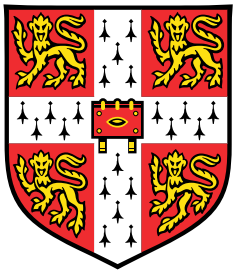
Bayesian Neural Networks for K-Shot Learning



Jakub Bartłomiej Świątkowski

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Homerton College

August 2017

Declaration

I, Jakub Bartłomiej Świątkowski, of Homerton College, being a candidate for the M.Phil in Machine Learning, Speech and Language Technology, hereby declare that this thesis and the work described in it are my own work, unaided except as may be specified below, and that the thesis does not contain material that has already been used to any substantial extent for a comparable purpose. This thesis contains 14119 words.

Jakub Bartłomiej Świątkowski
August 2017

Acknowledgements

I would like to thank my supervisor Dr. Richard Turner for his guidance and support that has been invaluable to me.

I also thank Matthias Bauer and Mateo Rojas for the fruitful collaboration and their support during this project.

Abstract

This thesis introduces multiple extensions and improvements to the probabilistic framework for k-shot image classification proposed by Bauer et al. [2017]. The aim of their framework is to transfer knowledge from a classifier trained on a large dataset to new classes with few training examples. We address three key considerations in this setting: the effectiveness of the weights-based concept transfer, the ability of the classifier to transform the feature space, and the influence of the feature-based representational transfer. More precisely, we propose a multitask training procedure for robust concept estimation, develop a probabilistic framework for non-linear neural network classifiers, and analyse the impact of a bottleneck layer on the model performance respectively. The proposed extensions show improvement over the results from the original framework in terms of both effectiveness of the concept transfer and the final k-shot learning performance.

Table of contents

Chapter 1

Introduction

1.1 Motivation

Humans are able to rapidly learn to recognise new categories of objects even after observing a single or a handful of examples from the new category. On the other hand, current state-of-the-art deep learning techniques require large number of labeled examples and long training times to achieve similar recognition capabilities. These techniques also face difficulties when the data is scarce or when they need to adapt to a new task.

The extraordinary learning ability of humans can be partially attributed to the prior knowledge gained through-out a person's life. Such knowledge is applied to efficiently learn a new task at hand rather than learning from scratch. For instance, children learn to rapidly recognise new classes of objects by accumulating knowledge about the world [Bloom, 2000].

The intuitive interpretation is that we use the prior knowledge in two ways. First, by observing large number of objects we learn to extract high-level features that allow us to efficiently and accurately recognise objects. The high-level features can represent different shapes, material structures etc. Second, we learn concepts that model our knowledge about the world and describe which high-level features or other concepts are likely to co-occur together. For instance, a dog is likely to have a furry skin and co-occur together with concepts of a paw and a tail.

This prior knowledge allows us to efficiently generalise from even a single example of a new class to new examples of this class. This is supported by the ability to infer missing information about an object based on other similar classes. For instance, after observing just a head of a new breed of dog, we expect it will also have four paws and a tail.

The motivation of this work is to apply the described intuition to construct a learning algorithm that performs well both in high and low data regime. We build on top of a powerful, but simple, general and extendible framework for k-shot learning proposed in Bauer et al. [2017]. K-shot learning has recently received increased attention from the machine learning community [Mishra et al., 2017; Qiao et al., 2017; Ravi and Larochelle, 2017; Snell et al., 2017; Vinyals et al., 2016] followed by the inspiring work of Lake et al. [2015]. While many of the recently proposed methods focus on specific tasks or tailor for particular test scenario, the motivation here is to propose a general and extensible approach.

We aim to combine the strengths of deep learning methods with probabilistic modelling by extending the k-shot image classification framework from Bauer et al. [2017]. While current deep learning techniques for object classification such as Convolutional Neural Networks (CNN) achieve above-human performance on large datasets, they often fail when given just a few data points. On the other hand, probabilistic modelling is particularly well suited for low data regime, by providing informative priors and well calibrated uncertainty estimates.

In this work we analyse different ways to perform the most informative knowledge transfer from the large data setting to the low data setting using probabilistic modelling.

1.2 K-shot learning: introduction to the task

1.2.1 K-shot learning task

K-shot learning is about learning from extremely small number k of labelled examples (shots). We consider a discriminative k-shot learning task for classification of images. We first train a model on a large dataset $\widetilde{\mathcal{D}} = \{\widetilde{\mathbf{u}}_i, \widetilde{y}_i\}_{i=1}^{\widetilde{N}}$ of images $\widetilde{\mathbf{u}}_i$ and labels $\widetilde{y}_i \in \{1, \ldots, \widetilde{C}\}$ that indicate which of the $\widetilde{C}$ classes each image belongs to. Then, using knowledge from the model trained on the large dataset, we perform k-shot learning with a small dataset $\mathcal{D} = \{\mathbf{u}_i, y_i\}_{i=1}^{N}$ with C new classes, labels $y_i \in \{\widetilde{C}+1, \widetilde{C}+C\}$, and k images from each new class. During test time we classify unseen images $\mathbf{u}^*$ from the new classes C and evaluate the predictions against ground truth labels y^*.

The knowledge transfer during the k-shot learning time is supported by probabilistic models trained on both $\widetilde{\mathcal{D}}$ and $\mathcal{D}$. One of the objectives is to find the models that perform best in transferring the concept knowledge.

	Few shots	**Many shots**
Batch	Multitask learning with few shots [Srivastava and Salakhutdinov, 2013]	Multitask learning Bakker and Heskes [2003] Caruana [1998]
Online	K-shot learning Bauer et al. [2017] Vinyals et al. [2016] Ravi and Larochelle [2017] Snell et al. [2017] Qiao et al. [2017] Mishra et al. [2017]	Transfer learning Pan and Yang [2010] Weiss et al. [2016]

Table 1.1 K-shot learning relatively to other learning paradigms. Batch: learning all tasks at the same time. Online: learning tasks sequentially. Few shot: learning with just few examples. Many shots: learning not constrained with low number of examples.

1.2.2 Comparison to alternative learning paradigms

Our approach to k-shot learning is similar in nature to other related learning paradigms. We aim to clarify the distinction between the different paradigms. Table 1.1 provides a structured view that situates the described k-shot learning approach relatively to the other learning paradigms. The distinction is made based on the learning procedure (batch or online) and the number of training examples (few or many shots). Batch algorithms learn many tasks at the same time from scratch and share knowledge during training. On the other hand, online algorithms learn the tasks sequentially. This means that consecutive tasks can reuse the knowledge learned from already trained tasks rather than having to start training from scratch. This type of learning methods is different from active learning [Settles, 2010] because we do not update the original model after learning the new task. Orthogonal distinction between the learning paradigms is made based on the number of training examples used to train the tasks. While some methods are designed to work well with small data (few shots), other methods do not adhere to such constraint (many shots).

The k-shot learning approach discussed in this work aims to use few data points during k-shot learning and trains models sequentially. More precisely, we first train a model on the large dataset $\widetilde{C}$, and then transfer the gained knowledge to the small dataset C.

1.3 Why is k-shot learning hard?

K-shot learning with just a handful of examples poses many problems that either do not exist or are less critical in the larger data settings. We can speculate that the additional difficulty of the k-shot learning problems can be seen as one of the reasons why research related to algorithms that operate in the low data regime is gaining traction only recently.

1.3.1 Why is it hard?

The diversity of examples within a single class can be very high. For instance, Fig. 1.1 shows examples of objects that belong to the same class, but are highly diverse. For this reason, classification is often a challenging problem even with a large number of training examples. In k-shot learning, where the learning algorithm is presented with just a few examples for each class, this problem becomes even more severe. With just k training examples, complex models will likely overfit the training set and generalise poorly to the test set. Furthermore, the lower number of examples might be not enough to estimate well the model parameters. Therefore, non-parametric methods have found successfull applications in k-shot learning [Snell et al., 2017; Vinyals et al., 2016] due to their built-in ability to adapt to the complexity of the input dataset.

To prevent overfitting in the parametric models we often use informative priors for the model parameters. In the low data setting the use of such priors becomes critical for the performance of our algorithms. Well chosen priors for the models can significantly increase the robustness of the model. However, modelling and tuning of the priors introduces additional burden on the modeller.

Another practical challenge is the need for robust and sensible procedures to evaluate the k-shot learning algorithms. In k-shot learning the results are highly depended on the splits between training and test classes, k training examples and the examples used for testing. Therefore, a well defined evaluation protocol is required that defines the exact used splits and applied averaging schemes so that results between different methods can be comparable. Finally, due to relatively recent increase in interest in the k-shot learning research, the standard testbeds for evaluation of the algorithms are only starting to emerge.

Fig. 1.1 Illustration of example challenges resulting from large variation in images from the same class. From left: illumination, deformation, occlusion, background clutter and intra class variability.

1.3.2 Why is it hard for Deep Learning?

K-shot learning is particularly hard for the current state-of-the-art deep learning models. Such models often contain very large number of parameters and the most common techniques for training these parameters use MLE estimates that require large amount of data to be well calibrated. With smaller dataset such models rapidly overfit the training examples. At the same time, full Bayesian treatment is often not applicable due to intractability of the resulting distributions and the large size of the problems.

Even with large datasets the deep learning methods that have shown to provide poor certainty estimates [Guo et al., 2017]. While Bayesian Neural Networks [Gal and Ghahramani, 2016] show better calibration estimates and are more robust to overfitting, they still requires large number of parameters to train.

Finally, transfer learning from deep neural networks to smaller datasets is often challenging due to high dimensionality of the network weights. For a small dataset the number of dimensions in the network weights is likely to be larger that then number of examples is such dataset. This can introduce a problem where the number of parameters to estimate is higher than the number of training examples.

1.4 Basic introduction to the method

The method for k-shot learning described in this work and Bauer et al. [2017] aims to combine the strengths of the deep learning models for features extraction with probabilistic modelling for transferring the knowledge from a large data setting to the k-shot learning setting. The main focus is the efficient transfer of the knowledge.

The probabilistic approach to k-shot learning consists of multiple phases. In the first phase we train a convolutional neural network (CNN) Φ_φ as feature extractor. To train the CNN we use only images from the large data set $\widetilde{\mathcal{D}}$ corresponding to classes

$\widetilde{C}$. The training results in parameters φ for all the layers before the last layer and parameters $\widetilde{\mathrm{W}}$ for the last softmax layer that compute $p(\widetilde{y}_n \,|\, \widetilde{\mathbf{x}}_n, \widetilde{\mathrm{W}}) = \mathrm{softmax}(\widetilde{\mathrm{W}}\widetilde{\mathbf{x}}_n)$. Given parameters $\widetilde{\mathrm{W}}$ we model a distribution over possible softmax weights, which forms a prior knowledge base for the k-shot learning.

During the k-shot learning time we fix parameters φ, extract final hidden layer features $\mathbf{x} = \Phi_\varphi(\mathbf{u})$ for k examples $\mathbf{u}$ from $\mathcal{D}$ for each class in C and retrain the parameters W in softmax $p(y_n \,|\, \mathbf{x}_n, \mathrm{W}) = \mathrm{softmax}(\mathrm{W}\mathbf{x}_n)$ for the new classes. During retraining we use both likelihood of the activations $\mathbf{x}$ and the prior distribution over weights trained based on $\widetilde{\mathrm{W}}$.

