix A.7.5). This result underlines the need to consider both forgetting and accuracy, as a near eradication of forgetting ($< 1\%$ average drop in accuracy, table 7) is not necessarily indicative of optimal overall performance. In summary, balanced training with one recall batch per real batch was found to perform best in our experiments.

A.7.6 Gradients study

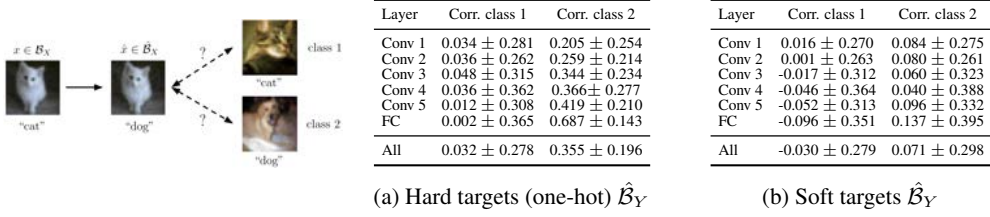


Figure 7: Despite $\hat{x}$ resembling class 1 at pixel level (a), training on $\hat{x}$ produces gradients more correlated with training on real images of class 2 than real images of class 1, with correlation being strongest in the highest layers of the network. Hard targets (b) are considered as well as soft (c) for fairness, as real samples use hard targets. Results shown on CIFAR10 using 1K random samples of $\hat{x}$.

As discussed in section 3.2.1, training on recalled samples produced gradients that were consistently more correlated with those produced from training on real images from the target class than real images from the originator class, in particular in the highest layers (fig. 7), even when recalled samples resembled the originator class at pixel level to the human eye. This was observed across all datasets and architectures, including both convolutional ResNet and multilayer perceptrons. Gradients within each layer were normalized for correlation, so this behavior is not attributable to varying magnitudes. Rather, it indicates that similarities in output targets have greater influence than similarities in input space in the computation of gradients, supporting the argument that recalled targets are material and should be optimized rather than arbitrary (appendix A.1.4). In short, recalled samples can be deceptive at pixel level and require an analysis of gradients to understand the relation to training on real images.

Computing correlations for fig. 7 and tables 9 and 10 involved gradients computed from 1K real samples and 1K recalled samples per dataset. The underlying models were taken from the end of training and parameters were fixed. Gradients for each parameter block (i.e. weights and biases) were linearized and normalized. 2 dot products were computed per parameter block and recalled sample, between the gradients induced by recall and the gradients induced by training on each of 2 randomly selected real samples belonging to the recalled sample’s originator class (class 1) and target class (class 2) respectively. Results were collected per layer and averaged over recalled samples and parameter blocks.

Layer	Corr. class 1	Corr. class 2
Conv 1	0.061 ± 0.316	0.246 ± 0.287
Conv 2	0.032 ± 0.274	0.282 ± 0.241
Conv 3	0.028 ± 0.326	0.414 ± 0.258
Conv 4	-0.031 ± 0.264	0.474 ± 0.198
Conv 5	-0.181 ± 0.195	0.556 ± 0.142
FC	-0.014 ± 0.084	0.726 ± 0.084
All	-0.036 ± 0.218	0.439 ± 0.177

Layer	Corr. class 1	Corr. class 2
Conv 1	0.065 ± 0.319	0.212 ± 0.302
Conv 2	0.022 ± 0.273	0.262 ± 0.259
Conv 3	0.034 ± 0.315	0.381 ± 0.273
Conv 4	-0.040 ± 0.247	0.433 ± 0.216
Conv 5	-0.200 ± 0.190	0.505 ± 0.186
FC	-0.019 ± 0.070	0.655 ± 0.156
All	-0.043 ± 0.212	0.401 ± 0.197

(a) Hard targets (one-hot) $\hat{B}_Y$ (b) Soft targets $\hat{B}_Y$

Table 9: MiniImageNet.

Layer	Corr. class 1	Corr. class 2
FC 1	0.022 ± 0.178	0.255 ± 0.176
FC 2	0.012 ± 0.168	0.325 ± 0.228
FC 3	0.002 ± 0.264	0.675 ± 0.160
All	0.012 ± 0.181	0.418 ± 0.181

Layer	Corr. class 1	Corr. class 2
FC 1	0.013 ± 0.182	0.055 ± 0.196
FC 2	0.015 ± 0.207	0.072 ± 0.246
FC 3	0.004 ± 0.311	0.100 ± 0.293
All	0.011 ± 0.216	0.076 ± 0.233

(a) Hard targets (one-hot) $\hat{B}_Y$ (b) Soft targets $\hat{B}_Y$

Table 10: MNIST.

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

Title of Paper	Automatic Recall Machines: Internal Replay, Continual Learning and the Brain
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	Xu Ji, Joao Henriques, Tinne Tuytelaars, Andrea Vedaldi. Automatic Recall Machines: Internal Replay, Continual Learning and the Brain. Published in the Workshop on BabyMind at the Conference on Neural Information Processing Systems (NeurIPS) 2021.

Student Confirmation

Student Name:	Xu Ji		
Contribution to the Paper	The initial idea was mine and I performed all experiments. I wrote most of the paper, with feedback on experiments and writing from Tinne Tuytelaars, Andrea Vedaldi and Joao Henriques.		
Signature		Date	24/05/2021

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Andrea Vedaldi			
Supervisor comments The description is accurate.			
Signature		Date	25/05/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

7

On the Role of Neurogenesis in Overcoming Catastrophic Forgetting

This paper was published in NeurIPS Workshop on Continual Learning 2018.

