One-shot Learning in Discriminative Neural Networks

Jordan Burgess
Queens' College

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Machine Learning, Speech and Language Technology

University of Cambridge
Engineering Department

August, 2016

Declaration

I, Jordan Burgess, of Queens' College, being a candidate for the MPhil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 11 400

Signed:

Date:

Abstract

Learning a concept from few examples remains a difficult problem in machine learning. Many of the breakthough successes have relied on applying gradient-based optimisation to high-capacity models. This has required large labelled datasets and long offline training times. However, there are many tasks where such data is not available and we need to quickly learn from only a few examples.

In the limit, this task is known as 'one-shot learning'. In this thesis we propose a novel approach to improve the accuracy of deep discriminative networks in this ultra-low data regime.

Humans have been shown to be able to learn object categories rapidly from few examples, especially if the object is composed of familiar components. Crucially, humans do not start as a blank model. Whether inate or learned, there is existing structure that can be transfered to new tasks; to quickly understand the structure of the concept, break it down into its constituant parts and draw anologies with past experience. This forms the basis for our approach.

Our key insight is that there is information contained in a model's parameters that can be exploited for new tasks. In this work, we illustrate explicitly (potentially for the first time), how the weights in a pretrained network form a compelling taxonomy of visual objects.

By examining these weights and their interaction with an example of a novel class, we determine an 'informed prior' over the weights for a new task. We then present a Bayesian learning procedure that is able to establish a classifier after few examples.

Compared to other one-shot learning techniques, we use the same general architecture as the convolutional neural networks used in the state of the art. Furthermore, our approach does not require access to the training data from a transferred task. Instead we exploit information already learned and available in publicly available pretrained models.

Our model achieved high one-shot learning classification performance, comparable to published work in this area. Unlike other methods, our model also performs highly in the presence of lots of data, making it of significant practical importance.

Contents

Chapter 1

Introduction

The rapid assimilation of new information is a trait demonstrated in humans of all ages. Show a three-year-old a giraffe for the first time, and immediately they'll be able to recognise other instances of the same animal. Show a deep learning model that same image, and there's little hope it'd be able to do the same. Most machine learning models rely on thousands of training examples in order to gradually build up comparable recognition accuracy.

The breakthrough performances of deep learning in speech recognition, image classification, game playing tasks amongst others [Hinton et al., 2012, He et al., 2015, Silver et al., 2016] have demonstrated the capabilities of high-capacity models optimised through gradient descent. However, all of these examples has relied on an abundance of labelled data to learn from. For many real world tasks, the availability of large amounts of labelled data can not be depended upon. Training deep neural networks on a small dataset will cause it overfit significantly, giving wrong predictions with no indication of uncertainty. Regularisation techniques such as weight penalties and dropout help, but are often insufficient to resolve this issue. This motivates our exploration into novel techniques that enable rapid learning from sparse data.

The observation humans frequently demonstrate the ability to learning from little data provides further motivation. Children are well-practised in this task, learning roughly ten new words a day after beginning to speak until the end of high school [Bloom, 2001]; largely grasping the meaning of words after just one or two examples [Carey and Bartlett, 1978].

Clearly, the ability to do so requires strong inductive biases. Crucially, humans do not start with a blank state, but with a combination of pre-existing brain structures developed through evolution and experience. It is thought that it is through transferring these abstract representations to novel but related tasks which enables the quick expansion of learned abilities.

This motivates our research into transfer learning (exploiting information learned elsewhere for help on a new task), and specifically, 'one-shot learning' (learning information about object categories from only a single example). This is a challenging task for machine learning models. It relies on being able to extract maximal information from the new example while drawing analogies to previous experience on other tasks.

In recent years there has been growing interest in one-shot learning. In part because of the successes and limitations of deep learning. Image classification, given enough data and computation, is essentially a solved problem. Convolutional neural networks have been extraordinarily successful. The best performing single model on the ImageNet-1k classification challenge achieves 4.5% top-5 error. In comparison, a trained human was only able to reach 5.1% error rate[1]. It is notable that the human classifier trained on only 500 images before test evaluation while the ConvNet needed all 1.2 million images thousands of times over multiple weeks. While neither of these numbers represent a lower bound of what's possible, it is becoming a less interesting problem to achieve improved accuracy if it relies purely on more data and more computation. Instead the community looks towards methods that are more data efficient and achieve more human-like performance.

1.1 Outline of our approach to one-shot learning

A trained network is a summary of everything it's learned so far. Our hypothesis is that there is structure within the weights of pretrained network that can be exploited for learning new tasks. The novelty of our work is in using this information to inform a prior belief over the network weights to classify a new task.

One of the goals of our approach is to make full use of the existing work in deep

[1]Performed by Andrej Karparthy and described on his blog: 'What I learned from competing against a ConvNet on ImageNet' [Karparthy, 2014]

convolutional networks and suggest a practical way to expand these tasks from very little data. Rather than train from scratch, we use publicly available pretrained ConvNets. And rather than requiring access to the pretraining data, we exploit the information contained in the network weights.

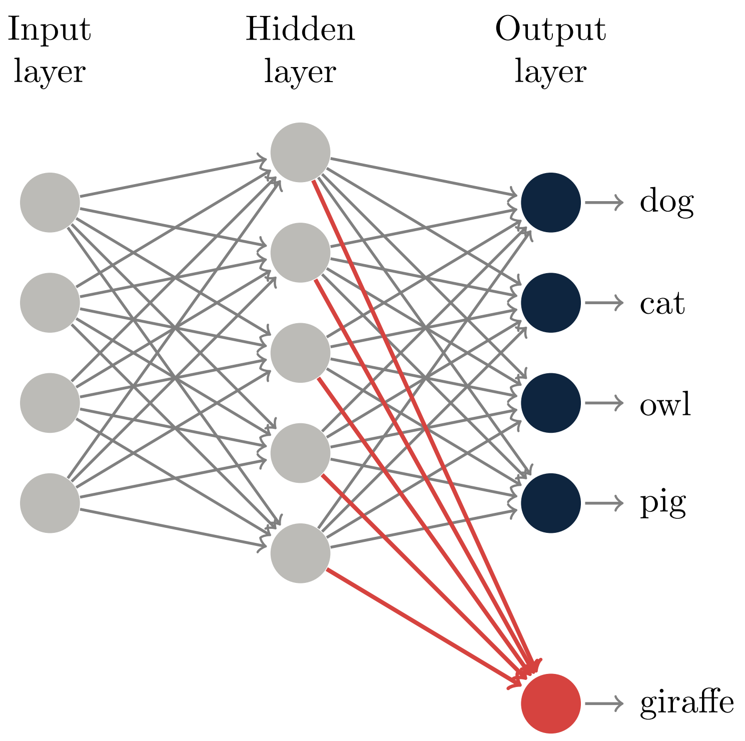


Figure 1.1: Simplified diagram of our approach. Given a pretrained network and a new category we wish to learn (giraffe) with only a few examples, we need to find good network weights (shown in red) that are able to accurately classify an a unseen example from the high level features in the final hidden layer. We set a prior on the new network weights based either on all the final layer weights learned elsewhere or on a subset of them determined by some inferred 'super-categories'. These weights are then updated in light of few training examples through a Bayesian update.

Given that these ConvNets can contain many millions of parameters, crucial to our approach is constraining the degrees of freedom. It has been shown that the features learned in a pretrained ConvNet are well suited for transfer to novel tasks [Yosinski et al., 2014] and vary in generality depending on the layer. The base and novel tasks will both be to classify images of objects, and so the features in the topmost hidden layer are likely to be suitable. We use the pretrained network as a feature extractor for the new tasks. This means we only need to learn the parameters for the classifier on top.

To do so with little data is still a challenging task. We set an inductive bias by placing a strong prior belief on what those network weights might be. To inform that prior, we examine the weights of the pretrained network (see Figure 4.1). For a more specific prior, we can cluster the pre-learned classes together in weightspace to form 'supercategories'. We can determine which supercategory a new example may belong to by examining how the weight-clusters respond to the the high-level features present in the image.

We model the distribution over the weights as a multivariate Gaussian and use this forms the prior on the new weights used in a softmax classifier. As the dimensionality of the weightspace is typically high, we use a variational Bayesian inference technique, ADVI [Kucukelbir et al., 2016], to quickly approximate the posterior weight distribution. For more accurate estimates and competitive performance, a slower MCMC-based sampling method can be used.

1.2 Document Overview

In the following sections give some background information on image classification and Bayesian learning (chapter 2), followed by a quick survey of related work, in chapter 3. In chapter 4, we give the full details of our model and Bayesian framework. In section 4.6, we illustrate (potentially for the first time) how the weights in a trained convolutional neural network create a compelling taxonomy of visual objects. In section 4.6.1 we discuss possible ways to exploit that. Finally, in chapter 5 we demonstrate how we achieve comparable one-shot learning performance to leading techniques with our substantially simpler and less specialised model.

Chapter 2

Background

To understand our approach to one-shot learning in discriminative networks, it is helpful to give a quick review of the current techniques used for image classification. Specifically, we briefly explain how convolutional neural networks have been so successful in image recognition by learning a hierarchy of reusable features and how they are currently trained and applied to new tasks.

Classifiers can be split into three broad classes of models:

- discriminative models learn a probability distribution of an outcome given an input, $p(y|x)$;
- discriminative functions map directly from x to output class y, $y = f(x)$;
- and generative models learn the joint distribution of data and latent variables $p(x, y)$ – with Bayes' decision rule applied to make predictions.

While some work on one-shot learning use generative models, in this work we focus on the discriminative feed-forward neural network models that form the basis for most image classification systems.

2.1 Deep Learning

The success of any machine learning method generally depends on having good data representations. The broad research area of *deep learning* is primarily concerned with learning good data representations *from the data* using an expressive hierarchical structure. The input data is transformed by multiple layers of non-linear functions into high-level, increasingly abstract, concepts. The parameters of the transformations are learned rather than designed. Given a target distribution at the output, the error between the target and the current output is backpropagated through the network and each parameter takes a small step in the downward gradient to reduce the error.

The design is loosely motivated from visual processing system in mammals. Hubel and Wiesel demonstrated in the 1960s that the neurons in the cortex exhibit a layed structure. They observed how neurons in the early visual layers respond to specific patterns of light, such as angled edges. This has led to one of the most successful architectures of machine learning: the artificial neural network [Rosenblatt, 1958].

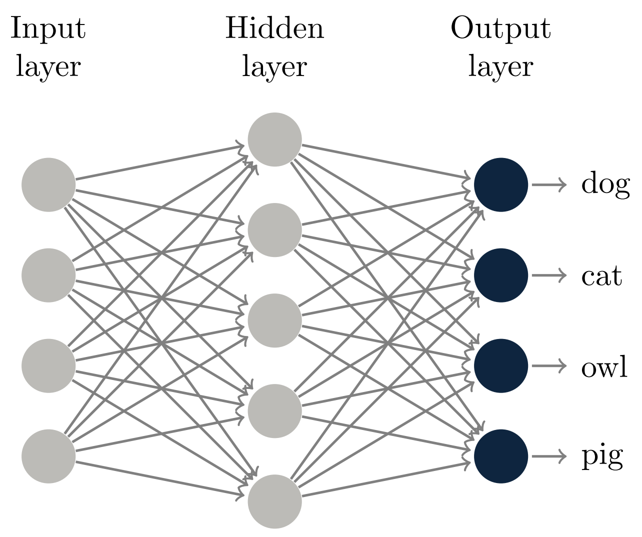


Figure 2.1: A feed-forward neural network is a directed graphical model which transforms input data through non-linear activation functions into a desired output distribution. A 'deep' network has more than one hidden layer.

Each layer contains multiple 'neurons' or 'units' that accumulate input from the layers below and produce a single scalar output which is fed to the layers above. The parameters of the model are the connection weights between each neuron and the neurons in the layer below.

Three common activation functions are given in Figure 2.2, the most common now being the rectified linear unit ("ReLU"): $f(z_j) = \max(0, z_j)$, where the input to the j-th neuron, is a weighted linear combination of the output of the previous layer,

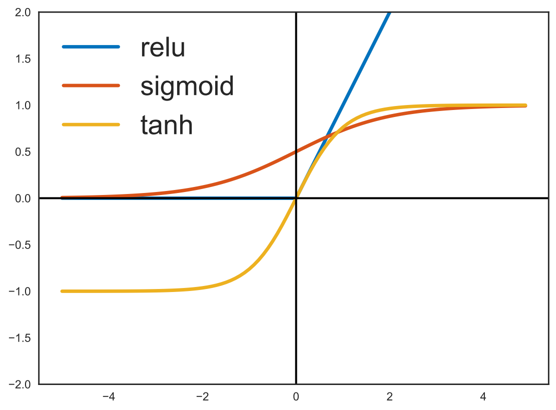


Figure 2.2: Common activation functions. The rectified linear unit (ReLU) has become the default in most deep learning tasks due to its non saturating activations and simple computation. However, care needs to be taken that their input is forever too negative or the unit can be effectively dead.

$z_j = \mathbf{w}_j^T\mathbf{x}$. Although non-differentiable precisely at zero, in practice this isn't an issue. The efficient computation and non-vanishing gradients of these units, mean that ReLUs are the most often used units for training deep neural networks.