We also propose multiple extensions to this method that aim to improve the quality of the learned prior distribution over weights and final k-shot learning accuracies.

First, instead of a single original task, which consist of training the CNN Φ_φ to obtain the weights $\widetilde{\mathrm{W}}$, we generate many tasks by retraining new weights $\widetilde{\mathrm{W}}_m$ based on subsets of classes from the dataset $\widetilde{\mathcal{D}}$. To train the new softmax weights we use all training examples from the selected classes. This results in higher number of the softmax weights to train the prior knowledge base.

Second extension involves substituting the linear softmax classifier with a non-linear function approximator (neural network with a single hidden layer). This aims to improve the k-shot classification accuracies on the new classes C by transforming the activations from the small dataset $\mathcal{D}$ into more linearly separable space.

Third and final extension, analyses the impact of the dimensionality of the last hidden layer on the quality of the joint prior distribution and the k-shot learning accuracies.

1.5 Contributions

The basis for the work presented in this thesis was created as a joint work with Matthias Bauer, Meteo Rojas-Carulla, Berhnard Schölkopf and Richard E. Turner in Bauer et al. [2017]. Whilst the initial research of the available datasets, related literature and comparable experimental setups was done by me, most of the implementation and related experimentation was done by Matthias and Meteo. Richard and Bernhard provided useful guiding thoughts and help. Key contributions of the work presented in Bauer et al. [2017] include:

- Development of a framework for k-shot learning that combines deep learning and probabilistic modelling.

- Extensive analysis of distributions to model the joint prior distribution over weights and related inference methods, including adversarial setting and calibration analysis.
- Comparison to related k-shot learning methods, which shows that a very simple variant of the framework is competitive with other state-of-the-art k-shot learning approaches. The probabilistic framework is also more general and extensible than the alternative methods.

The contributions of this thesis build on top of the original joint work by extending the proposed k-shot learning framework, providing further in depth analysis and improving the original results. To be precise, the contributions of solely this work include:

- Improvement in the held-out likelihoods for the GMM distributions by introducing regularisation parameter on the diagonal of component covariance. The regularisation encourages isotropic covariances in a low data setting while allowing for more flexibility than the strictly isotropic covariances.
- Extension of the training procedure to multitask training that results in higher number of example weights for estimating the distribution parameters. Resulted in further improvement of the GMM models relatively to the Gaussian models. The experiments are concluded with a principled explanation of the observed empirical results, which indicate that softmax weights for a given class are invariant to the context of other classes and depend purely on the data for this class.
- Development of a probabilistic framework for non-linear neural network classifiers and in-depth analysis of the obtained k-shot learning results.
- Analysis of k-shot learning performance w.r.t the dimensionality of the feature space. Resulted in improved k-shot learning accuracies by 3% on 1-shot (56% up from 53%) and 2% on 5-shot (69% up from 67%) 5-way classification when compared to the original results in Bauer et al. [2017].
- Other smaller contributions include: providing a structured view of the learning paradigms, comprehensive review of the k-shot learning literature, visualisation of the clustering structure in the softmax weights, updating the GMM likelihood from Bauer et al. [2017] with correct values.

Chapter 2

Background

In this chapter we provide background for the rest of the thesis. We briefly describe convolutional neural networks and their application to transfer learning. Then we explain the derivations for a Gaussian model used in probabilistic modelling to train a prior distribution over model weights. Finally, provide a literature review for k-shot learning.

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) [LeCun et al., 1989; Rumelhart et al., 1985] is a type of a deep neural network that has become a standard tool for image classification. Popularity of the CNNs can be attributed to their outstanding performance on the complex image recognition tasks. For instance, CNNs achieved above-human performance on the difficult ImageNet challenge (ILSVRC) [Russakovsky et al., 2015]. Due to this success, CNNs have now found application in many other areas.

Convolutional neural networks are composed of a sequence of three types of layers. Fully connected layers which are the same as in a regular neural network. Convolutional layers that instead of full connections use local connectivity to make use of spatial alignment in inputs and reduce number of model parameters. Finally, pooling layers perform downsampling operation again to reduce number of model parameters.

CNNs have proven to be a powerful general feature extractor that is able to provide easily separable features spaces. More recently, the best performing models such as ResNet [He et al., 2016] or Inception [Szegedy et al., 2016] are characterised by their very deep architectures that are trained on large datasets such as ImageNet. However, training such models often requires large computational resources. Therefore, a common practice is to use pre-trained CNNs as feature extractors and apply them to

other image datasets (see Fig. 2.1 and Fig. 2.2), which is a form of the representational knowledge transfer. We then use the obtained features to train classifiers for the new classes. However, when the number of training examples for the new classes is very small, as in k-shot learning, we are also interested in the concept transfer from the weights of the pre-trained network to the weights for the new classes. Therefore, in the next section we briefly introduce probabilistic modelling responsible for the concept transfer in the probabilistic framework for k-shot learning.

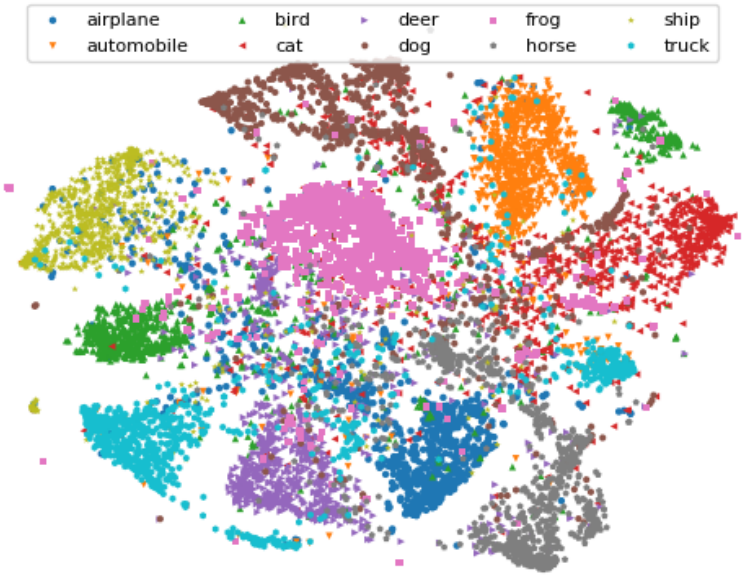


Fig. 2.1 T-SNE [Maaten and Hinton, 2008] visualisation of last hidden layer activations for the CIFAR-10 dataset obtained using Inception architecture trained on ImageNet.

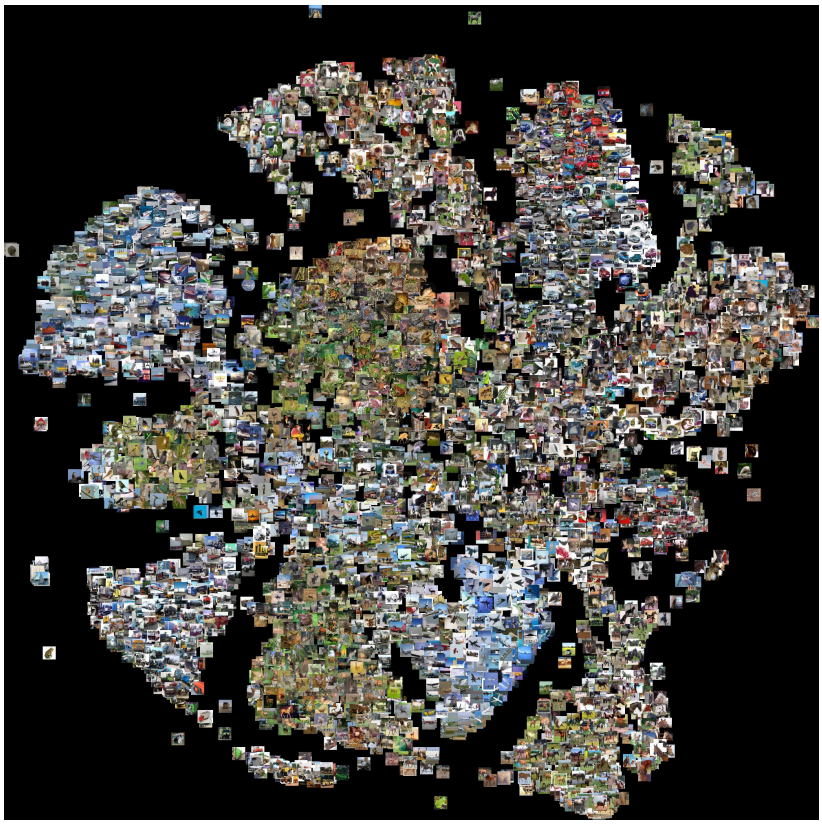


Fig. 2.2 Plot of images based on locations in a t-SNE embedding of activations obtained using pre-trained Inception for the CIFAR-10 dataset.

2.2 Probabilistic modelling

While CNNs are able to perform well the representational transfer, probabilistic modelling is used for concept transfer between the model weights. The concept transfer is performed by training a prior distribution over the weights of a well-trained model.

In this work we consider three distributions over the weights: Gaussian, Gaussian Mixture Model (GMMs) and Laplace. We describe their application in more detail in the next chapter. While the GMMs and Laplace distributions are trained using standard maximum likelihood estimation techniques, the parameters of a Gaussian model can estimated in a closed form when used with a conjugate prior. In this section we include a summary of the derivations that are performed to obtain the closed form solution for the Gaussian.

The enclosed derivations are following that in Murphy [2012], but are presented in a distilled form that is directly applicable. Furthermore, we include additional comments to ease the understanding of the calculations.

Conjugate prior for a Gaussian

We first show that a Normal-inverse-Wishart (NIW) distribution is the conjugate prior for the multivariate Gaussian. In this setting weights are treated as data and we denote them as $\mathbf{w}$. We start the derivation from the standard formulation of the Gaussian:

$$p(\mathcal{D}\,|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i}^{N} \mathcal{N}(\mathbf{w}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.1}$$

$$= (2\pi)^{-ND/2} |\boldsymbol{\Sigma}|^{-N/2} \exp\Big(-\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w}_i - \boldsymbol{\mu})\Big) \tag{2.2}$$

and using the "trace trick":

$$\mathbf{w}^T\mathbf{A}\mathbf{w} = \mathrm{tr}(\mathbf{w}^T\mathbf{A}\mathbf{w}) = \mathrm{tr}(\mathbf{w}\mathbf{w}^T\mathbf{A}) = \mathrm{tr}(\mathbf{A}\mathbf{w}\mathbf{w}^T) \tag{2.3}$$

we transform the exponent of the Gaussian into a form more suitable for comparison with the NIW distribution:

$$\sum_{i=1}^{N}(\mathbf{w}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{w}_i - \boldsymbol{\mu}) = \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{S}_{\overline{\mathbf{w}}}) + N(\overline{\mathbf{w}} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\overline{\mathbf{w}} - \boldsymbol{\mu}). \tag{2.4}$$

and apply it to the likelihood equation obtaining:

$$p(\mathcal{D}\,|\,\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{ND}{2}}} |\boldsymbol{\Sigma}|^{-\frac{N}{2}} \exp\Big(-\frac{N}{2}(\overline{\mathbf{w}} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\overline{\mathbf{w}} - \boldsymbol{\mu}) - \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_{\overline{\mathbf{w}}})\Big) \tag{2.5}$$