On the role of neurogenesis in overcoming catastrophic forgetting

German I. Parisi
Apprente, Inc.
Mountain View, CA, USA
german@apprente.com

Xu Ji
University of Oxford
Oxford, UK
xuji@robots.ox.ac.uk

Stefan Wermter
University of Hamburg
Hamburg, Germany
wermter@informatik.uni-hamburg.de

Abstract

Lifelong learning capabilities are crucial for artificial autonomous agents operating on real-world data, which is typically non-stationary and temporally correlated. In this work, we demonstrate that dynamically grown networks outperform static networks in incremental learning scenarios, even when bounded by the same amount of memory in both cases. Learning is unsupervised in our models, a condition that additionally makes training more challenging whilst increasing the realism of the study, since humans are able to learn without dense manual annotation. Our results on artificial neural networks reinforce that structural plasticity constitutes effective prevention against catastrophic forgetting in non-stationary environments, as well as empirically supporting the importance of neurogenesis in the mammalian brain.

1 Introduction

Artificial agents and autonomous robots should be able to learn in a lifelong manner, continually acquiring and fine-tuning knowledge through interaction with the environment [1]. Since real-world data is naturally non-stationary - for example, video may focus on different scenes or objects in adjacent temporal blocks - it is desirable for learning algorithms to operate on non-stationary input distributions without suffering from *catastrophic forgetting* [2], i.e. forgetting modes of the function that had previously been retained. For instance, if a trained object classification model is subsequently shifted to a different set of objects, it would proceed to forget the original set due to a representation shift, which is quantitatively observable via decreasing classification accuracy on previously learned classes. In addition to catastrophic forgetting, the requirement for dense human annotations used by standard regression techniques is also undesirable as such labels are typically not present in real-world learning scenarios. Furthermore, since storing previously encountered data samples is expensive, learning algorithms should be able to replay learned schema to consolidate internal representations, allowing for continual learning in the absence of external sensory stimulation [3].

The mammalian brain remains the best model of unsupervised incremental learning, which makes biologically-inspired learning algorithms a compelling approach. Proposed solutions for mitigating catastrophic interference in this domain have focused on regulating intrinsic levels of plasticity to protect acquired knowledge given a fixed number of neurons [5, 6, 13], dynamically allocating new neurons to accommodate novel dissimilar knowledge [7, 8, 16], and sample replay to simulate a stationary distribution in its absence [9, 15]. The general notion of structural plasticity is widely used across machine learning literature, for example in fine-tuning learned networks by appending

new layers [16], making it a promising solution to lifelong learning in its own right, even when disregarding biological desiderata. In previous work [9], we introduced a self-organizing neural architecture which combines the latter two ideas, updating synaptic connectivity patterns using an unsupervised competitive Hebbian learning rule, augmenting input via memory replay, and growing new neurons when activation levels fall short of an empirically defined neural activity threshold. The growing self-organizing network is composed of a dynamic set of recurrent neurons for learning the spatiotemporal structure of input sequences and has been successfully applied to incremental object and action recognition from videos [8, 9]. However, the question arises whether the process of neurogenesis functionally contributes to achieving better performance or if the same performance could be achieved by the use of sufficiently large networks in similar experimental set-ups.

In this work, we disentangle the factor of neurogenesis in mitigating catastrophic forgetting. We test GWR on both static and growing architectures, with the capacity of the latter bounded by the size of the former for fairness, otherwise any improvements in performance could be attributed to a greater memory capacity. Allowing neurogenesis results in an accuracy improvement on the CORe50 dataset [10] for incrementally learned object recognition, compared to a static network with memory equal to the maximum bound, showing that neurogenesis functionally contributes to preventing forgetting. The network trained with neurogenesis was also able to retain newly trained classes faster, which is particularly interesting given that the *stability-plasticity dilemma* suggests lower retention rates are required to mitigate forgetting [4], as fast retention contravenes representation stability.

2 Growing Self-Organizing Memory

Neurogenesis and synaptic rewiring via structural plasticity play a crucial role in the formation of memory from episodic experience [11]. A biological function of structural plasticity is to increase information storage efficiency in terms of space and energy demands. The definition of storage capacity is the number of memory associations represented by activity patterns at different times, so that memories are identified with patterns of neural activity that can be updated, stabilized, and evoked by modified synaptic connectivity. From a modeling perspective, neural networks that grow neurons and develop connectivity patterns over time have the advantage of mitigating the disruptive interference of existing internal representations when learning from novel sensory observations.

Here, we focus on growing recurrent self-organizing networks that learn the spatiotemporal structure of the input in an unsupervised fashion. The recurrent Grow-When-Required (GWR) network [8] can dynamically grow or shrink in response to the input distribution. New neurons will be created to better represent the input and connections between neurons will develop according to competitive Hebbian learning, i.e. neurons that are activated simultaneously will be connected to each other. The network is composed of a dynamic set of neurons, A , with each neuron consisting of a weight vector $\mathbf{w}_j$ and a number K of context descriptors $\mathbf{c}_{j,k}$ ($\mathbf{w}_j, \mathbf{c}_{j,k} \in \mathbb{R}^n$). Given the input $\mathbf{x}(t) \in \mathbb{R}^n$, the index of the best-matching unit (BMU), b , is computed as:

$$b = \arg \min_{j \in A} (d_j), \quad (1)$$

$$d_j = \alpha_0 \|\mathbf{x}(t) - \mathbf{w}_j\|^2 + \sum_{k=1}^K \alpha_k \|\mathbf{C}_k(t) - \mathbf{c}_{j,k}\|^2, \quad (2)$$