Provided that the response from each neuron is non-linear, a single hidden-layer model can approximate any function with an accuracy determined by the number of neurons. However, using more than one hidden-layer (making the model a 'deep' neural net) can increase the statistical efficiency exponentially. It has been shown that any function that can be modelled with k layers, can (theoretically) be achieved with exponentially fewer units in k+1 layers [Hastad, 1986, Montúfar et al., 2014].

Deep learning as a concept is not new. The first experiments with artificial neural networks were conducted in the the 1950s and deep networks have been used for commercial applications since the 1990s. However, for a long period the technology was sidelined due to the difficulty in training the model's many parameters - a difficulty which only increased with depth. However, it is now apparent the skill required to get good performance is inversely proportional to the amount of training data available. The main cause of the breakthrough success of deep learning has been an abundance of data and compute power. The learning algorithms used today are fundamentally the same as those that struggled to solve toy problems in the 80s.

While a neural network can take an arbitrary network structure, we focus only on the discriminative models provided by feed-forward networks, whose structure forms a directed acyclic graph. However, by incorporating cyclic of feedback connections one can create a *recurrent neural network* that lend themselves to modelling sequential data, such as speech. If we were to allow the connections to be undirected we can form a graphical model such a Deep Boltzmann Machine [Salakhutdinov, 2009]. DBM's model the data distribution as well as multiple layers of latent states, and are used as a generative model. We discuss some of the work in one-shot-learning which explore using a deep generative models in chapter 3.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs or ConvNets) are feed-forward nets developed for visual recognition. Instead of fully-connected layers (where each neuron is connected to all the neurons above and below) it has convolutional layers which pass a fixed 'filter' over the layer below. The filters are small grids of weights which are the same when applied to each spatial location of the input features.

By exploiting the spatial structure in the data, convolutional nets can share weights and significantly reduce in the number of parameters to train. It also makes the model somewhat invariant to spatial positioning; a model of a visual object will respond to that object regardless of its location in the input image.

Often these convolution layers will be be paired with 'pooling' layers which subsample the output of the convolution. The intuition is that once a feature has been found, its exact location isn't as important as its rough location relative to other features. This enables a further reduction in the number of parameters. Usually the final few layers are fully-connected and, for classification purposes, finished with a softmax layer to enforce a sum-to-one constraint of the output multinomial scores.

Thanks to the massive reduction number of parameters, ConvNets have a much smaller VC dimension that fully-connected nets. Learning theory suggests they can therefore be learned with fewer examples, yielding improvements in computation efficiency (fewer nodes to visit) and statistical efficiency (fewer parameters to learn). This meant that they were one of the few early successes of deep learning (see Figure 2.3). However, their use for anything beyond simple digit recognition was limited due to the supposed difficulty in training the large number of parameters.

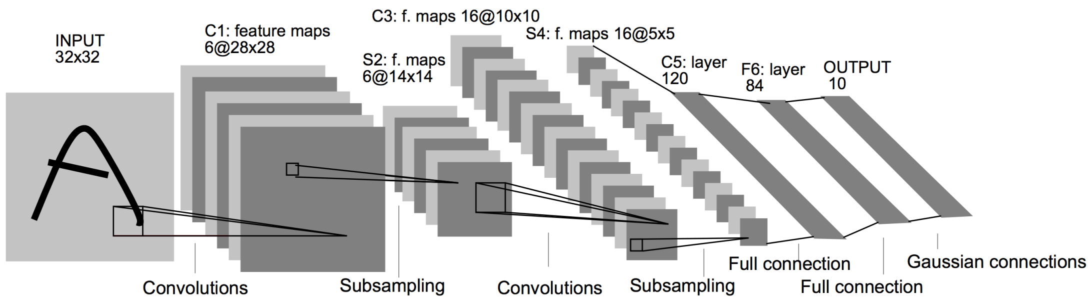


Figure 2.3: The convolutional neural network used for handwritten digit recognition known as 'LeNet-5'. Weight sharing through convolutional filters and subsampling means that this network has 340,000 connections yet only 60,000 learnable parameters (dominated by the fully-connected layers at the end). *Figure taken from LeCun et al. [1998]*

However, parallel processing on GPUs, the discovery of dropout [Srivastava et al., 2014], and the availability of large labelled datasets reintegrated convolution networks back into the computer vision community. The ImageNet dataset [Deng et al., 2009] is probably one of the leading causes of progress in computer vision in recent years. The classification challenge dataset contains 1.2 million labelled images of 1000 object classes. In 2011, a good top-5 error rate was around 25%. In 2012, we saw the first deep convolution neural net entry and a paradigm shift in computer vision research. AlexNet [Krizhevsky et al., 2012] (named after its author) was an 8-layer net with a 16% error rate. In the years since, refinements such as BatchNorm, and ResNets have allowed the networks to become deeper and more accurate. In 2015 the winning entry had 152-layers and a 4.49% error rate [Ioffe and Szegedy, 2015, He et al., 2015].

2.3 Weight initialisation

Deeper neural networks have been desirable since their initial conception as multilayer perceptrons. However, one of the great challenges was training them. For a long time, most neural networks used either the sigmoid or tanh activation functions, both of which have gradients in the range [0, 1). This meant that the backpropagated error signal would decrease exponentially with depth and training would become ineffective.

The idea of *greedy layer-wise pre-training* using restricted Boltzmann machines [Hinton et al., 2006] was one of the first succesful ways to train deep networks. By initialisiting the weights with stacked RBMs, the neuron activations were such that gradients could propagate through.

Since then, the deep learning world has effectively abandoned the use of layerwise pre-training. Switching the activation function to ReLUs has meant the gradients are less likely to vanish and the computation is easier meaning they can be trained for longer. It has been shown with an abundance of data, with proper initialisation and choice of activation function, deep supervised networks can be successfully trained, and pre-training does not aid performance [Mishkin and Matas, 2015]. Faster computation allowed by GPUs has meant that its possible to 'brute force' your way into good parameters provided you have enough data and a sensible init. Refinements over the initial parameter setting has led to 'Xavier' initialisation Glorot and Bengio [2010], which sets set the scale according to relationship between the number of input units and output units and sped up training times:

$$\mathrm{Var}(W_i) = \frac{2}{n_{in} + n_{out}} \tag{2.1}$$

2.4 Learned features

The features learned in a ConvNet tend to form low-level features at the early layers and increasingly abstract features.

An investigation of what is learned in the different layers in a deep neural network by Yosinski et al. informs our approach. The DNN learns a hierarchy of features, beginning from generic features similar to Gabor filters and colour blobs useful to all tasks, to increasingly specific features to the target task. In case of ImageNet, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.

2.5 Transfer Learning and Fine Tuning

In practice, it is rare to train an entire ConvNet from scratch. Most new tasks do not have a dataset of sufficient size. Instead one uses a pre-trained network that has already learned good features from a very large dataset. [Oquab et al., 2014] showed that for many tasks these high level features present in these networks are generic enough to be useful for multiple tasks. One then uses the ConvnNet either as a fixed feature extractor for the task of interest or as an initialisation for learning the new task.

Feature extractor

For example, take a ConvNet pretrained on ImageNet and use the activations layer before the last fully-connected layer as generic features that can be useful for a new dataset. With AlexNet this would compute a 4096 dimensional vector – known as a CNN code – for each input image. Once the features are extracted it is simple to train a classifier such as softmax or an SVM for the new dataset.

Fine-tuning

A second strategy is to continue to train the ConvNet on the new dataset. This usually involves throwing away the top-most layers and applying a suitable classifier. The whole network can then be trained through backpropagation, or some low level features can be held fixed to reduce the concern of overfitting and because these features are likely to be more generic, e.g. edge detectors or colour blob detectors that are useful to many tasks.

2.6 Regularisation

It appears that it is also preferable to use deep, powerful models – even with small data and let the data prune the structure as necessary. However, this can easily lead the model to overfit and generalise poorly. Limiting the capacity of the model in order to avoid overfitting has the effect of limiting the flexibility of the model to capture interesting and important patterns in the data.

Even with ImageNet, this is a concern. For example, AlexNet has eight layers and 60 million parameters – yet it was trained on “only” 1.2 million images. The 1000 target classes impose 10 bits of constraint on the mapping from image to label, but this is still insufficient to prevent severe overfitting.

Therefore the following tactics can be used:

- **Data augmentation**
  The simplest method to reduce overfitting is to create more data with label-preserving transformations. Often this involved random crops, rotations and horizontal reflections. This is often supplemented by altering the intensities of the RGB channels in the training images. This scheme captures an important property of natural images, that object identity is invariant to changes in image positioning, left-right axis, and intensity and colour of the illumination.

- **Early stopping**
  During training, performance is periodically assessed on a held-out validation set, and training halted if it declines. This has several shortcomings, such as biasing the estimator towards the validation set. This also requires withholding training data making it unsuitable for one-shot learning.

- **Weight penalties**
  Weight penalties such as L1/L2 regularisation prevent weights from becoming unneccesarily large and specific to features present in the training data. This is achieved by adding penalty term to the loss function. Setting an penalty based on the L2-norm is structurally equivalent to setting a prior on the weights to be Gaussian distributed around zero. This raises the question of how much regularisation to add, which in practice, is usually determined by rule-of-thumb or cross-validation error. A more satisfying solution was given in MacKay [1992], which showed that a principled Bayesian learning approach can automatically set regularisation coefficients as well obtaining estimates of the weights and output variances.

- **Dropout** [Srivastava et al., 2014]
  With dropout a random fraction of the network is deactivated during training. This helps prevent interdependence between weights and reduces the risk of overfitting. Recent work of implementing dropout at test time has

given rise to Bayesian ConvNets [Gal and Ghahramani, 2015].

- **BatchNorm** [Ioffe and Szegedy, 2015]
  Although not used in AlexNet as it wasn't invented at the time, Batch Normalisation has now become an integral part of all deep networks. Data normalisation is always performed at the input to prevent any dimension with a larger scale from having an overstated effect. However, after several layers of transformations, the activations within the network can be unevenly distributed. As normalisation is a differentiable operation, BatchNorm layers can be included *within* the network to force the activations at each layer to be distibuted with approximately unit Gaussian width and zero mean. This enables a higher learning rate and less concern about weight initialisation. The authors view it as a type of regularisition as it reduces the need for dropout and L2 weight penalty.

2.7 Bayesian learning

Most neural networks are treated as an optimisation problem: *"what are the parameters of this network which minimise the error function?"*. In the standard neural network paradigm, this involves making a point estimate of the network weights W based on the training data $\mathcal{D}$. Almost invariable, this is attempted through stochastic gradient descent on the error surface. Although this is a non-convex function in high dimensional space this generally works out ok if there's enough data. Contrary to common intuition, local minima with with high error are exponentially rare in high dimensions [Dauphin et al., 2014]. Instead there is a proliferation of saddle points which can drastically slow down learning and give the impression of local minima. It is believed that the stochastic nature of the updates can help escape some of these saddle points.

However, this approach can cause two types of errors, namely approximation and estimation errors. For a regression network, the approximation error is between between our parametric estimator $f(x; W)$ and the true regression function $h(x)$. This is determined by the capacity of the network - inversely proportional to the number of parameters which determines how much the model will underfit the data. The estimation error is the result of a lack of knowledge about the conditional distribution $p(y|W, x)$ – directly proportional to the number of parameters, and determines how much the model will overfit a certain dataset. This is commonly known as the bias/variance tradeoff [Bishop, 2006].

The issue of overfitting (high variance) is really an unfortunate consequence of maximum likelihood training. The Bayesian approach is based on the premise that all variables should be treated as random variables and our uncertainty about them expressly given. At the core is the expression known as Bayes' Rule. This incorporates a prior belief about a variable with the observed data to give an updated expression for our new probabilistic belief about that variable:

$$p(W|\mathcal{D}) = \frac{p(\mathcal{D}|W)p(W)}{p(\mathcal{D})} \tag{2.2}$$

The various distributions are known as:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$$

Our subjective beliefs of uncertainty are expressed in the prior. Once some data has been observed, the likelihood of the data under the various models allows us to update our belief about the parameters.

Whilst theoretically elegant, critics point to the use of a subjective prior. This shouldn't be a concern, as we should incorporate our uncertainty. However, it is often the case that the normalisation term, known as the 'evidence' that is computationally intractable. It often requires integration over the entire space of models. Analytical solutions only exist for certain conjugate distributions of likelihoods and priors.

Since the posterior embodies all the known statistical information about the parameters given the observations and the prior, one can theoretically obtain all features of interest by the standard probability transformations.

For example, we can estimate the predictive distribution of a new datapoint x_i by marginalising out our uncertainty over the model parameters:

$$p(y_i|x_i, \mathcal{D}) = \int p(y_i|W, \mathcal{D})p(W|\mathcal{D})dW \tag{2.3}$$

In addition, it is possible to find the MAP (maximum a posteriori) estimate of the weights by finding the peak of $p(W|\mathcal{D})$. For fixed, uniform priors the resulting solution is the maximum likelihood estimate. For fixed, zero-centered Gaussian priors, the MAP solution is akin to the maximum likelihood solution when there is an L2 weight penalty applied.