We then formulate the NIW distribution, which represents a joint prior for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The NIW assumes that $\boldsymbol{\mu}$ is dependent on $\boldsymbol{\Sigma}$. This makes intuitive sense because the higher the variance in the data the less sure we are about the mean of the data. The NIW is formulated using a multivariate Gaussian as a prior for $\boldsymbol{\mu}$ and the

Inverse-Wishart distribution as a prior for $\boldsymbol{\Sigma}$:

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\theta) = \text{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{m}_0, \kappa_0, \nu_0, \mathbf{S}_0) \triangleq \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \frac{1}{\kappa_0}\boldsymbol{\Sigma}) \times \text{IW}(\boldsymbol{\Sigma}|\mathbf{S}_0, \nu_0) \tag{2.6}$$

$$= \frac{1}{Z_{NIW}}|\boldsymbol{\Sigma}|^{\frac{1}{2}} \exp(-\frac{\kappa_0}{2}\Big(\boldsymbol{\mu} - \mathbf{m}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{m_0})\Big) \tag{2.7}$$

$$\times |\boldsymbol{\Sigma}|^{-\frac{\nu_0+D+1}{2}} \exp\Big(-\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\Big) \tag{2.8}$$

$$= \frac{1}{Z_{NIW}}|\boldsymbol{\Sigma}|^{-\frac{\nu_0+D+2}{2}} \tag{2.9}$$

$$\times \exp\Big(-\frac{\kappa_0}{2}(\boldsymbol{\mu} - \mathbf{m}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{m_0}) - \frac{1}{2}\mathbf{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\Big) \tag{2.10}$$

where $Z_{NIW} = 2^{\nu_0 D/2}\Gamma_D(\nu_0/2)(2\pi/\kappa_0)^{D/2}|\mathbf{S}_0|^{-\nu_0/2}$. Comparison of the transformed form of the Gaussian likelihood and the above form of the NIW distribution reveals a clear conjugate relationship between the two. The $\mathbf{m}_0$ is the prior mean for $\boldsymbol{\mu}$, κ_0 is the sudo-count for the belief in the mean, while $\mathbf{S}_0$ is proportional to the prior mean for the $\boldsymbol{\Sigma}$ and ν_0 represents how certain we are about that prior. In this work the hyper-parameters of the NIW distribution are tuned in a principled way using cross-validation.

Inferring parameters of a Gaussian

We show that the posterior distribution obtained from the product of the Gaussian likelihood and the NIW prior is also a NIW distribution. We form the posterior equation and substitute the likelihood by the Gaussian and the prior by the NIW as described above:

$$\begin{aligned}
p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) &\propto \Big(\prod_n p(\mathbf{w}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma}\Big)p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\theta) \\
&\propto |\boldsymbol{\Sigma}|^{-N/2}\exp\Big(-\frac{N}{2}(\overline{\mathbf{w}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\overline{\mathbf{w}} - \boldsymbol{\mu}) - \frac{N}{2}\text{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_{\overline{\mathbf{w}}})\Big) \\
&\times |\boldsymbol{\Sigma}|^{-\frac{\nu_0+D+2}{2}}\exp\Big(-\frac{\kappa_0}{2}(\boldsymbol{\mu} - \mathbf{m_0})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{m}_0) - \frac{1}{2}\text{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\Big) \\
&= |\boldsymbol{\Sigma}|^{-\frac{N+\nu_0+D+2}{2}}\exp\Big(-\frac{N+\kappa_0}{2}\Big(\boldsymbol{\mu} - \frac{\kappa_0\mathbf{m}_0 + N\overline{\mathbf{w}}}{\kappa_N}\Big)^T\boldsymbol{\Sigma}^{-1}\Big(\boldsymbol{\mu} - \frac{\kappa_0\mathbf{m}_0 + N\overline{\mathbf{w}}}{\kappa_N}\Big) \\
&- \frac{1}{2}\text{tr}\Big(\boldsymbol{\Sigma}^{-1}\Big(\mathbf{S}_{\overline{\mathbf{w}}} + \mathbf{S}_0 + \frac{\kappa_0 N}{\kappa_0 + N}(\overline{\mathbf{w}} - \mathbf{m}_0)(\overline{\mathbf{w}} - \mathbf{m}_0)^T\Big)\Big)\Big)
\end{aligned}$$

This form can be summarised as:

$$
\begin{aligned}
p(\boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{D}) &= \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathbf{m}_N, \kappa_N, \nu_N, \mathbf{S}_N) \\
\mathbf{m}_N &= \frac{\kappa_0 \mathbf{m}_0 + N\overline{\mathbf{w}}}{\kappa_N} \\
\kappa_N &= \kappa_0 + N \\
\nu_N &= \nu_0 + N \\
\mathbf{S}_n &= \mathbf{S}_0 + \mathbf{S}_{\overline{\mathbf{w}}} + \frac{\kappa_0 N}{\kappa_0 + N}(\overline{\mathbf{w}} - \mathbf{m}_0)(\overline{\mathbf{w}} - \mathbf{m}_0)^T
\end{aligned}
$$

where $\mathbf{S}$ is the sample covariance of $\widetilde{\mathrm{W}}$ calculated as $\mathrm{S} = \sum_{i=1}^{N} \mathbf{w}_i \mathbf{w}_i^T$. The results show that the parameters of the posterior NIW are the weighted combination of the prior and the likelihood. The posterior scatter matrix is the sum of the prior and the empirical scatter matrices plus additional term for the variance in the mean.

Given the distribution over $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the Gaussian it is possible to estimate the values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ as MAP estimates, which are the NIW posterior modes:

$$
\underset{\boldsymbol{\mu}^{\mathrm{MAP}}, \boldsymbol{\Sigma}^{MAP}}{\arg\max} \quad p(\boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathcal{D}) = \Big(\mathbf{m}_N, \frac{\mathbf{S}_N}{\nu_N + D + 2}\Big).
$$

The MAP estimates are later used as $p(\mathbf{w}|\widetilde{\mathcal{D}}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}^{\mathrm{MAP}}, \boldsymbol{\Sigma}^{\mathrm{MAP}})$. Alternatively, it is possible to integrate out the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, and obtain the posterior over weights analytically:

$$
\begin{aligned}
p(\mathbf{w}|\widetilde{\mathcal{D}}) &= \int\int \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma} \,|\, \mathbf{m}_N, \kappa_N, \nu_N, \mathrm{S}_N) d\boldsymbol{\mu} d\boldsymbol{\Sigma} && (2.11) \\
&= \mathcal{T}(\mathbf{w} \,|\, \mathrm{m}_N, \frac{\kappa_N + 1}{\kappa_N(\nu_N - D + 1)}\boldsymbol{\Sigma}, \nu_N - D + 1). && (2.12)
\end{aligned}
$$

We observe that the student-t distribution accounts for the additional uncertainty in the estimates of the variance $\boldsymbol{\Sigma}$ due to its heavier tails.

2.3 Related work

In this section we describe work related to the probabilistic k-shot learning framework. We start with introducing foundational work in the area of probabilistic transfer learning and inspiring work on human-level few-shot learning. Then we refer to alternative k-shot learning approaches and conclude with comparison to the most related deep probabilistic methods.

2.3.1 Foundational work

The most related foundational work was developed by Bakker and Heskes [2003]. The authors of this work also aim to achieve synergy between neural networks and statistical modelling.

Similarly to our approach they fix the initial part of a neural networks and loosely couple task specific softmax weights using joint distribution over possible weight values. They also consider similar prior distributions to model both the relationship between dimensions of the weight vectors (Gaussian) and multi-modality of the weights (GMM). However, their approach is more related to batch multitask learning as described in Section 1.2.2 or work of Srivastava and Salakhutdinov [2013]. In their method the shared parameters of the neural network are trained jointly across all the tasks, while our approach is online with the shared weights trained only on the large data $\widetilde{\mathcal{D}}$. Furthermore, they do not focus on the few-shot scenario.

Probabilistic modelling applied to one-shot image classification was initially proposed by Fei-Fei et al. [2006] (to the best of our knowledge). They also build prior distributions over model parameters, but their models are generative in nature rather discriminative as in our case (CNN).

More recently an inspiring work by Lake et al. [2015] have shown that their method was able to achieve human-level[1] learning abilities on the dataset of handwritten characters from the world's alphabet. Their method called Bayesian program learning (BPL) uses probabilistic generative models in hierarchical procedures. It aims to model real world concepts by compositional structure, causality and meta-learning to improve the learning process.

2.3.2 Alternative approaches to k-shot learning

In this section be describe methods that address the same problem of k-shot learning, but using alternative approaches.

Embedding methods We first discuss the group of k-shot learning methods that we refer to as embedding methods. These methods perform non-linear transformation of the original data space, and then perform classification in the transformed space based on distance between training k-shot examples and test examples according to a metric. Similarly to our method, these methods use CNNs trained on classes available before the k-shot learning as the non-linear function.

[1] Indistinguishable from humans.

One of the initial embedding methods for one-shot learning were Siamese neural networks proposed by Koch et al. [2015]. The Siamese networks were originally introduced for signature verification by Bromley et al. [1994]. They compute whether the images are the same or not. Siamese networks are built from two identical neural networks with tied parameters and joined by an energy function at the top. In Koch et al. [2015] the energy function is a weighted L_1 distance metric followed by a sigmoid function. In the k-shot setting given images x and new classes C the model predicts the class from C that contains an image most similar to x.

Vinyals et al. [2016] propose another embedding method that instead of the binary same/different classification performs nearest neighbour matching in the embedding space. They use cosine distances from the support set to perform classification. They also find improvement when training with embedding function using so called Full Context Embedding that is used to the learn to embed whole support set jointly instead of each data point independently. The authors also propose episodic training procedure that constructs batches of training data for training the embedding function in a settings that resembles that expected during test time. To be precise, they construct training batches (episodes) that contain the same number of distinct classes and examples per class as expected during the test time. This training procedure has proved to provide improvement on the test data and has been adopted by following k-shot learning work.

Later introduced Prototypical Networks Snell et al. [2017] are similar to the matching networks, but use even more simple classification procedure that results in higher model performance. Instead of computing distances from a test example to individual points in the support set, they calculate distances to prototypes for each class in the support sets. Prototypes are calculated as a mean in the embedding space of the class examples. Instead of cosine distance they used Euclidian distances, which results in better k-shot learning accuracies in their setting. Similarly to Matching Networks they also find significant performance improvements from episodic training.

Meta-learning The second group of related k-shot learning approaches are the meta-learning algorithms that aim to improve the learning process on the k examples of the new classes by training on distribution of similar tasks. They learn the high-level strategy that intends to capture the essence of the task. Ravi and Larochelle [2017] propose an LSTM-Meta learner architecture that learns the optimal initialisation and optimisation algorithm to train another classification network using such few

examples. Their training procedure also uses batches of training episodes that match test conditions.

More recently Mishra et al. [2017] propose a temporal-convolution-based meta-learner (TCML) that aims to address the problem of handcrafting meta-learning algorithms for a particular application. They apply their method to few-shot classification as an example test bed. The temporal convolutions [Yu and Koltun, 2015] in their model are used to learn how to evaluate similarity between test examples and support data points without any predefined comparison strategy. During k-shot learning time they use this similarity measure for classification. Their method also use episodic training and CNNs as feature extractors when applied to images.