$$\mathbf{C}_k(t) = \beta \cdot \mathbf{w}_b^{t-1} + (1 - \beta) \cdot \mathbf{c}_{b,k}^{t-1}, \quad (3)$$

where $\|\cdot\|^2$ denotes the Euclidean distance, α_i and β are constant values that modulate the influence of the temporal context, $\mathbf{w}_b^{t-1}$ is the weight vector of the BMU at $t - 1$, and $\mathbf{C}_k \in \mathbb{R}^n$ is the global context of the network with $\mathbf{C}_k(t_0) = 0$. If $K = 0$, then Eq.2 resembles the learning dynamics of the standard GWR without temporal context [12]. For a given input $\mathbf{x}(t)$, the activity of the network, $a(t)$, is defined in relation to the distance between the input and its BMU (Eq. 1) as follows:

$$a(t) = \exp(-d_b), \quad (4)$$

thus yielding the highest activation value of 1 when the network can perfectly match the input sequence ($d_b = 0$). Each neuron is equipped with a habituation counter $h_j \in [0, 1]$ expressing how frequently it has fired based on a simplified model of how the efficacy of a habituating synapse reduces over time. Newly created neurons start with $h_j = 1$, with the habituation counter of the

BMU, b , and its neighboring neurons, n , iteratively decreased towards 0. The habituation rule [12] for habituating a neuron i is given by:

$$\Delta h_i = \tau_i \cdot \kappa \cdot (1 - h_i) - \tau_i, \quad (5)$$

with $i \in \{b, n\}$ and where τ_i and κ are constants that control the monotonically decreasing behavior. Typically, h_b is habituated faster than h_n by setting $\tau_b > \tau_n$. The network is initialized with two neurons, which is the minimum required for the discriminative ability.

At each learning iteration, a new neuron is created whenever the activity of the network, $a(t)$, in response to the input $\mathbf{x}(t)$ is smaller than a given insertion threshold a_T . Furthermore, h_b must be smaller than a habituation threshold h_T in order for the insertion condition to hold, thereby fostering the training of existing neurons before new ones are added. The new neuron is created halfway between the BMU and the input. The training of the existing neurons is carried out by adapting the BMU b and the neurons n to which the BMU is connected:

$$\Delta \mathbf{w}_i = \epsilon_i \cdot h_i \cdot (\mathbf{x}(t) - \mathbf{w}_i), \quad (6)$$

$$\Delta \mathbf{c}_{i,k} = \epsilon_i \cdot h_i \cdot (\mathbf{C}_k(t) - \mathbf{c}_{i,k}), \quad (7)$$

with $i \in \{b, n\}$ and where ϵ_i is a constant learning rate ($\epsilon_n < \epsilon_b$). Furthermore, the habituation counters of the BMU and the neighboring neurons are updated according to Eq. 5. Connections between neurons are updated on the basis of neural co-activation, i.e. when two neurons fire together, a connection between them is created if it does not yet exist.

Our previously reported experiments [9] show that growing self-organizing networks outperform a variety of proposed supervised lifelong learning approaches [5,10,13,14] evaluated on the CORE50 dataset [10] (see Table 1 and 2 in Supplementary Material for a quantitative comparison with other approaches). The performance improved partly due to the recurrent learning dynamics of the networks that better capture the spatiotemporal structure of the input sequences and the replay of statistically significant neural activity patterns to mitigate the catastrophic forgetting during incremental learning. The latter mechanism relies on the periodic re-activation of previously learned neural patterns. This allows the consolidation of internal representations in the absence of external sensory input. However, an empirical assessment of the contribution of neurogenesis to the overall performance has not been conducted. (For details on the generation of neural activation trajectories and input sequence classification, see Sec. 5 in Supplementary Material.)

3 Experiments

We performed a series of experiments with two different models, a growing self-organizing network and a static self-organizing network (see Fig. 2.A in Supplementary Material). There are no separate training and test phases, i.e. the model learns and triggers behavior at the same time. For our catastrophic forgetting experiments, sequential input becomes incrementally available over time and direct access to previously encountered data samples is restricted as each data sample is seen only once. The maximum number of neurons for both models is equal, i.e. the growing network starts with two neurons and incrementally grows (as described in Sec. 2) with N_{max} being the upper-bound number of neurons, whereas the static network will be trained with a fixed number N_{max} of randomly initialized neurons.

We evaluated the performance of the models on the CORE50 dataset for continuous object recognition [10]. The CORE50 comprises 50 objects (instances) within 10 categories with image sequences captured in 11 different sessions containing multiple views of the same objects, varying background, object pose, and degree of occlusion (Fig. 2.B). We used 128×128 RGB images at a frame rate of 5Hz. For a direct comparison with previously reported experiments [9, 10], the feature extraction module consists of a VGG model pre-trained on the ILSVRC-2012 dataset to which we applied a convolutional operation with $256 \times 1 \times 1$ kernels on the output of the fully-connected hidden layer, reducing its dimensionality from 2048 to 256. Therefore, the self-organizing network receives a 256-dimensional feature vector per sequence frame. We conducted our experiments on the task of instance-level object classification, using the samples from sessions #3, #7, #10 for testing and the samples from the remaining 8 sessions for training.