2.7.1 Approximate Bayesian Inference

Within the Bayesian paradigm, learning is presented as an integration problem. Whenever we wish to carry out normalisation, marginalisation or calculate expectations, we have to integrate. Analytical solutions only exist for certain conjugate distributions of likelihoods and priors. Most real-word problems involve aspects of non-Gaussianity, non-linearity and non-stationarity that preclude the use of analytical integration. For most distributions, the exact solution is intractable.

Often you are left with a choice between approximation methods. Variational methods can be computationally efficient, yet they may not take into account all the salient statistical features and therefore give misleading results. Monte Carlo methods are very flexible and do not require any assumptions about the probability distributions of the data. Eventually, they should eventually converge to the exact solution but in practice this can take an unhelpfully long time. From a Bayesian perspective, Monte Carlo methods allow one to compute the full posterior probability distribution.

Recent advances in probabilisitic programming and automatic differentiation have made applying these techniques easier than ever. Monte-Carlo methods can already be treated as a black box tool to be applied to almost any distribution. However, some of the most effective techniques such as Hamiltonian MCMC which exploits gradient information have hyperparameters which require tuning. The invention of the the No U-Turn Sampler (NUTS) [Hoffman and Gelman, 2014], removes this need by adaptively setting the path length. Similarly, most variational inference algorithm still requires tedious model-specific derivations and implementations. A new black-box approach, Automatic Differentiation Variational Inference (ADVI) [Kucukelbir et al., 2016] automatically selects a suitable variational distribution to optimise.

To perform Bayesian inference on hundreds or thousands of network weights, computing the posterior is not trivial. To do so, we try using both MCMC and variational methods and make use of these recent advances.

Chapter 3

Related Work

One-shot learning is sufficiently broad and challenging that it can involved many different research areas. Transfer learning is concerned with applying knowledge learned elsewhere on a new task or domain. However, the literature in this area usually predisposes sufficiently large data sets on both tasks. ‘Zero-shot learning’ attempts to learn a classifiers based information from a separate modality. For example, attempting to classify a never-before-seen example of a horse, after only having read a paragraph describing what one looks like. The intention is the same to learn a classifier with less data – although the training case is more extreme. For our purposes, we happily assume at least one training image is available.

In this section, we give a brief overview of these alternative approaches to one-shot learning and techniques that have influenced our design.

3.1 Generative Models

A significant proportion of the work on one-shot learning makes use of generative models. This includes the seminal paper Fei-Fei et al. [2006] which largely popularised one-shot learning to the machine learning community. One of the benefits of using a probabilistic model is that they can easily incorporate priors over their parameters.

In Fei-Fei et al. [2003, 2006], the authors develop a bayesian framework with the premise that previously learned classes can inform a prior on parameters for a new class. They use single global prior on all new categories induced from three hand-selected previously learned categories. Variational inference was used to approximate the posterior distribution. Before the ConvNet 'revolution', they use probabilistic, generative model known as a 'constellation model'. In several ways, our work is an extension of this idea, but we combine the Bayesian one-shot learning paradigm with modern classification models.

Another inspiration comes from Salakhutdinov et al. [2013], where instead of a single global prior, they attempt to select from multiple 'supercategory'-level priors. Salakhutdinov et al. fit a Hierarchical Dirichlet Process (HDP) prior over the activations in the final layer of a Deep Boltzmann Machine (DBM) generative model. Similar to a ConvNet, the DBM learns a layered hierarchy of increasingly abstract features. The HDP learns a a tree-structured hierarchy of categories and supercategories.

Conceptually, their process is similar to a hierarchical version of Latent Dirichlet Allocation, where the 'words' are the activations, which come from a distribution based on the 'topic'/category (which itself comes from a supercategory of the dataset). By using a non-parametric approach for creating the supercategory structure, they can infer when a example likely forms its own category or supercategory. In section 4.6 we show how we learn clusters of supercategories, and unlike this paper, we do so without access to all the training data.

Other work has explored other 'one-shot learning' skills present in humans, such as inferring causal programs and generating novel examples. These have typically required restricted domains such as hand-written characters [Lake et al., 2015, Rezende et al., 2016] and access to a dictionary of parts and stroke data. These models have had success in these restricted datasets, however, the unrestricted visual world is still too challenging for these models.

3.2 Discriminative Models

For discriminative models, some work has attempted to use a single example of the novel class to adapt a classifier from similar base class [Bart and Ullman, 2005]. However, this work used shallow, hand engineered features and simply determined which ones would be useful for a training a binary classifier on a new task.

Hoffman et al. [2013] explored one-shot learning for the adaptation of DNNs of the same categories but between different domains. For example taking a model trained on classes of office equipment from the ImageNet dataset, and applying it to the same classes of objects but from product photos from Amazon. They do this by attempting to learn a mapping from all the data, and use this to transform the one-shot example. This is a different task to ours. We attempt to expand the classifier to new categories but, for now, using the same source dataset.

3.2.1 Representation regularisation

Other research has explored whether 'low-shot' learning can be achieved by modifications to the learning objective of a standard ConvNet architecture [Hariharan and Girshick, 2016]. Similarly to us, the authors use a pretrained ConvNet as a feature extractor and apply a softmax on top to classify novel categories. They pose the one-shot learning problem as attempting to find the network weights that would be found given lots of data, rather than just achieving low loss. This means that the target weights W are characterised by having zero gradient in the loss function.

To do so they modify the loss function to include a term which penalises the squared L2-norm of the *feature representations* of x_i. This term includes a coefficient that is higher for examples that are misclassified, and near zero if they classified correctly. This seems like an important inclusion as high norm examples that are misclassified might be outliers. In a low-shot learning scenario, we are extremely vulnerable to sampling bias and such an outlier can direct the posterior weight vectors far away from the optimal solution.

Curiously, they found that this extra term achieved little beyond the gains available from penalising all features indiscriminately (Figure 3.1). They found that L2 penalty of features can speed up learning "2x", achieving accuracy usually found after two examples after only a single example. However, the L2 regularisation parameter "has a large impact and needs to be cross-validated" and to do so, they use five examples of each class. This means that these methods were privy to more examples than the claim of 'one-shot learning' would imply. And again, unlike ours, their approach relies on having the pretraining data to hand.

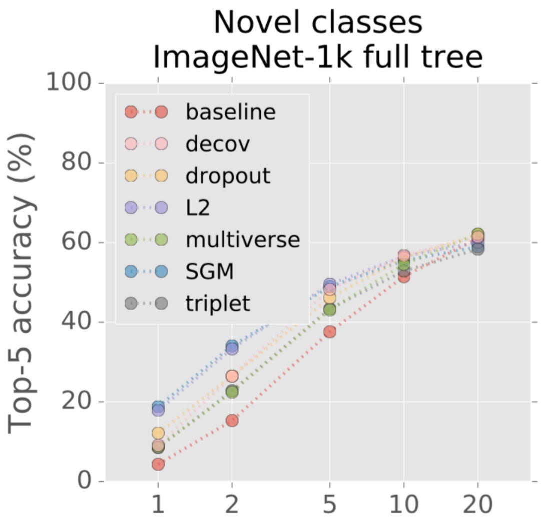


Figure 3.1: Hariharan and Girshick find that regularising extracted features weights (L2, 'SGM') can speed up low-shot learning "2x" compared to a standard softmax classifier, and outcompete other regularisation techniques such as dropout. The x-axis gives the number of training examples. However, to determine their technique's hyperparmaters required cross-validation for which they use five extra examples of each class – removing them from the strictly one-shot learning regime. *Figure from Hariharan and Girshick [2016].*

3.2.2 Metric Learning

Simple non-parametric methods are one of the most successful approaches to one-shot learning. A high baseline, on which many techniques fail to surpass, is to learn a useful representation (with a ConvNet or similar) and then apply k-nearest neighbours for classification. Recent work has looked at ways at improving the metric, such as triplet loss and magnet loss [Rippel et al., 2016] or deep non-linear transformations which learn embeddings such that objects of the same class are near each other, such as siamese networks [Koch et al., 2015] and matching networks [Vinyals et al., 2016].

This latter example of matching networks combines the idea of metric learning with the rapidly developing research area of memory augmented neural networks. These models go beyond "static" classification of fixed vectors by augmenting the network with addressable memory. The memory can act as a rapid short-term memory while the network itself gradually learns by incrementally updating its parameters. This is compelling in that it may (very loosely) be similar to how one-shot learning is achieved in the brain.

They use a non-parametric approach where the predictive label, $\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$, is determined by an attention mechanism a over the support set of k samples and labels $\mathcal{S} = (x_i, y_i)_{i=1}^{k}$. The output for a new class is essentially a linear combination of the labels in the support set. Which makes it akin to a kernel density estimator. As it is non-parametric in nature, as the support set size grows, so does the memory used.

The simplest form of $a(\cdot)$ is to use the softmax over the cosine distance of their embeddings. Like our approach, they create the embeddings using a pretrained ConvNet.

As noted in this paper, most groups' work on one-shot learning have evaluated their performance with test criteria favourable to their specific design. In Vinyals et al. [2016], the authors specify a test procedure of sampling 5 unseen classes, with 5 examples from each and recommend this a standard test proceedure for evaluating one-shot learning. They also tackle complex datasets such as ImageNet. We attempt to follow their test procedure so we can benchmark our performance against theirs (see chapter 5).

Chapter 4

Method

Our intention is to demonstrate that by placing a 'informed prior' on classifier weights for a new task, we can get high performance after only a few examples. And that by maintaining our uncertainty over these weights as a posterior distribution, that our classifier should yield better generalisation than competing approaches.

Our proposed approach is to use a pre-trained deep model and expand the topmost layer to include novel categories. We use the pretrained net as a fixed feature extractor and then test the performance on the one-shot task in a 5-way classification test of novel categories, after having seen $n \in \{1, 2, 5, 10, 20\}$ labelled examples in training. This 5-way test structure was chosen to mimmick a similar test procedure given in Vinyals et al. [2016]. Few experiments in one-shot learning have converged on testing conditions which has made like-for-like comparison difficult.

The novel categories will come from unseen examples of the same dataset, so as to reduce the amount of dataset bias introduced. We split the dataset into $\mathcal{D}_{\text{base}} = \{(x_i, t_i)|i = 1, ..., N\}$ and $\mathcal{D}_{\text{novel}} = \{(x_i, t_i)|i = 1, ..., M\}$ which contain examples of images, x_i, and target labels, t_i, from the category sets C_{base} and C_{novel} respectively.

In our n-shot 5-way learning task, we create a small training data set $\mathcal{S} \in \mathcal{D}_{\text{novel}}$ by sampling five categories at random from the unseen categories C_{novel}, then sampling n labelled images of each of the five categories: $\mathcal{S} = \{(x_i, t_i)|i = 1, ..., 5n\}$. With so little data, our model is particularly vulnerable to sampling bias. So we repeat this draw and evaluation multiple times to establish the performance.

The prior on new weights will initially be set to be a multivariate Gaussian based on the existing learned weights. It is assumed that the distribution of high-level features present in the network can represent the new categories and only the new weights will be adjusted. The posterior model for an object category will be obtained though updating the prior in light n observations of the new class. The posterior distribution can be approximated using sampling or variational inference.

The project objective will be to show the applicability of Bayesian techniques for neural network expansion and that it results in a substantial increase in learning rate.

In this work, we focus on image classification. There is suitable test data available and comparison to others' work. However, the techniques presented are general and should equally be valid on speech, text and other modalities with minor modifications.

4.1 Pre-training

Our proposal exploits a model pretrained on a different task. It does not matter whether we train this is explicitly for our purposes, or use an available model that was trained for another task, provided that we think that the features learned will be helpful for our novel task.

The pretraining procedure is to learn good feature representations, and a distribution on classifier weights. We use the base dataset $\mathcal{D}_{\text{base}} = \{(x_i, t_i) | i = 1, ..., N\}$ to train a ConvNet. We decompose this ConvNet into two parts: a transformation function $\phi(\cdot)$ which converts the input images into D-dimensional features, and a softmax classifier over the base classes with weight matrix $W_{\text{base}} = [\mathbf{w}_1, ..., \mathbf{w}_{|C_{\text{base}}|}]$.

The model is trained using stochastic gradient descent (SGD) with a cross-entropy loss function:

$$\phi^*, W^*_{\text{base}} = \underset{\phi, W_{\text{base}}}{\arg\min} \sum_{i=1}^{N} L(W_{\text{base}}, \phi(x_i), t_i) + \lambda_{L_2} ||W_{\text{base}}||^2 \tag{4.1}$$

where L is the cross-entropy loss function, $L = -t_i \log y_i$, i.e. the negative log probability of the target class. The output probability of a particular class k is given by the softmax function over the output scores:

$$P(y = k|x_i) = \frac{\exp[\mathbf{w}_k \cdot \phi(x_i)]}{\sum_j \exp[\mathbf{w}_j \cdot \phi(x_i)]} \tag{4.2}$$

The second term in Equation 4.1 is an L2 penalty applied to the network weights, applied as a form of regularisation.