2.3.3 Deep probabilistic methods

Out work is most similar to other deep probabilistic methods. In particular, our inspiration was the method proposed by Srivastava and Salakhutdinov [2013]. While their approach is not directly an online k-shot learning method, but rather a batch learning algorithm with small number of examples, the two methods are closely related. Both methods learn a joint prior distribution over softmax weights in a CNN to increase generalisation in low data settings. Their tree-based prior structure is similar to a GMM prior model presented in our work. While their inference technique for the new weights is restricted to a MAP estimate, we additionally analyse HMC sampling.

The most related to ours is the method described by Burgess et al. [2016]. Their approach very closely resembles the specific case of our framework. To be more precise, their prior model is similar to our multivariate Gaussian with parameters obtained using MLE estimate, but with a additional regularisation parameter on the diagonal of covariance that induces the isotropic behaviour. This regularisation parameter is identical to the one we use for the covariances of the Gaussian components in a GMM model prior. Furthermore, while our work uses the estimates of the weights using both MAP and HMC estimates, they report results just for the HMC treatment.

More recently a different approach was proposed by Qiao et al. [2017] that, instead of modelling the prior based weights for old classes $\widetilde{C}$, they aims to leverage the relationship between the structure of the softmax weights and last hidden layer activation to build robust priors for the new weights. Their method has shown state-of-the-art results on a large and challenging ImageNet dataset.

Chapter 3

General setup

In the previous chapter we briefly introduced probabilistic modelling techniques and convolutional neural networks that are combined in this work to form the probabilistic k-shot learning framework. We also described the related and alternative approaches to k-shot learning.

In this chapter we describe in detail each phase of for the probabilistic k-shot learning framework that forms a basis for extensions described in the next chapters. In particular, we introduce a graphical model and mathematical formulation for the task, paying special attention to the concept modelling stage. Furthermore, we provide the experimental setup and results for the described framework, that are referred to in the following chapters. Additionally, to the original results we also describe improvements to one of the models for the weights prior distribution (GMM).

The major part of the work described in this chapter is a result of a joint work with Matthias Bauer, Meteo Rojas-Carulla, Berhnard Schölkopf and Richard E. Turner that was submitted to NIPS 2017 in Bauer et al. [2017]. Whilst the initial research of the available datasets, related literature and comparable experimental setups was done by me, most of the implementation and related experimentation was done by Matthias and Meteo. Richard proposed the original idea and both Richard and Bernhard provided useful guiding thoughts and help. The improvements to the GMM models, which are also described in this chapter, are a result of my sole work.

3.1 Probabilistic k-shot learning framework

The probabilistic k-shot learning framework consist of four distinct phases. The first phase is representational learning when a CNN is trained on a large dataset $\widetilde{D}$. The second phase is concept learning that results in a knowledge base in a form of joint

prior distribution over model weights. The third and fourth phase constitute k-shot learning and testing respectively and both use the few shot dataset D. Fig. 3.1 shows a graphical model that gives an overview of the framework and illustrate dependencies between variables in the model. The rest of this section describes each of the four phases in detail, but paying most attention to the concept learning phase. The notation used in this chapter is the same as introduced in Section 1.4.

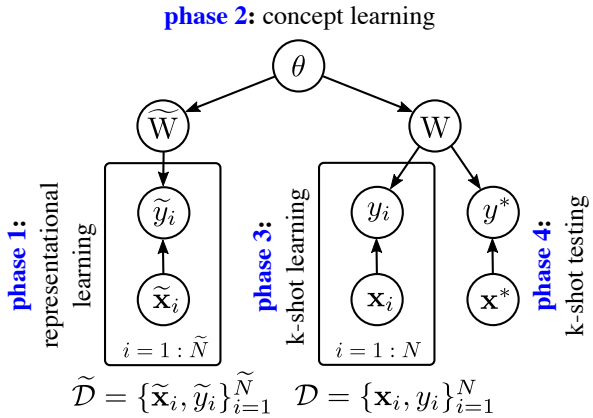


Fig. 3.1 Graphical model for probabilistic k-shot learning

3.1.1 Phase 1: representation learning

During representational learning phase we train a CNN Φ_φ on the large dataset $\widetilde{\mathcal{D}}$. This results in obtaining both parameters φ, which represent parameters of all the layers in the CNN until the last hidden layer, and softmax weights $\widetilde{\mathrm{W}}$ for the old classes C. The softmax weights are calculated using MAP estimation and thus denoted as $\widetilde{\mathrm{W}}^{\mathrm{MAP}}$ in later discussion. From that point the parameters φ are fixed and only used for feature extraction. The extraction can be expressed as $\mathbf{x} = \Phi_\varphi(\mathbf{u})$ and is performed for both images $\widetilde{\mathbf{u}}$ and $\mathbf{u}$ from respectively the large dataset $\widetilde{\mathcal{D}}$ and the small dataset $\mathcal{D}$. Fig. 3.2 illustrates this setup of the feature extractor.

3.1.2 Phase 2: concept learning

During the concept learning phase we are interested in learning the parameters θ of a probability distribution over all possible softmax weights. In this setting the softmax weights $\widetilde{\mathrm{W}}$ from the CNN trained on the large data $\widetilde{\mathcal{D}}$ are used as data to estimate the distribution parameters. This requires making a simplifying assumption that the

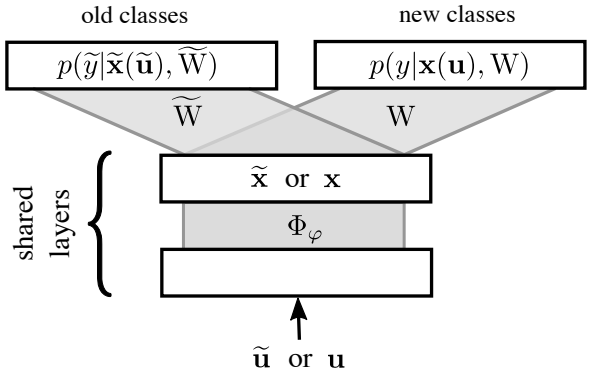


Fig. 3.2 Shared feature extractor Φ_φ and separate softmax units with associated weights for the original task (old classes) and the k-shot learning task (new classes).

distribution of the old weights $\widetilde{\mathrm{W}}$ and the new weights W is identical:

$$p(\widetilde{\mathrm{W}},\mathrm{W},\theta) = p(\theta)\prod_{c'=1}^{\widetilde{C}} p(\widetilde{\mathbf{w}}_{c'}|\theta)\prod_{c=1}^{C} p(\mathbf{w}_c|\theta) \text{ where } p(\widetilde{\mathbf{w}}_{c'}|\theta) \overset{\text{dist}}{=} p(\mathbf{w}_c|\theta). \tag{3.1}$$

Given such assumption we want to find a distribution that best models the old weights $\widetilde{\mathrm{W}}$ so that it can be used effectively during the k-shot learning time for knowledge transfer. The graphical model in Fig. 3.1 lets us express the relation between the distribution parameters θ and the softmax weights as $p(\widetilde{\mathrm{W}},\mathrm{W},\theta) = p(\theta)p(\widetilde{\mathrm{W}}|\theta)p(\mathrm{W}|\theta)$.

We train the parameters θ by formulating the posterior distribution over θ given the initial dataset $\widetilde{\mathcal{D}}$ $(p(\theta\,|\,\widetilde{\mathcal{D}}))$. Computing such distribution is challenging, but can be simplified by assuming that the weights $\widetilde{\mathrm{W}}$ trained on the large data $\widetilde{\mathcal{D}}$ are well approximated by the MAP solution $p(\widetilde{\mathrm{W}}\,|\,\widetilde{\mathcal{D}}) \approx \delta(\widetilde{\mathrm{W}} - \widetilde{\mathrm{W}}^{\mathrm{MAP}})$. Given the large data size this assumption is realistic. After making this assumption we arrive at a simplified posterior distribution $p(\theta\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}}) \propto p(\theta)p(\widetilde{\mathrm{W}}^{\mathrm{MAP}}|\theta)$ by using the weights $\widetilde{\mathrm{W}}^{\mathrm{MAP}}$ calculated using MAP estimation during the representational learning phase. This distribution can be computed analytically for conjugate priors $p(\theta)$ or using MAP estimate $p(\theta\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}}) \approx \delta(\theta - \theta^{\mathrm{MAP}})$ otherwise.

Specific distributions over weights

We analyse three possible choices for the distribution over the softmax weights. The first one is a multivariate Gaussian $p(\mathbf{w}|\theta) = \mathcal{N}(\mathbf{w}|\mu,\Sigma)$ which has a conjugate Normal-inverse-Wishart prior $p(\theta) = p(\mu,\Sigma) = \mathcal{NIW}(\mu,\Sigma\,|\,\mu_0,\kappa_0,\Lambda_0,\nu_0)$. Using the conjugate prior we can estimate the predictive distribution for the weights analytically, which results in a multivariate Student t-distribution [Gauss (integr. prior)]. Alternatively, it

is possible use a MAP estimate of the Normal-inverse-Wishart distribution $\theta^{\text{MAP}} = \{\mu^{\text{MAP}}, \Sigma^{\text{MAP}}\}$ [Gauss (MAP prior)]. We also analyse isotropic covariance obtained using MAP estimate to evaluate the impact of modelling relation between dimensions of the weights and to use it as a prior for biases. We skip the biases in our descriptions for clarity of notation.

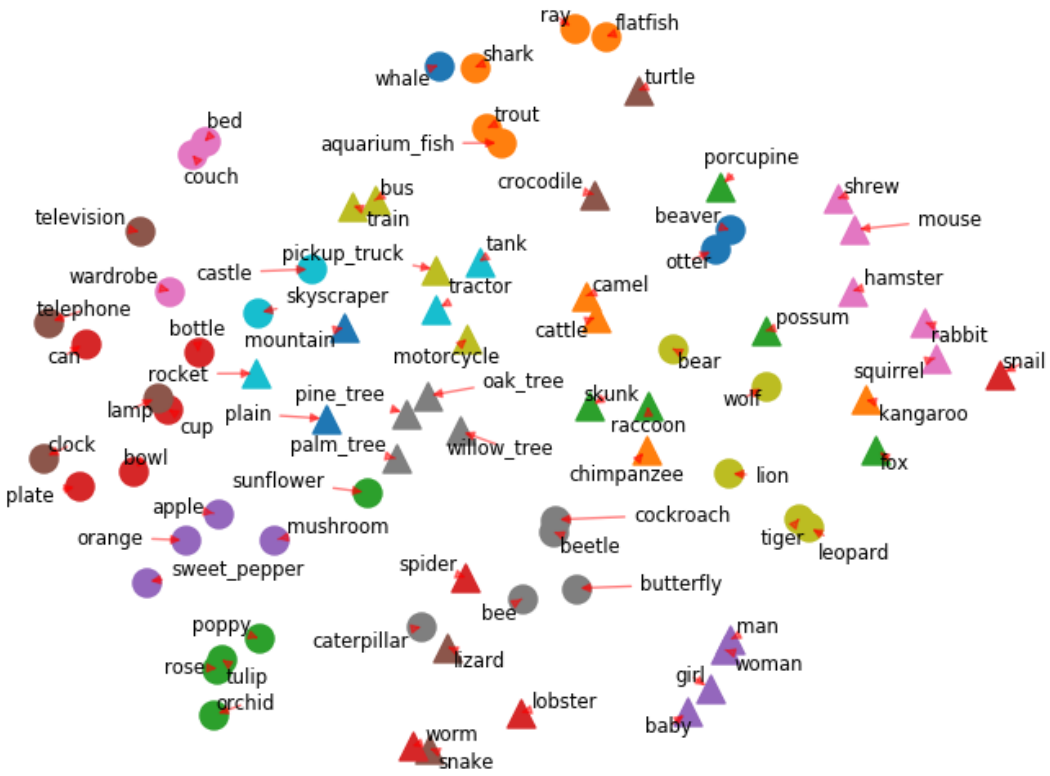


Fig. 3.3 T-SNE visualisation of softmax weights for each of the 80 training classes selected from the CIFAR-100 dataset. Combination of colour and shape of the markers indicates superclass of each class.