For comparison, we first trained static and dynamic networks with standard iid-sampled batches, increasing N_{max} from 500 to 2500 in intervals of 100. In general, a greater number of neurons

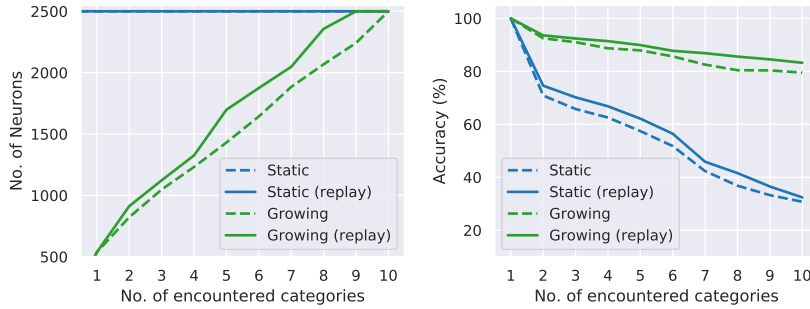


Figure 1: Incremental learning of object instances with a static and a growing model. Accuracy averaged across 10 learning trials with randomly shuffled training sessions.

corresponded to a better overall accuracy. For details see Table 3 and Figure 3. These experiments demonstrate that the static networks took longer to achieve competitive performance compared to dynamic networks (33 epochs versus 16), because the latter are able to add new neurons initialized between the input and BMUs, i.e. quickly interpolating the input data instead of needing to be iteratively adapted. Fast insertion does not lead to catastrophic interference with the learned representation since at the point of insertion, the rest of the neural representation does not depend on the weights of the novel neuron (see [9] for an extensive discussion).

In the incremental learning experiment, we trained the models with mini-batches containing 5 instances per category, with one training iteration per mini-batch (i.e. previously encountered mini-batches are not shown again to the model). We trained the models with and without memory replay and set $N_{max} = 2500$ for a comparison with the batch learning strategy. Fig. 1 shows the average number of neurons and overall accuracy across 10 trials of randomly shuffled acquisition sessions for the incremental learning task. For both models, the overall accuracy decreases as the number of encountered categories increases. However, the static model shows the worst performance since one training epoch per mini-batch is not sufficient to consolidate the representations. While in the batch learning strategy the static model required a larger number of epochs to achieve a similar performance to the dynamic model, in the incremental case the former leads to significant disruptive interference as the number of encountered categories increases. After having encountered 10 categories, the static model shows an overall accuracy of 32.41% with respect to the 83.24% achieved by the dynamic model (i.e. a difference of approx. 50%). The use of memory replay did not significantly improve the performance of the static network. This is because memory replay relies on the retrieval of neural activation trajectories which, in the case of the static network, do not represent the input well.

4 Open Issues and Future Work

We showed that self-organizing networks with additive neurogenesis show a better performance than a static network with the same number of neurons, thereby providing insights into the design of neural architectures in incremental learning scenarios when the total number of neurons is fixed. However, there are additional research directions to complement this study. First, we have focused on the process of neurogenesis whereas we have not investigated the dynamic development of synaptic patterns. Neurophysiological studies evidence that network connectivity is a crucial parameter because the ability of a network to retain memories scales in proportion to the number of synapses per neuron [11]. Consequently, one could quantitatively assess the contribution of different degrees of network connectivity and representational sparsity. Second, we have considered additive neurogenesis without investigating the aging and removal of rarely activated neurons and connections. This aspect is crucial for allowing to forget (not catastrophically) in networks of limited size. Finally, additional theories suggest that one function of neurogenesis in the mammalian hippocampus is the encoding of time for the formation of temporal associations in memory [17]. While it has been shown that unsupervised multisensory binding is possible through the use of unisensory self-organizing networks, e.g., from co-occurring audio-visual input [18], the use of neurogenesis for temporal memory binding remains to be investigated.

Acknowledgments

This research was partially supported by the German Research Foundation (DFG) under project Transregio Crossmodal Learning (TRR 169).

References

- [1] Thrun, S. and Mitchell, T.M. (1995). Lifelong robot learning. *Robot. Auton. Syst.*, 15:25–46.
- [2] Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., and Wermter, S. (2018). Continual Lifelong Learning with Neural Networks: A Review. *arXiv preprint arXiv:1802.07569*.
- [3] Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534.
- [4] Mermillod, M., Bugaiska, A. and Bonin, P. (2013). The Stability-Plasticity Dilemma: Investigating the Continuum from Catastrophic Forgetting to Age-Limited Learning Effects. *Frontiers in Psychology*, 4(504).
- [5] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13).
- [6] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. *ICML’17*, pages 3987–3995.
- [7] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. *arXiv:1606.04671*.
- [8] Parisi, G. I., Tani, J., Weber, C. and Wermter, S. (2017). Lifelong Learning of Humans Actions with Deep Neural Network Self-Organization. *Neural Networks*, 96:137–149.
- [9] Parisi, G. I., Tani, J., Weber, C. and Wermter, S. (2018). Lifelong Learning of Spatiotemporal Representations with Dual-Memory Recurrent Self-Organization. *arXiv preprint arXiv:1805.10966*.
- [10] Lomonaco, V. and Maltoni, D. (2017). CORe50: A New Dataset and Benchmark for Continuous Object Recognition. *CoRL’17*.
- [11] Knoblauch, A. (2017). Impact of structural plasticity on memory formation and decline. In: A. van Ooyen, M. Butz (eds.): *Rewiring the Brain: A Computational Approach to Structural Plasticity in the Adult Brain*, 17:361–386, Elsevier/Academic Press, London, UK.
- [12] Marsland, S., Shapiro, J., and Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Networks*, 15(8–9):1041–1058.
- [13] Li, Z. and Hoiem, D. (2016). Learning without forgetting. *ECCV’16*, pages 614–629.
- [14] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). iCaRL: Incremental Classifier and Representation Learning. *CVPR’14*, pages 2001–2010.
- [15] Kemker, R. and Kanan, C. (2018). FearNet: Brain-Inspired Model for Incremental Learning. *ICLR’18*.
- [16] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive Neural Networks. *arXiv:1606.04671*.
- [17] Aimone, J. B., Wiles, J., Gage, F. H. (2009). Computational influence of adult neurogenesis on memory encoding. *Neuron*, 61:187–202.
- [18] Parisi, G. I., Tani, J., Weber, C., Wermter, S. (2016). Emergence of Multimodal Action Representations from Neural Network Self-Organization. *Cognitive Systems Research*, 43:208–221.

