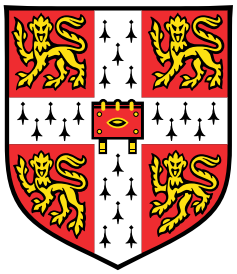
Adapting deep neural networks as models of human visual perception



Patrick S. McClure

MRC Cognition and Brain Sciences Unit
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Trinity College July 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 60,000 words including appendices, bibliography, footnotes, tables and equations.

Patrick S. McClure
July 2018

Acknowledgements

I would like to thank Dr. Nikolaus Kriegeskorte for giving me the opportunity to pursue a Ph.D. at the University of Cambridge. His encouragement and insightful comments have been instrumental in my growth as a researcher. I would also like to thank all of the Visual Objects lab members. Specifically, I want to thank Johannes Mehrer, Courtney Spoerer, and Tim Kietzmann for their invaluable assistance throughout my time in Cambridge. I would also like to thank Sergii Strelchuk and Francisco Pereira for their extremely helpful comments regarding this thesis.

My deepest thanks goes to my family, especially my parents. Their love and support has been instrumental not just in my completion of a Ph.D., but also in every aspect of my life. I also want to thank my brother for his comments on my research and papers and my sister for her encouragement.

Too many people remain for me to acknowledge them all. However, I want to thank everyone who has given their time and energy to help me throughout my life. They have all contributed to allowing me to complete this achievement.

Abstract

Deep neural networks (DNNs) have recently been used to solve complex perceptual and decision tasks. In particular, convolutional neural networks (CNN) have been extremely successful for visual perception. In addition to performing well on the trained object recognition task, these CNNs also model brain data throughout the visual hierarchy better than previous models. However, these DNNs are still far from completely explaining visual perception in the human brain. In this thesis, we investigated two methods with the goal of improving DNNs' capabilities to model human visual perception: (1) deep representational distance learning (RDL), a method for driving representational spaces in deep nets into alignment with other (e.g. brain) representational spaces and (2) variational DNNs that use sampling to perform approximate Bayesian inference. In the first investigation, RDL successfully transferred information from a teacher model to a student DNN. This was achieved by driving the student DNN's representational distance matrix (RDM), which characterises the representational geometry, into alignment with that of the teacher. This led to a significant increase in test accuracy on machine learning benchmarks. In the future, we plan to use this method to simultaneously train DNNs to perform complex tasks and to predict neural data. In the second investigation, we showed that sampling during learning and inference using simple Bernoulli- and Gaussian-based noise improved a CNN's representation of its own uncertainty for object recognition. We also found that sampling during learning and inference with Gaussian noise improved how well CNNs predict human behavioural data for image classification. While these methods alone do not fully explain human vision, they allow for training CNNs that better model several features of human visual perception.

Table of contents

List of figures **xi**

List of tables **xvii**

1 Deep neural networks in computational neuroscience **1**
1.1 The brain is a deep neural network 2
1.2 Brain-inspired neural network models are promising for artificial intelligence and computational neuroscience 3
1.3 Deep neural network models can be tested with brain and behavioural data . 7
1.4 Drawing insights from complex models 9
1.5 What neurobiological details matter to brain computation? 14
1.6 What is next? 17

2 Adapting neural networks with deep representational distance learning **19**
2.1 Introduction 20
2.2 Methods 23
2.2.1 Representational Distance Matrices 23
2.2.2 Representational Distance Learning 25
2.3 Experiments 27
2.3.1 MNIST 27
2.3.2 CIFAR-100 30
2.4 Discussion 34

3 Adapting deep neural networks by using stochasticity to robustly represent uncertainty **37**
3.1 Introduction 38
3.2 Methods 39
3.2.1 Bayesian Deep Neural Networks 39
3.2.2 Variational Distributions 40

3.3 Experiments 42
3.3.1 Logistic Regression 42
3.3.2 Convolutional Neural Networks 46
3.4 Discussion 49

4 Adapting Bayesian deep neural networks to model human visual perception 51
4.1 Introduction 52
4.2 Methods 53
4.2.1 Approximating Bayesian neural networks using Monte Carlo Gaussian dropout 53
4.2.2 Relationship between Monte Carlo Gaussian dropout and deep latent Gaussian models 53
4.2.3 Architecture and datasets 54
4.3 Results 56
4.3.1 Sampling improves accuracy for large-scale object recognition . . . 56
4.3.2 Sampling improves the representation of uncertainty for large-scale object recognition 58
4.3.3 Sampling improves the prediction of human confidence for image classification 58
4.4 Discussion 59

5 Conclusion 61

References 63

Appendix A 77
A.1 L2 regularisation and the KLD between Gaussians 77
A.2 Gaussian "reparameterization trick" 78
A.3 MC Gaussian Dropconnect and Dropout 78
A.3.1 MC spike-and-slab Dropout 79
A.3.2 MC spike-and-slab Dropout 81
A.4 Additional Section 3.3.2 Results 83

List of figures

1.1 **Testing the internal representations of DNNs against neural data.** (A) An example of neuron-level encoding with a convolutional neural network where the CNN-based neural response prediction (red) closely matches the recorded biological neural response (black) (adapted from (Yamins and DiCarlo, 2016)). (B) The representational geometries of a trained CNN's representations (center) and human (left) and monkey (right) brain activation patterns as defined by the distance matrices between the representation/activation patterns for different stimuli (adapted from Khaligh-Razavi and Kriegeskorte (2014)). 4

1.2 **Convolutional neural network structure.** (A) An example feed forward convolutional neural network (CNN) with 3 convolutional layers followed by a fully-connected layer. Bottom-up filters for selected neurons are illustrated with blue boxes. (B) The bottom-up (blue), lateral (green), and top-down (red) filters for two example neurons in different layers of a recurrent convolutional neural network (RCNN). 5

1.3 **Visualising the preferred features of internal artificial neurons.** Activations in a random subset of feature maps across layers for strongly driving ImageNet images projected down to pixel space (adapted from Zeiler and Fergus (2014)). 13

2.1 **Example CNN-based representational distance matrices (RDMs).** The RDMs of the output layer of CNNs for ten random images of each class from MNIST (left) and CIFAR-10 (right) made using the RSA toolbox (Nili et al., 2014). 24

2.2 **Train and test errors of the MNIST trained CNNs throughout training as the tested convolutional neural.** 28

2.3 **Representational distance matrices (RDMs) for different layers of the MNIST trained CNNs.** RDMs using the Euclidean distance for the first and second convolutional layers as well as the fully connected (FC) and softmax layers of the CNN tested methods, the raw pixel data, and the target labels for 10 random class exemplars from MNIST. Note that the target RDM was generated by computing the RDM of the one-hot vectors used as labels during training. 29

2.4 **RDL pulls internal representations towards the representations of the teacher for MNIST.** 2-D multi-dimensional scaling (MDS) visualisation of the distances between the representational distance matrices (RDMs) for selected layers of the MNIST trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the correlation distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot. 31

2.5 **Train and test errors of the CIFAR-100 trained NiNs throughout training as the tested convolutional neural.** 31

2.6 **RDL pulls internal representations towards the representations of the teacher for CIFAR 100.** 2-D multi-dimensional scaling (MDS) visualisation of the distances between the representational distance matrices (RDMs) for selected layers of the CIFAR-100 trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the normalised Euclidean distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot. 33

3.1 **Sampling either weights or units from different variational distributions can be constructed as multiplicative noise (of various statistical structure) imposed on the weight matrix.** 41

3.2 **Independent weight (i.e. dropconnect-based) sampling during training and testing makes models much more uncertain further away from the training data.** The probabalistic logistic regression decision boundaries of a linear network for: (top row) the MAP network and the dropconnect and dropout methods that only sample during training and (bottom row) the stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011) network and the Monte Carlo (MC) dropconnect and dropout methods that sample during training and testing. 42

3.3 **Sampling during training and testing prevents overfitting on MNIST.** The MNIST (a) training error, (b) training loss, (c) test error, and (d) test loss for for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. 44

3.4 **Sampling during training and testing prevents overfitting on CIFAR-10.** The CIFAR-10 (A) training error, (B) training loss, (C) test error, and (D) test loss for for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. 45

3.5 **Sampling during training and testing improves CNN calibration on MNIST in the presecence of input noise.** The MNIST (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 1, 2, 3, 4, and 5, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. 47

3.6 **Sampling during training and testing improves CNN calibration on CIFAR-10 in the presecence of input noise.** The CIFAR-10 (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 0.25, 0.5, 0.75, and 1, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. 47

3.7 **Sampling during training and testing improves CNN calibration curves.** The $x = y$ line (Ideal) and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD. for the (a) MNIST or (b) CIFAR-10 trained Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) networks with and without MC sampling using 10 samples. 48

3.8 **Sampling during training and testing (1) improves the robustness of CNNs to dropout hyperparmeter choice and (2) allows for increased dropout regularisation.** The classification error for the CIFAR-10 test set using different p hyperparameter values. For spike-and-slab dropout, p_{do} was varied, while p_{dc} was fixed. 49

4.1 **Sampling during training and testing improves CNN calibration for large-scale object recognition.** The CNN calibration curves for (A) the ImageNet trained CNN on the ImageNet validation set and (B) the Eco-set trained CNNs on the Eco-set test set. 57

4.2 **Sampling during training and testing improves the correlation between CNN-based predicted probabilities and human confidence scores.** The mean and standard errors for the (A) accuracies and (B) correlations with human confidence scores for the logistic regression models trained at each CNN. $*$ denotes the "MC Training and Testing" models have significantly higher (p-value<0.05,uncorr.) correlations with human confidence scores than both the "MAP" and "MC Training" models per a paired t-test across the five image sets. 58

A.1 **Sampling at test time allows for higher variance sampling to be used during training without inducing network failure.** The CIFAR-10 (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 0.25, 0.5, 0.75, and 1, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. For all dropconnect and dropout methods, $p = 0.5$. For spike-and-slab, $p_{do} = 0.5$ and $p_{dc} = 0.1$. 83

A.2 **Sampling at test time allows for higher variance sampling to be used during training without causing underfitting and underconfidence.** The $x = y$ line (Ideal) and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD. for the CIFAR-10 trained Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) networks with and without MC sampling using 10 samples. For all dropconnect and dropout methods, $p = 0.5$. For spike-and-slab, $p_{do} = 0.5$ and $p_{dc} = 0.1$. 84

List of tables

2.1 The convolutional neural network (CNN) architecture used for MNIST. . . 26
2.2 Test errors for MNIST trained convolutional neural networks (CNNs) and the CIFAR-100 trained "Network in Network" (NiN) models. (Note: The performance of the teacher for the CIFAR-100 classification is not shown, since it was trained on CIFAR-10 and, therefore, predicted across 10 not 100 classes, making it unable to perform the CIFAR-100 task.) 27
2.3 The McNemar exact test p-values for the tested CNNs trained on MNIST. Arrows indicate a significant difference ($p < 0.05$,uncorr.) and point to the better model. 29
2.4 The "Network in Network" (NiN) architecture with batch-normalisation (BN) (Ioffe and Szegedy, 2015) used for CIFAR-100. 32
2.5 The McNemar exact test p-values for the tested "Network in Network" (NiN) models trained on CIFAR-100. Arrows indicate a significant difference ($p < 0.05$,uncorr.) and point to the better model. 34

3.1 MNIST and CIFAR-10 mean and standard deviation of test errors for the trained convolutional neural networks (CNNs) with and without Monte-Carlo (MC) across 5 runs, each MC run using 10 samples. 43
3.2 The convolutional neural network (CNN) architecture used for MNIST. . . 43
3.3 The convolutional neural network (CNN) architecture used for CIFAR-10. . 43

4.1 The convolutional neural network (CNN) architecture used for ImageNet and Ecoset. 55
4.2 The accuracies for the ImageNet trained CNN on the ImageNet validation set and the Eco-set trained CNNs on the Eco-set test set. For MC sampling, the mean and standard deviation of the accuracies across 5 MC runs, each computed with 10 MC samples. 56

Chapter 1

Deep neural networks in computational neuroscience

This chapter is based on a manuscript by Kietzmann, McClure, and Kriegeskorte (2017b). Sections 1.1-1.5 of the paper were jointly written by all three authors. Section 1.6 was solely authored by Patrick McClure.

Summary

One of the goals of computational neuroscience is to find mechanistic explanations of how the nervous system processes information to support cognitive function and behaviour. At the heart of the field are its models, i.e. mathematical and computational descriptions of the system being studied. These models typically map sensory stimuli to neural responses and/or neural to behavioural responses and range from simple to complex. Recently, deep neural networks (DNNs) have come to dominate several domains of artificial intelligence (AI). As the term "neural network" suggests, these models are inspired by biological brains. However, current DNN models abstract away many details of biological neural networks. These abstractions contribute to their computational efficiency, enabling them to perform complex feats of intelligence, ranging from perceptual tasks (e.g. object recognition) to cognitive tasks, and on to motor control tasks. In addition to modelling complex intelligent behaviours, the learned representations of DNNs have been shown to predict neural responses to novel sensory stimuli that cannot be predicted with any other currently available type of model. DNNs can have millions of parameters (connection strengths), which are required to capture the domain knowledge needed for task performance. These parameters are often set by task training using stochastic gradient descent. The advances with neural nets in engineering

provide the technological basis for building task-performing models of varying degrees of biological realism that promise substantial insights for computational neuroscience.

1.1 The brain is a deep neural network

One of the goals of computational neuroscience is to find mechanistic explanations for how the nervous system processes information to support cognitive function and adaptive behaviour (Marr and Poggio, 1976). Computational models, i.e. mathematical and computational descriptions of component systems, capture the mapping of sensory input to neural responses and explain representational transformations, neuronal dynamics, and the way the brain controls behaviour.

The overarching challenge is therefore to define models that explain neural measurements as well as complex adaptive behaviour. Computational neuroscientists have had early successes with shallow, linear-nonlinear "tuning", modelling lower-level sensory processing (e.g. Hubel and Wiesel (1959)). Relatively shallow models have fueled progress in the past and will continue to do so. Yet, the brain is a deep recurrent neural network that exploits multistage non-linear transformations and complex dynamics. It therefore seems inevitable that computational neuroscience will come to rely increasingly on deep recurrent models. The need for multiple stages of nonlinear computation has long been appreciated in the domain of vision, by both experimentalists (Hubel and Wiesel, 1959, 1962) and theorists (Fukushima and Miyake, 1982; LeCun and Bengio, 1995; Riesenhuber and Poggio, 1999; Wallis and Rolls, 1997).

The traditional focus on shallow models was motivated both by the desire for simple explanations and by the difficulty of fitting complex models. Hand-crafted features, which laid the basis of modern computational neuroscience (Jones and Palmer, 1987), do not carry us beyond restricted lower-level tuning functions. As an alternative approach, researchers started directly using neural data to fit model parameters (Dumoulin and Wandell, 2008; Marmarelis and Marmarelis, 1978; Wu et al., 2006).

Despite its elegance, importance, and success, this approach is limited by the amount of neural observations that can be collected, both on an individual and group level. Even with neural measurement technology advancing rapidly (multi-site array recordings, two-photon imaging, or neuropixels, to name just a few), the amount of recordable data does not provide enough constraints to fit realistically complex neural models. For instance, while novel measurement techniques may record separately from hundreds of individual neurons, and the number of stimuli used may approach 10,000, the numbers of parameters in deep neural networks (DNNs) used to perform complex tasks, such as object recognition, are many

orders of magnitude larger (e.g. the influential object recognition network "AlexNet" has 60 million parameters (Krizhevsky et al., 2012) and a more recent object recognition network, VGG-16, has 138 million parameters (Simonyan et al., 2013)). While these networks may not directly model the receptive field of individual neurons, they have been successfully utilized as models of neuron activity patterns (Yamins and DiCarlo, 2016) and population-level representations (Khaligh-Razavi and Kriegeskorte, 2014) (Figure 2.1).

An important lesson in the history of AI is that intelligence requires a lot of domain knowledge. Transferring this knowledge into the model parameters through the bottleneck of neural measurements alone is too inefficient for complex models. A key insight that opened the path for the use of very complex models for prediction of neural responses is the idea that rather than fitting parameters based on neural observations, DNNs could instead be trained to perform relevant behaviour in the real world. This approach brings machine learning to bear on models for computational neuroscience, enabling researchers to constrain the model parameters via training on a variety of tasks. In the domain of vision, for instance, category-labelled sets of training images can easily be assembled using web-based technologies, and the amount of available data can therefore be expanded more easily than for measurements of neural activity. Of course a model trained to excel at a relevant task (such as object recognition, if we are trying to understand the computations in the primate ventral stream) might not be able to explain neural data. Testing which model architectures, input statistics, and learning objectives yield the best predictions of neural activity in novel experimental conditions (e.g. a set of images that has not been used in fitting the parameters) is a powerful way to learn about the computational mechanisms that might underlie the neural responses. The combined use of task training- and neural data enables us to build complex models with massive knowledge about the world in order to explain how biological brains implement cognitive function. Deep learning provides a very efficient tool to transfer this knowledge into the parameters of the model.

1.2 Brain-inspired neural network models are promising for artificial intelligence and computational neuroscience

Neural network models inspired by biological brains have become a central class of models in machine learning (Figure 1.2). Driven by optimizing task-performance, they developed and improved model architectures, hardware and training schemes that eventually led to today's high-performance deep neural network models. These models have revolutionized several domains of AI, including computer vision (LeCun et al., 2015). Starting with the seminal

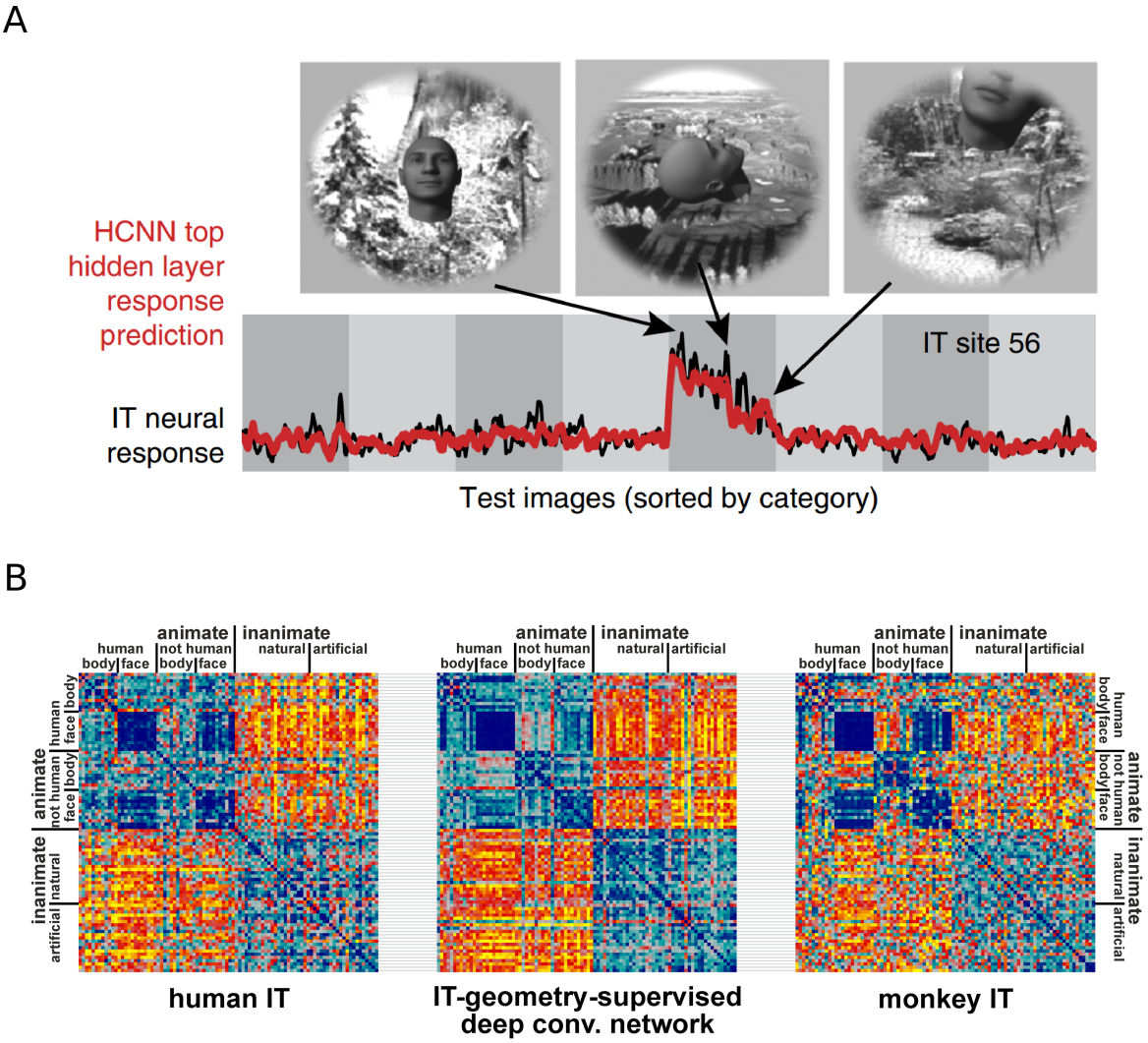


Fig. 1.1 Testing the internal representations of DNNs against neural data. (A) An example of neuron-level encoding with a convolutional neural network where the CNN-based neural response prediction (red) closely matches the recorded biological neural response (black) (adapted from (Yamins and DiCarlo, 2016)). (B) The representational geometries of a trained CNN's representations (center) and human (left) and monkey (right) brain activation patterns as defined by the distance matrices between the representation/activation patterns for different stimuli (adapted from Khaligh-Razavi and Kriegeskorte (2014)).