Second analysed type of distribution is the Gaussian Mixture Model (GMM) that aims to leverage the clustering behaviour shown by the weights of conceptually similar classes (see Fig. 3.3). We use GMMs with varying number of components e.g. 3 [GMM (3)] and 10 [GMM (10)], and compute the component parameters using MLE estimation with EM algorithm. For the CIFAR-100 dataset [Krizhevsky and Hinton, 2009] we also analyse using the provided superclasses as priors for the component means and variances [GMM (supercl.)].

The third considered distribution over weights is a Laplace distribution. The motivation for using this distribution is that the softmax weights show highly sparse structure and the Laplace distribution with peaked density function should fit well such values and enforce sparsity in modelling the weights ([Laplace (iso)] and [Laplace (diag)]).

3.1.3 Phase 3: k-shot learning

During the k-shot learning phase we use the prior distribution over weights from the concept modelling phase and last hidden layer activations for the small dataset $\mathcal{D}$ from the representational learning phase to form the posterior distribution $p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathcal{D}})$ over the softmax weights W for the C new classes from $\mathcal{D}$.

We begin deriving the posterior distribution by applying the product rule:

$$p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathcal{D}}) \propto p(\mathrm{W},\mathcal{D},\widetilde{\mathcal{D}}) = p(\widetilde{\mathcal{D}})p(\mathrm{W}\,|\,\widetilde{\mathcal{D}})p(\mathcal{D}\,|\,\mathrm{W},\widetilde{\mathcal{D}}). \tag{3.2}$$

Then based on the graphical model in Fig. 3.1 we know that $\mathcal{D}$ is conditionally independent from $\widetilde{\mathcal{D}}$ given W and thus:

$$p(\mathcal{D}\,|\,\mathrm{W},\widetilde{\mathcal{D}}) = p(\mathcal{D}\,|\,\mathrm{W}) = \prod_n p(y_n\,|\,\mathbf{x}_n,\mathrm{W}) \tag{3.3}$$

Using this equation and moving $p(\widetilde{\mathcal{D}})$ to a constant term we obtain:

$$p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathcal{D}}) \propto p(\mathrm{W}\,|\,\widetilde{\mathcal{D}})\prod_n p(y_n\,|\,\mathbf{x}_n,\mathrm{W}) \text{ where } p(\mathrm{W}\,|\,\widetilde{\mathcal{D}}) = \int p(\mathrm{W}\,|\,\theta)p(\theta\,|\,\widetilde{\mathcal{D}})\mathrm{d}\theta. \tag{3.4}$$

Using the simplifying assumption $p(\widetilde{\mathrm{W}}\,|\,\widetilde{\mathcal{D}}) \approx \delta(\widetilde{\mathrm{W}} - \widetilde{\mathrm{W}}^{\mathrm{MAP}})$ described for the concept learning phase and substituting $\widetilde{\mathcal{D}}$ with $\widetilde{\mathrm{W}}^{\mathrm{MAP}}$ results in:

$$p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathrm{W}}^{\mathrm{MAP}}) \propto p(\mathrm{W}\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}})\prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\mathrm{W}). \tag{3.5}$$

Although this formulation is generally intractable, to find the new weights W we can approximate it using MAP estimation $p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathrm{W}}^{\mathrm{MAP}}) \approx \delta(\mathrm{W} - \mathrm{W}^{\mathrm{MAP}})$ or given the small size of the dataset $\mathcal{D}$ using sampling $\mathrm{W}_m \sim p(\mathrm{W}\,|\,\mathcal{D},\widetilde{\mathrm{W}}^{\mathrm{MAP}})$. For the concept learning models with conjugate priors, the prior for the new weights $p(\mathrm{W}\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}}) = \int p(\mathrm{W}|\theta)p(\theta\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}})\mathrm{d}\theta$ can be estimated analytically. On other other hand, for the models that provide a MAP estimate θ^{MAP} the prior takes form $p(\mathrm{W}\,|\,\widetilde{\mathrm{W}}^{\mathrm{MAP}}) \approx p(\mathrm{W}|\theta^{\mathrm{MAP}})$.

3.1.4 Phase 4: k-shot testing

The final phase is k-shot testing where the representational learning is again used to extract the last hidden layer activations $\mathbf{x}^*$ for the unlabelled examples $\mathbf{u}^*$ and the posterior distribution of the softmax weights from the k-shot learning phase is used to perform classification. The predictive distribution averages the softmax outputs over the posterior distribution:

$$p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathrm{W}}^{\mathrm{MAP}}) = \int p(y^* \,|\, \mathbf{x}^*, \mathrm{W}) p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}^{\mathrm{MAP}}) \mathrm{dW}. \tag{3.6}$$

For MAP estimates of W the predictions are calculated using $p(y^* \,|\, \mathbf{x}^*, \mathrm{W}^{\mathrm{MAP}})$, while for samples using $\frac{1}{M} \sum_{m=1}^{M} p(y^* \,|\, \mathbf{x}^*, \mathrm{W}_m)$.

3.2 Implementation and experimental setup

In this section we describe experimental setup and include results for each phase of the probabilistic k-shot learning framework described in the previous section. This setup is also assumed in the following chapters unless stated otherwise. Additionally, we report in this section improvements developed as my sole work that directly relate to the general setup of the framework.

3.2.1 Phase 1: representational learning

We evaluate the performance of our method on a well-studied CIFAR-100 dataset [Krizhevsky and Hinton, 2009]. This dataset consist of 100 classes each with 500 training and 100 test images of size 32×32. The classes are grouped into 20 superclasses with 5 classes each. For the k-shot learning task the classes are randomly split into 80 training classes and 20 k-shot classes.

Architecture of the CNN used for representational learning is shown in Table 3.1. For non-linear activation function we use exponential linear units (ELUs) [Clevert et al., 2015] that have shown to shorten training times compared to the commonly used ReLUs [Clevert et al., 2015]. To prevent overfitting we use weight decay on fully connected layers and dropout [Gal and Ghahramani, 2016] with the probability that each element is kept at 50%.

The CNN is trained using Adam optimiser [Kingma and Ba, 2014] with decaying learning rate on the 80 training classes using all 500 training examples per classes. Evaluation of the trained CNN on the test set with 100 examples for the 80 classes

(80-way) results in 58.2% accuracy. Even though improving the feature extractor is not a direct aim of this work, we note that potential improvements in classification accuracies for our method are expected after employing more powerful types of CNN architectures such as ResNet [He et al., 2016] or Inception [Szegedy et al., 2016] that achieve accuracies as high as 79.6% [Zagoruyko and Komodakis, 2016] on 100-way classification (classification with 100 possible classes) for the CIFAR-100 dataset.

Network architecture	
Output size	**Layers**
$32 \times 32 \times 3$	Input patch
$16 \times 16 \times 64$	2× (Conv2D, ELU), Pool
$8 \times 8 \times 64$	2× (Conv2D, ELU), Pool
$4 \times 4 \times 128$	2× (Conv2D, ELU), Pool
$2 \times 2 \times 128$	2× (Conv2D, ELU), Pool
$2 \times 2 \times 128$	Dropout (0.5)
256	FullyConnected, ELU
256	Dropout (0.5)
128	FullyConnected, ELU
$\widetilde{C}$	FullyConnected, SoftMax

Table 3.1 Network architecture. All 2D convolutions have kernel size 3×3 and max-pooling is performed with stride 2. The output of the shaded layer corresponds to $\Phi_\varphi(u)$, the feature space representation of the image u, which is used as input for probabilistic k-shot learning.

3.2.2 Phase 2: concept learning

During concept learning we learn the parameters θ of the distribution over the softmax weight values $\widetilde{\mathrm{W}}$ as explained in the previous section. We evaluate the quality of the distributions based on negative log likelihood of randomly selected 10 held-out weights after training the distribution parameters on the remaining $\widetilde{C} - 10$ weights. The resulting log likelihoods are averaged over 50 random splits of the training and held-out weights. We also use this approach as a principled way of setting the hyperparameters of the prior distributions.

Table 3.2 shows the comparison of various analysed distributions. The best performing model is the multivariate Gaussian with the Normal-inverse-Wishart prior that allows to integrate over the model parameters. This indicates that the probabilistic approaches is in fact improving the model performance.

Gaussian mixture models (GMMs) with full covariances and additional regularisation term added to the covariance diagonal are also performing well. While the regularisation parameter encourages isotropic behaviour, the additional flexibility of the full covariances results in higher performance than that of the models with pure isotropic covariances despite higher number of model parameters. Fig. 3.4 shows that the performance of the GMM models trained using maximum likelihood EM strongly depends on the value of the regularisation parameter. The regularisation term is particularly important for the GMM models with a higher number of components. We expect such GMM distributions to provide better fit to the weights, due to the clustering structure of the weights for similar concepts (see Fig. 3.3). However, the results in Table 3.2 do not show such tendency. We attribute such behaviour to both the small number of training examples $\widetilde{C} - 10 = 70$ and a large number of parameters to estimate in the more complex models. Therefore, in the following chapters of this work we investigate ways to address the problem of the mismatch between the number of training examples and the model complexity. We also note that the results for the GMMs with the tuned regularisation show significant improvement over that reported in Bauer et al. [2017] (no tuning of regularisation parameter). These improved results are the outcome of my sole work as part of this thesis.

Finally, the Laplace distribution shows performance similar to that of the isotropic Gaussian, which suggests that modelling of the sparsity in the weights is not as beneficial as intended.

3.2.3 Phase 3 & 4: k-shot learning and testing

In the k-shot learning phase we use the posterior distribution over softmax weights trained on the large dataset $\widetilde{\mathcal{D}}$ and the last hidden layer activations for the small dataset $\mathcal{D}$ to infer the softmax weights for the new classes from $\mathcal{D}$.