Supplementary Material

5 Recurrent GWR and Memory Replay

For the purpose of memory replay in the absence of external sensory input, we generate pseudo-patterns in terms of temporally-ordered trajectories of neural activity [9]. We use asymmetric temporal links learned by the network to recursively reactivate sequence-selective neural activity trajectories (RNATs). We implemented temporal connections as synaptic links that are incremented between consecutively activated neurons [18]. When the two neurons i and j are activated at time $t - 1$ and t respectively, their temporal synaptic link $P_{(i,j)}$ is increased by $\Delta P_{(i,j)} = 1$. For each neuron $i \in A$, we can retrieve the next neuron v of a prototype trajectory

by selecting $v = \arg \max_{j \in A \setminus i} P_{(i,j)}$. After each learning episode, i.e. a learning epoch over a mini-batch of sensory observations, we generate a RNAT, S_j , of length $\lambda = K + 1$ for each neuron j as follows:

$$S_j = \langle \mathbf{w}_{s(0)}, \mathbf{w}_{s(1)}, \dots, \mathbf{w}_{s(\lambda)} \rangle, \quad (8)$$

$$s(i) = \arg \max_{n \in A \setminus j} P_{(n,s(i-1))}, i \in [1, \lambda], \quad (9)$$

where $P_{(i,j)}$ is the matrix of temporal synapses and $s(0) = j$. The class labels of the pseudo-patterns in S_j can be retrieved according to Eq. 10.

For the task of classification, an associative matrix $H(j, l)$ stores the frequency-based distribution of sample labels during the learning phase so that each neuron j stores the number of times that an input with label l had j as its BMU. The predicted label ξ_j for a neuron j can be computed as:

$$\xi_j = \arg \max_{l \in L} H(j, l), \quad (10)$$

where l is an arbitrary label. Therefore, the unsupervised network can be used for classification without requiring the number of label classes to be predefined.

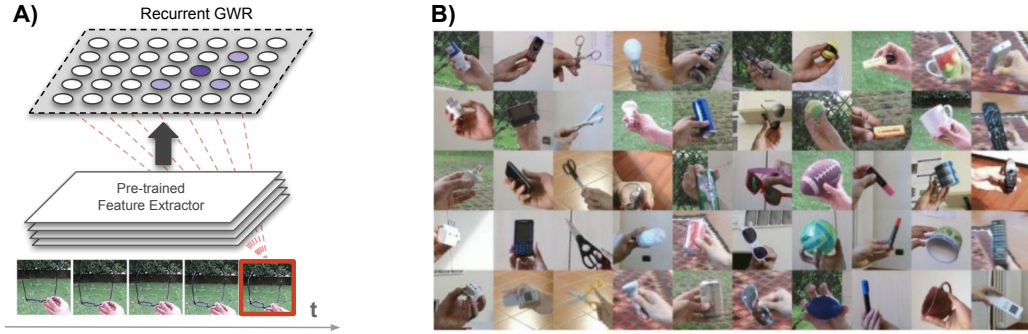


Figure 2: A) Neural architecture with a deep feature extractor and a recurrent GWR (static or dynamic). B) Example frames from the CORE50 dataset showing the 10 categories (columns) comprising 5 object instances each. Adapted from [10].

Table 1: Comparison of batch learning performance for instance-level classification. We show the accuracy for the pre-trained VGG with fine-tuning (VGG+FT) and the Growing Self-Organizing Memory for two different configurations: growing networks with temporal context (TC) and without TC. Best results in bold. Adapted from [9].

Approach	Accuracy (%)
	(Instance level: 50 classes)
VGG + FT [10]	69.08
Growing Self-Organizing Memory (No TC) [9]	70.42
Growing Self-Organizing Memory [9]	79.43

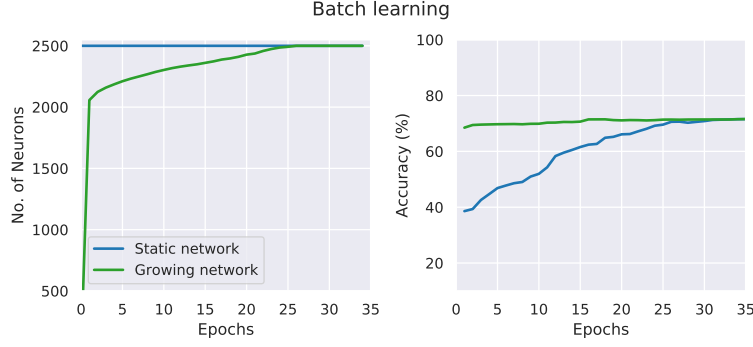


Figure 3: Batch learning of instance-level classification with a static and a growing model ($N_{max} = 2500$) over 35 training epochs.

Table 2: Accuracy results with the CORE50 on 3 incremental learning scenarios. Approaches denoted with * indicate that the results for that method are based on its re-implementation by [10].