After this network is trained on the base dataset, we use the (frozen) function $\phi(\cdot)$ as a fixed feature extractor on our target dataset $\mathcal{D}_{\text{novel}} = \{(x_i, y_i)|i = 1, ..., N\}$.

The trained classifier weights W^*_{base} will be used to inform our prior belief over the new weights needed for the classifier for C_{novel}.

4.2 Informing our prior

We use the pretrained ConvNet as a fixed feature extractor of the images in the new dataset $\mathcal{D}_{\text{novel}}$ to get D-dimensional CNN codes for each image $\phi(x_i)$. We train a softmax classifier directly on these features to the classes in the set C_{novel}. The softmax is parameterised by weight matrix $W_{\text{novel}} = [\mathbf{w}_1, ..., \mathbf{w}_{|C_{\text{novel}}|}]$.

The task is to find a set of weights W that can successfully categorise new examples after having seen only $n \in \{1, 2, 5, 10, 20\}$ training examples of the novel categories.

For each category, we need to define $D + 1$ weights (connections to the input features and a bias term). In the standard training paradigm, the weights would be initialised with Gaussian noise around zero. Then, with each observation of a training data, the parameters would be moved in the direction that reduces the loss function (Equation 4.1). With so few training data, we expect placing point estimates on the weights would generalise poorly to unseen test data (we test this assumption in section 5.3).

Instead, we use the Bayesian paradigm where we include our uncertainty about the model parameters, W_{novel}, as a probability distribution. By updating our distribution in light of evidence as it sees training examples, we maintain an accurate depiction of what the weights may be. By using sampled weights from this posterior distribution, we can find the distribution on the class predictions on unseen test data. For test purposes we can pick the modal prediction.

When enough data has been seen, the posterior distribution on the weights should have collapsed enough so as to be well approximated by a point estimate.

To implement this, we need to first set the prior distribution on W_{novel}. A good prior belief should significantly speed up training as it will predispose the weights into a certain distribution. A bad prior will require a lot of data to escape from.

Generally, weights are distributed around zero so a uniform Gaussian prior wouldn't be unreasonable. The MAP estimate would be equivalent to optimisation with an L2 weight penalty, but we'd likely see some additional benefit from maintaining a distribution over the weights.

However, integral to our approach is use of an 'informed' prior, to attempt to bootstrap the weights into a sensible regime. We have access to W^*_{base} which gives $K = |C_{\text{base}}|$ examples of how the weight-vectors are distributed. If we have reason to believe that the new task bears some resemblance to the pretraining task, this is a good basis for our prior belief over the weights.

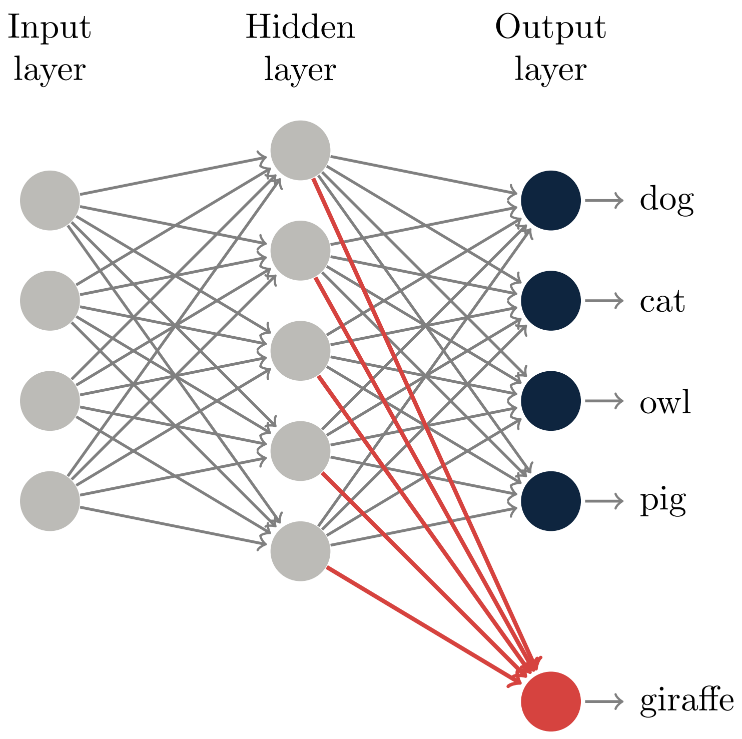


Figure 4.1: The prior distribution on $\mathbf{w}_{\text{giraffe}}$ is given by the mean and covariance structure of $W^*_{\text{base}} = [\mathbf{w}_{\text{dog}}, \mathbf{w}_{\text{cat}}, \mathbf{w}_{\text{owl}}, \mathbf{w}_{\text{pig}}]^T$.

We choose to model the distribution of weights as a multivariate Gaussian. This can incorporate any covariance structure among the features which we believe will be important to create strong inductive biases. We hypothesise that there is a high correlation between certain features in natural images. For example, an image with wheel-like features is likely to also have shiny metal-edge features, less likely to have wing-like features. This correlation should be reasonably consistent between most datasets.

Thus our prior belief is that each new weight vector $\mathbf{w}_i$ for $i \in \{1..5\}$ is distributed as:

$$\mathbf{w}_i \sim \mathcal{N}(\bar{\mu}, \bar{\mathbf{\Sigma}} + \lambda\mathbb{I}) \tag{4.3}$$

Where $\bar{\mu} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}_k$ and $\bar{\mathbf{\Sigma}} = \frac{1}{K-1}\sum_{k=1}^{K}(\mathbf{w}_k - \bar{\mu})(\mathbf{w}_k - \bar{\mu})^{\mathrm{T}}$; the sample mean and the sample covariance matrix of W^*_{base} respectively.

The parameter λ determines how much identity matrix to add. The covariance matrix will not be full rank if $K < D$, and so this is necessary to make it positive definite. For example, if we were to use AlexNet [Krizhevsky et al., 2012], the width of the final hidden layer, $D = 4096$, is greater than the examples we have of the weight vectors, $K = |C_{\text{base}}| = 1000$, so the covariance matrix would be degenerate without this added padding. The parameter also acts as a regulariser: the larger the λ, the weaker our prior belief.

4.3 Parameter update

The likelihood term is given by our 5-way softmax:

$$P(y = j|x_i, \mathbf{W}_{\text{novel}}) = \frac{\exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_j\right]}{\sum_{k=1}^{5}\exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_k\right]} \tag{4.4}$$

Given a single training example $\{x_i, t_i = j\}$, the posterior weight distribution for a weight vector $\mathbf{w}_j$ can be found through Bayes' rule:

$$P(\mathbf{w}_j|\{x_i, t_i\}) = \frac{P(\{x_i, t_i\}|\mathbf{w}_j)P(\mathbf{w}_j)}{P(\{x_i, t_i\})} \tag{4.5}$$

$$= \frac{P(y = j|\mathbf{x_i}) \cdot \mathcal{N}(\mathbf{w}_j; \bar{\mu}, \bar{\mathbf{\Sigma}} + \lambda\mathbb{I})}{\int P(y = j|\mathbf{x_i}) \cdot \mathcal{N}(\mathbf{w}_j; \bar{\mu}, \bar{\mathbf{\Sigma}} + \lambda\mathbb{I})\, d\mathbf{w}_j} \tag{4.6}$$

$$\propto \frac{\exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_j\right]}{\sum_{k=1}^{5} \exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_k\right]} \cdot \mathcal{N}(\mathbf{w}_j; \bar{\mu}, \bar{\mathbf{\Sigma}} + \lambda\mathbb{I}) \tag{4.7}$$

Even in its unnormalised form, this is a complicated expression because of the interaction between the weights in the softmax classifier. To estimate one weight-vector's posterior requires marginalising out our uncertainty over all the other parameters.

We will explore both sampling and variational methods to approximate this posterior. Due to the high-dimensionality of D, Metropolis-Hastings based MCMC sampling may be prohibitively slow. Variational inference methods replace the unbiased estimate of MCMC with a biased, but computationally efficient algorithm.

4.4 Predictions

To calculate the predictive distribution for a test case, we must marginalise out our uncertainty over the parameters given our 'low-shot' training set $\mathcal{S}$:

$$P(y = t_i|x_i, \mathcal{S}) = \int \frac{\exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_j\right]}{\sum_{k=1}^{5} \exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_k\right]} P(W|\mathcal{S})dW \tag{4.8}$$

which we can compute as an expectation by sampling weights from the posterior, and classifying a test point by the model's modal prediction:

$$P(y = t_i|x_i, \mathcal{S}) = \mathbb{E}_{p(W|S)}\left[\frac{\exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_j\right]}{\sum_{k=1}^{5} \exp\left[\phi(\mathbf{x}_i) \cdot \mathbf{w}_k\right]}\right] \tag{4.9}$$

4.5 Data and models

Unlike other work on one-shot learning [Lake et al., 2011, Rezende et al., 2016], which has focussed on handwritten characters or faces, we plan to evaluate our technique on complex datasets with diverse categories and large intra-class variation. We focus on natural images of objects, as this follows our intuitive expectation of how to exploit the high level features present in images.

One of the key appeals of our approach, is that we exploit existing models and classification architectures. We use pretrained networks that are available freely online and we do not rely on access to their training data. This makes this technique more practical as training these models from scratch can take several weeks on high-end GPUs. For datasets where no pretrained models are available,we train our own ConvNets.

The models are trained on a large dataset D_{base} on a set of many object categories C_{base}. The ConvNet is then used as a fixed feature extractor to a softmax classifier. The pretraining dataset, if we have it, D_{base} is no longer needed, as a summary of that informatoin should be contained in the network weights.

We evaluate our model on a 5-way classification task of novel object categories C_{novel}. For each novel category, our model has access to only n positive examples, where $n \in \{1, 2, 5, 10, 20\}$.

4.5.1 CIFAR-100 dataset

The CIFAR-100 data set is a collection of 60,000 tiny (32 x 32px) images of 100 object categories. This is a dataset which easily fits in memory, yet is still challenging in terms of the high intra-class variance of the images.

We split this into 'CIFAR-80' and 'CIFAR-20' datasets by using the first 80 categories (in alphabetical order) as the pretraining set C_{base} and leaving the remaining 20 for use as novel categories C_{novel}. To create the five-way n-shot learning process C_{novel} is randomly sampled to select five categories and n training images.

As there weren't pretrained networks on CIFAR-80, we created our own, adapted from the 'quick' CIFAR-10 model distributed with Caffe [Jia et al., 2014]. This network consisted of three (conv-pool-relu) layers followed by two fully-connected

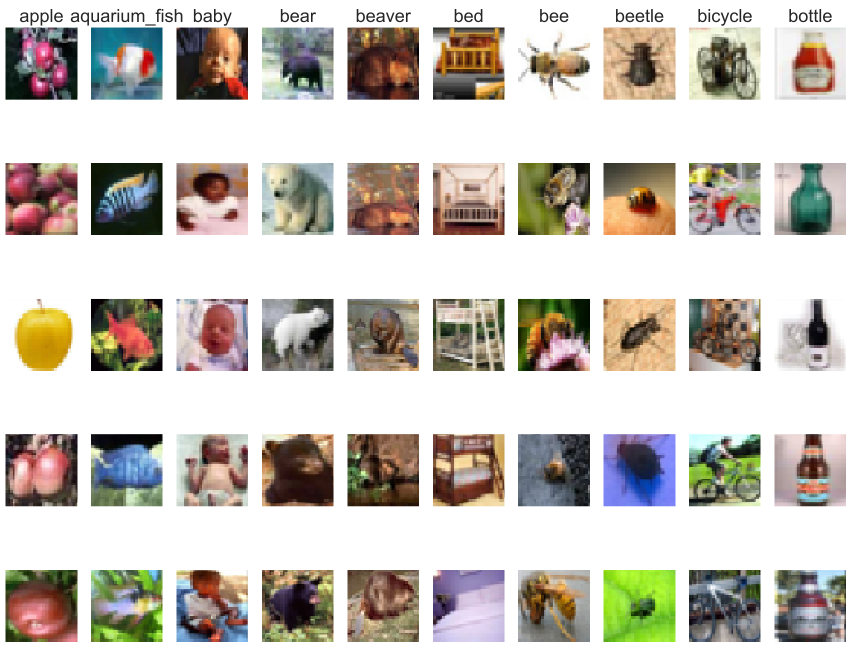


Figure 4.2: Samples of the first 10 categories in the CIFAR-100

linear layers and a softmax non-linearity. The final layer before the softmax has 64 units. The model was trained with 5/6ths of the data, without any augmentation, for 40,000 epochs. An L2 weight penalty of 0.004 was used. The model achieved 30.4% top-1 test accuracy.

4.5.2 ImageNet

ImageNet [Deng et al., 2009] is a collection of over 14 million images of around 22,000 'synsets' (synonym sets) that follow the lexical structure of WordNet [Fellbaum, 1998]. A 1000 category, 1.2 million image, subset has been used as part of the 'ImageNet classification challenge' since 2010.