Experimental setup

We evaluate the k-shot learning performance on a 5-way classification task. We construct the k-shot training datasets by randomly selecting 5 classes from the $C = 20$ new classes of the small data set $\mathcal{D}$. For each of the selected classes we use $k \in \{1, 5, 10\}$ training examples. We average our results over 20 sets of classes and 5 restarts for each set with different training examples across the restarts. During the k-shot testing we evaluate the inferred softmax weights for the new classes using all 100 test examples for each of the classes in the set. Additionally to test accuracies, we also report the

Model	Optimised held-out neg. log prob.
Gauss (iso)	-175.9 ± 0.3
Gauss (MAP prior)	-196.1 ± 0.5
Gauss (integr. prior)	-200.6 ± 0.4
GMM 1-mean (iso)	-175.5 ± 0.2
GMM 1-mean (diag)	-177.8 ± 0.3
GMM 1-mean (full)	-191.1 ± 0.3
GMM 3-means (iso)	-179.0 ± 0.3
GMM 3-means (diag)	-181.2 ± 0.3
GMM 3-means (full)	-189.5 ± 0.3
GMM 10-means (iso)	-181.6 ± 0.4
GMM 10-means (diag)	-181.6 ± 0.4
GMM 10-means (full)	-186.9 ± 0.5
Laplace (iso)	-173.8 ± 0.4
Laplace (diag)	-176.6 ± 0.5

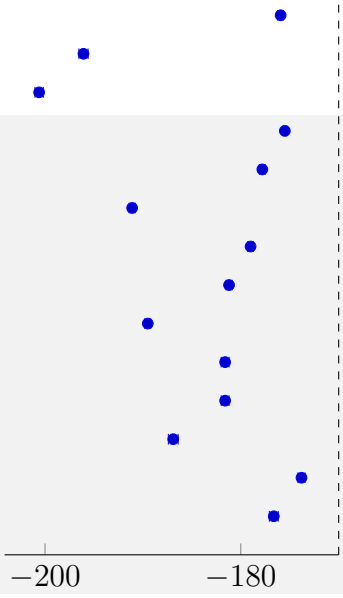


Table 3.2 Held-out negative log probabilities on random 70/10-splits of the training weights on CIFAR-100 (lower is better). Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

log-likelihoods of the predictions and calibration curves that measure the quality of uncertainty estimates for the predictions. The well calibrated uncertainty estimates are critical in decision making applications such as active learning [Ghahramani, 2015].

Approximate inference schemes

During the k-shot learning we approximate the posterior distribution over the new softmax weights W using either MAP estimate $p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}^{\mathrm{MAP}}) \approx \delta(\mathrm{W} - \mathrm{W}^{\mathrm{MAP}})$ or sampling $\mathrm{W}_m \sim p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}^{\mathrm{MAP}})$.

For the MAP estimation we use the fact that the gradients of the densities w.r.t W are available and use gradient-based optimisation to find the values of W. More precisely, we use the L-BFGS [Liu and Nocedal, 1989] optimisation algorithm. For computational efficiency and numerical stability when computing the likelihoods and the gradients we employ Cholesky decomposition of the covariances $\Sigma = LL^T$. For instance, the Gaussian likelihoods are efficiently computed using $-0.5(L^{-1}(x-\mu))^T L^{-1}(x-\mu))$ and its gradients as $-\Sigma^{-1}(x-\mu) = -L^{T^{-1}} L^{-1}(x-\mu)$.

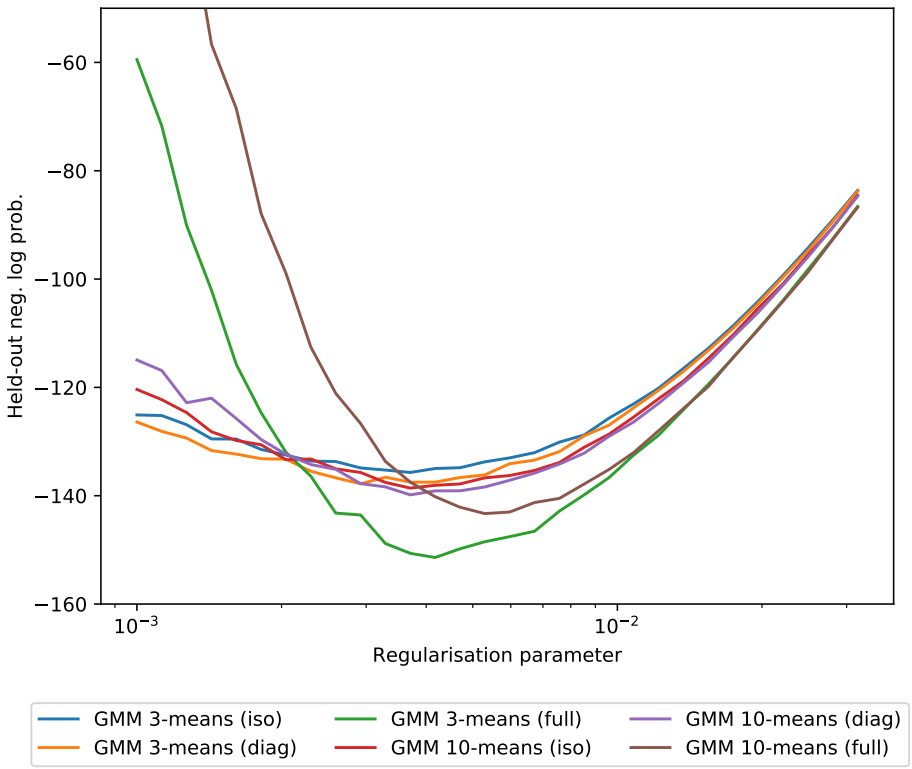


Fig. 3.4 Held-out negative log probabilities for the GMM models on random 70/10-splits of the training weights on CIFAR-100 w.r.t the diagonal covariance regularisation parameter. Values are averaged over 50 splits.

When sampling we use Hybrid Monte Carlo (HMC) in the form of a NUTS sampler [Hoffman and Gelman, 2014] that is able to automatically adapt sampling parameters such as the step size and the number of leapfrog steps. We use a burn-in period of 500 samples and 2000 samples to approximate target the distribution.

Baselines

We compare the performance of various combinations of the prior distributions for the weights W and the approximate inference schemes with the standard baselines for k-shot learning. We use two types of baselines, both of which operate in the embedding space from the representational learning phase. The first baseline is a non-parametric nearest neighbour classifier. The second baseline is a logistic regression in two versions: using pure MLE estimation and with regularisation term tuned for optimal performance. Both types of baselines provide competitive results despite their simplicity. It is important to mention that in the extreme low data setting as with

$k = 1$ using held-out training examples to tune the logistic regression regularisation parameter is problematic due to further reduction in the training dataset size.

Results

In this section we report results that form basis for comparison of the described general setup for the probabilistic k-shot learning framework with the extensions described in the following chapters. For more extensive evaluation of the general setup refer to Bauer et al. [2017],

Fig. 3.5 shows k-shot testing results for various combinations of the described prior distributions for the weights and the explained inference schemes, but also baselines for comparison. While the nearest neighbour classifier forms a strong baseline for $k = 1$, the logistic regression is a strong baseline for larger k. The results show that the best performing models for each k are the multivariate Gaussian models with an informative prior on the weights (MAP or integrated prior). The Laplace diagonal model with HMC sampling also achieves competitive results. Performance of the GMM models decreases with the increase in their complexity (number of components), but the gap between the simple and the more complex models diminishes with the increase in k. We also indicate that the results reported here for the more complex GMMs are improved over that reported in Bauer et al. [2017] due the application of the regularisation parameter on the component covariance diagonal.

While for Gaussian models there is nearly no difference in performance between MAP estimation and HMC sampling, the Laplace models perform better with HMC sampling.

The calibration curves show that the uncertainty estimates provided by the Gaussian models with the integrated prior outperform the MAP prior. This reinforces the argument that the probabilistic approach yields better uncertainty estimates. Furthermore, the logistic regression, even with an added regularisation term, shows one of the worst calibration performances across the models and very poor uncertainty estimates when used without the regularisation term (see Bauer et al. [2017]).

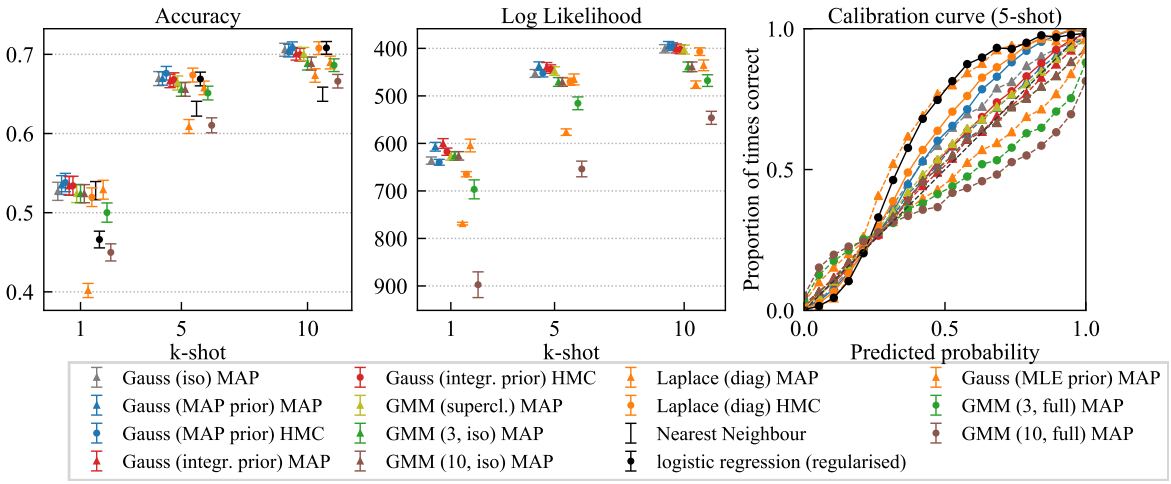


Fig. 3.5 Results for 5-way classification on CIFAR-100 averaged over 20 splits and 5 restarts with different training images. Our methods outperform the baselines in terms of accuracy and test log-likelihood, and lead to better calibrated classifiers.

Chapter 4

Extension 1 - Multi-set training

4.1 Introduction

In the previous chapter we discussed the general setup of the probabilistic k-shot learning framework introduced in Bauer et al. [2017]. While this approach performs well in the low data regime thanks to its ability of transfer the knowledge from the large data setting $\widetilde{\mathcal{D}}$ to the small data setting $\mathcal{D}$, we still see room for improvement in the quality of the prior distributions responsible for the knowledge transfer. In particular, we want to address the mismatch between the small number of examples to train the distributions and the complexity of the models that are expected to be well suited for the knowledge transfer. To be more precise, in the general framework setup, the number of the training examples for the prior is equal to the number of the training classes, which is usually lower than the dimensionality of the feature space. This poses a challenging scenario for estimating the distribution parameters.

While the multivariate Gaussian distribution over weights uses an informative conjugate Normal-Inverse-Wishart priors to remedy this problem, other distributions trained using MLE estimation such as GMMs are facing a more difficult task. At the same time, the models such as GMMs are expected to be most suited for the task (see Fig. 3.3).

Given this motivation, in this chapter we investigate ways to address the described mismatch problem. In particular, we look into ways of generating higher number of training examples for estimating the parameters of the distributions over the softmax weights. We aim to achieve this by generating many large data setting tasks instead of the single original task. The tasks are constructed by selecting subsets of classes from the $\widetilde{C}$ training classes and retraining the top softmax weights of the CNN Φ_{φ}. The hypothesis is that the weights for a given class take a different form depending on the

context of other classes and should provide higher number of examples for training the distributions. For instance, under such hypothesis, a softmax weight vector for a class 'car' is expected to use different image features when trained to discriminate between a car and a bicycle than when performing classification between a car and a dog. With the higher number and additional variety of the training weights it is possible to obtain more calibrated estimates for the parameters of the distribution over weights. This in turn results in improved knowledge transfer during the k-shot learning phase.