Method	Avg. Acc. (%)	Std. Dev. (%)
<i>New Instances (NI)</i>		
Growing Self-Organizing Memory (with replay) [9]	87.94	1.72
Growing Self-Organizing Memory [9]	74.87	2.54
Cumulative [19]	65.15	0.66
LwF* [13]	59.42	2.71
EWC* [5]	57.40	3.80
Naïve [10]	54.69	6.18
<i>New Classes (NC)</i>		
Growing Self-Organizing Memory (with replay) [9]	86.14	2.03
Growing Self-Organizing Memory [9]	73.02	2.91
Cumulative	64.65	1.04
iCaRL* [14]	43.62	0.66
CWR [10]	42.32	1.09
LwF*	27.60	1.70
EWC*	26.22	1.18
Naïve	10.75	0.84
<i>New Instances and Classes (NIC)</i>		
Growing Self-Organizing Memory (with replay) [9]	87.06	2.13
Growing Self-Organizing Memory [9]	72.57	2.96
Cumulative	64.13	0.88
CWR	29.56	0
LwF*	28.94	4.30
EWC*	28.31	4.30
Naïve	19.39	2.90

Table 3: Training hyperparameters for the static and dynamic recurrent GWR networks.

Hyperparameters	Value
Insertion thresholds	$a_T = 0.3$
Habituation counters	$h_T = 0.1, \tau_b = 0.3, \tau_n = 0.1, \kappa = 1.05$
Temporal depth	$K = 2$
Temporal context	$\alpha = [0.67, 0.24, 0.09], \beta = 0.7$
Learning rates	$\epsilon_b = 0.5, \epsilon_n = 0.005$

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor (**only required where there isn't already a statement of contribution within the paper itself**).

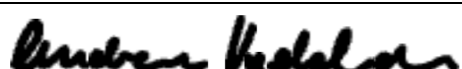
Title of Paper	On the Role of Neurogenesis in Overcoming Catastrophic Forgetting
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and unsubmitted work written in a manuscript style
Publication Details	German Parisi, Xu Ji, Stefan Wermter. On the Role of Neurogenesis in Overcoming Catastrophic Forgetting. Published in the Workshop on Continual Learning at the Conference on Neural Information Processing Systems (NeurIPS) 2018.

Student Confirmation

Student Name:	Xu Ji		
Contribution to the Paper	The initial idea belongs to German Parisi, who performed all experiments and wrote the first draft of the paper. I provided feedback and direction and helped with writing the paper.		
Signature 	Date	24/05/2021	

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title: Professor Andrea Vedaldi		
Supervisor comments The description is accurate.		
Signature 	Date	25/05/2021

This completed form should be included in the thesis, at the end of the relevant chapter.

8

Conclusions

Unsupervised learning and continual learning are often framed as separate concerns in the literature, but are deeply connected problems. We have reviewed a number of specific ideas in each category, but at a high level, they revolve around the same question of generalization: how to learn on empirically observed data such that performance on some superset of data is maximised, without complete information of the latter. The robustness and sample efficiency problems pointed out by Chollet in the opening quote (chapter 1) are both subsumed by this question.

Occam’s Razor (section 2.1.4) directly targets this question, arguing that simpler models are superior for maximising incidental generalization, however it is not clear exactly what constitutes a “simpler” or smaller neural network. The double descent paradigm has exposed that generalization can improve with increases in parameter size, which either invalidates using number of parameters as a size metric or invalidates Occam’s Razor. In chapter 3 we suggested the number of activation regions as a more meaningful size metric, but this is a post-hoc measure that is not directly optimizable with current methods. Furthermore, we discussed in section 2.1.4 how the size or complexity of the test data is a material consideration when assessing generalization. The smaller the test sets generalized to, the less valuable the learning algorithm even if it attains perfect performance, because the purpose of the trained model is to automate inference, and the smaller or simpler the

test set the smaller the benefit of automation. Thus the problem of generalization is not just about test set performance, but test set diversity and complexity: the extent to which access to a relatively small number of training samples is leveraged for good performance on a much larger number of samples. Awareness that measures of generalization are more meaningful given diverse and difficult test sets, where difficult means distance from the training data by some metric (e.g. gradient decorrelation (section 2.1.1) or human-determined semantic distance (fig. 5, chapter 3)), should encourage the field to move away from experimenting on narrow or relatively easy test sets. This has proved challenging however, because the prevailing attitude is to discount the value of work that does not improve on established benchmarks (Chollet, 2019), which are overwhelmingly non-diverse in content. A thorough evaluation of generalization would involve assessing a single model on classification, captioning, text generation, visual question answering and navigation tasks, for instance.

It appears that poor robustness and narrow generalization go hand in hand; the average error of a network trained specifically for image classification on a well-curated dataset appears respectable until its wildly erroneous behaviour on images with minor pixel corruptions is discovered (Nguyen, Yosinski, and Clune, 2015). Meanwhile, human behaviour is suboptimal in many real-world settings, but is not wildly erroneous in the presence of novelty; humans *fail well*. This suggests that average error does not matter as much as low ceilings on error; that human learning prioritizes the retention of highly generalizable, generic behaviours that may not attain maximal rewards in every scenario but ensure the avoidance of worst cases. Furthermore, this is attained with a neuronal model using distributed memory, where the number of stimuli that can be represented increases exponentially with the number of neurons, themselves numbering in the billions (Rolls, Treves, and Tovee, 1997), which contravenes the idea that such simple, generalizable behaviours should be obtained through parameter size constraints, or that catastrophic forgetting is inevitable in distributed memory. Thus, despite their roots in biological neuronal models, major differences remain in the way that generalization, including in unsupervised learning and continual

learning, is approached by current neural network training methods. How this gap will be convincingly bridged remains to be seen.