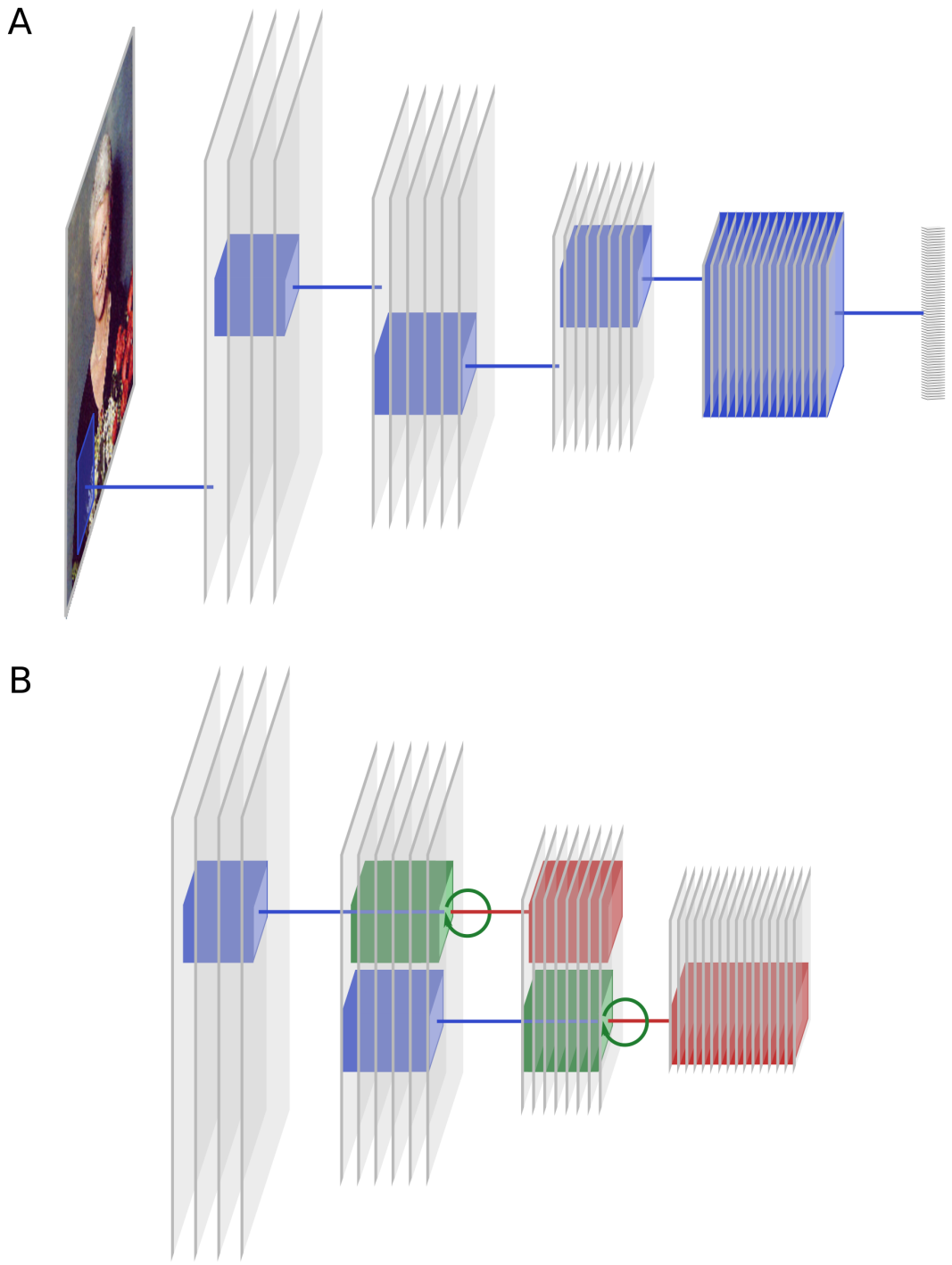


Fig. 1.2 Convolutional neural network structure. (A) An example feed forward convolutional neural network (CNN) with 3 convolutional layers followed by a fully-connected layer. Bottom-up filters for selected neurons are illustrated with blue boxes. (B) The bottom-up (blue), lateral (green), and top-down (red) filters for two example neurons in different layers of a recurrent convolutional neural network (RCNN).

work by Krizhevsky et al. (2012), who won the ImageNet competition in visual object recognition by a large margin, deep neural networks now dominate computer vision (He et al., 2016; Simonyan et al., 2013; Szegedy et al., 2015), and drove reinforcement learning (Lange and Riedmiller, 2010; Mnih et al., 2014, 2015), speech-recognition (Sak et al., 2014), machine translation (Sutskever et al., 2014; Wu et al., 2016), and many other domains to unprecedented performance levels. In terms of visual processing, deep convolutional, feed-forward networks now achieve human-level classification performance (VanRullen, 2017).

Although inspired by biology, current DNNs abstract from all but the most essential features of biological neural networks. They are composed of simple units that typically compute a linear combination of their inputs and pass the result through a static nonlinearity (e.g. setting negative values to zero). To what extent they can nevertheless bring insights to computational neuroscience is controversial (Kay, 2017; Kriegeskorte, 2015; VanRullen, 2017). Optimised to perform, DNNs differ substantially from biological neural networks, but they also exhibit architectural similarities. Consider the particularly successful variant of feedforward convolutional neural networks. Inspired by biological vision, these networks process images through a sequence of visuotopic representations. Each unit "sees" a restricted local region of the visuotopic map in the previous layer (its receptive field). Moreover, units are grouped into sets that detect the same visual feature all over the image (feature maps). The units within a feature map jointly learn a single connection weight template. The restriction to local receptive fields and sharing of weights among units in the same feature map greatly reduce the number of parameters that need to be learned. Like the primate visual system, convolutional neural networks perform a deep cascade of non-linear computations, their neurons exhibit spatially restricted receptive fields that increase in size, invariance, and complexity along the hierarchical levels and similar feature detectors exist for different spatial locations in a given layer (although this is only approximately true in the primate brain). At the same time, however, these models are simplified in radical ways. They do typically not include lateral or top-down connections, compute continuous outputs (real numbers that could be interpreted as firing rates) rather than spikes. The list of features of biological neural networks not captured by these models is endless.

Despite abstracting from many features of biology, deep convolutional neural networks predict functional signatures of primate visual processing at multiple hierarchical levels. Trained to recognize objects, they develop V1-like receptive fields in early layers, and are predictive of single cell recordings in macaque IT (Cadieu et al., 2014; Khaligh-Razavi and Kriegeskorte, 2014; Yamins and DiCarlo, 2016; Yamins et al., 2014). In particular, the explanatory power of DNNs was on par with the performance of linear prediction based on

an independent set of IT neurons and beyond linear predictions based directly on the category labels on which the networks were trained (Yamins et al., 2014). DNNs thereby constitute the only model class in computational neuroscience that is capable of predicting responses to novel images in IT with reasonable accuracy. DNNs explain about 50% of the variance of windowed spike counts in IT across individual images (Yamins et al., 2014), a performance level comparable to that achieved with Gabor models in V1 (Olshausen and Field, 2005). DNN modelling has also been shown to improve predictions of intermediate representations in area V4 over alternative models (Yamins and DiCarlo, 2016). This indicates that, in order to solve the task, the trained network transforms the image through a similar sequence of intermediate representations as the primate brain.

In human neuroscience similarly, DNNs proved capable of predicting representations measured with functional magnetic resonance imaging across multiple levels of processing in a hierarchical fashion: lower network levels better predict lower level visual representations, and subsequent, higher-levels better predict activity in higher- more anterior cortical areas (Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014). In line with results from macaque IT, DNNs were furthermore able to explain within-category neural similarities, despite being trained on a categorisation task (Khaligh-Razavi and Kriegeskorte, 2014). At a lower spatial, but higher temporal resolution, DNNs have also been shown to be predictive of visually evoked magnetoencephalography (MEG) data (Cichy et al., 2016, 2017; Seeliger et al., 2017). On the behavioural level, deep networks exhibit similar behaviour (Hong et al., 2016; Kheradpisheh et al., 2016; Kubilius et al., 2016) and are currently the best-performing model in explaining human eye-movements in free viewing paradigms (Kümmerer et al., 2015). These early examples clearly illustrate the power of DNN models for computational neuroscience.

1.3 Deep neural network models can be tested with brain and behavioural data

DNNs are typically trained to optimise external behavioural objectives rather than being derived from neural data. Thus, model testing with activity measurements is crucial to assess how well a network matches cortical responses. DNNs excel at task performance, but even human-level performance does not imply that the underlying computations employ the same mechanisms. In particular, no one-to-one mapping from a DNN unit to a biological neuron can be guaranteed. Fortunately, computational neuroscience has a rich toolbox at its disposal that allows researchers to probe even highly complex models, such as DNNs

(Diedrichsen and Kriegeskorte, 2017). One such tool is encoding models, which use external, fixed feature spaces in order to model neural responses across a large variety of experimental conditions (e.g. different stimuli, Figure 1.2A). The underlying idea is that if the model and the brain compute the same features, then linear combinations of the model features should enable successful prediction of the neural responses for independent experimental data (Naselaris et al., 2011). For visual representations, the model feature space can derive from simple filters, such as Gabor-wavelets (Kay et al., 2008), from human labeling of the stimuli (Mitchell et al., 2008; Naselaris et al., 2009), or responses in different layers of a DNN (Güçlü and van Gerven, 2015). Probing the system on the level of multivariate response patterns, representational similarity analysis (RSA) (Kriegeskorte et al., 2008) provides another approach to comparing internal representations in DNNs and the brain (Figure 1.2B). RSA characterizes the representational geometry in a given system by the representational pattern dissimilarities among the stimuli. A model representation is considered similar to a brain representation to the degree that it emphasizes the same distinctions among the stimuli. Stimulus-by-stimulus representational dissimilarity matrices can be directly compared between brain regions and model layers, side-stepping the problem of defining the correspondency mapping between the units of the model and the channels of brain-activity measurement (e.g. voxels in fMRI) (Khaligh-Razavi and Kriegeskorte, 2014; Kietzmann et al., 2012), single-cell recordings (Kriegeskorte et al., 2008; Leibo et al., 2017), M/EEG data (Cichy et al., 2017; Kirkpatrick et al., 2017), and behavioural measurements including perceptual judgements (Mur et al., 2013).

On the behavioral level, recognition performance (Cadieu et al., 2014; Hong et al., 2016; Majaj et al., 2015), perceptual confusions, and illusions provide valuable clues as to how representations in brains and DNNs may differ. For instance, it can be highly informative to understand the detailed patterns of errors (Walther et al., 2009) and reaction times across stimuli, which may reveal subtle functional differences between systems that exhibit the same overall level of task performance. Visual metamers (Freeman and Simoncelli, 2011; Wallis et al., 2016) provide a powerful tool to test for similarities in internal representations across systems. Given an original image, a modified version is created that matches the original in the model representation (for instance, a layer of a DNN), while features that do not change the representation are altered. If the human brain processed the stimuli through the same stages, it should similarly be insensitive to the two stimuli that are indistinguishable ("metameric") to the model. Conversely, an adversarial example (Goodfellow et al., 2014; Nguyen et al., 2015) is a minimal modification of an image that elicits a different category label from a DNN. For convolutional feedforward networks, minimal changes to an image (say of a bus), which are imperceptible to humans, lead the model to classify the image incorrectly (say as an

ostrich). Adversarial examples can be generated using the backpropagation algorithm down to the level of the image, to find the gradients in image space that change the classification output. This method requires omniscient access to the system, making it impossible to perform a fair comparison with biological brains, which might likewise be confused by stimuli designed to exploit the idiosyncratic aspects (Kriegeskorte, 2015). The more general lesson for computational neuroscience is that metamers and adversarial examples provide methods for designing stimuli for which different representations disagree maximally. This may enable us in the future to optimise our power to adjudicate between alternative models experimentally.

Ranging across levels of description and modalities of brain-activity measurement, from responses in single neurons, to array recordings, fMRI and MEG data, and behavioral responses, the above methods enable computational neuroscientists to investigate the similarities and differences between brains and DNNs. Future studies can explore a wide range of model units and network architectures, adding features consistent with neurobiology so as to best predict brain activity and behaviour.

1.4 Drawing insights from complex models

Deep learning has transformed machine learning and only recently found its way back into computational neuroscience, where it originated. Despite their high performance, DNNs have met with scepticism regarding their explanatory value as models of brain information processing (e.g. Kay (2017)). One of the arguments commonly put forward is that DNNs merely exchange one impenetrably complex system with another (the “black box” argument). That is, while DNNs may be able to predict neural data, researchers now face the problem of understanding what exactly the network is doing.

The black box argument is best appreciated in historical context. Shallow models are easier to understand and supported by stronger mathematical results. For example, the weight template of a linear-nonlinear model can be directly visualized and understood in relation to the concept of an optimal linear filter. Simple models can furthermore enable researchers to understand the role of each individual parameter. Overall, a model with fewer parameters is considered more parsimonious as a theoretical account.

It is certainly true that simpler models should be preferred over models with excessive degrees of freedom. Many seminal explanations in neuroscience have been derived from simple models. This argument only applies, however, if the two models provide similar predictive power. Models should be as simple as possible, but no simpler. Because the brain is a complex system with billions of parameters (presumably containing the domain knowl-

edge required for adaptive behaviour) and complex dynamics (which implement perceptual inference, cognition, and motor control), computational neuroscience will eventually need complex models. The field has to find ways to draw insight from such models. One way to draw insight from complex models is to consider their constraints at a higher level of abstraction. The computational properties of DNNs are the result of four manipulable elements: the network architecture, the input statistics, the objective function, and the learning algorithm.

A worthwhile thought experiment for neuroscientists is to consider what cortical representations would develop if the world were different. Governed by different input statistics, for instance, a different distribution of category occurrences or different temporal dependency structure, the brain may develop quite differently. This knowledge would provide us with principal insights into the objectives that it tries to solve during development. Deep learning allows computational neuroscientists to make this thought experiment a simulated reality. Investigating which aspects of the simulated world are crucial to render the learned representations more similar to the brain thereby serves an essential function in understanding of representational characteristics.

In addition to experiments with different input statistics, the network architecture can be altered to test how anatomical structure gives rise to computational function, and which features of the biological brain are required to explain a given neural phenomenon. For instance, it can be asked whether neural responses in a given paradigm are best explained by a feed-forward or a recurrent network architecture. Moreover, starting from an abstract level, biological details can be integrated into DNNs in order to see which ones prove to be required ingredients for predicting neural responses and behaviour. Current DNNs derive their power from bold abstractions. Although complex in terms of their parameter count, they are simple in terms of their component mechanisms. Biological brains draw from a richer set of dynamical primitives. It will be interesting to see to what extent incorporating more biologically inspired mechanisms can further enhance the power of DNNs and their ability to explain neural activity and animal behaviour.

Given input statistics and architecture, the missing determinants that transform the randomly initialized model into a trained DNN are the objective function and the learning algorithm. The idea of normative approaches is that neural representations in the brain can be understood as being optimised with regard to one or many overall objectives. These define what the brain should compute, in order to provide the basis for successful behaviour. While experimentally difficult to investigate, deep learning based on different cost functions allows researchers to ask the directly related inverse question: what cost functions need to be optimised such that the resulting internal representations best predict neural data? Various objectives have been suggested in both the neuroscience and machine learning

community. Feed-forward convolutional DNNs are often trained with the objective to minimize classification error (Krizhevsky et al., 2012; Simonyan et al., 2013; Yamins and DiCarlo, 2016). This focus on classification performance has proven quite successful, leading researchers to observe an intriguing correlation: classification performance is positively related to the ability to predict neural data (Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014). That is, the better the network performed on a given image set, the better it could predict neural data, although the latter was not part of the training objective.

The objective to minimize classification error in a DNN for visual object recognition requires millions of labelled training images. Although the trained DNN provides the best current predictive model of ventral stream visual processing (Khaligh-Razavi and Kriegeskorte, 2014), the process by which the model is obtained is not biologically plausible. The image labels are best viewed as a crutch for semantics, which replaces the contribution of the rest of the brain and the body in interaction with a complex dynamic environment. It is unlikely that the human brain is only trained using supervised learning, which limits the conclusions that can be made about biological learning using most current DNNs. As a result, investigating other learning paradigms is needed (Marblestone et al., 2016). Objective functions from the unsupervised domain have been suggested, which would allow the brain (and DNNs) to create error signals without external feedback. One influential suggestion is that neurons in the brain aim at an efficient sparse code, while faithfully representing the external information (Olshausen and Field, 1996; Simoncelli and Olshausen, 2001). Similarly, compression-based objectives aim to represent the input with as few neural dimensions as possible (Barlow, 1961; Bell and Sejnowski, 1997; Hyvärinen et al., 2004). Autoencoders are one example of this coding principle (Hinton and Salakhutdinov, 2006).

Harnessing information from the temporal domain, the temporal stability or slowness objective is based on the insight that latent variables that vary slowly over time are useful for adaptive behaviour. Neurons should therefore detect the underlying, slowly changing signals, while disregarding fast changes likely due to noise, potentially simplifying readout from downstream neurons (Berkes and Wiskott, 2005; Kayser et al., 2001; Kording et al., 2004). Slow feature analysis methods can be viewed as a probabilistic modelling approach fit using maximum likelihood estimates (Turner and Sahani, 2007). Stability can be optimised across layers in hierarchical systems, if each subsequent layer tries to find an optimally stable solution from the activation profiles in previous layer. This approach was shown to lead to invariant codes for object identity (Franzius et al., 2008) and viewpoint-invariant place-selectivity (Franzius et al., 2007; Wyss et al., 2006). Experimental evidence in favour of the temporal stability objective has been provided by electrophysiological and behavioral studies (Li and DiCarlo, 2008, 2010; Wallis and Bülthoff, 2001).

Many implementations of classification, sparseness and stability objectives ignore the action repertoire of the agent. Yet, different cognitive systems living in the same world may exhibit different neural representations because the requirements to optimally support action may differ. Deep networks optimizing the predictability of the sensory consequence (Weiller et al., 2010), or cost of a given action (Mnih et al., 2015) have started incorporating the corresponding information. Finally, there does not have to be one true objective that the brain optimises, as neural cost functions are not necessarily constant across regions or time (Marblestone et al., 2016).

As a result, one way to draw theoretical insights from DNN models is to explore what architectures, input statistics, objective functions, and learning algorithms yield models predictive of neural activity and behaviour. This approach does not elucidate the role of individual units or connections in the brain, but it can reveal what features of biological structure support what aspects of a system's function and what objectives the biological system might be optimizing.

In addition to contextualizing the black box in this way, we can also open the black box and look inside. Given a model that accounts for neural activity and behaviour, much is won. Unlike a biological brain, a model is entirely accessible to scrutiny and manipulation, enabling, for example, high-throughput "in silico" electrophysiology. One method for visualizing a unit's preferences is to approximately undo the operations performed by a convolutional DNN (Zeiler and Fergus, 2014) to visualize what image features drive a given unit deep in a neural network in the context of a particular image. This results in visualisations such as those shown in Figure 1.3. A closely related technique is to use backpropagation (Rumelhart et al., 1986) to calculate the change in the input needed to drive or inhibit the activation of any unit in a DNN (Simonyan and Zisserman, 2014; Yosinski et al., 2015). We can select an image that strongly drives the unit and compute the gradient in image space that corresponds to enhancing the unit's activity even further. The gradient image shows how small adjustments to the pixels affect the activity of the unit. For example, if the image strongly driving the unit is a person next to a car, the corresponding the gradient image might reveal that it is really the face of the person driving the unit's response. In that case, the gradient image would deviate from zero only in the region of the face and adding the gradient image to the original image would accentuate the facial features. To understand the unit's response, we might have to look at its gradient in image space for many different test images to get a sense of the orientation of its tuning surface around multiple reference points (test images).

Backpropagation can also be used to iteratively optimise images to strongly drive a particular unit, starting from a noise image. This yields complex psychedelic looking patterns

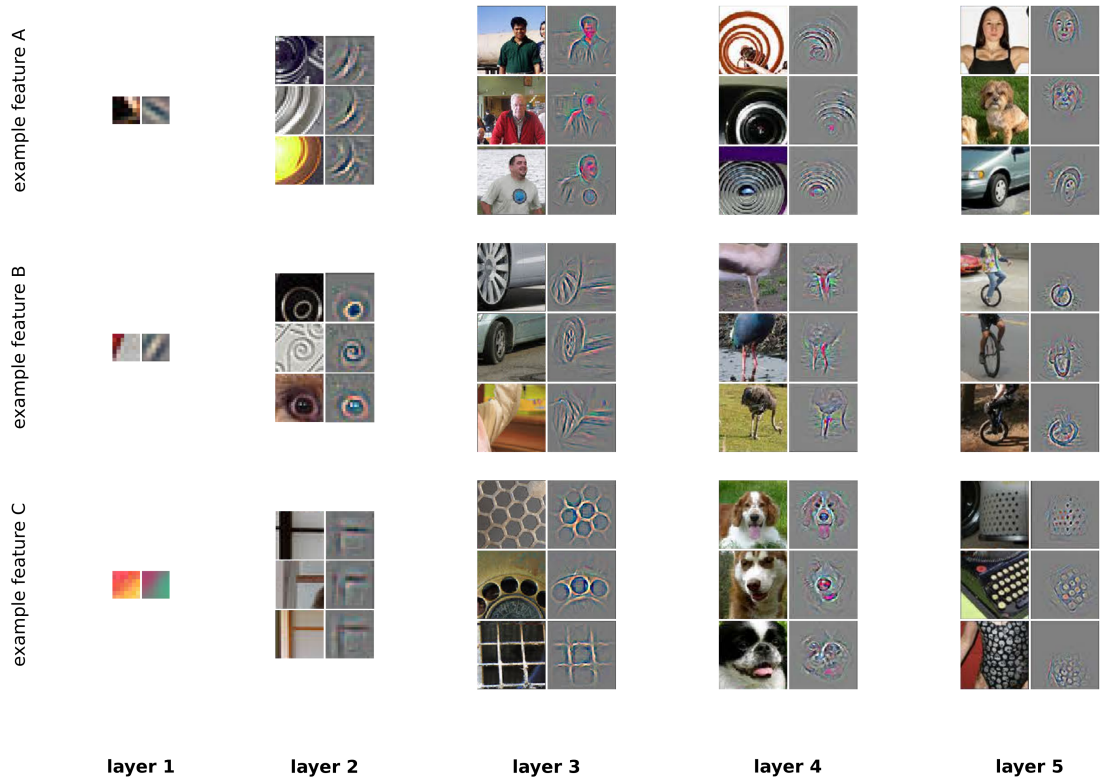


Fig. 1.3 Visualising the preferred features of internal artificial neurons. Activations in a random subset of feature maps across layers for strongly driving ImageNet images projected down to pixel space (adapted from Zeiler and Fergus (2014)).

containing features and forms, that a network has learned through its task training. It is important to note that the tuning function of a unit deep in a network cannot be characterized by a single visual template. If it could, there would be no need for multiple stages of nonlinear transformation. However, visualisations of receptive field properties provide intuitions about the neuronal selectivity at different layers or time-points. In summary, "in silico" electrophysiology enables researchers to measure and manipulate every single neuron, if required. In addition, researchers can gain an understanding at a more abstract level, by observing the effects of predictive performance of changes to the architecture, input statistics, objective function, and learning algorithm.