In this chapter we explore this hypothesis and gain valuable insights. We show that the training of the softmax weights is in fact invariant to the context of the different classes they are trained in. We provide both empirical results and a mathematical formulation to explain such behaviour. However, we also show that with smaller set sizes (number of classes in the context) the variance increases and results in improved estimates for the more complex models.

The work described in this chapter is completed solely by me with the guidance from Richard E. Turner.

4.2 Multi-set k-shot learning framework

This section describes in more detail the multi-set k-shot learning framework briefly introduced in the previous section. In particular, we focus on modifications to the general setup from the previous chapter.

4.2.1 Representational learning

In this approach the representational learning phase consists of two sub-phases. We first train the feature extractor CNN Φ_φ on all $\widetilde{C}$ training classes, which is similar to the general setup. However, we add an additional step when we retrain the softmax weights for random combination of the training classes using last hidden layer activations from Φ_φ. This setting is equivalent to multitask learning where the initial parameters of the network are shared across all tasks and the softmax weights are retrained for each of the tasks separately (see Section 1.2.2).

Feature extractor training

The first sub-phase is identical to the complete phase 1 in the original framework. We are given a large dataset $\widetilde{\mathcal{D}} = \{\widetilde{\mathbf{u}}_i, \widetilde{y}_i\}_{i=1}^{\widetilde{N}}$ of images $\widetilde{\mathbf{u}}_i$ and labels $\widetilde{y}_i \in \{1, \ldots, \widetilde{C}\}$ that indicate which of the $\widetilde{C}$ classes each image belongs to. Using this dataset we train a

CNN Φ_φ as a feature extractor. The feature extractor is used to obtain the last hidden layer activations $\widetilde{\mathbf{x}} = \Phi_\varphi(\widetilde{\mathbf{u}})$ for the large dataset $\widetilde{\mathcal{D}}$ corresponding to classes $\widetilde{C}$.

Weights retraining

The second sub-phase is an extension to the previous framework, where instead of just using the softmax weights $\widetilde{\mathrm{W}}$ computed during the feature extractor training, we compute m softmax weight matrices $\widetilde{\mathrm{W}}_1, \widetilde{\mathrm{W}}_2, ..., \widetilde{\mathrm{W}}_m$ based on m subsets of activations $\widetilde{\mathbf{x}}_i$ of l randomly selected labels from $\widetilde{y}_i \in \{1, \ldots, \widetilde{C}\}$. Therefore, each $\widetilde{\mathrm{W}}_m$ corresponds to a set of randomly selected l classes. In particular, the approach from the general setup is a specific instance of this framework when $m = 1$ and $l = \widetilde{C}$. Fig. 5.1 illustrates the proposed representational learning architecture.

The weights are computed using MAP estimation maximising the likelihood of the training data for selected l classes calculated using $p(\widetilde{y}_n \,|\, \widetilde{\mathbf{x}}_n, \widetilde{\mathrm{W}}_m) = \mathrm{softmax}(\widetilde{\mathrm{W}}_m \widetilde{\mathbf{x}}_n)$.

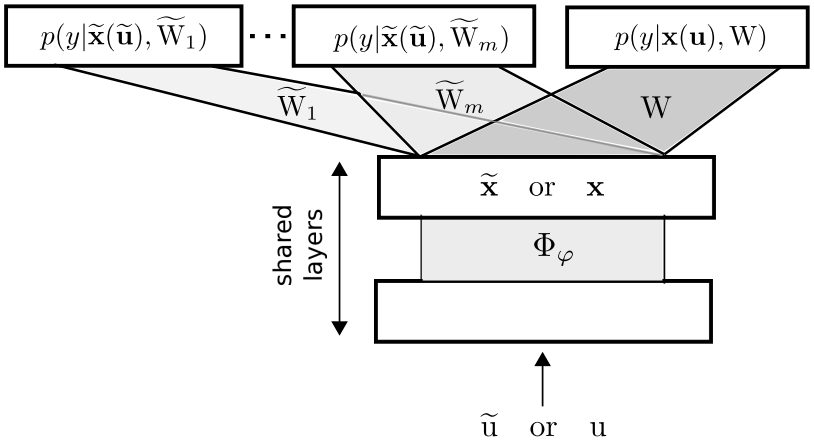


Fig. 4.1 Shared feature extractor Φ_φ trained on all training classes $\widetilde{C}$ as in the general setup, but instead of obtaining a single set of weights $\widetilde{\mathrm{W}}$ for all training classes $\widetilde{C}$, we retrain many $\widetilde{\mathrm{W}}_m$ weights for randomly selected subsets of classes from $\widetilde{C}$.

The expected average number of examples for a single class is equal to on $m \times l/\widetilde{C}$. Intuitively, each class will appear in in ever $\widetilde{C}/l$ set of classes, and with m sets there will be $m \times l/\widetilde{C}$ occurrences of any single class. The exact number of weight examples per class can vary in this procedure due to randomness in selection of the l classes to form a set. It is important to consider such imbalance in the number of weights amongst classes introduces a bias into the learned distribution over weights. Such behaviour can also be used to intentionally model a prior bias for the weights. In this case the count of weights for each class can be loosely interpreted as a sudo-count for such weight values. However, in our experiments we use relatively large number of

randomly selected sets of classes, which results in approximately equal distribution of weight examples per class.

In the experimental part of this chapter we verify whether the weights $\mathbf{w}'_c$ for a class $c' \in \{1, \ldots, \widetilde{C}\}$, which correspond to a single row in $\widetilde{\mathrm{W}}_m$, trained in m different contexts of $l-1$ changing other classes, show variability in their values. We also analyse how this variability depends on the number of classes l in each subset and the number of subsets m.

4.2.2 Concept learning

Concept learning phase is equivalent to that in the general setup, but instead of a single weight vector $\mathbf{w}'_c$ per class, we use one or many example weights for the same class coming from different training class subsets. This difference does not change the concept learning procedure because learning of the prior for the weights does not directly take into account the classes. However, the resulting distribution over weights depends on the weights given as input to the concept learning algorithm.

In the experimental section for concept learning, we analyse the log-likelihoods of the held-out weights with respect to the parameters l and m, and finally compare the resulting log-probabilities with the log-likelihoods for the general setup ($l = 80$ and $m = 1$). Similarly to the general setup, we tune the optimal values of the hyper-parameters for the weight distributions using cross-validation on the held-out weights.

4.2.3 K-shot learning and testing

The k-shot learning and testing procedure is the same as in the general setup. Given the prior trained in the concept modelling phase we perform k-shot learning using k examples from the test classes. However, the expectation is that the prior distribution over weights will be more informative during the learning time than in the general setup.

In the experimental section we verify whether the distributions trained on the larger set of examples of the weights results in more efficient knowledge transfer. This is evaluated based on the comparison of k-shot testing results for selected configurations of l and m with the results obtained in the general setup.

4.3 Representational learning results

In this section we empirically analyse the weights obtained using weights retraining during representational learning phase. In particular, we pay attention to the variance of the weights for the same class as it indicates potential increase in information provided to the concept learning.

Experimental setup

We first briefly introduce parts of the experimental setup that differ from that in the general setup from Section 3.2. The representation learning phase starts with preparing the training tasks for the multitask learning. Out of the $\widetilde{C} = 80$ training classes we randomly sample without returns l classes m times. In our experiments l commonly takes values of 5, 20, 40 and 80, and m takes values 1, 100 and 1000. Using $l = 5$ and $k = 5$ for 5-way classification resembles the episodic training used in alternative k-shot learning methods described in Section 2.3.2.

For retraining of the last softmax layer for the m generated subsets of classes we use all the 500 training examples per class. The examples are passed though the featuer extractor to obtained hidden layer activations $\widetilde{\mathbf{x}}_i$. The retraining is performed using MAP estimation with a tuned weight decay on the weights. In this scenario the regularisation of the weight values is critical because it ensures that the weights trained across different sets of classes converge to the same local modality and thus are comparable.

Experiments

We start the experiments by comparing the weights trained for the same class across different contexts (sets of classes). Fig. 4.2 shows that the weights for a single class obtain very similar values across the subsets with small variance in feature values across the examples of the weights. However, the plots indicate that the size l of the classes in the context (set) impacts the variance of the resulting weights. In particular, the weights trained in contexts of size $l = 40$ show smaller variance than the weights with $l = 5$ classes in a subset. This is also confirmed quantitatively with the two types of weights having similar range of mean, minimum and maximum values while having a difference of two orders of magnitudes in the scale of the variance across the examples for a single class. Therefore, in the later experiments we use $l = 5$ because we are interested in using weights with high variance within their possible values to approximate well the distribution over all possible softmax weights.

We also perform k-means clustering on the resulting weights to verify that the semantic relationship between the weights for a conceptually similar classes was preserved after the retraining procedure. Fig. 4.3 confirms that the resulting clusters contain examples of conceptually similar classes. We also observe that the k-means clustering was able to find alternative sensible clusters than the ones suggested by the CIFAR-100 superclasses. This indicates that in fact the GMM distribution over weights trained using MLE estimation should be able to model the weights better given large enough samples size than a GMM with components defined by the CIFAR-100 superclasses.

4.4 Concept learning results

In the concept learning phase we train the distributions over the softmax weights obtained from the previous phase.

Experimental setup

All the weights from the multiset training stage are stacked together without any special treatment of the weights that came from the same or a different subset. Similarly to the general setup we tune the hyperparameters of the distributions using cross-validation and evaluate them using held-out negative log-likelihood. The training and held-out examples are split based on classes. This means that all weights that correspond to a given class are exclusively either in the training set or the held-out set. The split between training and held-out classes is 70 training classes and 10 validation classes. The results are averaged across 50 random splits of classes.

Held-out log likelihood results

Table 4.1 shows held-out log-likelihoods for different distributions trained on $m = 1000$ subsets of weights with size $l = 5$, which corresponds to on average 62.5 weights per class. We observe that the multiset training procedure resulted in much better performance of the GMM models with higher number of components than the performance of the Gaussian models. The GMM with 10 components has proved to provide the best generalisation ability to the held-out weights, while the GMM with 70 components (number of training classes) overfitted the weights for the training classes. This is a significant improvement over the results reported in Section 3.2.2 where only as single set of weights with 80 classes was trained. These results indicate that in fact retraining

of the softmax weights allows to obtain better estimates for training more expressive distributions.

We also analyse the impact of the number of sets m on the held-out log likelihood for various models. We observe that for smaller number of weights to train the distributions (avg. 6 weights per class, $m = 5$ and $l = 5$) GMMs show much worse fit to the weights and do not outperform the baseline Gaussian. See the supplementary material in Section A.1 for detailed results. Furthermore, a smaller number of training weights to estimate the parameters of distributions increases the bias from unequal number of examples per class and is expected to be less calibrated for knowledge transfer to the new classes during k-shot learning.