References

- [1] Hirotugu Akaike. “A new look at the statistical model identification”. In: *IEEE transactions on automatic control* 19.6 (1974), pp. 716–723.
- [2] Rahaf Aljundi et al. “Memory aware synapses: Learning what (not) to forget”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 139–154.
- [3] Rahaf Aljundi et al. “Online continual learning with maximal interfered retrieval”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 11849–11860.
- [4] D. Alan Allport. “Distributed memory, modular subsystems and dysphasia”. In: *Current perspectives in dysphasia* (1985).
- [5] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. “Self-labelling via simultaneous clustering and representation learning”. In: *arXiv preprint arXiv:1911.05371* (2019).
- [6] Frederico AC Azevedo et al. “Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain”. In: *Journal of Comparative Neurology* 513.5 (2009), pp. 532–541.
- [7] David Bau et al. “Gan dissection: Visualizing and understanding generative adversarial networks”. In: *arXiv preprint arXiv:1811.10597* (2018).
- [8] Suzanna Becker and Geoffrey E Hinton. “Self-organizing neural network that discovers surfaces in random-dot stereograms”. In: *Nature* 355.6356 (1992), pp. 161–163.
- [9] Jens Behrmann et al. “Invertible residual networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 573–582.
- [10] Mikhail Belkin et al. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.
- [11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.
- [12] Alfred Binet and Th Simon. “Méthodes nouvelles pour le diagnostic du niveau intellectuel des anormaux”. In: *L’année Psychologique* 11.1 (1904), pp. 191–244.
- [13] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [14] Mathilde Caron et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149.
- [15] Mathilde Caron et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *arXiv preprint arXiv:2006.09882* (2020).

- [16] Margaret F Carr, Shantanu P Jadhav, and Loren M Frank. “Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval”. In: *Nature neuroscience* 14.2 (2011), p. 147.
- [17] Arslan Chaudhry et al. “Continual learning with tiny episodic memories”. In: *arXiv preprint arXiv:1902.10486* (2019).
- [18] Arslan Chaudhry et al. “Efficient lifelong learning with a-gem”. In: *arXiv preprint arXiv:1812.00420* (2018).
- [19] Ricky TQ Chen et al. “Residual flows for invertible generative modeling”. In: *arXiv preprint arXiv:1906.02735* (2019).
- [20] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- [21] In-Koo Cho and David M Kreps. “Signaling games and stable equilibria”. In: *The Quarterly Journal of Economics* 102.2 (1987), pp. 179–221.
- [22] François Chollet. “On the measure of intelligence”. In: *arXiv preprint arXiv:1911.01547* (2019).
- [23] Elijah Cole et al. “When Does Contrastive Visual Representation Learning Work?” In: *arXiv preprint arXiv:2105.05837* (2021).
- [24] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [25] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).
- [26] Carl Doersch, Abhinav Gupta, and Alexei A Efros. “Unsupervised visual representation learning by context prediction”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1422–1430.
- [27] Robert M French. “Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks”. In: 1991.
- [28] Matteo Gamba et al. “Hyperplane Arrangements of Trained ConvNets Are Biased”. In: *arXiv preprint arXiv:2003.07797* (2020).
- [29] Hagar Gelbard-Sagiv et al. “Internally generated reactivation of single neurons in human hippocampus during free recall”. In: *Science* 322.5898 (2008), pp. 96–101.
- [30] Malcolm Gladwell. *Outliers: The story of success*. Little, Brown, 2008.
- [31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “Machine learning basics”. In: *Deep learning* 1 (2016), pp. 98–164.
- [32] Ian J Goodfellow et al. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In: *arXiv preprint arXiv:1312.6211* (2013).
- [33] Ian J Goodfellow et al. “Generative adversarial networks”. In: *arXiv preprint arXiv:1406.2661* (2014).
- [34] Anirudh Goyal and Yoshua Bengio. “Inductive Biases for Deep Learning of Higher-Level Cognition”. In: *arXiv preprint arXiv:2011.15091* (2020).

- [35] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).
- [36] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.
- [37] Boris Hanin and David Rolnick. “Deep ReLU networks have surprisingly few activation patterns”. In: *Advances in Neural Information Processing Systems 32* (2019), pp. 361–370.
- [38] Demis Hassabis et al. “Neuroscience-inspired artificial intelligence”. In: *Neuron* 95.2 (2017), pp. 245–258.
- [39] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [40] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [41] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.
- [42] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.
- [43] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).
- [44] Geoffrey E Hinton. “Learning distributed representations of concepts”. In: 1986.
- [45] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [46] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.
- [47] Phillip Isola et al. “Learning visual groups from co-occurrences in space and time”. In: *arXiv preprint arXiv:1511.06811* (2015).
- [48] P. Jonas and G. Buzsaki. “Neural inhibition”. In: *Scholarpedia* 2.9 (2007). revision #91565, p. 3286.
- [49] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.
- [50] Ronald Kemker and Christopher Kanan. “Fearnnet: Brain-inspired model for incremental learning”. In: *arXiv preprint arXiv:1711.10563* (2017).
- [51] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [52] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