1.5 What neurobiological details matter to brain computation?

A second concern about DNNs is that they abstract too much from biological reality to be of use as models for neuroscience. Whereas the black box argument states that DNNs are too complex, the biological realism argument states that they are too simple. Both arguments have merit. It is conceivable that a model is simultaneously too simple (in some ways) and too complex (in other ways). However, this raises a fundamental question: Which features of the biological structure should be modelled and which omitted to explain brain function? Abstraction is the essence modelling and is the driving force of understanding. If the goal of computational neuroscience is to understand brain computation, then we should seek the simplest models that can explain task performance and predict neural data. The elements of the model should map onto the brain at some level of description. However, what biological elements must be modelled is an empirical question. Large-scale models should enable an exploration of the level of detail required (Eliasmith and Trujillo, 2014). DNNs are important not because they capture the biological features that matter to brain computation, but because they provide a minimal functioning starting point for exploring what biological details matter to brain computation. If, for instance, spiking models outperformed rate-coding models at explaining neural activity and task performance (for example in tasks requiring probabilistic inference (Buesing et al., 2011), then this would be strong evidence in favour of spiking models.

Convolutional DNNs, like AlexNet and VGG, were built to optimise performance, rather than biological plausibility. However, these models draw from a history of neuroscientific insight and share many qualitative features with the primate ventral stream. The defining property of convolutional DNN is the use of convolutional layers. These have two main characteristics: (1) local connections that define receptive fields and (2) parameter sharing between neurons across the visual field. Whereas spatially restricted receptive fields are a prevalent biological phenomenon, parameter sharing is biologically implausible. However, biological visual systems learn qualitatively similar sets of basis features in different parts of a retinotopic map, and similar results have been observed in models optimizing a sparseness objective (Güçlü and van Gerven, 2014; Olshausen and Field, 1996).

Moving toward greater biological plausibility with DNNs, locally connected layers that have receptive fields without parameter sharing were suggested (Uetz and Behnke, 2009). Researchers have already started exploring this type of DNN, which was shown to be very successful in face recognition (Sun et al., 2015; Taigman et al., 2014). One reason for this is that locally connected layers work best in cases where similar features are frequently

present in the same visual arrangement, such as faces. In the brain, retinotopic organisation principles have been proposed for higher-level visual areas (Levy et al., 2001), and similar organisation mechanisms may have led to faciotopy, the spatially stereotypical activation for facial features across the cortical surface in face-selective regions (Henriksson et al., 2015).

Another aspect in which convolutional AlexNet and VGG deviate from biology is the focus on feed-forward processing. Feedforward DNNs compute static functions, and are therefore limited to modelling the feed-forward sweep of signal flow through a biological visual system. Yet, recurrent connections are a key computational feature in the brain, and represent a major research frontier in neuroscience. In the visual system, too, recurrence is a ubiquitous phenomenon. Recurrence is likely the source of representational transitions from global to local information (Matsumoto et al., 2004; Sugase et al., 1999). The timing of signatures of facial identity (Barragan-Jason et al., 2013; Freiwald and Tsao, 2010) and social cues, such as direct eye-contact (Kietzmann et al., 2017a), point towards a reliance on recurrent computations. Finally, recurrent connections likely play a vital role in dealing with occlusion (Spoerer et al., 2017; Wyatte et al., 2012, 2014).

The first generation of DNNs focused on feed-forward, but the general class of DNNs can implement recurrence. Lateral recurrent connections are often used in neural networks to normalize representations, as in local response normalisation (Krizhevsky et al., 2012) and divisive normalisation (Carandini and Heeger, 2012). By using lateral recurrent connections, DNNs can implement visual attention mechanisms (Mnih et al., 2014). Lateral recurrent connections can also be added to convolutional DNNs (Liang and Hu, 2015; Spoerer et al., 2017), increasing the effective receptive field size of each unit. In addition to local feed-forward and lateral recurrent connections, the brain also uses local feedback, as well as long-range feedforward and feedback connections. While missing from the convolutional DNNs previously used to predict neural data, DNNs with these different connection types have been implemented (He et al., 2016; Liao and Poggio, 2016; Srivastava et al., 2015). The field of recurrent convolutional DNNs is still in its infancy, and the effects of lateral and top-down connections on the representational dynamics in these networks, and their predictive power for neural data are yet to be fully explored. Nevertheless, recurrent connections are an exciting tool for computational neuroscience and will likely allow for insights into the recurrent computational dynamics of the brain.

Apart from architectural considerations, backpropagation, the most successful learning algorithm for DNNs, has classically been considered neurobiologically implausible. Rather than as a model of biological learning, backpropagation may be viewed as an efficient way to arrive at reasonable parameter estimates, which are then subject to further tests. That is, even if backpropagation is considered a mere technical solution, the resulting model

may still be a good model of the dynamics in the system after learning. However, there is also a growing literature on biologically plausible forms of error-driven learning. If the brain does optimise cost functions during development and learning (which can be diverse, and supervised, unsupervised, or reinforcement-based), then it will have to use a form of optimisation mechanism, such as stochastic gradient descent techniques. The current literature suggests several neurobiologically plausible ways in which the brain could adjust its internal parameters to optimise such objective functions (Lee et al., 2015; Lillicrap et al., 2016; O'Reilly, 1996; Whittington and Bogacz, 2017). These methods have been shown to allow deep spiking neural networks to learn simple vision tasks (Guerguiev et al., 2017). The brain might not be performing the exact algorithm of backpropagation, but it might have a mechanism for modifying synaptic weights in order to optimise one or many objective functions (Marblestone et al., 2016).

In addition to architectural considerations and optimisation, there are other ways in which DNNs abstract from biological detail. For instance, DNNs are generally deterministic, while biological networks are stochastic. While much of this stochasticity is commonly thought to be noise, it has been hypothesized that this variability could code for uncertainty (Fiser et al., 2010a; Hoyer and Hyvärinen, 2003; Orbán et al., 2016a). Furthermore, current recurrent convolutional DNNs often only run for a few time steps, and the roles of dynamical features found in biological networks, such as oscillations, are only beginning to be tested (Finger and König, 2014; Reichert and Serre, 2013; Siegel et al., 2012). The non-recurrent DNNs also only consider static images, whereas humans receive time series sensory inputs. Another abstraction is the omission of spiking dynamics. However, DNNs with spiking neurons can be implemented (Tavanaei and Maida, 2016) and represent an exciting frontier of deep learning research. These considerations show that it would be hasty to judge the merits of DNNs based on the level of abstraction chosen in the first generation. The usage of DNNs in computational neuroscience is still in its infancy. Integration of biological detail will require close collaboration between and experimental neuroscientists and anatomists.

Computational neuroscience comprises a wide range of models, defined at various levels of biological and behavioral detail. For instance, many conductance-based models contain large amounts of parameters to explain single or few neurons at great level of detail but are typically not geared towards behaviour. DNNs, at the other end of the spectrum, use their high number of parameters not to account for effects on the molecular level, but to achieve behavioral relevance, while accounting for overall neural selectivity. Explanatory merit is not only gained by biological realism (because this would render human brains the perfect explanation for themselves), nor does it directly follow from simplistic models that cannot account for complex animal behaviour. The space of models is continuous and

neuroscientific insight works across multiple levels of explanation, following top-down and bottom-up approaches (Craver, 2007).

1.6 What is next?

Deep neural networks provide a flexible framework for modelling neurocomputation. However, the DNN models built by the machine learning community will need to be adapted for them to become more meaningful models of neural computation. There are many adaptations that can be made to classical DNNs to either make them model neural or behavioural data better or to make them more neurobiologically plausible (e.g. recurrent connections, more biologically plausible objective functions, and stochasticity). While DNNs could potentially be used to model many neural information processing systems in the future, DNNs have thus far been most successful as models of human visual perception, particularly AlexNet (Krizhevsky et al., 2012) and VGG-16 (Simonyan and Zisserman, 2014). In this thesis we explore two methods for adapting DNNs to better explain neural data for human visual perception: (1) actively constraining internal layers of a DNN using a known representational space and (2) using stochasticity and sampling to model human confidence for image classification.

DNNs trained to perform a task are not guaranteed to have internal representations similar to those found in the human brain. In Chapter 2, we propose deep representational distance learning (RDL), a method for driving internal representational spaces of DNNs into alignment with other representational spaces defined by a teacher (e.g. the human brain). In this chapter, we use DNNs as the teacher, but in the future we plan to use fMRI activation patterns as the teacher.

As discussed in Section 1.5, stochasticity and sampling have been proposed as potential methods for biological neural networks to code for uncertainty. In Chapter 3, we evaluate how well Bayesian DNNs can represent their own predictive uncertainty by sampling during training and testing. We demonstrate that approximate variational DNNs perform the trained task well and robustly represent their own uncertainty. In order to compare to human behaviour, we applied these methods to large-scale object recognition in Chapter 4. We show that these Bayesian CNNs with sampling have improved accuracy and better represent their own uncertainty. We also demonstrate that the stochastic representations of Bayesian CNNs better explain human confidence scores during image classification. Our results demonstrate two potential paths for adapting DNNs, a brain-inspired AI model class, in order to better model biological brains.

Chapter 2

Adapting neural networks with deep representational distance learning

This chapter is based on a publication by McClure and Kriegeskorte (2016).

Summary

Deep neural networks (DNNs) provide useful models of visual representational transformations. We present a method that enables a DNN (student) to learn from the internal representational spaces of a reference model (teacher), which could be another DNN or, in the future, a biological brain. Representational spaces of the student and the teacher are characterised by representational distance matrices (RDMs). We propose representational distance learning (RDL), a stochastic gradient descent method that drives the RDMs of the student to approximate the RDMs of the teacher. We demonstrate that RDL is competitive with other transfer learning techniques for two publicly available benchmark computer vision datasets (MNIST and CIFAR-100), while allowing for architectural differences between student and teacher. By pulling the student's RDMs towards those of the teacher, RDL significantly improved visual classification performance when compared to baseline networks that did not use transfer learning. In the future, RDL may enable combined supervised training of deep neural networks using task constraints (e.g. images and category labels) and constraints from brain-activity measurements, so as to build models that replicate the internal representational spaces of biological brains.

2.1 Introduction

Deep neural networks (DNNs) have recently been highly successful for machine perception, particularly in the areas of computer vision using convolutional neural networks (CNNs) (Krizhevsky et al., 2012) and speech recognition using recurrent neural networks (RNNs) (Deng et al., 2013). The success of these methods depends on their ability to learn good, hierarchical representations for these tasks (Bengio, 2012). DNNs have not only been useful in achieving engineering goals, but also as models of computations in biological brains. Several studies have shown that DNNs trained only to perform object recognition learn representations that are similar to those found in the human ventral stream (Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014). The models benefit from task training, which helps determine the large number of parameters and bring the domain knowledge required for feats of intelligence such as object recognition into the models. This is in contrast to the earlier approach in visual computational neuroscience of using nonlinear systems identification techniques to set the parameters exclusively on the basis of measured neural responses to large sets of stimuli (Naselaris et al., 2011). The latter approach is challenging for deep neural networks, because the high cost of brain-activity measurement limits the amount of data that can be acquired (Yamins and DiCarlo, 2016). Ultimately, task-based constraints will have to be combined with constraints from brain-activity measurements to model information processing in biological brains.

Here we propose a method that enables the training of DNNs with combined constraints on the desired outputs and the internal representations. We demonstrate the method by using another neural net model as the reference system whose internal representations the DNN is to emulate. One method for doing so would be to have a layer in a DNN linearly predict individual measured responses (e.g. fMRI voxels or neurons), and backpropagate the error derivatives from the linear measured-response predictors into the DNN. However, the linear measurement prediction model has a large number of parameters ($n_{units} \times n_{responses}$). An alternative approach is to constrain the DNN to replicate the representational distance matrices (RDMs) estimated from brain responses. In this chapter, we take a step in that direction by considering the problem of training a DNN (student) to model the sequence of representational transformations in another artificial system (teacher), a CNN trained on different data.

Our technique falls in the class of transfer learning methods. In the deep learning literature, several such techniques have been proposed both for pulling a DNN's internal representations towards the task target and for transferring knowledge from a teacher DNN to a student DNN. We begin by briefly considering the previous transfer learning approaches.

Pulling internal representations toward the desired output using auxiliary classifiers

Recently, it has been investigated how the error signal reaching an internal layer through backpropagation can be complemented by auxiliary error functions. These more directly constrain internal representations using auxiliary optimization goals. A variety of methods using auxiliary error functions to pull representations toward the desired output have been proposed.

Weston et al. (2012) proposed semi-supervised embeddings to augment the error from the output layer. A reference embedding of the inputs was used to guide representational learning. The embedding constraint was implemented in different ways: inside the network as a layer, as part of the output layer, or as an auxiliary error function that directly affected a particular hidden layer. Weston et al. discussed a variety of embedding methods that could be used, including multidimensional scaling (MDS) (Kruskal, 1964) and Laplacian Eigenmaps (Belkin and Niyogi, 2003). The addition of these semi-supervised error functions led to increased accuracy compared to DNNs trained using output layer backpropagation alone.

Lee et al. (2014) also showed that auxiliary error functions improve DNN representational learning. Instead of using semi-supervised methods, they performed classification with a softmax or L2SVM readout at a given intermediate hidden layer. The softmax layer allowed the output of a network to be treated as a probability distribution by performing normalised exponentiation on the previous layer's activations ($y_i = e^{x_i} / \sum_j e^{x_j}$). The error of the intermediate-level readout was then backpropagated to earlier layers to drive intermediate layers directly towards the target output. The gradients from these classifiers were linearly combined with the gradients from the output layer classifier. This technique resulted in improved accuracies for several datasets.

A challenge in training very deep networks is the problem of vanishing gradients. Layers far from the output may receive only a weak learning signal via conventional backpropagation. Auxiliary error functions were successfully applied to these very deep networks by Szegedy et al. (2015) to inject a complementary learning signal at internal layers by constraining representations to better discriminate between classes. This was implemented in a very large CNN which won the ILSVRC14 classification competition (Russakovsky et al., 2015). In this DNN, two auxiliary networks were used to directly backpropagate from two intermediate layers back through the main network. Similar to the method used in Lee et al. (2014), the parameters for the layers in the main network directly connected to auxiliary networks were updated using a linear combination of the backpropagated gradients from later layers and the auxiliary network.

Wang et al. (2015) investigated the effectiveness of auxiliary error functions in very large CNNs and their optimal placement. They selected where to place these auxiliary functions by

measuring the average magnitude of the conventional backpropagation error signal at each layer. Auxiliary networks, similar to those used in Szegedy et al. (2015), were placed after layers with vanishing gradients. These networks consisted of a convolutional layer followed by three fully connected layers and a softmax classifier. As in Lee et al. (2014) and Szegedy et al. (2015), the auxiliary gradients were linearly combined to update the model parameters. Adding these supervised auxiliary error functions led to an improved accuracy for two very large datasets, ILSVRC12 (Russakovsky et al., 2015) and MIT Places (Zhou et al., 2014).

Pulling the representations of a student towards those of a teacher using transfer learning

In transfer learning (Bengio, 2012), knowledge learned during training on one data distribution is then reused while training on a different data distribution. This is sometimes framed as a teacher model trained on one task being used to improve training of new student model on a new task. Different methods within transfer learning focus on improving different aspects of the student's training. Transfer learning can be used to increase accuracy on the new task, allow generalization to new classes, increasing training sample efficiency, and decreasing training time.

We seek to develop a method that can eventually constrain the internal representations of an NN trained on a large number of classes using neural data. The method must enable generalization from a few input examples from a few categories to many input examples from many categories, since neural data often is only collected for a small number of stimuli. The transferred knowledge would ideally be useful for the task that the NN is being trained on and would ideally lead to an accuracy increase. Importantly, the method must not rely on architectural similarity between the teacher and the student, since the network architecture of the brain is unknown and certainly different from the exact model NN architecture.

One of the most prominent techniques for performing transfer learning is to initialise the weights of the student network to those of the teacher. The network is then trained on a different task or using different data. This can lead to improved training speed, and improved network accuracy (Yosinski et al., 2014). However, this requires that the teacher and student have the same, or very similar, architectures, which may not be desirable, especially if the teacher is a biological neural network.

Another popular group of transfer learning methods is one-shot/k-shot learning (Burgess et al., 2016; Li et al., 2006; Srivastava and Salakhutdinov, 2013). These techniques primarily seek to increase sample efficiency when generalizing to new categories. As with weight initialization, most of these methods assume very similar architectures between the teacher and the student. This makes these methods non-ideal for our end goal.

One method that seeks to improve student accuracy, while not requiring similar architectures, is linear prediction of internal teacher representations from selected internal student representations. Romero et al. (2014) proposed a model compression method for transferring the knowledge of a wide and shallow teacher network to a thin and deep student network, called FitNet. This method Pre-trained a network by constraining an intermediate layer of the student network to have representations that could linearly predict 'hints' from the teacher network (i.e. activation patterns at a corresponding layer in the teacher network). After this, the network was fine-tuned using the technique proposed in Hinton et al. (2015). The FitNet method was shown to improve the student's classification accuracy. This method, however does not actively constrain the internal representations of the student and doing so at multiple hidden layers would lead to a dramatic increase in learnable parameters.

In this chapter, we introduce an auxiliary error function that enables a student network to learn from the internal representational spaces of a teacher that has either a similar or a different architecture. The method constrains the student's representational distances in a set of layers to approximate those of the teacher. The student can thus learn the computational transformations discovered by the teacher, leading to improved representational learning during training.

2.2 Methods

Our method, representational distance learning (RDL), enables DNNs to learn from the representations of other models to improve performance. As in Lee et al. (2014); Szegedy et al. (2015); Wang et al. (2015), we utilise auxiliary error functions to train internal layers directly in conjunction with the error from the output layer found via backpropagation. We propose an error function that maximises the similarity between the representational spaces of a student DNN and that of a teacher model.

2.2.1 Representational Distance Matrices

In order to compare the representational spaces of models, a method must be used to describe them. As discussed in Weston et al. (2012), a representational space can be characterised by the pairwise distances between representations. This idea has been used in several methods such as MDS, which seeks to reduce the dimensionality of data while minimizing the error between the pairwise distance matrix of the original data and the reduced dimensionality data (Kruskal, 1964). Kriegeskorte et al. (2008) proposed using the matrix of pairwise dissimilarities between representations of different inputs, which they

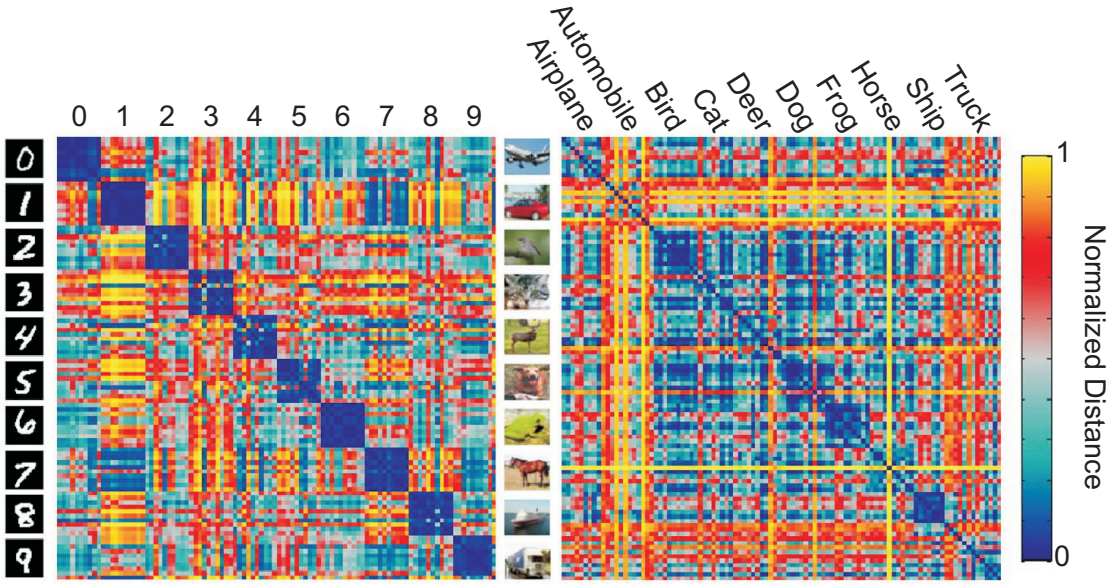


Fig. 2.1 Example CNN-based representational distance matrices (RDMs). The RDMs of the output layer of CNNs for ten random images of each class from MNIST (left) and CIFAR-10 (right) made using the RSA toolbox (Nili et al., 2014).

called representational distance, or dissimilarity, matrices (RDMs), to compare computational models and neurological data. More recently, Khaligh-Razavi and Kriegeskorte (2014) used this technique to analyse several computer vision models, including the CNN proposed in Krizhevsky et al. (2012), and neurological data. Any distance function could be used to compute the pairwise dissimilarities, for instance the Euclidean or correlation distances. An RDM for a DNN can be defined by:

$$RDM(X;f_m)_{i,j} = d(f_m(x_i;W_m), f_m(x_j;W_m)) \tag{2.1}$$

where X is a set of n inputs (e.g. a mini-batch or a subset of a mini-batch), f_m is the neuron activations at layer m, x_i and x_j are single inputs, W_m is the weights of the neural network up to layer m, and some distance, or dissimilarity, measure d.

In addition to characterizing the information present in a particular layer of a DNN, RDMs can be used to visualise the representational space of a layer in a DNN (Figure 2.1). Currently, understanding and visualizing the information captured by internal layers in a DNN is challenging. Zeiler and Fergus (2014) proposed a method for visualizing the input features which activate internal neurons at varying layers using deconvolutional neural

networks. Yosinski et al. (2015) also proposed methods for visualizing the activations of a DNNs for a given input. However, these methods do not show the categorical information of each representational layer. Visualizing the similarity of labelled inputs at layers of interest, via an RDM, allow clusters inherent to the learned representational transformations to be viewed.

2.2.2 Representational Distance Learning

RDL uses an auxiliary error functions that maximises the similarity between the RDMs of a student and the RDMs of a teacher at several layers. This is motivated by the idea that RDMs, or distance matrices in general, can characterise the representational space of a model. DNNs seek to learn a set of hierarchical representations. For classification, this culminates in finding a representational space where different classes are separable. RDL allows a DNN to learn from the representations of a different, potentially better, model by maximizing the similarity between the RDMs of the DNN being trained and the target model at several layers. Unlike in Ba and Caruana (2014); Bucilua et al. (2006) and Hinton et al. (2015), RDL not only directly trains the output representation, but also the representations of hidden layers. As discussed in Bengio (2012), however, large datasets can prohibit the use of pairwise techniques, since the number of comparisons grows quadratically with dataset size. To address this, our technique only uses a random subset of all pairwise distances for each parameter update. This allows the speed of our method to be constrained by the subset size and not the overall number of training examples, which is usually several orders of magnitude larger.