Many networks pretrained on the classification dataset are publicly available. We use the VGG models from Oxford Visual Geometry Group because of their simple architecture and high performance. The original VGG 16-layer network (top-5 error rate of 7.4%) produces 4096-dimension CNN codes that are unwieldy for rapid iteration. We instead use a version that was modified so that the top-most hidden layer has only 128 units. Impressively, this still achieves a sub-10% error rate [Chatfield et al., 2014].

C_{base} consists of the 1000 ImageNet-1k categories. This is a diverse dataset, with images of foods, landscapes, man-made objects and animals. A significant proportion of the classes are breeds of dogs, meaning the pretrained network spends a lot of its capacity on fine-grain features to distinguish similar looking animals. The 1000 classes have been selected for mutual exclusiveness so that no synset is a subset of another (e.g. no 'mammal' which subsumes the many dog categories).

To create a set of unseen categories C_{novel}, the ImageNet API was queried for a list of synsets. Synsets present in the C_{base} were removed and the rest sampled at random and saved if they contained at least 100 valid, downloadable images. We saved 30 categories.

Care was intially not taken to ensure that these were leaf nodes in the synset hierarchy or that they were neither a parent or child node of C_{base} list. This means that some of the categories corresponding to quite visually nebulous concepts such 'greeter/saluter', 'insectivore' and 'melancholic'. This was initially left as it is as it seemed like a suitable challenge to attempt one-shot learning on difficult concepts. However, this essentially increased the variance of the results, without giving a clearer indication of the one-shot performance. Instead we tested on a hand-selected subset of these that refer to objects, such as 'spice rack', 'yak' and 'Dalai Lama'.

4.6 Information in the weights

One of our assumptions is that the pretrained model is an summary of everything learned so far, and that within the weights there is structure that can be exploited.

We apply hierarchical clustering to the weight vectors of W^*_{base}. To do, we use the 'bottom up' agglomerative approach that minimises the total within-cluster variance.

We start with singleton clusters for each categories weight vector w_i for $i \in C_{\text{base}}$. At each step we find the pair of clusters that when merged lead to minimum increase in total within-cluster variance. The distance metric is the weighted squared Euclidean distance between cluster centres.

Class	Class	Distance
`657 missile`	`744 projectile, missile`	0.1376
`248 Eskimo dog, husky`	`250 Siberian husky`	0.1672
`435 bathtub, bathing tub,`	`876 tub, vat`	0.1696
`066 horned viper, cerastes`	`068 sidewinder, horned rat`	0.1719
`482 cassette player`	`848 tape player`	0.1980
`461 breastplate, aegis, eg`	`524 cuirass`	0.2001
`265 toy poodle`	`266 miniature poodle`	0.2035
`836 sunglass`	`837 sunglasses, dark glass`	0.2297
`040 American chameleon, an`	`046 green lizard, Lacerta`	0.2311
`240 Appenzeller`	`241 EntleBucher`	0.2346
`638 maillot`	`639 maillot, tank suit`	0.2369
`987 corn`	`998 ear, spike, capitulum`	0.2532
`166 Walker hound, Walker f`	`167 English foxhound`	0.2540
`155 Shih-Tzu`	`204 Lhasa, Lhasa apso`	0.2550
`620 laptop, laptop compute`	`681 notebook, notebook com`	0.2551

Table 4.1: The closest 16 weight vectors by variance minimisation in VGGNet

$$d(u,v) = \sqrt{\frac{|v|+|s|}{|v|+|s|+|t|}d(v,s)^2 + \frac{|v|+|t|}{|v|+|s|+|t|}d(v,t)^2 - \frac{|v|}{|v|+|s|+|t|}d(s,t)^2} \tag{4.10}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster, and $|\cdot|$ gives the membership of the cluster.

The results of applying Equation 4.10 to the 1000 weight vectors of VGGNet can be seen in Table 4.1. This shows the ImageNet-1k categories that are closest together in weightspace. Evident is that i) we can discover similar objects by their high-level weight vectors ii) there is redundancy in the object categories.

If we depict the whole hierarchy as a dendrogram, we demonstrate that these weight vectors can provide a compelling taxonomy of visual objects. The result of of clustering CIFAR-80 trained weights is shown in Figure 4.3. As far as we're aware, this has never been shown explicitly before. The results from ImageNet trained model are even more compelling, and this is given in the Appendix A (this is large and best viewed on a screen). Each major branch is semantically meaningful. There is a branch for 'food', 'sealife', 'trees', 'vehicles', 'man made objects', 'natural landscapes' etc.

Clustering by alternative metrics, such as distance between centroids, or by nearest points, did not create a compelling taxonomy.

Our initial experiment is to model all the weights with a Gaussian distribution, $W^*_{\text{base}} \sim \mathcal{N}(\mu, \Sigma)$. We can later make this more specific by selecting an appropriate subset of the weights.

By cutting the hierarchical tree at a given point we can form an arbitrary number of 'supercategories'. Projecting these into two-dimensions using Fisher's Linear Discriminant Analysis, we can get a feel for how separable these clusters are and therefore how distinct and useful it could be to use a supercategory based prior. LDA will seek the projection which to maximises inter-cluster variance, while minimising intra-cluster variance. Splitting the CIFAR-80 weights into 8 supercategories, and projecting into two-dimensions gives Figure 4.4. We can identify semantic groups corresponding to "trees", "man made objects", "landscapes" etc. which should form useful supercategories.

4.6.1 Supercategory assignment

Having discovered this structure in the pre-trained weights, it would be helpful for one-shot learning if we could use it to inform our prior. For example, if a novel class was 'birch tree', and we could estimate that it likely belongs to the "trees" supercategory, we could use $W^*_{\text{trees} \in C_{\text{base}}}$ as the basis for our prior. As a result, this prior could be more specific and provide a stronger inductive bias that hopefully brings the classifier into good parameter settings with less data.

We set this up as a hierarchical Bayesian model with a 'prior on the prior', however, in the interest of time, we implement it with a non-Bayesian heuristic.

We investigate 'soft' and 'hard' supercategory assignment for our training data $\mathcal{D}_{\text{novel}}$ to supplement each training point $\{x_i, t_i\}$ with its supercategory distribution/assignment z_i.

We set the prior on z to be the empirical membership size of the supercategories. We approximate each supercategory's weight distribution $\mathbf{w}_{C_i}$ as a point-estimate at its centroid. In the soft assignment we update the 'posterior' supercategory dis-

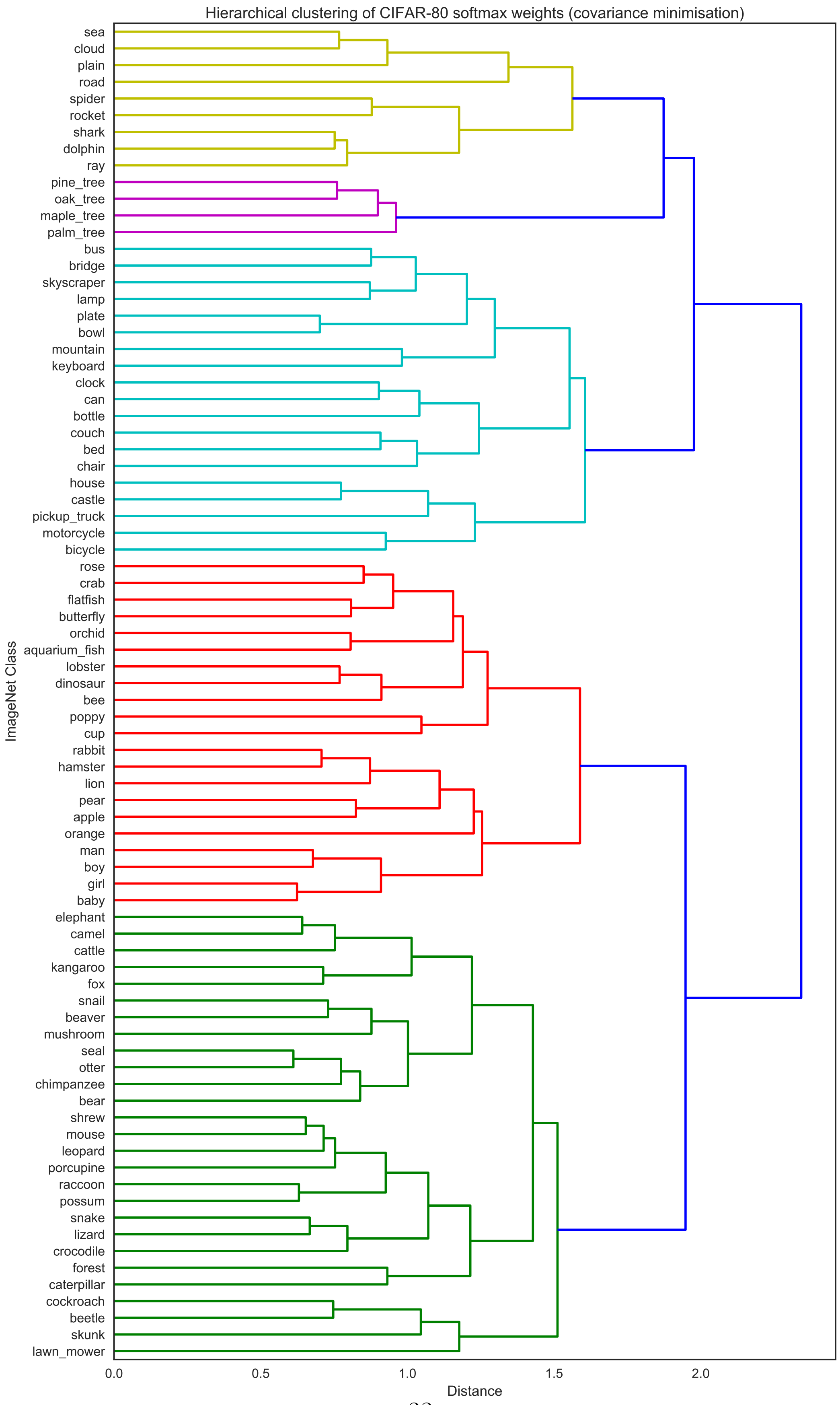


Figure 4.3: Hierarchical clustering of trained network weights

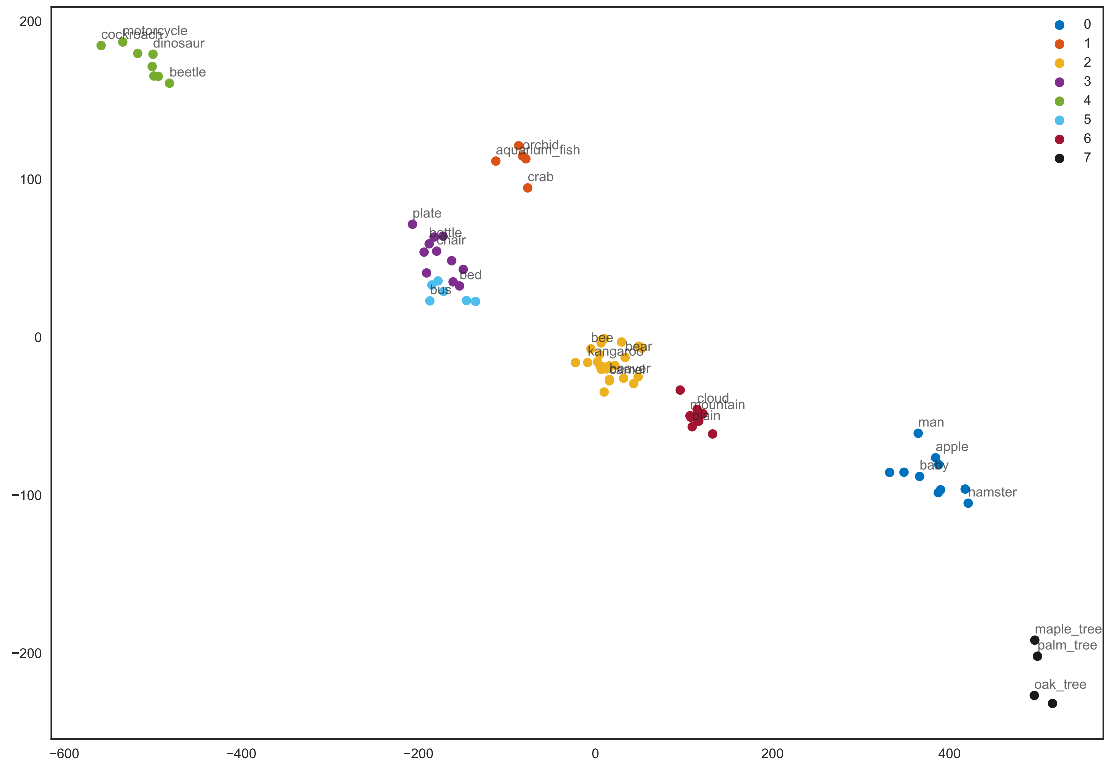


Figure 4.4: Fisher's Linear Discriminant Analysis of the 8 superclusters in *weight-space* as determined by cutting the hierarchical clustering tree. Separable semantic supercategories corresponding to "trees", "man made objects", "landscapes" are identifiable.

tribution as the prior multiplied by the likelihood of the data under that supercategory's mean weightvector $p(z_i|\{x_i, t_i\}) = p(\{x_i, t_i\}|z_i)p(z_i) = P(y = t_i|\mathbf{w}_{C_i})p(z_i)$ and renormalise so that it sums to one. In the hard assignment we simple select the 'MAP' supercategory – the supercategory whose mean weight, scaled by its membership size, gives the highest output score for that example. Where there are more than one training example, the average scores are used.