- [53] Klemen Kotar et al. “Contrasting Contrastive Self-Supervised Representation Learning Models”. In: *arXiv preprint arXiv:2103.14005* (2021).
- [54] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. “Handling imbalanced datasets: A review”. In: *GESTS International Transactions on Computer Science and Engineering* 30.1 (2006), pp. 25–36.
- [55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.
- [56] Dharshan Kumaran, Demis Hassabis, and James L McClelland. “What learning systems do intelligent agents need? Complementary learning systems theory updated”. In: *Trends in cognitive sciences* 20.7 (2016), pp. 512–534.
- [57] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [58] Zhizhong Li and Derek Hoiem. “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.
- [59] David Lopez-Paz and Marc’Aurelio Ranzato. “Gradient episodic memory for continual learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6467–6476.
- [60] Michael McCloskey and Neal J Cohen. “Catastrophic interference in connectionist networks: The sequential learning problem”. In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.
- [61] Takeru Miyato et al. “Spectral normalization for generative adversarial networks”. In: *arXiv preprint arXiv:1802.05957* (2018).
- [62] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).
- [63] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.
- [64] Preetum Nakkiran et al. “Deep double descent: Where bigger models and more data hurt”. In: *arXiv preprint arXiv:1912.02292* (2019).
- [65] Andrew Y Ng and Michael I Jordan. “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes”. In: *Advances in neural information processing systems*. 2002, pp. 841–848.
- [66] Anh Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.
- [67] Aaron van den Oord et al. “Conditional image generation with pixelcnn decoders”. In: *arXiv preprint arXiv:1606.05328* (2016).
- [68] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).
- [69] Sankar K Pal and Sushmita Mitra. “Multilayer perceptron, fuzzy sets, classification”. In: (1992).

- [70] Judea Pearl. *Causality*. Cambridge university press, 2009.
- [71] Alexander Pritzel et al. “Neural episodic control”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2827–2836.
- [72] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [73] Maithra Raghu et al. “On the expressive power of deep neural networks”. In: *international conference on machine learning*. PMLR. 2017, pp. 2847–2854.
- [74] Danilo Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1530–1538.
- [75] Anthony Robins. “Catastrophic forgetting, rehearsal and pseudorehearsal”. In: *Connection Science* 7.2 (1995), pp. 123–146.
- [76] Edmund T Rolls, Alessandro Treves, and Martin J Tovee. “The representational capacity of the distributed encoding of information provided by populations of neurons in primate temporal visual cortex”. In: *Experimental Brain Research* 114.1 (1997), pp. 149–162.
- [77] David Rolnick et al. “Experience replay for continual learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 348–358.
- [78] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.
- [79] Andrei A Rusu et al. “Progressive neural networks”. In: *arXiv preprint arXiv:1606.04671* (2016).
- [80] Tim Salimans et al. “Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications”. In: *arXiv preprint arXiv:1701.05517* (2017).
- [81] Monika S Schmid. *First language attrition: Interdisciplinary perspectives on methodological issues*. Vol. 28. John Benjamins Publishing, 2004.
- [82] Claude E Shannon. “Programming a computer for playing chess”. In: *first presented at the National IRE Convention, March 9, 1949, and also in Claude Elwood Shannon Collected Papers*. IEEE Press. 1993, pp. 637–656.
- [83] Hanul Shin et al. “Continual learning with deep generative replay”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2990–2999.
- [84] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [85] Linda Smith and Chen Yu. “Infants rapidly learn word-referent mappings via cross-situational statistics”. In: *Cognition* 106.3 (2008), pp. 1558–1568.
- [86] Jake Snell, Kevin Swersky, and Richard S Zemel. “Prototypical networks for few-shot learning”. In: *arXiv preprint arXiv:1703.05175* (2017).
- [87] Paul Vincent Spade et al. *The cambridge companion to ockham*. Cambridge University Press, 1999.

- [88] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
- [89] Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. “Local minima in training of neural networks”. In: *arXiv preprint arXiv:1611.06310* (2016).
- [90] Alan Turing. “Computing Machinery and Intelligence”. In: (1950).
- [91] Vladimir Vapnik. “On the uniform convergence of relative frequencies of events to their probabilities”. In: *Doklady Akademii Nauk USSR*. Vol. 181. 4. 1968, pp. 781–787.
- [92] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [93] Guido M van de Ven and Andreas S Tolias. “Generative replay with feedback connections as a general strategy for continual learning”. In: *arXiv preprint arXiv:1809.10635* (2018).
- [94] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.
- [95] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.
- [96] Justin N Wood. “A smoothness constraint on the development of object recognition”. In: *Cognition* 153 (2016), pp. 140–145.
- [97] Xiaohui Xie and H Sebastian Seung. “Equivalence of backpropagation and contrastive Hebbian learning in a layered network”. In: *Neural computation* 15.2 (2003), pp. 441–454.
- [98] Dong Yin et al. “Optimization and Generalization of Regularization-Based Continual Learning: a Loss Approximation Viewpoint”. In: *arXiv preprint arXiv:2006.10974* (2020).
- [99] Hongxu Yin et al. “Dreaming to Distill: Data-free Knowledge Transfer via DeepInversion”. In: *arXiv preprint arXiv:1912.08795* (2019).
- [100] Jure Zbontar et al. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *arXiv preprint arXiv:2103.03230* (2021).
- [101] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3987–3995.
- [102] Hongjie Zhang et al. “Hybrid models for open set recognition”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 102–117.
- [103] Richard Zhang, Phillip Isola, and Alexei A Efros. “Colorful image colorization”. In: *European conference on computer vision*. Springer. 2016, pp. 649–666.
- [104] Jun-Yan Zhu et al. “Generative visual manipulation on the natural image manifold”. In: *European conference on computer vision*. Springer. 2016, pp. 597–613.

- [105] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.