In order to maximise the similarity between the RDM of a DNN layer being trained and a target RDM, we propose minimizing the mean squared error between the two RDMs. This corresponds to making all possible pairwise distances as similar as possible:

$$E_{aux}(X;f_m;T_m) = \frac{2}{n(n-1)} \sum_{(i,j)|i<j} (RDM(X;f_m)_{i,j} - T_{m,i,j})^2 \tag{2.2}$$

where X is a set of n inputs (e.g. a mini-batch or a subset of a mini-batch), f_m is the neuron activations at layer m, and $T_{m,i,j}$ is the distance between the teacher's representations of input x_i and input x_j at layer m. The function d used to calculate the RDMs (Eq. 1) could be any dissimilarity or distance function, but we chose to use the mean squared error (MSE). This results in the average auxiliary error with respect to neuron k of f_m, $f_{m,k}$, for input x_i and the weights of the neural network up to layer m, W_m, being defined as:

$$\frac{\partial E_{aux}(x_i;X;f_m;T_m)}{\partial f_{m,k}} = \frac{8}{n(n-1)} \sum_{j|j\neq i} (RDM(X;f_m)_{i,j} - T_{m,i,j})(f_{m,k}|_{x_j}^{x_i}) \tag{2.3}$$

Table 2.1 The convolutional neural network (CNN) architecture used for MNIST.

Layer	Kernel Size	# Features	Stride	Non-linearity	Other
Conv-1	5x5	32	1	ReLU	-
MaxPool-1	3x3	32	3	Max	-
Conv-2	5x5	64	1	ReLU	-
MaxPool-2	2x2	64	2	Max	-
FC	1500	200	-	ReLU	Dropout ($p = 0.5$)
Linear	200	10	-	-	-

where $f_{m,k}|_{x_j}^{x_i} = f_{m,k}(x_i;W_m) - f_{m,k}(x_j;W_m)$.

However, calculating the error for every pairwise distance can be computational expensive, so we estimate the error using a random subset, P, of the pairwise distances for each update of a network's parameters. This leads to the auxiliary error gradient being approximated by:

$$\frac{\partial E_{aux}(x_i;X;f_m;T_m)}{\partial f_{m,k}} \approx \frac{8}{|X_P||P_{x_i}|} \sum_{(i,j)\in P_{x_i}} (RDM(X;f_m)_{i,j} - T_{m,i,j})(f_{m,k}|_{x_j}^{x_i}) \tag{2.4}$$

where X_P is the set of all images contained in P, P_{x_i} is the set of all pairs, (i,j), in P that include input x_i and another input, x_j. If an image is not sampled, its auxiliary error is zero.

The total error of $f_{m,k}$ for input x_i is calculated by taking a linear combination of the auxiliary error at layer m and the error from backpropagation of the output error function and any later auxiliary functions. These terms are combined using weighting hyper parameter α, similar to the method discussed in Lee et al. (2014), Szegedy et al. (2015), and Wang et al. (2015). In RDL, α is the weight of the RDL error in the overall error function. Subsequently, the error gradient at a layer with an auxiliary error function is defined as:

$$\frac{\partial E_{total}(x_i;y_i;X;f_m;T_m)}{\partial f_{m,k}} = \frac{\partial E_{backprop}(x_i;y_i;f_m)}{\partial f_{m,k}} + \alpha \frac{\partial E_{aux}(x_i;X;f_m;T_m)}{\partial f_{m,k}} \tag{2.5}$$

This error is then used to calculate the error of earlier layers in the DNN using back-propagation. As discussed by Lee et al. (2014) and Wang et al. (2015), the value of α was decayed as training progressed. Throughout RDL training, α was updated following $\alpha_{t+1} = \alpha_0 * (1 - t/t_{max})$ where t is the epoch number and t_{max} is the total number of epochs. By using this decay rule, the auxiliary error function initially helps drive the parameters to good values while allowing the DNN to converge predominantly using the output error by the end of training.

Table 2.2 Test errors for MNIST trained convolutional neural networks (CNNs) and the CIFAR-100 trained "Network in Network" (NiN) models. (Note: The performance of the teacher for the CIFAR-100 classification is not shown, since it was trained on CIFAR-10 and, therefore, predicted across 10 not 100 classes, making it unable to perform the CIFAR-100 task.)

MNIST	
Method	**Error (%)**
Baseline CNN	0.63
Teacher	0.56
Teacher with Fine-tuning	0.48
Student with Deep Supervision	0.55
Student with Hints	0.56
Student with RDL	0.49
CIFAR-100	
Method	**Error (%)**
Baseline NiN	30.68
Teacher with Fine-tuning	29.39
Student with Deep Supervision	29.46
Student with Hints	29.37
Student with RDL	28.77

2.3 Experiments

We performed two experiments with the goal of determining whether transferring knowledge from a teacher DNN to a student DNN led to a significant increase in the accuracy and led to internal representations that were more similar to the teacher than those of the baseline network, which was trained without any auxiliary error functions or transfer learning. We compared RDL to four different methods: (1) a baseline without any transfer learning or auxiliary functions, (2) fine-tuning after directly copying the weights of the teacher to test, (3) pre-training an internal layer of the student to linearly predict a corresponding layer in the teacher using 'hints', and (4) deep supervision using an auxiliary classifier.

For the two experiments, we used four different datasets, MNIST, InfiMNIST, CIFAR-10, and CIFAR-100. These experiments show that the knowledge stored in the weights of a teacher network can be transferred to a student network using the representational distances learned by a teacher trained on a related task.

2.3.1 MNIST

MNIST is a dataset of 28x28 images of handwritten digits from ten classes, 0 through 9 (LeCun et al., 1998). The dataset contains 60,000 training images and 10,000 test images.

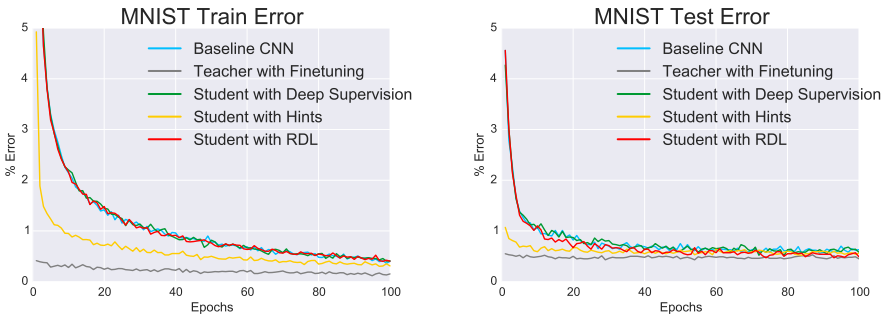


Fig. 2.2 Train and test errors of the MNIST trained CNNs throughout training as the tested convolutional neural.

A 10,000 image subset of the training data was used as a validation set for hyper-parameter tuning. No pre-processing or data augmentation was applied. InfiMNIST is a dataset that extends the MNIST dataset using pseudo-random deformations and translations (Loosli et al., 2007). The first 10,000 non-MNIST InfiMNIST examples were used as a validation set and the next 120,000 examples were used as a training set for the teacher network. Each tested network had the same architecture (Table 3.2), excluding any auxiliary error functions. The deeply supervised network had linear auxiliary softmax classifiers placed after the max pooling layers and α was decayed using $\alpha_{t+1} = \alpha_t * 0.1 * (1 - t/t_{max})$, as proposed in Lee et al. (2014). For the fine-tuning network, the weights were initialised as the weights of the teacher network instead of being randomly initialised. After this, the network was trained normally. The RDL network had auxiliary error functions after both max pooling layers and the fully connected layer. 5% (500) of the image pairs per mini-batch were used to calculate the RDL auxiliary errors. A momentum of 0.9 and a mini-batch size of 100 were used for all networks trained on MNIST and InfiMNIST.

In addition to the classification error (Figure 2.2 and Table 2.2), we used the McNemar exact test (Edwards, 1948) to evaluate whether a network was significantly more accurate in classifying a random image from the distribution from which the images in the training and test sets were drawn. The results (Table 2.3) show that the fine-tuning and RDL methods both significantly improve accuracy compared to the baseline CNN. They are, however, not significantly different, showing the ability of RDL to indirectly transfer the knowledge of the teacher network. The fine-tuned network is also significantly better than the teacher and the 'hint' network, unlike RDL, because RDL actively constrains the student network to imitate the teacher, while 'hint' pre-training only affects initialisation.

In order to further compare the trained networks, RDMs were generated for each fully trained model. Figure 2.3 shows RDMs for 100 random test images, 10 from each class. This visualisation emphasises the class clustering as inputs are transformed from pixel space to

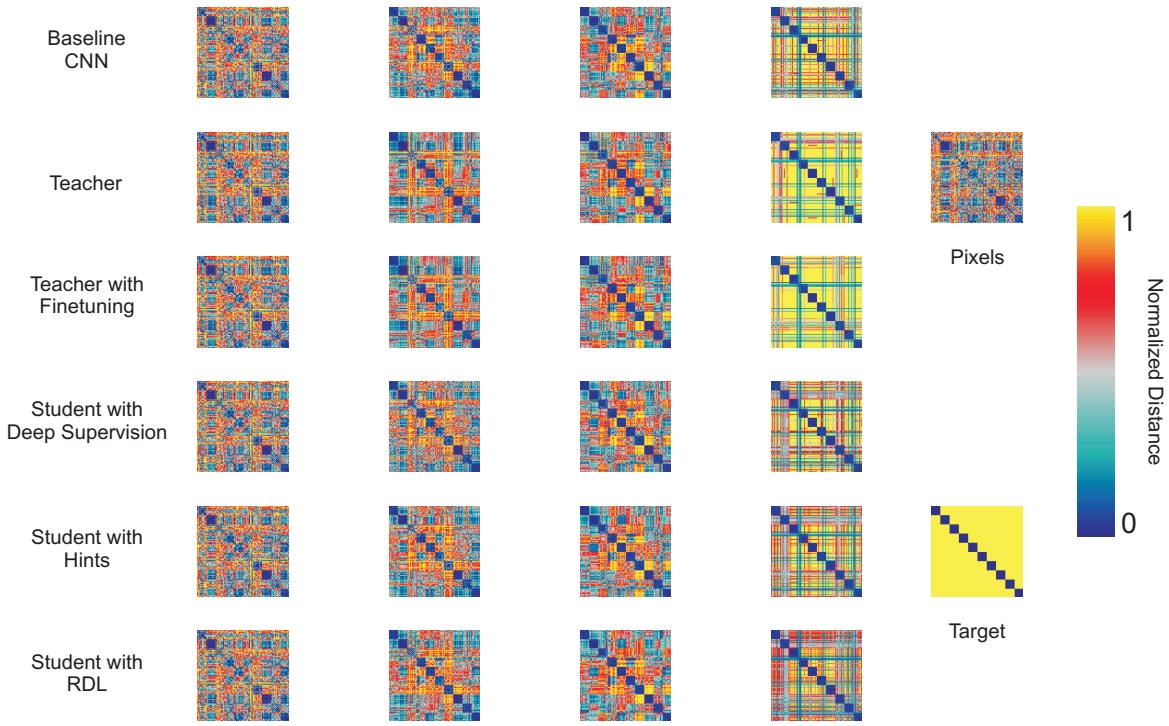


Fig. 2.3 Representational distance matrices (RDMs) for different layers of the MNIST trained CNNs. RDMs using the Euclidean distance for the first and second convolutional layers as well as the fully connected (FC) and softmax layers of the CNN tested methods, the raw pixel data, and the target labels for 10 random class exemplars from MNIST. Note that the target RDM was generated by computing the RDM of the one-hot vectors used as labels during training.

Table 2.3 The McNemar exact test p-values for the tested CNNs trained on MNIST. Arrows indicate a significant difference ($p < 0.05$,uncorr.) and point to the better model.

	Baseline	Teacher	Fine-tuning	Deep Supervision	Hints	RDL
Baseline	—	0.38	0.00 ↑	0.11	0.34	0.01 ↑
Teacher	0.38	—	0.01 ↑	0.66	0.89	0.20
Fine-tuning	0.00 ←	0.01 ←	—	0.14	0.04 ←	0.63
Deep Supervision	0.11	0.66	0.14	—	0.64	0.39
Hints	0.34	0.89	0.04 ↑	0.64	—	0.17
RDL	0.01 ←	0.20	0.63	0.39	0.17	—

label space. Some classes are already clustered in pixel space. For instance, 1s, 7s and 9s each have large blocks along the diagonal portion of the pixel RDM. However, by looking at the rows and columns we can see that these classes are difficult to separate from one another. After the first convolutional layer, class clustering increases, especially for the baseline CNN. After the second convolutional layer, class clustering increases for every model and other class relationships become apparent. For instance, 3s and 5s are becoming increasingly different from other classes, but are still similar to each other. Also, 1s remain similar to many other classes. The fully connected (FC) layer leads to stronger, but not perfect, class clustering. As expected, the softmax layer leads to extremely strong class distinction. However, most of the models still view 1s as similar to other classes, as seen by the large horizontal and vertical grey stripes. The notable exception is the fine-tuned CNN, which had the lowest testing error.

While viewing the RDMs directly can make certain facts about the transformations performed by the models evident, it can be hard to compare RDMs to each other by visual inspection. To better understand the relationships between the representations of the different models, we calculate the correlation distance between each pair of RDMs and use MDS to create a 2-D plot showing the relative position in representational space of the transformations learned by the various trained networks (Figure 2.4). This allows for drawing several qualitative conclusions. As expected, the RDMs of the networks start close to the pixel-based RDM and become more similar to the target RDM the deeper the layer. The differences between the evaluated techniques can most clearly be seen at the 2nd (Conv2) and 3rd (FC) layers. As expected: (1) the network initialised with the weights of the teacher and then fine-tuned has the most similar RDMs to the teacher, (2) deep supervision pulls the RDMs of the student towards the target, (3) RDL pulls the RDMs of the student toward and the RDMs of the teacher, especially at 3rd layer.

2.3.2 CIFAR-100

In order to test RDL on a more interesting problem, we performed transfer learning from CIFAR-10 to CIFAR-100. This experiment consists of transferring knowledge learned in an easier task to a harder one, something that is useful in many instances. CIFAR-100 is a dataset of 32x32 color images each containing one of one hundred objects. The dataset contains 50,000 training images and 10,000 test images. A 10,000 image subset of the training data was used as a validation set for hyper-parameter tuning. CIFAR-10 is also a dataset of 32x32 color images, but containing only ten distinct classes instead of one hundred. CIFAR-10 also contains 50,000 training images and 10,000 test images. For both datasets, the data were pre-processed using global contrast normalisation. During training, random

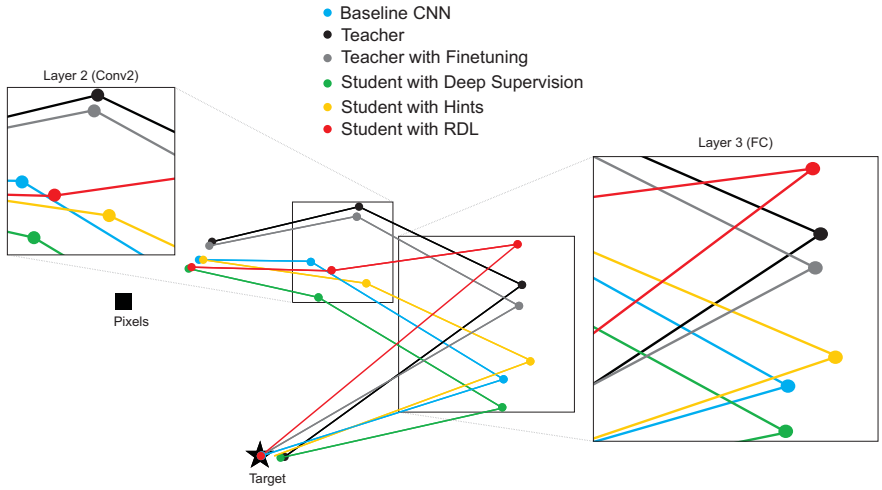


Fig. 2.4 RDL pulls internal representations towards the representations of the teacher for MNIST. 2-D multi-dimensional scaling (MDS) visualisation of the distances between the representational distance matrices (RDMs) for selected layers of the MNIST trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the correlation distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot.

Fig. 2.5 Train and test errors of the CIFAR-100 trained NiNs throughout training as the tested convolutional neural.

Table 2.4 The "Network in Network" (NiN) architecture with batch-normalisation (BN) (Ioffe and Szegedy, 2015) used for CIFAR-100.

Layer	Kernel Size	# Features	Stride	Non-linearity	Other
Conv-1	5x5	192	1	ReLU	BN
MLPConv-1-1	1x1	160	1	ReLU	BN
MLPConv-1-2	1x1	96	1	ReLU	BN
MaxPool	3x3	96	2	Max	-
Conv-2	5x5	192	1	ReLU	BN, Dropout ($p = 0.5$)
MLPConv-2-1	1x1	192	1	ReLU	BN
MLPConv-2-2	1x1	192	1	ReLU	BN
AveragePool-1	3x3	192	2	-	-
Conv-3	5x5	192	1	ReLU	BN, Dropout ($p = 0.5$)
MLPConv-3-1	1x1	192	1	ReLU	BN
MLPConv-3-2	1x1	100	1	ReLU	BN
AveragePool-2	8x8	100	-	-	-

horizontal flips of the images were performed and the learning rate was halved every 25 epochs.

To evaluate using RDL with a more complex network, we used a "Network in Network" (NiN) architecture (Lin et al., 2013), which use MLPConv layers, convolutional layers that use multi-layered perception (MLP) filters instead of linear filters (Table 2.4). The CIFAR-10 trained teacher network had the same architecture as the baseline CIFAR-100 NiN (Table 2.4) except with a 10-class output layer and had a testing error of 8.0%. The DSN had linear auxiliary softmax classifiers after the first and second pooling layers and α was decayed as proposed in Lee et al. (2014). The fine-tuning network's weights were initialised using those of the CIFAR-10 teacher network and a linear readout was added. The RDL network had the same architecture as the baseline CIFAR-100 network with randomly initialised weights and the addition of auxiliary error functions that used the RDMs from the CIFAR-10 teacher. For RDL, an additional linear readout was added after the last MLPConv layer since RDL does not specify that each neuron in a representation corresponds to an output class. For RDL, 2.5% (406) of the image pairs per mini-batch of 128 images were used to calculate the RDL auxiliary errors.

As in the previous experiment, the performances of the networks (Figure 2.5 and Table 2.2) were statistically compared using the McNemar test. The results are shown in Table 2.5. The networks that were trained with fine-tuning, deep supervision, 'hints', and RDL all significantly improved upon the baseline NiN. These results show that learning from RDMs can extract meaningful information from a teacher network, which led to improved

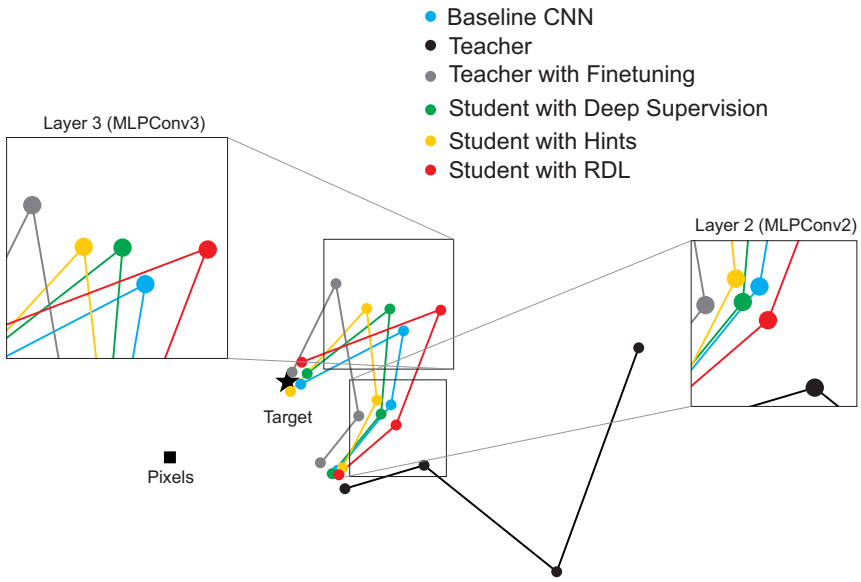


Fig. 2.6 RDL pulls internal representations towards the representations of the teacher for CIFAR 100. 2-D multi-dimensional scaling (MDS) visualisation of the distances between the representational distance matrices (RDMs) for selected layers of the CIFAR-100 trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the normalised Euclidean distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot.

Table 2.5 The McNemar exact test p-values for the tested "Network in Network" (NiN) models trained on CIFAR-100. Arrows indicate a significant difference ($p < 0.05$,uncorr.) and point to the better model.

	Baseline	Fine-tuning	Deep Supervision	Hints	RDL
Baseline	—	0.00 ↑	0.00 ↑	0.00 ↑	0.00 ↑
Fine-tuning	0.00 ←	—	0.86	0.70	0.12
Deep Supervision	0.00 ←	0.86	—	0.86	0.08
Hints	0.00 ←	0.70	0.86	—	0.05
RDL	0.00 ←	0.12	0.08	0.05	—

classification performance. However, these methods affected the representational spaces learned by the student differently.

To investigate the relationships between the representations of the different NiN models, we calculated the correlation between each pair of RDMs and use MDS to create a 2-D plot showing the relative position in representational space of the transformations learned by the various trained networks (Figure 2.6). The MDS plots shows that: (1) RDMs of the network initialised with the weights of the teacher and then fine-tuned were not close to the teacher's RDMs, likely due to the large class increase from CIFAR-10 to CIFAR-100 and the lack of an active constraint, (2) deep supervision pulled the RDMs of the internal representations of the student towards those of the target, (3) RDL pulled the RDMs of the student towards the RDMs of the teacher, and (4) despite learning a series of transformations that do not map directly to the target, the teacher contained useful information to the students' task. This demonstrates the ability of RDL to incorporate both the representational information from the teacher as well as from the classification task.