The prior on the weights for our 'hierarchical' n-shot learning is then:

$$p(\mathbf{w}_i) = \mathcal{N}(\mathbf{w}_i|\bar{\mu}_{\mathbf{z_i}}, \bar{\mathbf{\Sigma}}_{\mathbf{z_i}} + \lambda\mathbb{I})$$

Where $\bar{\mu}_{\mathbf{z_i}}$ and $\bar{\mathbf{\Sigma}}_{\mathbf{z_i}}$ are the sample means and covariance matrices for the weights of W^*_{base} weighted by the supercategory assignment z_i.

In the following experiments we investigate the effect of using this more specific prior combined the regularising factor λ. Our prior on $\mathbf{w}_i$ lies within the convex hull of W^*_{base}. If the optimal weight for the novel task also lies on that space, we should expect rapid learning. If it is outside, or if our use of a narrow prior confines the weights to the wrong area, then we should expect poor performance and the benefit of using a large regularising factor.

Chapter 5

Evaluation

We evaluate the models on a five-way classification task of CIFAR-100 and ImageNet categories. We compare the performance after n training examples, and with a prior on the weights based on the entirety of the pretrained W^*_{base} or a subset determined by an inferred 'supercategory' assignment. We test, arbitrarily, the performance when there are 1, 2, 3, 5 or 10 supercategory clusters and the effect of the prior regularisation parameter λ.

Five classes were selected at random from the C_{novel}, and $n = \{1, 2, 5, 10, 20\}$ training examples were sampled for n-shot learning evaluation. This process was repeated at least 20 times for each experiment. For CIFAR-100, dataset variation accounts for around 8 percentage points standard deviation as determined by comparing methods. This is an artefact of sampling few data points from a complex data set which has significant intra-class variance.

Research on one-shot learning has not yet coalesced around a shared evaluation metric, making it hard to benchmark techniques. We follow the test proceedure recommended in Vinyals et al. [2016] as best we can.

5.1 Approximate inference method

To estimate the posterior weight distribution, we tried a variational method and an MCMC sampling method.

The posterior weights were evaluated through 30 000 iterations of the ADVI algorithm implemented in PyMC3. The posterior predictive distribution was found by sampling 500 times from the posterior weight distribution, running through the softmax classifier, and selecting the modal class prediction. Each learning and testing episode took around 30 seconds on a CPU.

For MCMC based sampling we ran 2000 samples of the Hamiltonian-MC based No U-Turn Sampler [Hoffman and Gelman, 2014] which automatically selects an appropriate step size. Each learning/testing episode took around 45 seconds when we used a broad prior distribution ($\lambda = 0.1$) and up to 300 seconds when we used of a narrow prior distribution ($\lambda = 0.0001$), presumably caused by a low acceptance-rate of samples.

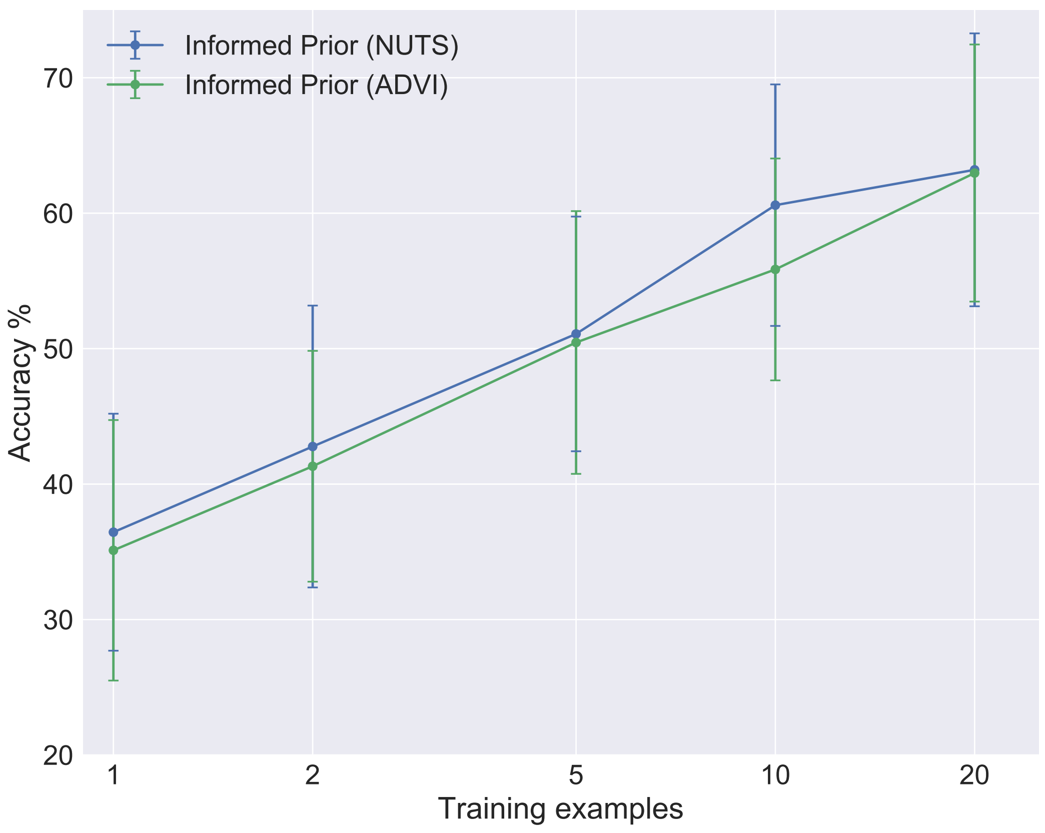
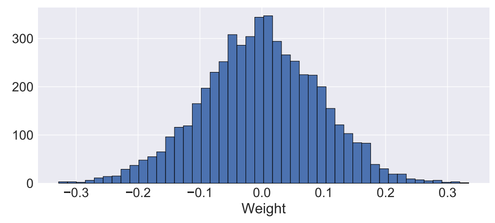
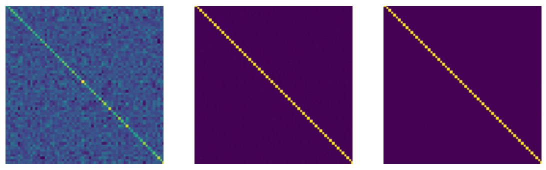


Figure 5.1: MCMC sampling takes up to six times longer than the variational approximation but enabled the best n-shot performance. The VI method uses a mean-field approximation which likely the caused the weaker performance. (Error bars give 1 s.d.)

Figure 5.1 shows the performance of both approximations. NUTS consistently

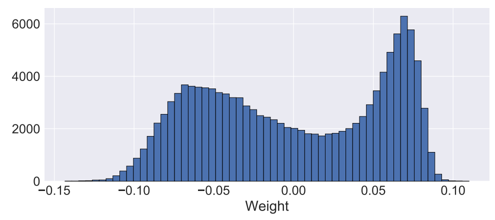


(a) Histogram of all weights

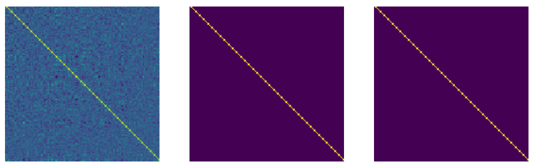


(b) Covariance matrix with $\lambda = \{0.0001, 0.001, 0.01\}$

Figure 5.2: $W^*_{\text{CIFAR-80}}$ weight distribution



(a) Histogram of all weights



(b) Covariance matrix with $\lambda = \{0.0001, 0.001, 0.01\}$

Figure 5.3: W^*_{VGG} weight distribution

outperformed ADVI, although took at least twice as long. ADVI uses a mean-field approximation, so that the posterior is a diagonal Gaussian. It is apparent that this approximation is a significant detriment to competitive one-shot learning. A full-rank implementation of the algorithm exists in Stan, which should be explored in future work as it has the potential of being suitable accurate, while still being reasonably fast.

5.2 Strength of prior

Recall from Equation 4.3, that we add a regularisation term λ to the covariance matrix of our prior. This term has the effect of making the prior weaker and more spherical. For both CIFAR-80 and ImageNet, most weights W^*_{base} lie in a small range around zero (see Figure 5.2a) with standard deviation around 0.1 and 0.05 respectively. Including a a prior larger than this reduces the influence of W^*_{base}, and the MAP solution will tend to the MLE solution. Furthermore, regularising beyond a small amount (0.0001) effectively removes any subtle covariance structure (see Figure 5.2b) although the relative widths in the different dimensions are still present.

The weight distribution in VGGNet Figure 5.3a follows a non-Gaussian distribution. This is likely due to the use of dropout, which encourages sparse representations and less co-dependence of weights.

For the CIFAR-20 dataset, we ran n-shot learning with $\lambda = \{1, 0.1, 0.01, 0.001, 0, 0001\}$ for the weight prior based on all of W^*_{base}, ten times each (Figure 5.4). No statistically significant effect was found as the variance is so large. However, it appears that an overly precise prior degraded performance, especially after 20 examples when the data has strong opinions about what the weights should be.

Note that for these experiments, the posterior was approximated using ADVI – which makes a mean-field approximation and (incorrectly) models the posteriors as a Gaussian with diagonal covariance matrix. This potentially makes our efforts to model the covariance structure redundant. Fewer tests were run using the MCMC sampler, but no obvious dependence between regularisation and performance was suggested. Middling values such as $\lambda = 0.01$ generally did well.

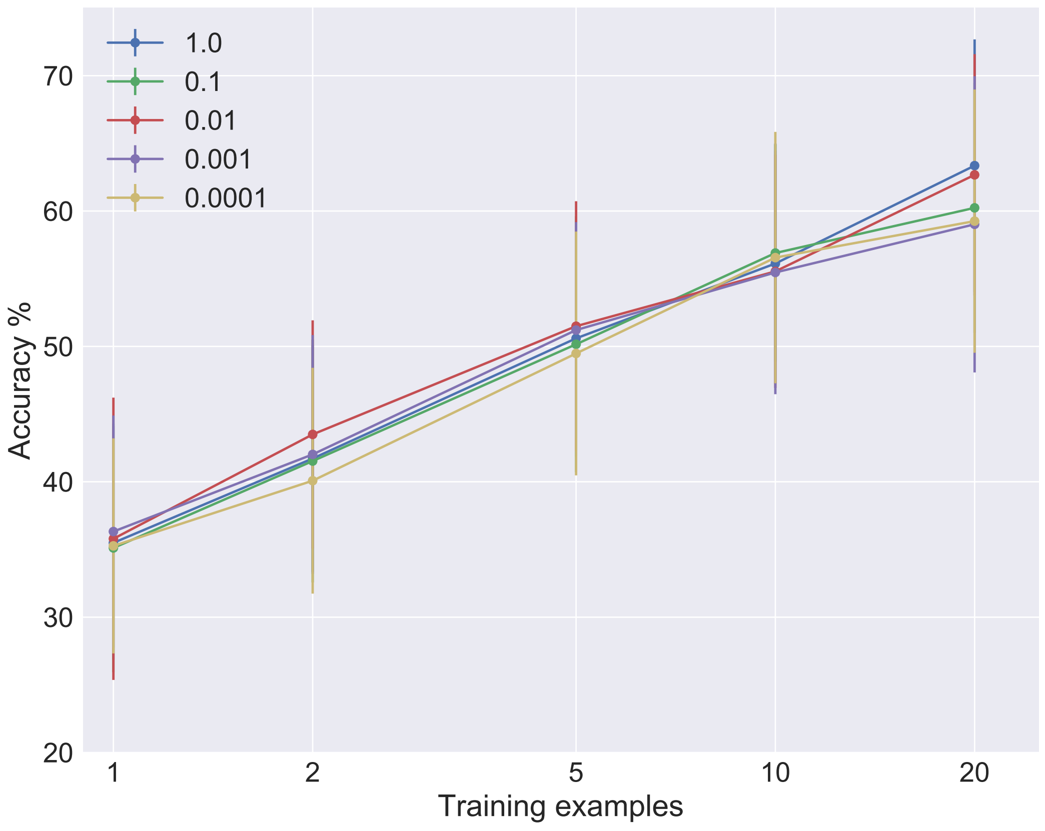


Figure 5.4: Performance is not strongly dependent on the regularisation of prior. (Error bars give 1 s.d.)