2.4 Discussion

In this chapter, we proposed RDL, a technique for transferring knowledge from a teacher model to a student DNN. The representational space of the student is pulled towards that of a teacher model during training using stochastic gradient descent. This was performed by minimizing the difference between the pairwise distances between representations of two models at selected layers using auxiliary error functions. Training with RDL was shown to improve classification performance by extracting knowledge from another model trained on a similar task, while allowing architectural differences between the student and teacher. This suggests that RDL can transfer the relationships between class examples learned by the teacher. This information is not present when only constraining internal layers using

class labels, as done in the deeply supervised method, since the target vectors for each class are orthogonal. In particular, RDL allows a student network to learn similar sequential transformations to those learned by a teacher network. This could be of potential use in learning transformations similar to those performed in the human visual ventral stream. Such a model might be able to generate brain-like RDMs for novel stimuli. In the future, we plan to train such a model by constraining large DNNs using fMRI-based RDMs from the human visual ventral stream. By learning from brain-activity patterns, RDL has the potential to help build more realistic models of computations in biological brains.

Chapter 3

Adapting deep neural networks by using stochasticity to robustly represent uncertainty

This chapter is based on a manuscript by McClure and Kriegeskorte (2017).

Summary

As deep neural networks (DNNs) are applied to increasingly challenging problems, they will need to be able to represent their own uncertainty. Modelling uncertainty is one of the key features of Bayesian methods. Using Bernoulli dropout with sampling at prediction time has recently been proposed as an efficient and well performing variational inference method for DNNs. However, sampling from other multiplicative noise based variational distributions has not been investigated in depth. We evaluated Bayesian DNNs trained with Bernoulli or Gaussian multiplicative masking of either the units (dropout) or the weights (dropconnect). We tested the calibration of the probabilistic predictions of Bayesian convolutional neural networks (CNNs) on MNIST and CIFAR-10. Sampling at prediction time increased the calibration of the DNNs' probabalistic predictions. Sampling weights, whether Gaussian or Bernoulli, led to more robust representation of uncertainty compared to sampling of units. However, using either Gaussian or Bernoulli dropout led to increased test set classification accuracy. Based on these findings we used both Bernoulli dropout and Gaussian dropconnect concurrently, which we show approximates the use of a spike-and-slab variational distribution without increasing the number of learned parameters. We found that spike-and-slab sampling

had higher test set performance than Gaussian dropconnect and more robustly represented its uncertainty compared to Bernoulli dropout.

3.1 Introduction

Deep neural networks (DNNs), particularly convolutional neural networks (CNNs), have recently been used to solve complex perceptual and decision tasks Krizhevsky et al. (2012); Mnih et al. (2015); Silver et al. (2016). While these models take into account aleatoric uncertainty via their softmax output (i.e. the uncertainty present in the training data), they do not take into account epistemic uncertainty (i.e. parameter uncertainty) (Kendall and Gal, 2017). Bayesian DNNs attempt to learn a distribution over their parameters, thereby allowing for the computation of the uncertainty of their outputs given the parameters. However, ideal Bayesian methods do not scale well due to the difficulty in computing the posterior of a network's parameters.

As a result, several approximate Bayesian methods have been proposed for DNNs. Using the Laplace approximation was proposed by MacKay (1992). Using Markov chain Monte Carlo (MCMC) has been suggested to estimate the posterior of the networks weights given the training data (Neal, 2012; Welling and Teh, 2011). Using expectation propagation has also been proposed (Hernández-Lobato and Adams, 2015; Jylänki et al., 2014). However, these methods can be difficult to implement for the very large CNNs commonly used for object recognition. Variational inference methods have also been applied to NNs. These have the advantage of making Bayesian NNs more tractable(Barber and Bishop, 1998; Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993). Gal and Ghahramani (2016) and Kingma et al. (2015) recently recently developed variational Bayesian DNN based on Bernoulli and Gaussian dropout (Srivastava et al., 2014), respectively. Independent weight sampling with additive Gaussian noise has been investigated (Barber and Bishop, 1998; Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993). However, independent sampling weights using multiplicative Bernoulli noise, i.e. dropconnect (Wan et al., 2013), or independent sampling weights multiplicative Gaussian noise has not been thoroughly evaluated.

In addition to Bernoulli and Gaussian distributions, spike-and-slab distributions, a combination of the two, have been investigated, particularly for linear models (George and McCulloch, 1997; Ishwaran and Rao, 2005; Madigan and Raftery, 1994; Mitchell and Beauchamp, 1988). Interestingly, Bernoulli dropout and dropconnect can be seen as approximations to spike-and-slab distributions for units and weights, respectively (Gal, 2016; Louizos, 2015). Spike-and-slab variational distributions have been implemented using Bernoulli dropout with additive weight noise sampled from a Gaussian with a learned standard deviation (Louizos,

2015). This approach more than doubled the number of learned parameters, since the mean and the standard deviation of each weight as well as the dropout rate for each unit were learned. However, this method did not consistently outperform standard neural networks. Gal (2016) also discussed motivations for spike-and-slab variational distributions, but did not suggest a practical implementation.

We evaluated the performance Bayesian CNNs with different variational distributions on MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009). We also investigate how adding Gaussian image noise with varying standard deviations to the test set affected each network's learned uncertainty. We did this to test how networks responded to inputs not drawn from the data distribution used to create the training and test sets. We also propose an approximation of the spike-and-slab variational inference based on Bernoulli dropout and Gaussian dropconnect, which combines the advantages of Gaussian dropconnect and Bernoulli dropout sampling leading to better uncertainty estimates and good test set generalisation without increasing the number of learned parameters.

3.2 Methods

3.2.1 Bayesian Deep Neural Networks

DNNs are commonly trained by finding the maximum a posteriori (MAP) weights given the training data (D_{train}) and a prior over the weight matrix W, $p(W)$. However, ideal Bayesian learning would involve computing the full posterior. This can be intractable due to both the difficulty in calculating $p(D_{train})$ and in calculating the joint distribution of a large number of parameters. Instead, $p(W|D_{train})$ can be approximated using a variational distribution $q(W)$. This distribution is constructed to allow for easy generation of samples. The objective of variational inference is to optimise the variational parameters V so that the Kullblack-Leiber (KL) divergence between $q_V(W)$ and $p(W|D_{train})$ is minimised (Barber and Bishop, 1998; Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993):

$$V^* = \underset{V}{\operatorname{argmin}} KL[q_V(W)||p(W)] - \int q_V(W) \log p(D_{train}|W) dW \tag{3.1}$$

Using Monte Carlo (MC) methods to estimate $E_{q_V(w)}[\log p(D_{train}|W)]$, using weight samples $\hat{W}^k \sim q_V(W)$, results in the following loss function:

$$\mathscr{L} := KL(q_V(W)||p(W)) - \frac{1}{n}\sum_{k=1}^{n} \log p(D_{train}|\hat{W}^k) \tag{3.2}$$

MC sampling can also be used to estimate the probability of test data:

$$p(D_{test}) \approx \frac{1}{n}\sum_{k=1}^{n} p(D_{test}|\hat{W}^k) \tag{3.3}$$

3.2.2 Variational Distributions

The number and continuous nature of the parameters in DNNs makes sampling from the entire distribution of possible weight matrices computationally challenging. However, variational distributions can make sampling easier. In deep learning, the most common sampling method is using multiplicative noise masks drawn from some distribution. Several of these methods can be formulated as variational distributions where weights are sampled by element-wise multiplication of the variational parameters V, the $n \times n$ connection matrix with an element for each connection between the n units in the network, by a mask $\hat{M}$, which is sampled from some probability distribution:

$$\hat{W} = V \circ \hat{M} \quad where \quad \hat{M} \sim p(M) \tag{3.4}$$

From this perspective, the difference between dropout and dropconnect, as well as Bernoulli and Gaussian methods, is simply the probability distribution used to generate the mask sample (Figure 3.1).

Bernoulli Dropconnect & Dropout

In Bernoulli dropconnect, each element of the mask is sampled independently, so $\hat{m}_{i,j} \sim Bernoulli(1-p)$ where p is the probability of dropping a connection. In Bernoulli dropout, however, the weights are not sampled independently. Instead, one Bernoulli variable is sampled for each row of the weight matrix, so $\hat{m}_{i,*} \sim Bernoulli(1-p)$ where p is the probability of dropping a unit.

Gaussian Dropconnect & Dropout

In Gaussian dropconnect and dropout, $\hat{w}_{i,j}$ is sampled from a Gaussian distribution centred at variational parameter $v_{i,j}$. This is accomplished by sampling the multiplicative mask using Gaussian distributions with a mean of 1 and a variance of $\sigma_{dc}^2 = p/(1-p)$, which matches the mean and variance of Bernoulli dropout when training time scaling is used Srivastava et al. (2014). In Gaussian dropconnect, each element of the mask is sampled independently, which results in $\hat{m}_{i,j} \sim \mathcal{N}(1, \sigma_{dc}^2)$. In Gaussian dropout, each element in a row has the same random variable, so $\hat{m}_{i,*} \sim \mathcal{N}(1, \sigma_{dc}^2)$. It can be shown that using Gaussian dropconnect or dropout with L2-regularisation leads to optimizing a stochastic lower-bound of the variational

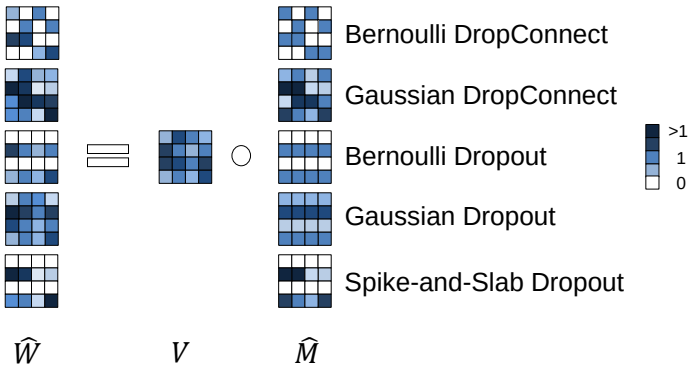


Fig. 3.1 Sampling either weights or units from different variational distributions can be constructed as multiplicative noise (of various statistical structure) imposed on the weight matrix.

objective function when using a mean zero Gaussian prior(See Section A.3 in the Appendix). This differs from the methods proposed by Kingma et al. (2015), which used an improper log uniform prior.

Spike-and-Slab Dropout

A spike-and-slab distribution is the normalised linear combination of a "spike" of probability mass at zero and a "slab" consisting of a Gaussian distribution. This spike-and-slab returns a 0 with probability p_{spike} or a random sample from a Gaussian distribution $\mathcal{N}(\mu_{slab}, \sigma^2_{slab})$ with probability $1 - p_{spike}$. We propose concurrently using Bernoulli dropout and Gaussian dropconnect to approximate the use of a spike-and-slab variational distribution and spike-and-slab prior by optimizing a lower-bound of the variational objective function (See Supplementary Material). In this formulation, $m_{i,j} \sim b_{i,*}\mathcal{N}(1, \sigma^2_{dc})$, where $b_{i,*} \sim Bern(1 - p_{do})$ for each mask row and $\sigma^2_{dc} = p_{dc}/(1 - p_{dc})$. As for Bernoulli dropout, each row of the mask M is multiplied by 0 with probability p_{do}, otherwise each element in that row is multiplied by a value independently sampled from a Gaussian distribution as in Gaussian dropconnect. During non-sampling inference, spike-and-slab dropout uses the mean weight values and, per Bernoulli dropout, multiplies unit outputs by $1 - p_{do}$.

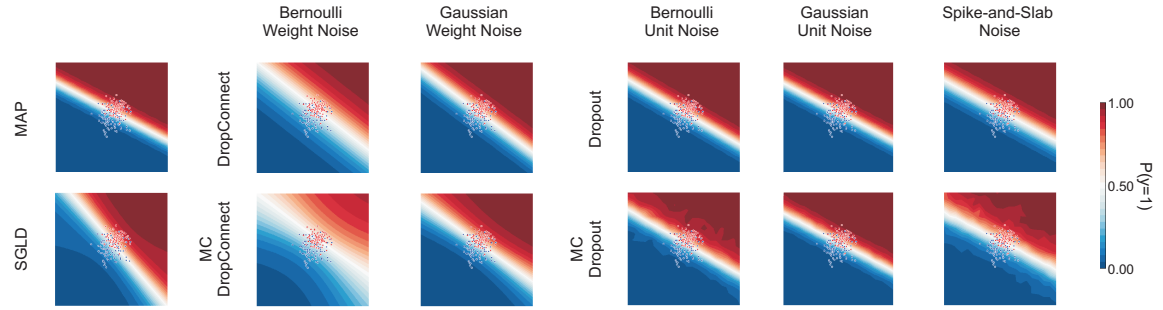


Fig. 3.2 Independent weight (i.e. dropconnect-based) sampling during training and testing makes models much more uncertain further away from the training data. The probabalistic logistic regression decision boundaries of a linear network for: (top row) the MAP network and the dropconnect and dropout methods that only sample during training and (bottom row) the stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011) network and the Monte Carlo (MC) dropconnect and dropout methods that sample during training and testing.

3.3 Experiments

3.3.1 Logistic Regression

In order to visualise the effects of each variational distribution, we trained linear networks with five hidden units to classify data drawn from two 2D multivariate Gaussian distributions. Multiple linear units were used so that Bernoulli dropout would not dropout the only unit in the network. For the dropout methods, unit sampling was performed on the linear hidden layer. For the dropconnect methods, every weight was sampled. Dropout and dropconnect probabilities of $p = 0.4$ were used for each of these networks, except for the spike-and-slab dropconnect probability which was 0.2. In Figure 3.2, we show the decision boundaries learned by the various networks. Higher variability in the decision boundaries corresponds to higher uncertainty. Ideally, the networks would predict with higher uncertainty as points become further away from the training data, as demonstrated by the the stochastic gradient Langevin dynamics (SGLD) network (Welling and Teh, 2011). All of the MC sampling methods predict with higher uncertainty as points become further away from the training data, but the dropconnect and spike-and-slab methods are much better than the methods that only use dropout.

Table 3.1 MNIST and CIFAR-10 mean and standard deviation of test errors for the trained convolutional neural networks (CNNs) with and without Monte-Carlo (MC) across 5 runs, each MC run using 10 samples.

	MNIST		**CIFAR-10**	
Method	**Mean Error (%)**	**Error Std. Dev.**	**Mean Error (%)**	**Error Std. Dev.**
MAP	0.76	-	25.86	-
Bernoulli DropConnect	0.56	-	16.46	-
MC Bernoulli DropConnect	0.56	0.03	16.59	0.11
Gaussian DropConnect	0.56	-	16.78	-
MC Gaussian DropConnect	0.58	0.02	16.65	0.11
Bernoulli Dropout	0.49	-	11.23	-
MC Bernoulli Dropout	0.48	0.03	9.95	0.08
Gaussian Dropout	0.42	-	9.07	-
MC Gaussian Dropout	0.36	0.04	9.00	0.10
Spike-and-Slab Dropout	0.48	–	10.64	–
MC Spike-and-Slab Dropout	0.46	0.01	10.05	0.06

Table 3.2 The convolutional neural network (CNN) architecture used for MNIST.

Layer	**Kernel Size**	**# Features**	**Stride**	**Non-linearity**
Conv-1	5x5	32	1	ReLU
MaxPool-1	2x2	32	2	Max
Conv-2	5x5	64	1	ReLU
MaxPool-2	2x2	64	2	Max
FC	1500	500	-	ReLU
FC	500	10	-	Softmax

Table 3.3 The convolutional neural network (CNN) architecture used for CIFAR-10.

Layer	**Kernel Size**	**# Features**	**Stride**	**Non-linearity**
Conv-1	3x3	64	1	ReLU
Conv-2	3x3	64	1	ReLU
MaxPool-1	2x2	64	2	Max
Conv-3	3x3	128	1	ReLU
Conv-4	3x3	128	1	ReLU
MaxPool-2	2x2	128	2	Max
Conv-5	3x3	256	1	ReLU
Conv-6	3x3	256	1	ReLU
Conv-7	3x3	256	1	ReLU
MaxPool-3	2x2	256	2	Max
Conv-8	3x3	512	1	ReLU
Conv-9	3x3	512	1	ReLU
Conv-10	3x3	512	1	ReLU
MaxPool-4	2x2	512	2	Max
Conv-11	3x3	512	1	ReLU
Conv-12	3x3	512	1	ReLU
Conv-13	3x3	512	1	ReLU
MaxPool-5	2x2	512	2	Max
FC	512	512	-	ReLU
FC	512	10	-	Softmax

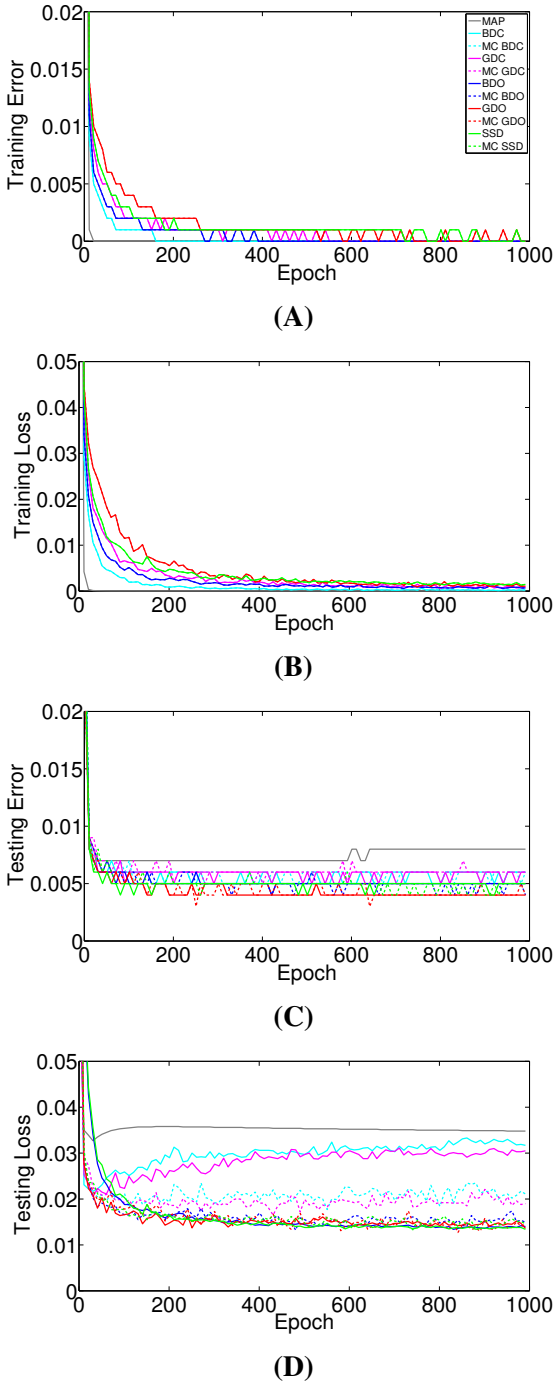


Fig. 3.3 Sampling during training and testing prevents overfitting on MNIST. The MNIST (a) training error, (b) training loss, (c) test error, and (d) test loss for for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples.

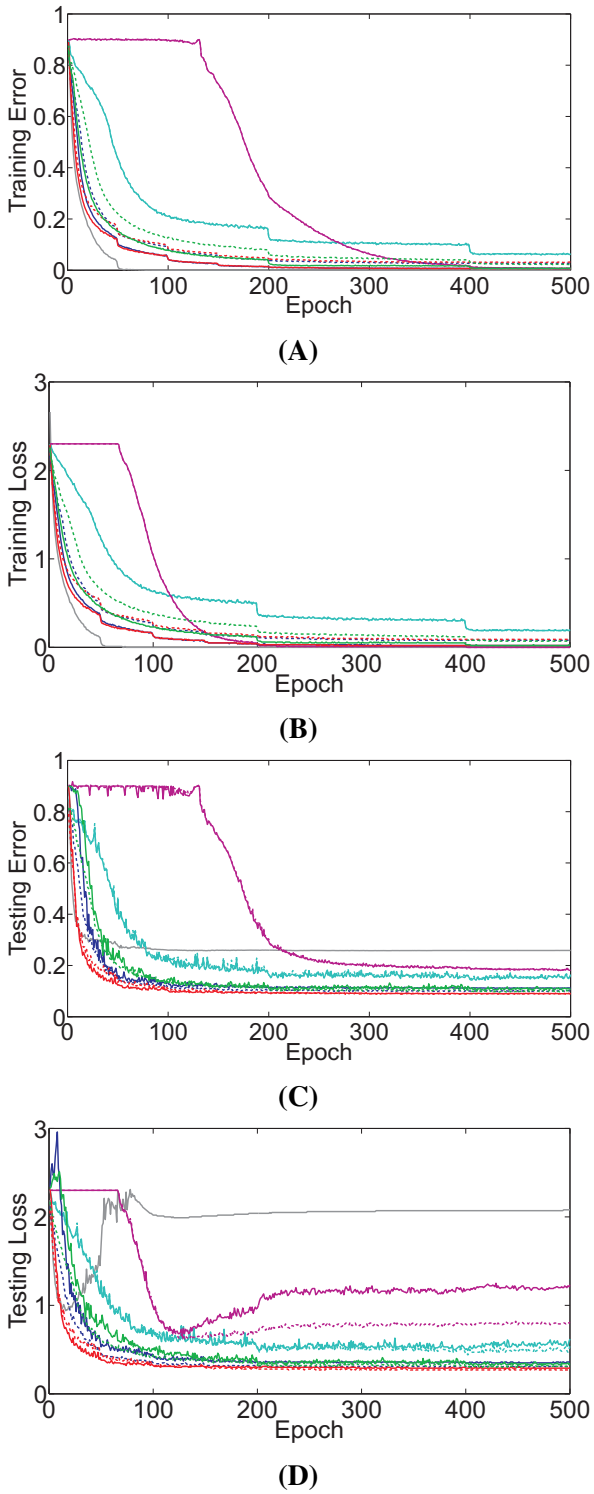


Fig. 3.4 Sampling during training and testing prevents overfitting on CIFAR-10. The CIFAR-10 (A) training error, (B) training loss, (C) test error, and (D) test loss for for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples.

3.3.2 Convolutional Neural Networks

We trained CNNs on MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009). For each dataset, a 10,000 image subset of the training set was used for validation, which was used to set the hyperparameters. For MNIST, each CNN had two convolutional layers followed by a fully connected layer and a softmax layer(3.2). For CIFAR-10, each CNN had 13 convolutional layers followed by a fully connected layer and a softmax layer (3.3). For the dropout networks, dropout was used after each convolutional and fully-connected layer, but before the non-linearity. For the dropconnect networks, all weights were sampled. All ps were treated as network-wide hyperparameters. No data augmentation was used for MNIST. Random horizontal flipping was used during CIFAR-10 training. We evaluated the trained CNNs using the original testing sets and using the testing images with added random Gaussian noise of increasing variance in order to test each network's uncertainty for the regions of input space not seen in the training set.

Sampling during training is often used as a form of regularisation. To evaluate how well the different methods achieved this, we plotted the training and testing classification error and cross-entropy loss for MNIST (Figure 3.3) and CIFAR-10 (Figure 3.4). We found that the dropout-based sampling during training consistently prevented overfitting, but sampling during testing did not consistently lead to better testing classification error or cross-entropy loss. However, only using dropconnect-based methods without test-time sampling often led to increased testing loss due to overfitting. Surprisingly, this increase in testing loss was not accompanied by an increase in classification error.

While the dropout-based methods were the most accurate on the test-set (Table 3.1), as image noise was added they became increasingly worse compared to the dropconnect-based networks (Figures 3.5.a and 3.6.A). Sampling only consistently improved the test accuracy for Bernoulli and spike-and-slab dropout. However, sampling did consistently improve the calibration of the networks as the image noise was increased (Figures 3.5.B and 3.6.B). For a given accuracy across each set of noisy test images, sampling also generally led to better calibration (Figures 3.5.C, 3.6.C, and 3.7). Gaussian dropout led to the highest test set accuracy, but it also led to reduced robustness to noise. While slightly less accurate on the test set, Bernoulli dropout and spike-and-slab dropout were much more robust.

Seemingly contradictory results have been reported in the literature regarding CIFAR-10 and MC Bernoulli dropout. Gal and Ghahramani (2015) found that standard Bernoulli dropout methods led to relatively inaccurate networks when dropout was used at every layer in a CNN, whereas MC sampling increased the accuracy of these networks. However, Srivastava et al. (2014) found that using dropout at every layer led to increased generalisation performance even without sampling at prediction time. In our CIFAR-10 experiments, but

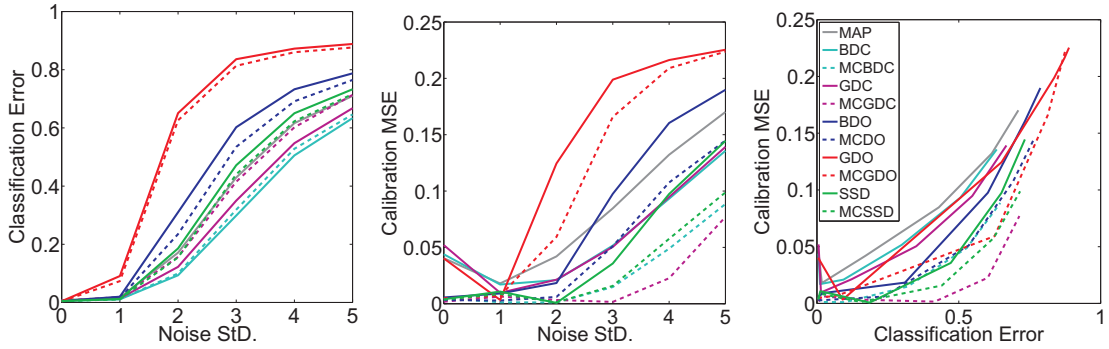


Fig. 3.5 Sampling during training and testing improves CNN calibration on MNIST in the presecence of input noise. The MNIST (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 1, 2, 3, 4, and 5, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples.

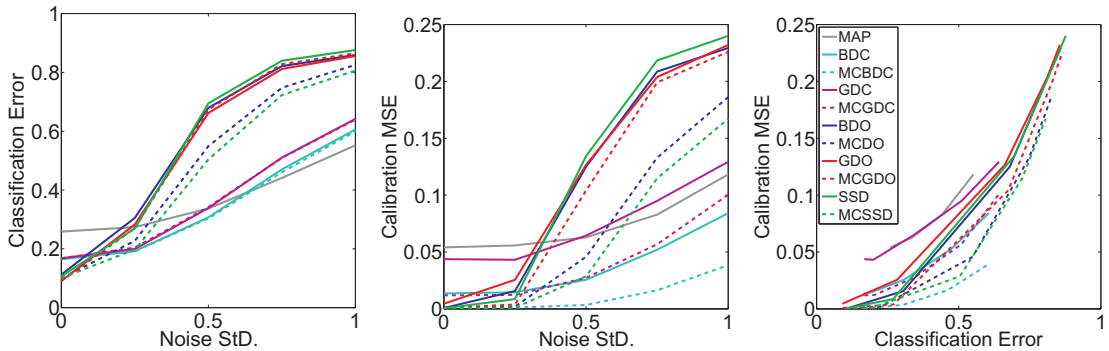


Fig. 3.6 Sampling during training and testing improves CNN calibration on CIFAR-10 in the presecence of input noise. The CIFAR-10 (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 0.25, 0.5, 0.75, and 1, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples.

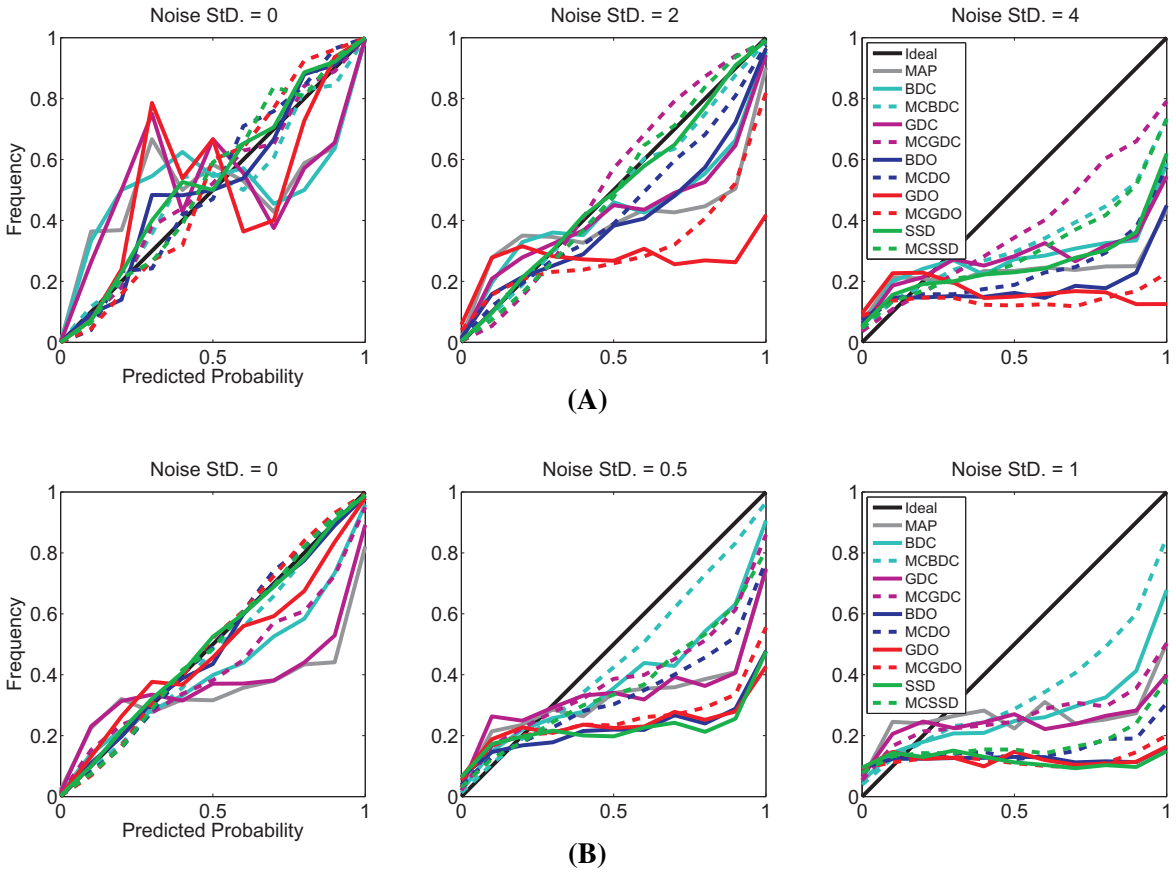


Fig. 3.7 Sampling during training and testing improves CNN calibration curves. The $x = y$ line (Ideal) and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD. for the (a) MNIST or (b) CIFAR-10 trained Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) networks with and without MC sampling using 10 samples.

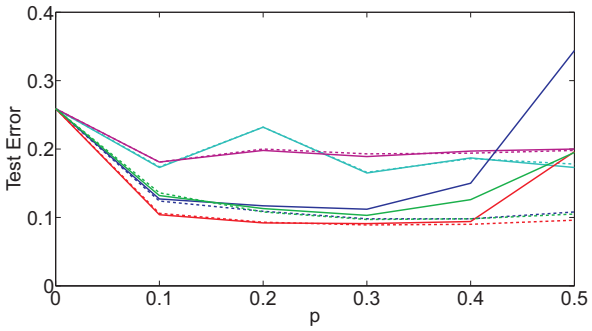


Fig. 3.8 Sampling during training and testing (1) improves the robustness of CNNs to dropout hyperparmeter choice and (2) allows for increased dropout regularisation. The classification error for the CIFAR-10 test set using different p hyperparameter values. For spike-and-slab dropout, p_{do} was varied, while p_{dc} was fixed.

not our MNIST experiments, we have found that using sampling at prediction time makes networks more robust to high variance dropout. Using lower variance dropout results in standard and MC methods having similar accuracies, while using higher variance distributions results in MC inference outperforming standard methods (Figure 3.8). (See the Appendix for more results when using $p = 0.5$.) These results indicate that Bernoulli or Gaussian dropout with MC sampling are less dependent on the exact value of p and can allow higher levels of dropout regularisation to be used.

3.4 Discussion

L2 regularisation and Bernoulli dropout are widely used for regularisation and routinely lead to increased testing accuracy. However, the predictive uncertainty learned when using these methods does not generalise well. However, performing approximate Bayesian inference via sampling during training and testing allowed CNNs to better model their uncertainty. Dropconnect-based CNNs performed worse on the unmodified test set, but were much more robust to deviations from the training distribution. On the other hand, dropout-based networks, particularly MC Gaussian dropout, performed well on the unmodified test set, but were not as robust. Using sampling and combining Bernoulli dropout and Gaussian dropconnect to approximate the use of spike-and-slab variational distributions led to a CNN that performed better near the test set than the dropconnect methods and more robustly represented its uncertainty compared to the dropout methods.

Chapter 4

Adapting Bayesian deep neural networks to model human visual perception

Summary

Dealing with sensory uncertainty is necessary for humans to operate in the world. Often, multiple interpretations of an event are possible given the sensory evidence, even if one interpretation is most likely. The exact neurobiological mechanism used for representing uncertainty is unknown, but there is increasing evidence that the human brain could use its inherent stochasticity to code for uncertainty. However, the prominent convolutional neural networks (CNNs) currently used to model human vision implement deterministic mappings from input to output. We seek to use stochasticity to improve CNNs as both computer vision models and models of human visual perception. In this chapter, we used Gaussian unit noise and sampling to implement Bayesian CNNs. We then tested how using sampling during learning and inference affects a CNN's accuracy, its ability to model its own uncertainty, and its prediction of human confidence scores. We found that sampling during both training and testing improved a CNN's accuracy and ability to represent its own uncertainty for large-scale object recognition. We also found that sampling during both training and testing improved the ability of linear classifiers trained on internal CNN representations to predict human confidence scores for natural image classification. These results add to the evidence that Bayesian models predict key aspects of human object categorisation behaviour and that sampling in biological neural networks could be a means of representing uncertainty for visual perception in the human brain.

4.1 Introduction

Humans must deal with sensory uncertainty to operate in the world (Vilares and Kording, 2011). Often, multiple interpretations are possible given some sensory input, even if one interpretation is most likely. This requires neural representations to code a distribution of interpretations. It has been hypothesised that humans and animals perform near optimal inference by integrating this probabilistically represented information using Bayesian decision theory (Griffiths et al., 2008; Knill and Pouget, 2004). For vision, it has been shown that humans model their own objective uncertainty (Barthelmé and Mamassian, 2009). Several probabilistic neural coding frameworks have been suggested, such as probabilistic population codes (PPC) (Ma et al., 2006) and neural sampling (Fiser et al., 2010b). For visual perception in particular, there is evidence for a sampling-based probabilistic representation (Berkes et al., 2011; Goris et al., 2014; Moreno-Bote et al., 2011; Orbán et al., 2016b).

Despite this, current neural network models of high level vision are deterministic and do not model the uncertainty of their learned representations. Specifically, deterministic deep convolutional neural networks (CNNs) have become prominent models in computational neuroscience for visual perception (Güçlü and van Gerven, 2015; Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014). These CNNs either use the maximum likelihood estimate (MLE) or the maximum a posteriori (MAP) solution for the parameters and do not model a distribution of parameters or representations. As shown in the previous chapter, utilizing stochasticity can improve a CNN's accuracy and its ability to represent its own uncertainty. This is important in building computer vision systems, but also in building better computational models of the human brain.

In the computational neuroscience literature, two main types of sampling-based models have been proposed: (1) latent variable models (Aitchison and Lengyel, 2016; Goris et al., 2014; Hoyer and Hyvärinen, 2003; Orbán et al., 2016b) and (2) synaptic sampling (Aitchison and Latham, 2015; Kappel et al., 2015). One major difference between the methods is that latent variable models treat the neural activity as a random variable, while synaptic sampling treats each weight as a random variable. Both of these methods can be implemented in CNNs. This is similar to using unit noise and weight noise, respectively, in Bayesian NNs (Aitchison and Latham, 2015; Rezende et al., 2014). The input can also be viewed as a random variable, with input noise leading to regularisation (Bishop, 1995; Holmstrom and Koistinen, 1992). However, we focus on internal network stochasticity.

In this chapter, we approximate a variational Bayesian CNN using MC Gaussian dropout, which was discussed in the Chapter 3. We also show that this method is highly similar to a deep latent variable CNN. We investigate how much using sampling affects a CNN's

classification accuracy, predicted uncertainty, and the ability to predict human confidence scores for natural image classification.

4.2 Methods

4.2.1 Approximating Bayesian neural networks using Monte Carlo Gaussian dropout

In machine learning, noise has been traditionally injected into neural networks as a form of regularisation during training followed by using the layerwise expectation during testing, as done for AlexNet and VGG-16 (Krizhevsky et al., 2012; Simonyan et al., 2013; Srivastava et al., 2014). However, sampling both during training and testing in Bayesian CNNs can lead to better representation of uncertainty, as discussed in the previous chapter. Monte Carlo (MC) sampling during training using Eq. 4.1 and testing using multiplicative Gaussian unit noise with a mean of 1 and a variance of $\alpha = (1-p)/p$, where p is the dropout probability, approximates Bayesian inference in neural networks. For an input image x and an image label y, the matrix V of weight means $v_{i,j}$, and weight noise $\sigma_{i,j} = \sqrt{\alpha} v_{i,j}$, the optimization objective for the Bayesian CNN is given by 4.1. (The derivation of this method is shown in Appendix A.1.)

$$\max_V \frac{1}{n} \sum_{k=1}^{n} \log p(y|x,V,\varepsilon^k) - \frac{\lambda}{2}||vec(V)||_2^2 \text{ where } \varepsilon_i^k \sim \mathcal{N}(0,1) \tag{4.1}$$

4.2.2 Relationship between Monte Carlo Gaussian dropout and deep latent Gaussian models

In a deep latent Gaussian model (DLGM) (Rezende et al., 2014), the goal is to learn a hierarchy of latent Gaussian variables. In the context of natural image recognition, these latent variable could define the relationship between images and class labels. Rezende et al. (2014) construct DLGMs using additive variable noise, we however use multiplicative Gaussian variable noise (Kingma et al., 2015; Srivastava et al., 2014). For input variable x, output variable y, the matrix V_m of layer m weight means $v_{i,j}$, unit noise $\varepsilon_i \sim \mathcal{N}(0,1)$, and non-linearity h, each unit, $h(z_{m,i})$, is defined using the latent variables:

$$z_{m,i} = \sum_{j=1}^{n_{m-1}} h(z_{m-1,j}) v_{i,j} + \varepsilon_i \sqrt{\alpha} \sum_{j=1}^{n_{m-1}} h(z_{m-1,j}) v_{i,j} \tag{4.2}$$

The weight means, V, are learned by optimising:

$$\max_V \log p(y|x) = \max_V \log \Big(\int p(y|x,z)p(z|x)dz \Big) \tag{4.3}$$

Using the Gaussian "repramaterization trick" (Appendix A.2) for each unit, $z = f(x,V,\varepsilon)$, so this objective can be rewritten as:

$$\max_V \log p(y|x) = \max_V \log \Big(\int p(y|x,z)p(\varepsilon)d\varepsilon \Big) \text{ where } \varepsilon_i \sim \mathcal{N}(0,1) \tag{4.4}$$

Approximating the integral with MC sampling leads to:

$$\max_V \log p(y|x) \approx \max_V \log \Big(\frac{1}{n} \sum_{k=1}^{n} p(y|x,z^k) \Big) \text{ where } \varepsilon_i^k \sim \mathcal{N}(0,1) \tag{4.5}$$

If unit-based multiplicative Gaussian noise and $n = 1$ is used for training, which is done for MC Gaussian dropout and is neurobiologically plausible (Goris et al., 2014), the loss of the DLGM (Eq. 4.5) is equivalent to the loss of MC Gaussian dropout (Eq. 4.1) without L2 regularisation on the means of the network weights, since the latent variables are deterministic functions of the inputs, weights, and random noise (i.e. $z = f(x,V,\varepsilon)$). Therefore, the trained CNN can be interpreted as either a synaptic sampling or a latent variable model.

4.2.3 Architecture and datasets

Large-scale object recognition

We tested three CNNs: (1) a baseline CNN with no sampling, (2) a CNN with Gaussian unit noise before each ReLU non-linearity only during learning, and (3) a CNN with Gaussian unit noise during learning and inference. For all CNNs with sampling during inference, 10 MC samples were used. Each CNN had 8 layers, 7 convolutional and a softmax readout layer, which transforms an activation pattern into a probability distribution (Table 4.1). CNNs with this architecture were trained on both ImageNet (Russakovsky et al., 2015) and Eco-set (Mehrer et al., 2017) using stochastic gradient descent with momentum and weight normalisation (Salimans and Kingma, 2016). ImageNet is a 1,000 class object recognition problem with 1.2 million training images and 150,000 validation images. However, the ImageNet categories are biased towards certain entry-level categories, such as birds and dogs. As a computer vision task, this is reasonable, but from a human visual neuroscience perspective models should be trained on the image distribution seen by humans. The Eco-set project seeks to create a datasets that more closely matches the human visual diet. This image set is a 578 class object recognition with 569,413 training images, 28,900 validation images, and 28,900 testing images. We evaluated how much using MC sampling during

Table 4.1 The convolutional neural network (CNN) architecture used for ImageNet and Ecoset.

Layer	Kernel Size	# Features	Stride	Non-linearity
Conv-1	3x3	64	1	ReLU
MaxPool-1	2x2	64	2	Max
Conv-2	3x3	128	1	ReLU
MaxPool-2	2x2	128	2	Max
Conv-3	3x3	256	1	ReLU
MaxPool-3	2x2	256	2	Max
Conv-4	3x3	512	1	ReLU
MaxPool-4	2x2	512	2	Max
Conv-5	3x3	512	1	ReLU
MaxPool-5	2x2	512	2	Max
Conv-6	3x3	1024	1	ReLU
MaxPool-6	2x2	1024	2	Max
Conv-7	3x3	1024	1	ReLU
AveragePool-1	3x3	1024	0	Max
FC	1024	$n_{classes}$	-	Softmax

testing, which approximates the expected prediction for an input, affected ImageNet and Eco-set trained CNNs. For all of the CNNs with MC sampling, $p = 0.2$ was found to be the best Gaussian dropout parameter value using validation testing.

Modelling human confidence using decision boundaries

The translation from internal representations in the human brain to decisions about object categories has been modelled using linear decision boundaries (Carlson et al., 2014; Ritchie and Carlson, 2016). These models successfully predict human reaction times, a proxy for human confidence, for object recognition. A similar approach can be used to predict human confidence scores from the internal representations of DNNs using the decision scores of linear classifiers (Eberhardt et al., 2016). Eberhardt et al. (2016) created five non-overlapping sets of 300 grey-scaled randomly sampled ImageNet images, each set containing 150 animal and 150 non-animal images. For each image, 50 participants were asked to classify it as animal or non-animal during a fixation task. Eberhardt et al. (2016) computed a human confidence score for each image by considering the fraction of correct animal/non-animal classifications across the 50 participants shown that image. In order to evaluate the ability of sampling to improve prediction of human uncertainties, we trained logistic regression models to classify animal and non-animal images from the internal representations of the

Table 4.2 The accuracies for the ImageNet trained CNN on the ImageNet validation set and the Eco-set trained CNNs on the Eco-set test set. For MC sampling, the mean and standard deviation of the accuracies across 5 MC runs, each computed with 10 MC samples.