5.3 Performance comparison

We compare our performance against alternative approaches for n-shot learning (see Figure 5.5):

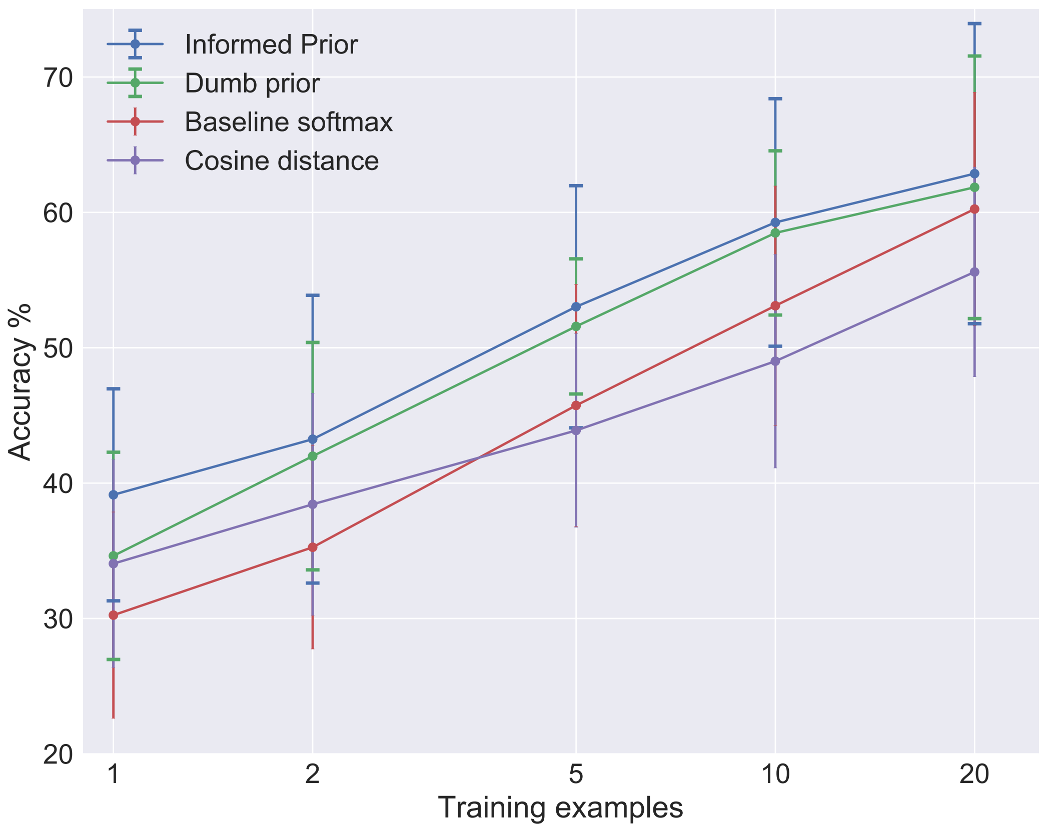


Figure 5.5: Results on a 5-way classification task of unseen categories in CIFAR-100 results (Error bars give 1 s.d.)

- '**Informed Prior**': this is our method of placing a multivariate Gaussian prior on the network weights of a softmax classifier. "(ADVI)" or "(NUTS)" refer to the variational and MCMC methods used to approximate the posterior.
- '**Dumb prior**': As above but fitting a spherical zero-centred Gaussian on the network weights, while still being Bayesian.
- '**Baseline softmax**': a standard softmax classifier initialised with Gaussian noise and optimised through gradient descent. L2 regularisation of 0.001 was applied.

Classifier	**1**	**2**	**5**	**10**	**20**
SGD	30.2	35.3	45.7	53.1	60.2
Cosine	34.1	38.4	43.9	49.0	55.6
Dumb prior (NUTS)	34.6	42.0	51.6	58.5	61.9
Informed prior (ADVI)	35.1	41.3	50.5	55.8	**63.0**
Informed prior (NUTS)	**39.1**	**43.2**	**53.0**	**59.3**	**62.9**

Table 5.1: CIFAR-20 n-shot accuracies.

- '**Cosine**': Nearest neighbour classifier on the extracted features of the images based on cosine distance.

In this 5-way task, a chance classifier would achieve 20% accuracy. Training a nearest neighbour classifer using cosine distance of the extracted features $\phi(x)$ achieves 34%. This is quite impressive after a single example. In contrast, training a softmax classifier by gradient descent, *with regularisation*, only manages 30%. Our one-shot learning method, achieves almost 40% accuracy on this challenging task.

Figure 5.5 shows that our *informed* prior is essentially the cause of the improvement in one-shot learning. This prior loosely constrains the weight distribution to a sensible area of weightspace, which in the absence of much data, significantly aids performance. The prior distributions become less important in the presence of more data and we can see the 'dumb' zero-centred Gaussian prior and our informed prior converging to similar solutions. Within the 20 training examples, the informed prior still maintains a performance improvement over all other methods.

The simple nearest neighbour classifier using extracted features is high baseline to beat, which some one-shot learning algorithms have been noted as failing to do so [Rippel et al., 2016]. However, as more data becomes available, this simple non-parametric method fails to make full use of it, and its relative performance declines.

We can compare our method against the 'Matching Networks' method proposed by Vinyals et al.. Although the datasets differ in source[1], the training/test procedure is comparable. Using a fixed feature extractor, a cosine distance metric on their dataset achieves 36.6% after one shot and their 'matching nets' achieve 41.2%.

[1]They make a new dataset derivied from ImageNet, *mini*ImageNet which consists of 100 classes, 600 examples each (like CIFAR-100), of 84 x 84 pixel images (bigger than CIFAR-100) which they split this 80/20 into base/novel classes (as we do).

On our dataset, cosine distance gets 34.1% and our 'informed prior' softmax gets 39.1%; a 15% increase which compares well to their 12.6%. After 5-shots, their model achieves 13.5% more than cosine distance whereas ours improves it by 18%.

However, part of their method is to fine-tune their feature extractors to purposefully embed differing categories further apart. This improves the performance of their cosine classifier and their model and they achieve an impressive 60% accuracy in 5-shots.

We weren't able to compare against other papers as most choose their own training sets. However, the matching nets were mentioned to outperform Siamese Nets [Koch et al., 2015] and so it is encouraging to see that our approach is competitive with the state-of-the-art.

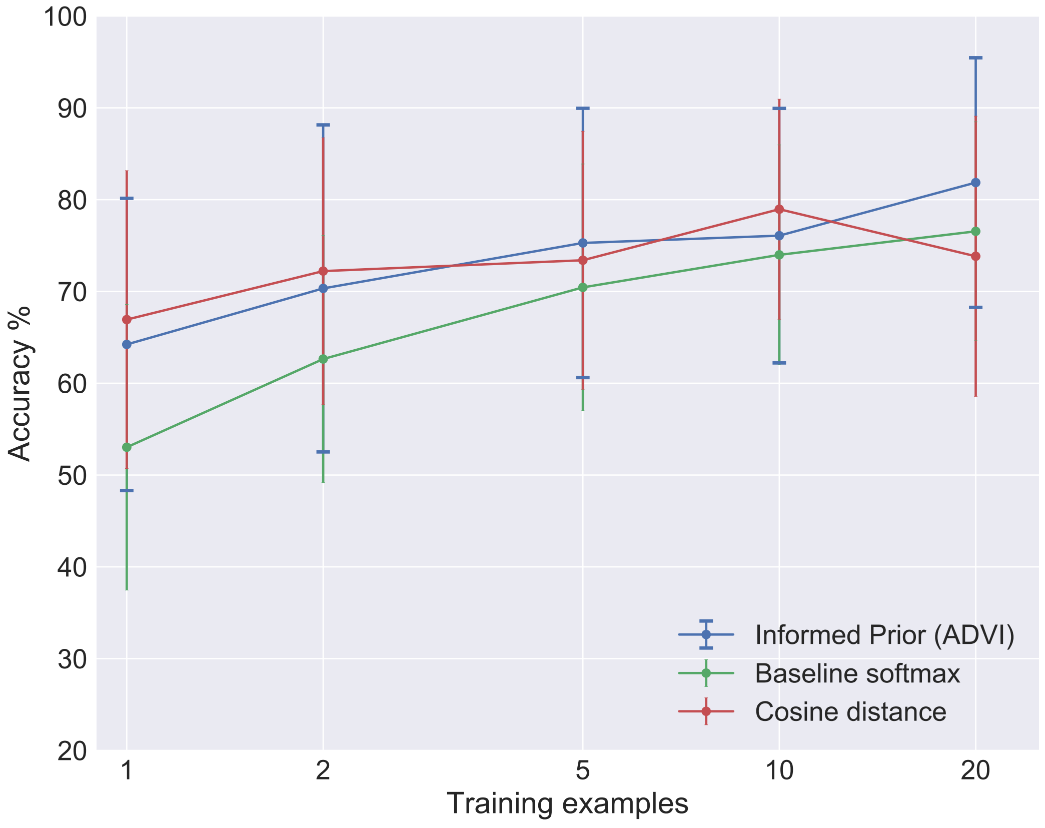


Figure 5.6: ImageNet Performance. Results were more variable than CIFAR-100, because of greater variance in the data set. The Informed prior (our technique) used the variational mean-field approximation. Performance would likely be improved from (slower) MCMC sampling. (Error bars are 1 s.d.)

The performance on our ImageNet dataset (Figure 5.6) is harder to interpret. For a start, the data is more variable, meaning more evaluations are required to asses relative performance of methods. Secondly, we did not run MCMC sampler on this dataset due to time constraints. This was shown to improve performance on CIFAR-20 by up to 4%. Still we can see that our performance is similar to the cosine metric after 1-shot, and outperforms it after 20-shots.

5.4 Supercategory Priors

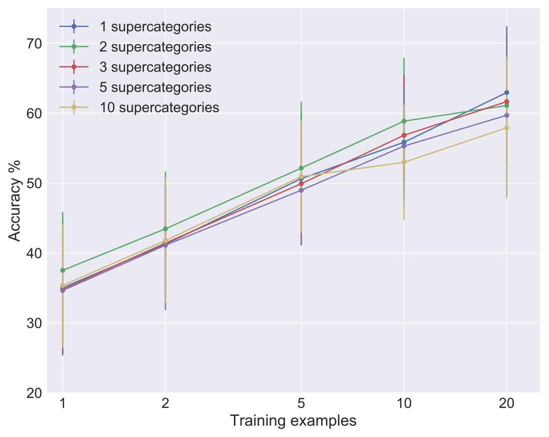
Having discovered the rich visual structure present in a trained network weights in section 4.6 we would like to use this to inform a specific prior over the network weights for a new class.

We split the W^*_{base} into $\{1, 2, 3, 5 \text{ or } 10\}$ clusters. For a new training example we infer which supercategory it likely belongs to. We then use the subset of weights in that supercategory to inform our prior on the weights.

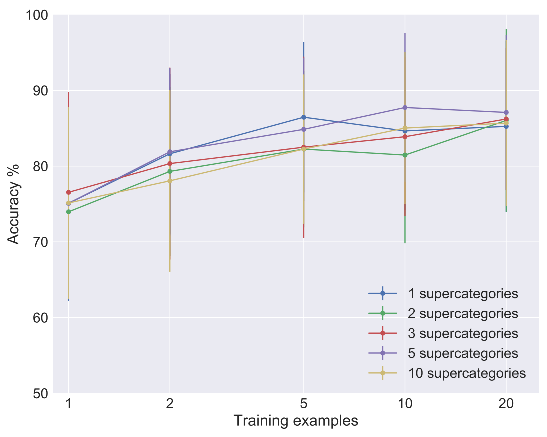
Results are shown in Figure 5.7a on the CIFAR-20 dataset. In this example we did 'hard' assignment to a cluster. Our heuristic based soft assignment was under-explored sufficiently to determine if it was better. In this instance there is an improvement when splitting W^*_{base} into two, but no meaningful improvement beyond that. We expect this is because it reasonably easy to assign correctly between the two most distant clusters, and this more specific prior aids learning. With ImageNet (Figure 5.7b) the results are too noisy.

It appears that although there is a potential benefit of having a more specific prior (Figure 5.8, there is a significant cost if that prior is wrong. I suspect that with our heuristic approach to assignment, and the inherent variability of the data, incorrect assignments had an overall detrimental affect on performance.

Small scale tests with 'oracle' assignment were performed and suggested a small increase in performance was possible. As proof of concept, we performed hard assignment for a given training example when given access to all the training examples of that class. This hard assignment was less likely to be wrong. The few experiments of this suggested a minor performance increase was possible (1-2 percentage points) however not enough runs were performed to be confident of this improvement.



(a) CIFAR-20



(b) ImageNet

Figure 5.7: Splitting W^*_{base} into N supercategories to inform a more specific prior did not meaningfully improve performance, presumably from difficulty assigning clusters. (Error bars a 1.s.d.)

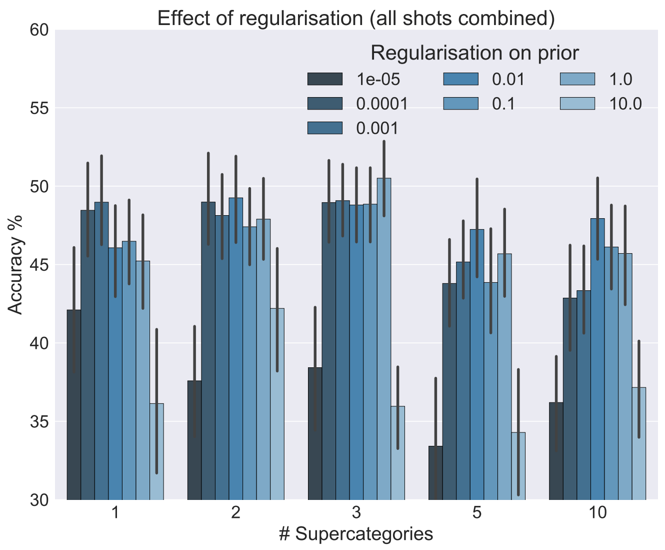


Figure 5.8: Very broad, or very specific priors had a negative effect on performance.(Error bars are 1 s.d.)