CNN	ImageNet Accuracy (%)	Eco-set Accuracy (%)
MAP	52.07	49.09
MC Training	55.43	53.36
MC Training and Testing	57.04 ± 0.13	55.06 ± 0.09

Eco-set trained CNNs. These logistic regression models were trained using leave-one-out crossvalidation across the five non-overlapping image sets created by Eberhardt et al. (2016). For the MAP CNN, the internal representation for layer m, z_m, for an image was deterministic, leading to the logistic regression optimising:
For the stochastic CNN, the internal representation was stochastic, leading to logistic regression optimising:

$$\max_W \int p(y|x, z_m) p(z_m|x) dz \tag{4.6}$$

This integration is approximated using MC sampling with n samples:

$$\max_W \frac{1}{n} \sum_{k=1}^{n} p(y|x, \tilde{z}_m^k) \text{ where } \tilde{z}_m^k \sim p(z_m|x) \tag{4.7}$$

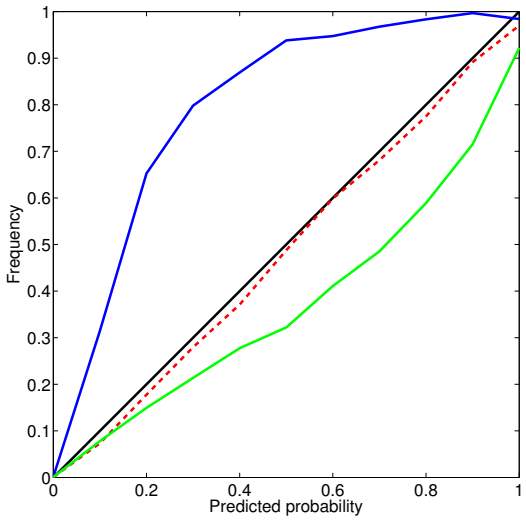
For training, one MC sample was used, as in MC Gaussian dropout. For testing, ten MC samples were used to approximate the predicted probability of class y for input x per:

$$p(y|x) \approx \frac{1}{n} \sum_{k=1}^{n} p(y|x, \tilde{z}_m^k) \text{ where } \tilde{z}_m^k \sim p(z_m|x) \tag{4.8}$$

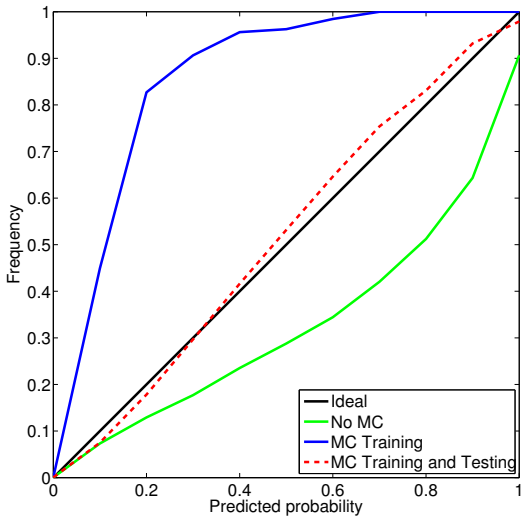
4.3 Results

4.3.1 Sampling improves accuracy for large-scale object recognition

Using random noise in deep neural networks during learning is often used to reduce overfitting and increase generalisation performance (Srivastava et al., 2014). However, as discussed in the previous chapter, sampling during testing can sometimes lead to accuracy improvements. We found that for large-scale object recognition CNN that we trained, MC sampling at test time led to significant accuracy improvements (Table 4.2) for both ImageNet and Eco-set.

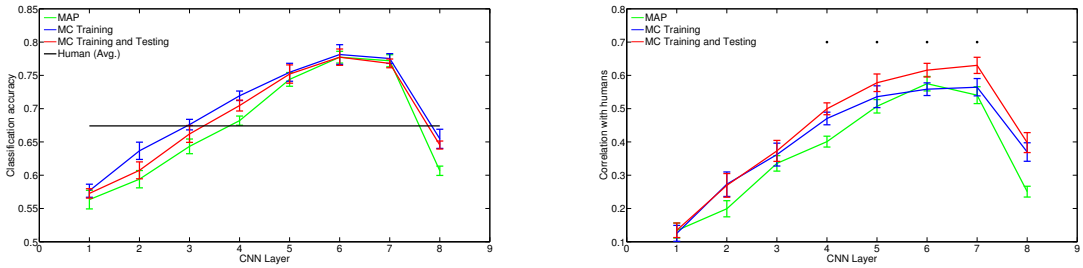


(A) ImageNet



(B) Eco-set

Fig. 4.1 Sampling during training and testing improves CNN calibration for large-scale object recognition. The CNN calibration curves for (A) the ImageNet trained CNN on the ImageNet validation set and (B) the Eco-set trained CNNs on the Eco-set test set.



(A) **(B)**

Fig. 4.2 Sampling during training and testing improves the correlation between CNN-based predicted probabilities and human confidence scores. The mean and standard errors for the (A) accuracies and (B) correlations with human confidence scores for the logistic regression models trained at each CNN. $*$ denotes the "MC Training and Testing" models have significantly higher (p-value<0.05,uncorr.) correlations with human confidence scores than both the "MAP" and "MC Training" models per a paired t-test across the five image sets.

4.3.2 Sampling improves the representation of uncertainty for large-scale object recognition

Humans can accurately estimate their objective uncertainty for visual perception (Barthelmé and Mamassian, 2009). A good computational model of human vision would also need to properly represent its own uncertainty. A model correctly models its own uncertainty (i.e. is well calibrated) if its predicted probabilities closely match the frequency of correctly predicting the true label. To evaluate a network's ability to model its own uncertainty, we calculated the calibration of each of these methods' probabilistic predictions to evaluate the quality of the learned representations. We evaluated how calibrated a prediction was by: (1) Binning test set predictions by predicted probability and then (2) calculating the frequency that predictions in each predicted-probability bin correctly predicted a target label. The larger the difference between these values and the $x = y$ line, the worse the calibration of the model. For both ImageNet and the Eco-set, sampling led to improved calibration of output predictions (Figure 4.1).

4.3.3 Sampling improves the prediction of human confidence for image classification

To evaluate whether sampling improved the prediction of human confidence scores, we calculated the Pearson correlation coefficient between the output probabilities of the logistic regression models trained using the representations of the Eco-set trained CNN layers and

human confidence scores for animal/non-animal classification from ImageNet images. We compared the correlations between the human confidence scores collected by Eberhardt et al. (2016) and the CNN representation-based predicted probabilities of the MAP and the two sampling models by performing pairwise paired t-tests across all layers and the five image sets. MC sampling during both training and testing led to significantly improved correlation between the human confidence scores collected by Eberhardt et al. (2016) and the CNN representation-based predicted probabilities compared to the MAP and MC sampling only during training. (p=1.31e-7 and p=2.23e-5, respectively). As found by Eberhardt et al. (2016), the accuracy and correlation with human confidence scores of the linear classifiers generally increases the deeper the layer, except that they decrease at the softmax layer (Figure 4.2.A). This might have been caused by the reduced generality (Yosinski et al., 2014) and size of the feature space. Using a paired t-test for each layer comparing across image sets showed that MC sampling during training and testing improved prediction of human confidence scores at deep hidden layers (Figure 4.2.B). This may be caused by the fact that sampling only during training relies on the assumption of a linear network when approximating the expectation of the output. This assumption is increasingly violated as we move up the layers of the network, which may explain the widening of the gap between the human confidence prediction accuracies of the two models.

4.4 Discussion

Biological neural networks are highly stochastic. This variance may code for uncertainty in neural representations. In this work, we evaluated the effect of adding stochasticity (in the form of Gaussian unit noise) and sampling-based inference on large scale CNNs. We tested the effects of sampling during learning and during learning and inference on ImageNet (Russakovsky et al., 2015) and Eco-set (Mehrer et al., 2017) trained CNNs. Sampling during training and testing not only improved the accuracy and the representation of uncertainty of CNNs for the ImageNet and Eco-set test sets, making the CNNs better computer vision models, but also increased the correlation between the predictions of classifiers trained on the internal representations of CNNs and human confidence scores, making them better models of human visual perception. These improvements were caused by simply injecting random Gaussian noise into the CNNs and integrating over different random noise samples. This mechanism is not only simple, but also neurobiologically plausible. However, this is likely not the only sampling paradigm that can lead to improvements, as discussed in the previous chapter. While these results are far from conclusive, they do add to the evidence that

Bayesian models predict human behaviour and that sampling in biological neural networks could be a means of representing uncertainty for visual perception in the human brain.

Chapter 5

Conclusion

Deep neural networks (DNNs) have revolutionised machine learning and AI, and have recently moved back into computational neuroscience. These models reach human-level performance in certain tasks, and early experiments indicate that they are capable of capturing characteristics of cortical function that cannot be captured with shallow models. DNNs offer an intriguing framework that enables computational neuroscientists to address fundamental questions about brain computation in the developing and adult brain, particularly for visual perception. In this thesis we focus on two areas of investigation: (1) constraining the representational spaces of DNNs using a reference representational distance matrix (RDM) and (2) using stochasticity in DNNs to implement variational approximations to Bayesian inference through sampling.

It is important for potential models of neural computation to both match human behaviour and explain neural data. Historically, models have been mainly constrained by neural data, but the difficulty of collecting, and thus the scarcity, of this data makes constraining the complex models models required for complicated tasks infeasible. DNNs are normally trained only to perform a task. In this thesis, we proposed deep representational distance learning (RDL), which constrains the internal representations of a DNN by pulling its RDM towards a reference RDM. We showed that for machine learning benchmarks this method can transfer information from a teacher model to a DNN through an RDM. This constraint led to a significant increase in test accuracy. In the future, this method could be used to constrain large scale DNNs using RDMs computed from neural data, such as single cell recordings or functional magnetic resonance imaging (fMRI) data.

High-level human behaviour and decision making often mirrors Bayesian models. A prominent computational neuroscience theory is that this is performed by sampling in the human brain, either at neurons or synapses. In this thesis, we evaluated how stochasticity and sampling affected DNNs trained to perform object recognition. We showed that sampling

DNN weights using a variety of Bernoulli- and Gaussian-based methods approximates Bayesian inference in neural networks and leads to better representation of uncertainty for benchmark object recognition datasets. We also found that sampling only using Gaussian unit noise during learning and inference resulted in better prediction of human confidence for image classification. This adds to the evidence that Bayesian methods can model human perceptual decision making at the computational level. The tested methods were trained only with supervised learning, which limits the claims that can be made about learning. An approach that would be more realistic from a neuroscience perspective would seamlessly interleave unsupervised and supervised training, since supervised examples are limited in the real world. This is an area of research that we plan to explore in the future. In the future, we will also test the effects of sampling on internal representations in DNNs and evaluate whether sampling increases the similarity between DNN representational spaces and those defined by neural data (e.g. RDMs calculated using fMRI recording of the visual ventral stream).

DNNs enhance the investigative repertoire of the computational neuroscientist. Understanding neural computations is ultimately an interdisciplinary endeavour. Experimental neuroscientists will have to collaborate with machine learning researchers if we are to understand how the brain works. With computers approaching the brain in computational power, we are entering a truly exciting phase of computational neuroscience.

References

Laurence Aitchison and Peter E. Latham. Synaptic sampling: A connection between psp variability and uncertainty explains neurophysiological observations. *arXiv preprint arXiv:1505.04544*, 2015.

Laurence Aitchison and Máté Lengyel. The hamiltonian brain: Efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics. *PLOS Computational Biology*, 12(12):e1005186, 2016.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, pages 2654–2662, 2014.

David Barber and Christopher M. Bishop. Ensemble learning in bayesian neural networks. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 168:215–238, 1998.

Horace B. Barlow. Possible principles underlying the transformations of sensory messages. 1961.

Gladys Barragan-Jason, Gabriel Besson, Mathieu Ceccaldi, and Emmanuel J. Barbeau. Fast and famous: looking for the fastest speed at which a face can be recognized. *Frontiers in Psychology*, 4:100, 2013.

Simon Barthelmé and Pascal Mamassian. Evaluation of objective uncertainty in the visual system. *PLoS Computational Biology*, 5(9):e1000504, 2009.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Anthony J. Bell and Terrence J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. *Unsupervised and Transfer Learning Challenges in Machine Learning*, 7:19, 2012.

Pietro Berkes and Laurenz Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5(6):9–9, 2005.

Pietro Berkes, Gergő Orbán, Máté Lengyel, and József Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013): 83–87, 2011.

Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the International Conference on Machine Learning*, pages 1613–1622, 2015.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, pages 535–541, 2006.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7(11):e1002211, 2011.

Jordan Burgess, James Robert Lloyd, and Zoubin Ghahramani. One-shot learning in discriminative neural networks. In *NIPS Bayesian Deep Learning Workshop*, 2016.

Charles F. Cadieu, Ha Hong, Daniel L. K. Yamins, Nicolas Pinto, Diego Ardila, Ethan A. Solomon, Najib J. Majaj, and James J. DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Computational Biology*, 10(12):e1003963, 2014.

Matteo Carandini and David J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51, 2012.

Thomas A. Carlson, J. Brendan Ritchie, Nikolaus Kriegeskorte, Samir Durvasula, and Junsheng Ma. Reaction time for object categorization is predicted by representational distance. *Journal of cognitive neuroscience*, 26(1):132–142, 2014.

Radoslaw M. Cichy, Aditya Khosla, Dimitrios Pantazis, Antonio Torralba, and Aude Oliva. Deep neural networks predict hierarchical spatio-temporal cortical dynamics of human visual object recognition. *arXiv preprint arXiv:1601.02970*, 2016.

Radoslaw Martin Cichy, Aditya Khosla, Dimitrios Pantazis, and Aude Oliva. Dynamics of scene representations in the human brain revealed by magnetoencephalography and deep neural networks. *NeuroImage*, 153:346–358, 2017.

Carl F. Craver. Explaining the brain: Mechanisms and the mosaic unity of neuroscience. 2007.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013.

Jörn Diedrichsen and Nikolaus Kriegeskorte. Representational models: A common framework for understanding encoding, pattern-component, and representational-similarity analysis. *PLoS Computational Biology*, 13(4):e1005508, 2017.

Serge O. Dumoulin and Brian A. Wandell. Population receptive field estimates in human visual cortex. *NeuroImage*, 39(2):647–660, 2008.

Sven Eberhardt, Jonah G. Cader, and Thomas Serre. How deep is the feature analysis underlying rapid visual categorization? In *Advances in Neural Information Processing Systems*, pages 1100–1108, 2016.

Allen L. Edwards. Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, 13(3):185–187, 1948.

Chris Eliasmith and Oliver Trujillo. The use and abuse of large-scale brain models. *Current Opinion in Neurobiology*, 25:1–6, 2014.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Holger Finger and Peter König. Phase synchrony facilitates binding and segmentation of natural images in a coupled neural oscillator network. *Frontiers in Computational Neuroscience*, 7:195, 2014.

József Fiser, Pietro Berkes, Gergő Orbán, and Máté Lengyel. Statistically optimal perception and learning: from behavior to neural representations, 2010a.

József Fiser, Pietro Berkes, Gergő Orbán, and Máté Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, 14(3): 119–130, 2010b.

Mathias Franzius, Henning Sprekeler, and Laurenz Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

Mathias Franzius, Niko Wilbert, and Laurenz Wiskott. Invariant object recognition with slow feature analysis. In *International Conference on Artificial Neural Networks*, pages 961–970, 2008.

Jeremy Freeman and Eero P. Simoncelli. Metamers of the ventral stream. *Nature Neuroscience*, 14(9):1195, 2011.

Winrich A. Freiwald and Doris Y. Tsao. Functional compartmentalization and viewpoint generalization within the macaque face-processing system. *Science*, 330(6005):845–851, 2010.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. pages 267–285, 1982.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *ICML Deep Learning Workshop*, 2015.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *ICLR Workshop*, 2016.

Edward I. George and Robert E. McCulloch. Approaches for bayesian variable selection. *Statistica Sinica*, pages 339–373, 1997.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Robbe L. T. Goris, J. Anthony Movshon, and Eero P. Simoncelli. Partitioning neuronal variability. *Nature Neuroscience*, 17(6):858, 2014.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

Thomas L. Griffiths, Charles Kemp, and Joshua B. Tenenbaum. *Bayesian models of cognition*. Cambridge University Press, 2008.

Umut Güçlü and Marcel A. J. van Gerven. Unsupervised feature learning improves prediction of human brain activity in response to natural images. *PLoS Computational Biology*, 10 (8):e1003724, 2014.

Umut Güçlü and Marcel A. J. van Gerven. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27):10005–10014, 2015.

Jordan Guerguiev, Timothy P. Lillicrap, and Blake A. Richards. Towards deep learning with segregated dendrites. *eLife*, 6, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016.

Linda Henriksson, Marieke Mur, and Nikolaus Kriegeskorte. Faciotopy—a face-feature map with face-like topology in the human occipital face area. *Cortex*, 72:156–167, 2015.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1861–1869, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E. Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the ACM Conference on Computational Learning Theory*, pages 5–13, 1993.

Lasse Holmstrom and Petri Koistinen. Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38, 1992.

Ha Hong, Daniel L. K. Yamins, Najib J. Majaj, and James J. DiCarlo. Explicit information for category-orthogonal object properties increases along the ventral stream. *Nature Neuroscience*, 19(4):613, 2016.

Patrik O Hoyer and Aapo Hyvärinen. Interpreting neural response variability as monte carlo sampling of the posterior. In *Advances in Neural Information Processing Systems*, pages 293–300, 2003.

David H. Hubel and Torsten N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, 148(3):574–591, 1959.

David H. Hubel and Torsten N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160(1):106–154, 1962.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*, volume 46. John Wiley & Sons, 2004.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456, 2015.

Hemant Ishwaran and J. Sunil Rao. Spike and slab variable selection: frequentist and bayesian strategies. *Annals of Statistics*, pages 730–773, 2005.

Judson P. Jones and Larry A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6): 1233–1258, 1987.

Pasi Jylänki, Aapo Nummenmaa, and Aki Vehtari. Expectation propagation for neural networks with sparsity-promoting priors. *Journal of Machine Learning Research*, 15(1): 1849–1901, 2014.

David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Synaptic sampling: A bayesian approach to neural network plasticity and rewiring. In *Advances in Neural Information Processing Systems*, 2015.

Kendrick N. Kay. Principles for models of neural information processing. *NeuroImage*, 2017.

Kendrick N. Kay, Thomas Naselaris, Ryan J. Prenger, and Jack L. Gallant. Identifying natural images from human brain activity. *Nature*, 452(7185):352, 2008.

Christoph Kayser, Wolfgang Einhäuser, Olaf Dümmer, Peter König, and Konrad Körding. Extracting slow subspaces from natural videos leads to complex cells. pages 1075–1080, 2001.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? pages 5580–5590, 2017.

Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computational Biology*, 10 (11):e1003915, 2014.

Saeed R. Kheradpisheh, Masoud Ghodrati, Mohammad Ganjtabesh, and Timothée Masquelier. Humans and deep networks largely agree on which kinds of variation make object recognition harder. *Frontiers in Computational Neuroscience*, 10:92, 2016.

Tim C. Kietzmann, Jascha D. Swisher, Peter König, and Frank Tong. Prevalence of selectivity for mirror-symmetric views of faces in the ventral and dorsal visual pathways. *Journal of Neuroscience*, 32(34):11763–11772, 2012.

Tim C. Kietzmann, Anna L. Gert, Frank Tong, and Peter König. Representational dynamics of facial viewpoint encoding. *Journal of Cognitive Neuroscience*, 2017a.

Tim C. Kietzmann, Patrick McClure, and Nikolaus Kriegeskorte. Deep neural networks in computational neuroscience. *bioRxiv*, page 133504, 2017b.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

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.

David C. Knill and Alexandre Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *Trends in Neurosciences*, 27(12):712–719, 2004.

Konrad P. Kording, Christoph Kayser, Wolfgang Einhauser, and Peter Konig. How are complex cell properties adapted to the statistics of natural stimuli? *Journal of Neurophysiology*, 91(1):206–212, 2004.

Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1:417–446, 2015.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4, 2008.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Joseph B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

Jonas Kubilius, Stefania Bracci, and Hans P. Op de Beeck. Deep neural networks as a computational model for human shape sensitivity. *PLoS Computational Biology*, 12(4): e1004896, 2016.

M. Kümmerer, L. Theis, and M. Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. In *International Conference on Learning Representations*, 2015.

Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *The International Joint Conference on Neural Networks*, pages 1–8, 2010.

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. 1995.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*, 2014.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 498–515, 2015.

Joel Z. Leibo, Qianli Liao, Fabio Anselmi, Winrich A. Freiwald, and Tomaso Poggio. View-tolerant face recognition and hebbian learning imply mirror-symmetric neural tuning to head orientation. *Current biology*, 27(1):62–67, 2017.

Ifat Levy, Uri Hasson, Galia Avidan, Talma Hendler, and Rafael Malach. Center–periphery organization of human object areas. *Nature Neuroscience*, 4(5):533, 2001.

Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

Nuo Li and James J. DiCarlo. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science*, 321(5895):1502–1507, 2008.

Nuo Li and James J. DiCarlo. Unsupervised natural visual experience rapidly reshapes size-invariant object representation in inferior temporal cortex. *Neuron*, 67(6):1062–1075, 2010.

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015.

Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

Christos Louizos. Smart regularization of deep architectures. Master's thesis, University of Amsterdam, 2015.

Wei Ji Ma, Jeffrey M. Beck, Peter E. Latham, and Alexandre Pouget. Bayesian inference with probabilistic population codes. *Nature Neuroscience*, 9(11):1432–1438, 2006.

David J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

David Madigan and Adrian E. Raftery. Model selection and accounting for model uncertainty in graphical models using occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546, 1994.

Najib J. Majaj, Ha Hong, Ethan A. Solomon, and James J. DiCarlo. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *Journal of Neuroscience*, 35(39):13402–13418, 2015.

Adam H. Marblestone, Greg Wayne, and Konrad P. Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10:94, 2016.

Panos Z. Marmarelis and Vasilis Z. Marmarelis. The white-noise method in system identification. In *Analysis of Physiological Systems*, pages 131–180. 1978.

David Marr and Tomaso Poggio. From understanding computation to understanding neural circuitry. 1976.

Narihisa Matsumoto, Masato Okada, Yasuko Sugase-Miyamoto, Shigeru Yamane, and Kenji Kawano. Population dynamics of face-responsive neurons in the inferior temporal cortex. *Cerebral Cortex*, 15(8):1103–1112, 2004.

Patrick McClure and Nikolaus Kriegeskorte. Representational distance learning for deep neural networks. *Frontiers in Computational Neuroscience*, 10:131, 2016.

Patrick McClure and Nikolaus Kriegeskorte. Representation of uncertainty in deep neural networks through sampling. *NIPS Bayesian Deep Learning Workshop*, 2017.

Johannes Mehrer, Tim C. Kietzmann, and Nikolaus Kriegeskorte. Deep neural networks trained on ecologically relevant categories better explain human it. In *Conference on Cognitive Computational Neuroscience*, 2017.

Toby J. Mitchell and John J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

Tom M. Mitchell, Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel Adam Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195, 2008.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Rubén Moreno-Bote, David C. Knill, and Alexandre Pouget. Bayesian sampling in visual perception. *Proceedings of the National Academy of Sciences*, 108(30):12491–12496, 2011.

Marieke Mur, Mirjam Meys, Jerzy Bodurka, Rainer Goebel, Peter A Bandettini, and Nikolaus Kriegeskorte. Human object-similarity judgments reflect and transcend the primate-it object representation. *Frontiers in Psychology*, 4:128, 2013.

Thomas Naselaris, Ryan J. Prenger, Kendrick N. Kay, Michael Oliver, and Jack L. Gallant. Bayesian reconstruction of natural images from human brain activity. *Neuron*, 63(6): 902–915, 2009.

Thomas Naselaris, Kendrick N. Kay, Shinji Nishimoto, and Jack L. Gallant. Encoding and decoding in fmri. *NeuroImage*, 56(2):400–410, 2011.

Radford M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

Hamed Nili, Cai Wingfield, Alexander Walther, Li Su, William Marslen-Wilson, and Nikolaus Kriegeskorte. A toolbox for representational similarity analysis. *PLoS Computational Biology*, 10(4):e1003553, 2014.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

Bruno A. Olshausen and David J. Field. How close are we to understanding v1? *Neural Computation*, 17(8):1665–1699, 2005.

Gergő Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2):530–543, 2016a.

Gergő Orbán, Pietro Berkes, József Fiser, and Máté Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92(2):530–543, 2016b.

Randall C. O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5):895–938, 1996.

David P. Reichert and Thomas Serre. Neuronal synchrony in complex-valued deep networks. *arXiv preprint arXiv:1312.6115*, 2013.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286, 2014.

Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019, 1999.

J. Brendan Ritchie and Thomas A. Carlson. Neural decoding and "inner" psychophysics: a distance-to-bound approach for linking mind, brain, and behavior. *Frontiers in Neuroscience*, 10:190, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

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. *International journal of Computer Vision*, 115(3):211–252, 2015.

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–901, 2016.

K. Seeliger, M. Fritsche, U. Güçlü, S. Schoenmakers, J.-M. Schoffelen, SE Bosch, and M. A. J. van Gerven. Cnn-based encoding and decoding of visual object recognition in space and time. *bioRxiv*, page 118091, 2017.

Markus Siegel, Tobias H. Donner, and Andreas K. Engel. Spectral fingerprints of large-scale neuronal interactions. *Nature Reviews Neuroscience*, 13(2):121–134, 2012.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Eero P. Simoncelli and Bruno A. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24(1):1193–1216, 2001.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Courtney J. Spoerer, Patrick McClure, and Nikolaus Kriegeskorte. Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in Psychology*, 8:1551, 2017.

Nitish Srivastava and Ruslan R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems*, pages 2094–2102, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Yasuko Sugase, Shigeru Yamane, Shoogo Ueno, and Kenji Kawano. Global and fine information coded by single neurons in the temporal visual cortex. *Nature*, 400(6747): 869–873, 1999.

Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 2015.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

Amirhossein Tavanaei and Anthony S. Maida. Bio-inspired spiking convolutional neural network using layer-wise sparse coding and stdp learning. *arXiv preprint arXiv:1611.03000*, 2016.

Michalis K Titsias and Miguel Lázaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2011.

Richard Turner and Maneesh Sahani. A maximum-likelihood interpretation for slow feature analysis. *Neural Computation*, 19(4):1022–1038, 2007.

Rafael Uetz and Sven Behnke. Locally-connected hierarchical neural networks for gpu-accelerated object recognition. *NIPS Workshop on Large-scale Machine Learning: Parallelism and Massive Datasets*, 2009.

Rufin VanRullen. Perception science in the age of deep neural networks. *Frontiers in Psychology*, 8:142, 2017.

Iris Vilares and Konrad Kording. Bayesian models: the structure of the world, uncertainty, behavior, and the brain. *Annals of the New York Academy of Sciences*, 1224(1):22–39, 2011.

Guy Wallis and Heinrich H. Bülthoff. Effects of temporal association on recognition memory. *Proceedings of the National Academy of Sciences*, 98(8):4800–4804, 2001.

Guy Wallis and Edmund T. Rolls. Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194, 1997.

Thomas S. A. Wallis, Matthias Bethge, and Felix A. Wichmann. Testing models of peripheral encoding using metamerism in an oddity paradigm. *Journal of Vision*, 16(2):4, 2016.

Dirk B. Walther, Eamon Caddigan, Li Fei-Fei, and Diane M. Beck. Natural scene categories revealed in distributed patterns of activity in the human brain. *Journal of Neuroscience*, 29 (34):10573–10581, 2009.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning*, pages 1058–1066, 2013.

Liwei Wang, Chen-Yu Lee, Zhuowen Tu, and Svetlana Lazebnik. Training deeper convolutional networks with deep supervision. *arXiv preprint arXiv:1505.02496*, 2015.

Daniel Weiller, Robert Märtin, Sven Dähne, Andreas K. Engel, and Peter König. Involving motor capabilities in the formation of sensory space representations. *PloS one*, 5(4): e10377, 2010.

Max Welling and Yee W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the International Conference on Machine Learning*, pages 681–688, 2011.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. pages 639–655, 2012.

James C. R. Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural Computation*, 29(5):1229–1262, 2017.

Michael C.-K. Wu, Stephen V. David, and Jack L. Gallant. Complete functional characterization of sensory neurons by system identification. *Annual Review of Neuroscience*, 29: 477–505, 2006.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Dean Wyatte, Tim Curran, and Randall O'Reilly. The limits of feedforward vision: recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, 24(11):2248–2261, 2012.

Dean Wyatte, David J. Jilk, and Randall C. O'Reilly. Early recurrent feedback facilitates visual object recognition under challenging conditions. *Frontiers in Psychology*, 5:674, 2014.

Reto Wyss, Peter König, and Paul F. M. J. Verschure. A model of the ventral visual system based on temporal stability and local memory. *PLoS Biology*, 4(5):e120, 2006.

Daniel L. K. Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356, 2016.

Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23): 8619–8624, 2014.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833, 2014.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

Appendix A

A.1 L2 regularisation and the KLD between Gaussians

The Kullback–Leibler divergence (KLD) between $\mathcal{N}(\mu_q, \sigma_q^2)$ and $\mathcal{N}(\mu_p, \sigma_p^2)$ can be calculated using:

$$KL(q(w_{i,j})||p(w_{i,j})) = \frac{(\mu_q - \mu_p)^2}{2\sigma_p^2} + \log\frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2}{2\sigma_p^2} - \frac{1}{2} \tag{A.1}$$

In the case where $\mathcal{N}(\mu_p, \sigma_p^2)$ is a pre-defined prior and σ_q is not a function of the learnable parameters V:

$$\underset{V}{\operatorname{argmin}}\, KL(q(w_{i,j})||p(w_{i,j})) = \underset{V}{\operatorname{argmin}}\, \frac{(\mu_q - \mu_p)^2}{2\sigma_p^2} \tag{A.2}$$

For $\mu_p = 0$, this is equivalent to L2 regularisation where the L2-coefficient is equal to $1/\sigma_p^2$. However, in the case where σ_q is a function of V, such as for Gaussian dropout/dropconnect, this equivalence does not hold. Kingma et al. (2015) used a log-uniform prior instead of a Gaussian prior in order to bypass this and make the KLD not a function of V. In our derivations, we minimise a lower bound of Equation A.1 constructed using the fact that the sum of the terms that include σ_q and the constant term is greater than or equal to 0:

$$KL(q(w_{i,j})||p(w_{i,j})) \geq \frac{(\mu_q - \mu_p)^2}{2\sigma_p^2} \tag{A.3}$$

Note that a similar lower bound can be derived for the KLD between two multivariate Gaussians.

A.2 Gaussian "reparameterization trick"

As discussed in Kingma et al. (2015), for a matrix W of Gaussian random variables can be sampled using the "reparamaterization trick":

$$w_{i,j} \sim \mathcal{N}(v_{i,j}, \alpha v_{i,j}^2) \tag{A.4}$$

$$w_{i,j} = f(v_{i,j}, \varepsilon_{i,j}) = v_{i,j} + \sqrt{\alpha} v_{i,j} \varepsilon_{i,j} \tag{A.5}$$

where $\varepsilon_{i,j}$ is sampled from $\mathcal{N}(0,1)$, $\alpha = p/(1-p)$, and p is the dropout or dropconnect drop probability. Given a deterministic, differentiable, and monotonic mapping $W = f(V, \varepsilon)$, $q_V(W)dW = p(\varepsilon)d\varepsilon$. As a result:

$$\int q_V(W) l(W) dW = \int p(\varepsilon) l(W) d\varepsilon = \int p(\varepsilon) l(f(V, \varepsilon)) d\varepsilon \tag{A.6}$$

A.3 MC Gaussian Dropconnect and Dropout

For approximate inference, variational distribution $q_V(W)$ is learned by maximising the log-evidence lower bound over parameters V (Barber and Bishop, 1998; Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993):

$$\log(p(D_{train})) \geq \int \log p(D_{train}|W) q_V(W) dW - KL(q_V(W)||p(W)) \tag{A.7}$$

For either Gaussian dropout or dropconnect, each element of W is sampled from the Gaussian distribution $\mathcal{N}(v_{i,j}, \alpha v_{i,j}^2)$, similar to the methods discussed by Kingma et al. (2015) and **?**. W can then be sampled using the Gaussian "reparameterization trick", which allows Equation A.20 to be rewritten as:

$$\log(p(D_{train})) \geq \int_{\varepsilon} \log p(D_{train}|W) p(\varepsilon) d\varepsilon - KL(q_V(W)||p(W)) \tag{A.8}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

This results in the following minimisation objective function:

$$\mathcal{L}_V := -\int_{\varepsilon} \log(p(D_{train}|W)) p(\varepsilon) d\varepsilon + KL(q_V(W)||p(W)) \tag{A.9}$$

By using L2 regularisation, we are optimising a lower-bound of the KLD between $q_V(W)$ and the prior $p(w_{i,j}) = \mathcal{N}(0, \lambda^{-1})$ as previously shown:

$$\mathscr{L}_V \geq \widetilde{\mathscr{L}_V} := -\int_{\varepsilon} \log p(D_{train}|W)p(\varepsilon)d\varepsilon + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.10}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

Approximating using Monte Carlo integration for training (Eq. A.24) and testing (Eq. A.25):

$$\widetilde{\mathscr{L}_{\mathbf{v}}} :\approx -\frac{1}{n}\sum_{k=1}^{n} \log p(D_{train}|W^k) + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.11}$$

$$p(D_{test}) \approx \frac{1}{n}\sum_{k=1}^{n} p(D_{test}|W^k) \tag{A.12}$$

where $W^k = V + V \circ \varepsilon^k$ and $\varepsilon_{i,j}^k \sim \mathscr{N}(0,1)$ for Gaussian dropconnect or $\varepsilon_{i,*}^k \sim \mathscr{N}(0,1)$ for Gaussian dropout.

A.3.1 MC spike-and-slab Dropout

For MC spike-and-slab dropout, the weight matrix $W = B \circ G$ where $b_{i,*} \sim Bern(1-p_{do})$ and $g_{i,j} \sim \mathscr{N}(v_{i,j}, \sigma_{v_{i,j}}^2)$, similar to the method discussed by Titsias and Lázaro-Gredilla (2011). Instead of directly performing variational inference for $p(W|D_{train})$, we find a variational distribution, $q_V(B,G)$ for $p(B,G|D_{train})$ using:

$$\begin{aligned} log(p(D_{train})) \geq & \sum_B \int_G \log p(D_{train}|B,G) q_V(B,G) dG \\ & - KL(q_V(B,G)||p(B,G)) \end{aligned} \tag{A.13}$$

Assuming independence between the random variables B and G, $q(B,G) = q(B)q(G)$, so:

$$\begin{aligned} log(p(D_{train})) \geq & \sum_B \int_G \log p(D_{train}|B,G) q(B) q_V(G) dG \\ & - KL(q(B)||p(B)) - KL(q_V(G)||p(G)) \end{aligned} \tag{A.14}$$

For a spike-and-slab distribution, each element of G is independently sampled from a Gaussian distribution, $\mathscr{N}(v_{i,j}, \sigma_{v_{i,j}}^2)$, where $\sigma_{v_{i,j}}^2 = \alpha \mu_{v_{i,j}}^2$. G can be sampled using the Gaussian "reparameterization trick". This allows Equation A.27 to be rewritten as:

$$\log(p(D_{train})) \geq \sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon \\ - KL(q(B)||p(B)) - KL(q_V(G)||p(G)) \tag{A.15}$$

ε is a vector containing each $\varepsilon_{i,j}$.

This results in the following minimisation objective function:

$$\mathscr{L}_V := -\sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon + KL(q(B)||p(B)) + KL(q_V(G)||p(G) \tag{A.16}$$

Using $Bern(1-p_{do})$ as a prior for each element of B, where p_{do} is also used for the spike-and-slab posterior, leads to $KL(q(B)||p(B))$ being equal to zero. Using a prior of $\mathscr{N}(0,\sigma_p^2)$ for each element of G leads to L2-regularisation being a lowerbound of the KLD between $q_V(G)$ and $\mathscr{N}(0,\lambda^{-1})$:

$$\mathscr{L}_V \geq \widetilde{\mathscr{L}_V} := -\sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.17}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

Approximating using Monte Carlo integration for training (Eq. A.31) and testing (Eq. A.32):

$$\widetilde{\mathscr{L}_V} :\approx -\frac{1}{n}\sum_{k=1}^{n} \log p(D_{train}|B^k,G^k) + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.18}$$

$$p(D_{test}) \approx \frac{1}{n}\sum_{k=1}^{n} p(D_{test}|B^k,G^k) \tag{A.19}$$

where $b_{i,*}^k \sim Bern(1-p_{do})$, $G^k = V + V \circ \varepsilon^k$, and $\varepsilon_{i,j} \sim \mathscr{N}(0,1)$.

For approximate inference, variational distribution $q_V(W)$ is learned by maximising the log-evidence lower bound over parameters V (Barber and Bishop, 1998; Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993):

$$\log(p(D_{train})) \geq \int \log p(D_{train}|W)q_V(W)dW - KL(q_V(W)||p(W)) \tag{A.20}$$

For either Gaussian dropout or dropconnect, each element of W is sampled from the Gaussian distribution $\mathscr{N}(v_{i,j}, \alpha v_{i,j}^2)$, similar to the methods discussed in **?** and Kingma et al. (2015). W can then be sampled using the Gaussian "reparameterization trick", which allows

Equation A.20 to be rewritten as:

$$\log(p(D_{train})) \geq \int_{\varepsilon} \log p(D_{train}|W)p(\varepsilon)d\varepsilon - KL(q_V(W)||p(W)) \tag{A.21}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

This results in the following minimisation objective function:

$$\mathscr{L}_V := -\int_{\varepsilon} \log(p(D_{train}|W))p(\varepsilon)d\varepsilon + KL(q_V(W)||p(W)) \tag{A.22}$$

By using L2 regularisation, we are optimising a lower-bound of the KLD between $q_V(W)$ and the prior $p(w_{i,j}) = \mathscr{N}(0, \lambda^{-1})$ as previously shown:

$$\mathscr{L}_V \geq \widetilde{\mathscr{L}_V} := -\int_{\varepsilon} \log p(D_{train}|W)p(\varepsilon)d\varepsilon + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.23}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

Approximating using Monte Carlo integration for training (Eq. A.24) and testing (Eq. A.25):

$$\widetilde{\mathscr{L}_{\mathbf{v}}} :\approx -\frac{1}{n}\sum_{k=1}^{n} \log p(D_{train}|W^k) + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.24}$$

$$p(D_{test}) \approx \frac{1}{n}\sum_{k=1}^{n} p(D_{test}|W^k) \tag{A.25}$$

where $W^k = V + V \circ \varepsilon^k$ and $\varepsilon_{i,j}^k \sim \mathscr{N}(0,1)$ for Gaussian dropconnect or $\varepsilon_{i,*}^k \sim \mathscr{N}(0,1)$ for Gaussian dropout.

A.3.2 MC spike-and-slab Dropout

For MC spike-and-slab dropout, the weight matrix $W = B \circ G$ where $b_{i,*} \sim Bern(1-p_{do})$ and $g_{i,j} \sim \mathscr{N}(v_{i,j}, \sigma_{v_{i,j}}^2)$, similar to the method discussed by Titsias and Lázaro-Gredilla (2011). Instead of directly performing variational inference for $p(W|D_{train})$, we find a variational distribution, $q_V(B,G)$ for $p(B,G|D_{train})$ using:

$$\begin{aligned} log(p(D_{train})) \geq & \sum_{B}\int_{G} \log p(D_{train}|B,G)q_V(B,G)dG \\ & - KL(q_V(B,G)||p(B,G)) \end{aligned} \tag{A.26}$$

Assuming independence between the random variables B and G, $q(B,G) = q(B)q(G)$, so:

$$\begin{aligned} log(p(D_{train})) \geq & \sum_B \int_G \log p(D_{train}|B,G)q(B)q_V(G)dG \\ & - KL(q(B)||p(B)) - KL(q_V(G)||p(G)) \end{aligned} \tag{A.27}$$

For a spike-and-slab distribution, each element of G is independently sampled from a Gaussian distribution, $\mathcal{N}(v_{i,j}, \sigma^2_{v_{i,j}})$, where $\sigma^2_{v_{i,j}} = \alpha\mu^2_{v_{i,j}}$. G can be sampled using the Gaussian "reparameterization trick". This allows Equation A.27 to be rewritten as:

$$\begin{aligned} \log(p(D_{train})) \geq & \sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon \\ & - KL(q(B)||p(B)) - KL(q_V(G)||p(G)) \end{aligned} \tag{A.28}$$

ε is a vector containing each $\varepsilon_{i,j}$.

This results in the following minimisation objective function:

$$\mathcal{L}_V := -\sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon + KL(q(B)||p(B)) + KL(q_V(G)||p(G) \tag{A.29}$$

Using $Bern(1-p_{do})$ as a prior for each element of B, wj, leads to $KL(q(B)||p(B))$ being equal to zero. Using a prior of $\mathcal{N}(0,\sigma_p^2)$ for each element of G leads to L2-regularisation being a lowerbound of the KLD between $q_V(G)$ and $\mathcal{N}(0,\lambda^{-1})$:

$$\mathcal{L}_V \geq \widetilde{\mathcal{L}_V} := -\sum_B \int_\varepsilon \log p(D_{train}|B,G)p(\varepsilon)q(B)d\varepsilon + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.30}$$

where ε is a vector containing each $\varepsilon_{i,j}$.

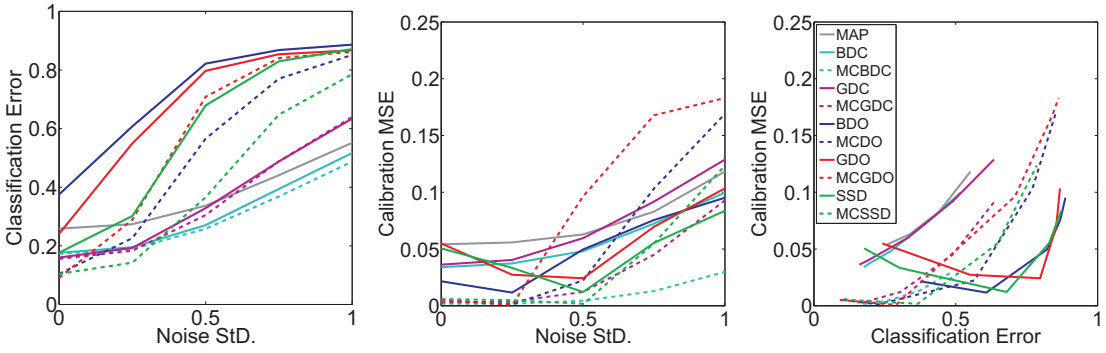
Approximating using Monte Carlo integration for training (Eq. A.31) and testing (Eq. A.32):

$$\widetilde{\mathcal{L}_V} :\approx -\frac{1}{n}\sum_{k=1}^{n} \log p(D_{train}|B^k,G^k) + \frac{\lambda}{2}||vec(V)||_2^2 \tag{A.31}$$

$$p(D_{test}) \approx \frac{1}{n}\sum_{k=1}^{n} p(D_{test}|B^k,G^k) \tag{A.32}$$

where $b^k_{i,*} \sim Bern(1-p_{do})$, $G^k = V + V \circ \varepsilon^k$, and $\varepsilon_{i,j} \sim \mathcal{N}(0,1)$.

A.4 Additional Section 3.3.2 Results



(A) **(B)** **(C)**

Fig. A.1 Sampling at test time allows for higher variance sampling to be used during training without inducing network failure. The CIFAR-10 (A) classification error for additive Gaussian noise using standard deviations St.D of 0, 0.25, 0.5, 0.75, and 1, (B) mean squared error (MSE) between the $x = y$ line and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD., and (C) calibration MSE versus the classification error for predicitons across all noise StD. for Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) with and without MC sampling using 10 samples. For all dropconnect and dropout methods, $p = 0.5$. For spike-and-slab, $p_{do} = 0.5$ and $p_{dc} = 0.1$.

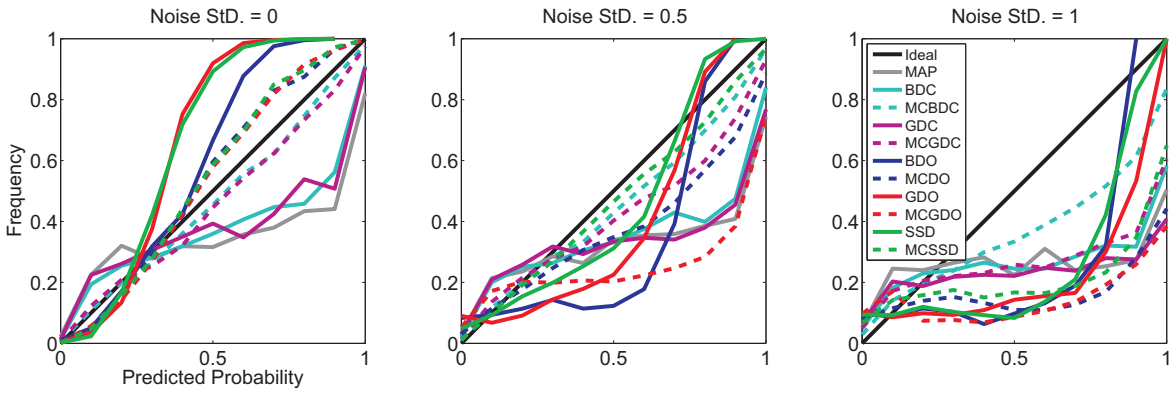


Fig. A.2 Sampling at test time allows for higher variance sampling to be used during training without causing underfitting and underconfidence. The $x = y$ line (Ideal) and the calibration plot (i.e. the frequency of the true label vs predicted probability of that label) for varying Gaussian image noise StD. for the CIFAR-10 trained Bernoulli dropconnect (BDC), Gaussian dropconnect (GDC), Bernoulli dropout (BDO), Gaussian dropout (GDO), and spike-and-slab dropout (SSD) networks with and without MC sampling using 10 samples. For all dropconnect and dropout methods, $p = 0.5$. For spike-and-slab, $p_{do} = 0.5$ and $p_{dc} = 0.1$.