Chapter 6

Summary and Conclusions

One-shot learning continues to be a challenging problem for machine learning models. Non-parametric methods (combined with deep non-linear embeddings) often perform best in this low data regime. However, these methods become relatively weak if more data becomes available. Our approach, which is presentated as a Bayesian learning procedure on a pretrained convnet, seemingly combines the best of both. After a single data example, our 'informed' prior enables the classifier to outperform or match models specialised for this task. When more data is observed, the distribution over the weights collapses to a point mass and we are left with precisely the same model as a well-trained deep convolutional network.

Our work has demonstrated not only the benefits of being Bayesian, but that a significant performance improvement is possible by expressing our belief over what the new parameters might be. This prior is currently informed by existing weights trained on separate categories of the same dataset. This should be effective as long there's reason to believe that there is similarity in the tasks, such that the extracted image features are useful and the optimal new weights are not far from the convex bowl of the existing weights.

Networks pretrained on ImageNet, will likely extend to most new tasks on natural images using our technique. However, natural images only occupy a small space in the possible pixel values of an image[1]. Extending this task to 'unnatural' generated images would likely require more data. However once the weight-space occupied by

[1]This forms the basis of how adversarial images can be generated - resting in pockets of image space that the network has not seen.

the model increases, so does our ability to perform one-shot learning on a broader range of tasks.

As the volume occupied by a the networks weights increases, a global prior on new weights would become increasingly generic. Our work into clustering weightvectors into supercategories is a promising technique to inform more specific priors. Our demonstration that the weights in a network can be clustered to form an unsupervised visual taxonomy is highly compelling, however, further work is needed to best exploit it.

Our heuristic based attempt to assign supercategories was disappointing. We performed hard assignment of an training point based on its likelihood under the weight supercategory centroid. Presumably this frequently got it wrong and led to a decrease in performance. Modelling each supercluster as a Gaussian, or testing against individual weight vectors would lead to a better result. Our initial investigations with 'oracle' assignment suggest that a 2-4 percentage-point increase might be possible. A similar approach would be to model the weight distribution over W^*_{base} as a multivariate Gaussian. With K Gaussians this will, at most, be K times more expensive to evaluate.

Similar to [Salakhutdinov et al., 2013], we could place a Dirichlet Process prior on the weight space to automatically infer the number of supercategory clusters. However, it is not worth doing so until we find a performance increase from clustering.

A possible performance improvement may be possible from regularising the extracted activation *features*. Hariharan and Girshick [2016] says linear classifiers are particularly sensitive to the normalisation of the data and Chatfield et al. [2014] note that L2-normalisation of the features accounted for up to 5% of their performance increase over VOC 2007. We could implement this by simply switching to a ConvNet with BatchNorm layers.

For future work, it would be practically useful to implement this technique as an online learning algorithm. As a use case, imagine a mobile device or robot pre-trained for object recognition on a set number of categories. We would like it to learn a new face or a new object after giving it an example. We could achieve this by expanding the model and using the Bayesian prior developed in this work. Maintaining distributions over weights which have to be marginalised out is computationally expensive, so once further data has caused the distribution has become sufficiently peaked, we should switch to a point-estimate and the frequentist rather than Bayesian domain.

As a final note, one of the reasons given for exploring this topic was the fact that humans consistently perform one-shot learning. Aiming to tackle machine learning tasks with as little training data as people need is a challenging and worthwhile task. Although biological plausibility should not be a constraint to our exploration (as it isn't here) developing learning algorithms that learn and think like a human is a high bar worth reaching for. By doing so, we not only develop more powerful algorithms, but we may also understand the human mind. We hope our demonstration of one-shot learning spurs further research into this challenging and practical problem.

Appendix A

Visual taxonomy of ImageNet

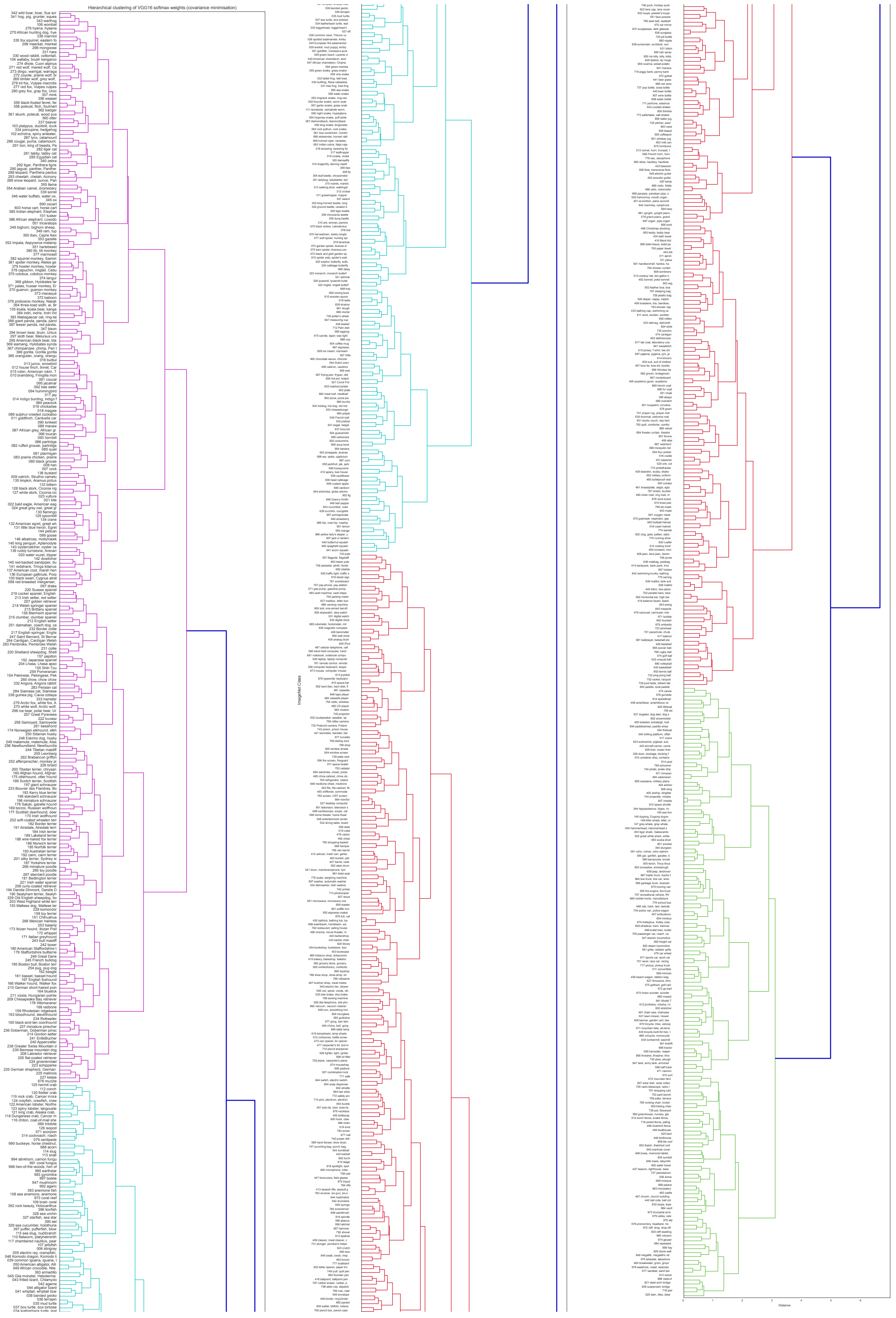


Figure A.1: Hierarchical clustering of trained network weights from VGGNet (best viewed on a screen, zoomed in.)

Bibliography

Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, I:672–679, 2005. ISSN 1063-6919. doi: 10.1109/CVPR.2005.117.

Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4 of *Information science and statistics*. Springer, 2006. ISBN 9780387310732. doi: 10.1117/1.2819119. URL http://www.library.wisc.edu/selectedtocs/bg0137.pdf.

Paul Bloom. Precis of how children learn the meaning of words. *Behavior and Brain Sciences*, 24:1095–1103, 2001. ISSN 0140-525X. doi: 10.1353/lan.2005.0073.

Susan Carey and Elsa Bartlett. Acquiring a single new word. *Papers and Reports on Child Language Development*, 15(August):17–29, 1978. URL http://www.mendeley.com/research/acquiring-single-new-word-1/.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv preprint arXiv: . . .*, pages 1–11, 2014. doi: 10.5244/C.28.6. URL http://arxiv.org/abs/1405.3531.

Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv*, pages 1–14, 2014. ISSN 10495258. URL http://arxiv.org/abs/1406.2572.

Jia Deng Jia Deng, Wei Dong Wei Dong, R. Socher, Li-Jia Li Li-Jia Li, Kai Li Kai Li, and Li Fei-Fei Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2–9, 2009. ISSN 1063-6919. doi: 10.1109/CVPR.2009.5206848.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. ISSN 01628828. doi: 10.1109/TPAMI.2006.79.

Li Fe-Fei Li Fei-Fei, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. *Proceedings Ninth IEEE Interna-*

tional Conference on Computer Vision, pages 0–7, 2003. doi: 10.1109/ICCV.2003.1238476.

Christiane Fellbaum. WordNet: An Electronic Lexical Database, 1998. ISSN 17508460.

Yarin Gal and Zoubin Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. page 12, jun 2015. URL http://arxiv.org/abs/1506.02158.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 9:249–256, 2010.

Bharath Hariharan and Ross Girshick. Low-shot visual object recognition. pages 1–10, 2016. URL http://arxiv.org/abs/1606.02819.

Johan Hastad. Almost Optimal Lower Bounds for Small Depth Circuits. *STOC '86 Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986. doi: 10.1145/12130.12132.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Arxiv.Org*, 7(3):171–180, 2015. ISSN 1664-1078. doi: 10.3389/fpsyg.2013.00124. URL http://arxiv.org/pdf/1512.03385v1.pdf.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *Ieee Signal Processing Magazine*, (November):82–97, 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2205597.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–54, 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL http://www.ncbi.nlm.nih.gov/pubmed/16764513.

Judy Hoffman, Eric Tzeng, and Jeff Donahue. One-Shot Adaptation of Supervised Deep Convolutional Models. *arXiv preprint arXiv: ...*, pages 1–10, 2013. URL http://arxiv.org/pdf/1312.6204v2.pdf$\delimiter"026E30F$nhttp://arxiv.org/abs/1312.6204.

Md Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(2008):30, 2014. ISSN 15337928.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, pages 1–11, 2015. ISSN 0717-6163. doi: 10.1007/s13398-014-0173-7.2. URL http://arxiv.org/abs/1502.03167.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014. ISSN 10636919. doi: 10.1145/2647868.2654889. URL http://arxiv.org/abs/1408.5093.

Andrei Karparthy. What I learned from competing against a ConvNet on ImageNet, 2014. URL https://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition. 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

Alp Kucukelbir, Andrew Gelman, and David M Blei. Automatic Differentiation Variational Inference. pages 1–38, 2016.

Brenden M Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B Tenenbaum. One shot learning of simple visual concepts. *Proceedings of the 33rd Annual Conference of the Cognitive Science Society (CogSci 2011)*, pages 2568–2573, 2011.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, dec 2015. URL http://science.sciencemag.org/content/350/6266/1332.abstract.

Yann LeCun, L??on Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791.

David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *Iclr*, pages 1–8, 2015. URL http://arxiv.org/abs/1511.06422.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the Number of Linear Regions of Deep Neural Networks . *Nips*, pages 1–12, 2014. ISSN 10495258.

Maxime Oquab, Maxime Oquab, Ivan Laptev, Josef Sivic Learning, Transferring Mid-level, Maxime Oquab, and Leon Bottou. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks To cite this version : Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. 2014.

Danilo J Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-Shot Generalization in Deep Generative Models. 2016.

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric Learning with Adaptive Density Discrimination. *International Conference on Learning Representations (ICLR)*, (Dml):1–15, 2016. URL http://arxiv.org/abs/1511.05939.

F Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519.

R. Salakhutdinov. Learning Deep Generative Models. *Mit.Edu*, pages 1–84, 2009. ISSN 2326-8298. doi: 10.1146/annurev-statistics-010814-020120. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Learning+deep+generative+models{#}0.

Ruslan Salakhutdinov, Joshua B Tenenbaum, and Antonio Torralba. Learning with Hierarchical-Deep Models. 35(X):1–14, 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.269.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961. URL http://dx.doi.org/10.1038/nature16961.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014. ISSN 15337928. doi: 10.1214/12-AOS1000.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. 2016. URL http://arxiv.org/abs/1606.04080.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems 27 (Proceedings of NIPS)*, 27:1–9, 2014. URL http://arxiv.org/abs/1411.1792.