Model	**Optimised held-out neg. log prob.**
Gauss (iso)	-351.0 ± 0.5
Gauss (MAP prior)	-428.4 ± 0.8
Gauss (integr. prior)	-432.4 ± 0.8
GMM 1-means (iso) reg.	-349.0 ± 0.4
GMM 1-means (diag) reg.	-350.6 ± 0.6
GMM 1-means (full) reg.	-431.1 ± 0.9
GMM 3-means (iso) reg.	-352.4 ± 0.4
GMM 3-means (diag) reg.	-354.5 ± 0.6
GMM 3-means (full) reg.	-463.5 ± 1.0
GMM 10-means (iso) reg.	-354.6 ± 0.6
GMM 10-means (diag) reg.	-357.6 ± 0.9
GMM 10-means (full) reg.	-483.8 ± 1.0
GMM 20-means (diag) reg.	-473.6 ± 1.1
GMM 40-means (diag) reg.	-434.0 ± 1.8
GMM 70-means (full) reg.	-298.0 ± 4.0

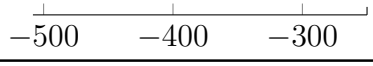


Table 4.1 Held-out negative log probabilities on random 70/10-splits of the training weights on CIFAR-100 (lower is better) obtained by retraining with $m = 1000$ sets each with $l = 5$ classes. Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

4.5 K-shot learning results

Experimental setup

During the k-shot learning phase we use the distributions over weights (with tuned hyperparameters) trained on the retrained $m = 1000$ sets of weights with $l = 5$ training classes in each set. We combine these distributions with the small dataset $\mathcal{D}$ to perform knowledge transfer to the new weights W. The experimental setup is the same as in the general setup.

K-shot learning results

Fig. 4.4 shows that in this setup the GMMs outperformed the Gaussian models both in terms of accuracy and log likelihood. Particularly high improvement is observed for the GMMs with a higher number of components. Such models achieve accuracies on pair with the best models from the general setup. Finally, while the GMMs show well calibrated uncertainty estimates, the Gaussian models tend to underestimate their certainty in predictions and show decrease in performance even after tuning of the hyperparameters of the distribution over weights. This can possibly be attributed to a high number of similar weights for each class, which is not well handled by the Gaussian. On the other hand, this setting can be well approximated by the GMM with tuned regularisation of component covariances on held-out classes.

4.6 Impact of context on the weights

In Section 4.3 we have shown empirically that the weights for a single class when retrained in the context of different classes results in similar values of the weights. While the reduction in the size of a set l introduced more variance across the weights for the same class, we can likely attribute this to a higher noise of the training process for the smaller sets. In this section we aim to provide a mathematically principled reasoning behind why the softmax weights are invariant to the context they are in. To explain this behaviour of the softmax function we frame softmax as a GMM with isotropic component covariances and show analogies between the two models. Fig. 4.5 provides an illustration of how GMM components relate to the softmax weights.

We start our derivations by transforming the softmax into a suitable form for our comparison:

$$p(\widetilde{y}_n = k \,|\, \widetilde{\mathbf{x}}_n, \widetilde{\mathrm{W}}) = \mathrm{softmax}(\widetilde{\mathrm{W}}\widetilde{\mathbf{x}}_n + \widetilde{\mathrm{b}}) = \frac{e^{\widetilde{\mathbf{w}}_k^T \mathbf{x}_n + \widetilde{b}_k}}{\sum_{c'=1}^{\widetilde{C}} e^{\widetilde{\mathbf{w}}_{c'}^T \mathbf{x}_n + \widetilde{b}_{c'}}} = \frac{1}{\sum_{c'=1}^{\widetilde{C}} e^{(\widetilde{\mathbf{w}}_{c'} - \widetilde{\mathbf{w}}_k)^T \mathbf{x}_n + \widetilde{b}_{c'} - \widetilde{b}_k}} \tag{4.1}$$

We now perform similar adaptation to the form of a GMM posterior. We assume the components of a GMM to have isotropic covariances. In such trained GMM model, the means and covariances of each component are defined by the data points that belong to a class assigned to the given component. In this model the prior probability of each class is defined as $sp(\widetilde{y}_n = k) = \pi_k$ and likelihood of the data is expressed as $p(\widetilde{\mathbf{x}} \,|\, \widetilde{y} = k, \widetilde{\boldsymbol{\mu}}_k, \widetilde{\sigma}_k^2) = \mathcal{N}(\widetilde{\mathbf{x}}; \widetilde{\boldsymbol{\mu}}_k, \mathbf{I}\widetilde{\sigma}_k^2)$. Given such assumptions the posterior that is defined by this GMM takes form:

$$\begin{aligned} p(\widetilde{y}_n = k \,|\, \widetilde{\mathbf{x}}_n) &= \frac{\pi_k p(\widetilde{\mathbf{x}}_n \,|\, \widetilde{y} = k, \widetilde{\boldsymbol{\mu}}_k, \widetilde{\sigma}_k^2)}{\sum_{c'}^{\widetilde{C}} \pi_{c'} p(\widetilde{\mathbf{x}}_n \,|\, \widetilde{y}_n = c', \widetilde{\boldsymbol{\mu}}_{c'}, \widetilde{\sigma}_{c'}^2)} = \frac{\pi_k (\widetilde{\mathbf{x}}_n; \widetilde{\boldsymbol{\mu}}_k, \widetilde{\sigma}_k^2)}{\sum_{c'}^{\widetilde{C}} \pi_{c'} \mathcal{N}(\widetilde{\mathbf{x}}_n; \widetilde{\boldsymbol{\mu}}_{c'}, \widetilde{\sigma}_{c'}^2)} \\ &= \frac{\pi_k (2\pi\widetilde{\sigma}_k^2)^{-D/2} \exp(-\frac{1}{2\widetilde{\sigma}_k^2}(\widetilde{\mathbf{x}}_n - \widetilde{\boldsymbol{\mu}}_k)^T(\widetilde{\mathbf{x}}_n - \widetilde{\boldsymbol{\mu}}_k)}{\sum_{c'}^{\widetilde{C}} \pi_{c'} (2\pi\widetilde{\sigma}_{c'}^2)^{-D/2} \exp(-\frac{1}{2\widetilde{\sigma}_{c'}^2}(\widetilde{\mathbf{x}}_n - \widetilde{\boldsymbol{\mu}}_{c'})^T(\widetilde{\mathbf{x}}_n - \widetilde{\boldsymbol{\mu}}_{c'})} \end{aligned}$$

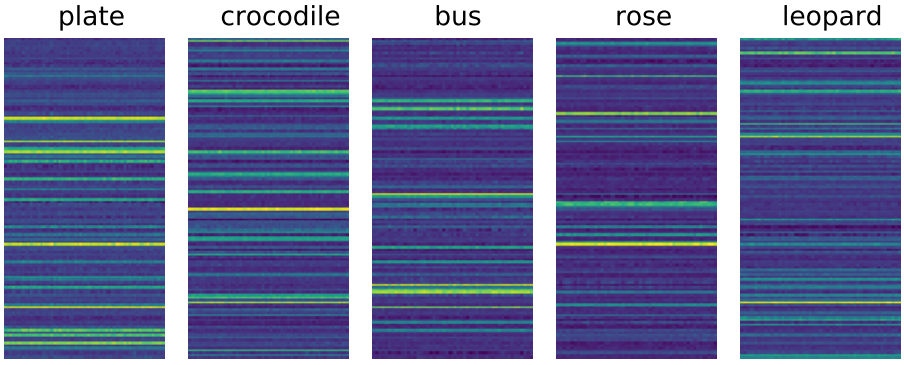
By assuming that the isotropic covariances for each class are the same $\widetilde{\sigma}_{c'} = \widetilde{\sigma} \forall c' \in \widetilde{C}$ we arrive at the form analogous to the softmax:

$$p(\widetilde{y}_n = k \,|\, \widetilde{\mathbf{x}}_n) = \frac{1}{\sum_{c'}^{\widetilde{C}} e^{\frac{1}{\widetilde{\sigma}^2}(\widetilde{\boldsymbol{\mu}}_{c'} - \widetilde{\boldsymbol{\mu}}_k)^T \widetilde{\mathbf{x}}_n - \frac{1}{2}(\widetilde{\boldsymbol{\mu}}_{c'}^T \widetilde{\boldsymbol{\mu}}_{c'} - \widetilde{\boldsymbol{\mu}}_k^T \widetilde{\boldsymbol{\mu}}_k) + \log \frac{\pi_{c'}}{\pi_k}}} \tag{4.2}$$

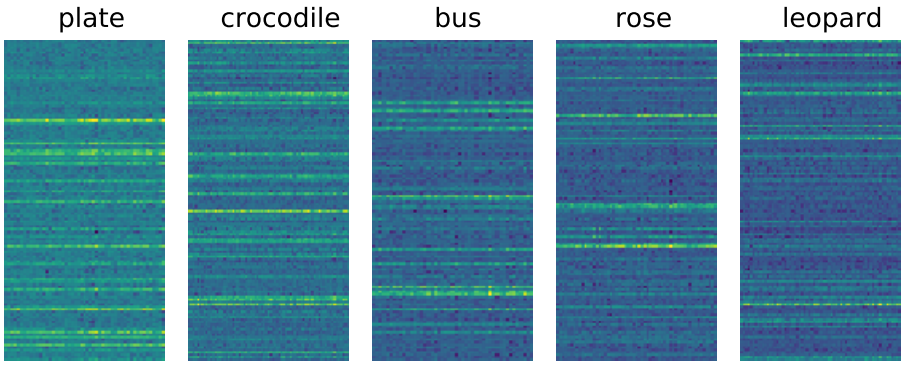
Given the transformed forms of Eq. (4.1) and Eq. (4.2) we observe clear analogies. The difference of the means of components in a GMM $\frac{1}{\widetilde{\sigma}^2}(\widetilde{\boldsymbol{\mu}}_{c'} - \widetilde{\boldsymbol{\mu}}_k)^T$ is equivalent to the difference of the weights in a softmax $\widetilde{\mathbf{w}}_{c'} - \widetilde{\mathbf{w}}_k$. Furthermore, the difference of the normalising constants in the GMM $-\frac{1}{2}(\widetilde{\boldsymbol{\mu}}_{c'}^T \widetilde{\boldsymbol{\mu}}_{c'} - \widetilde{\boldsymbol{\mu}}_k^T \widetilde{\boldsymbol{\mu}}_k) + \log \frac{\pi_{c'}}{\pi_k}$ is equivalent to the difference of the softmax biases $\widetilde{b}_{c'} - \widetilde{b}_k$. This allows us to conclude that the weights of a softmax for a given class are proportional only to the mean and the variance of the data points for a given class, and not the data point of other classes. Furthermore, while the mean of the data points in the assigned class defines the direction of the weight vector, the variance of the data defines the magnitude of the vector.

4.7 Conclusions

We show that retraining the network in a multitask learning fashion is able to produce higher number of examples for training the distribution over softmax weights. While the varying context of the weights does directly impact the obtained values of the retrained weights, the empirical variance improves with the smaller context sizes, possibly due to increased impact of noise in training process. The resulting larger number of training weights for concept learning improves both the quality of the prior distribution over weights and the final k-shot learning performance for more complex models such as GMMs. We conclude this chapter with a mathematically principled explanation for why the context of other classes does not directly impact the trained values of the softmax weights.



(a) Weights from $m = 100$ subsets of $l = 40$ classes. Each class on average occurs $40/80 \times 100 = 50$ times, thus around 50 columns per class.



(b) Weights from $m = 1000$ subsets of $l = 5$ classes. Each class on average occurs $5/80 \times 1000 = 62.5$ times, thus around 62.5 columns per class.

Fig. 4.2 Visualisation of weights for randomly selected classes from $\widetilde{C}$ retrained for $m_1 = 100$ subsets with $l_1 = 40$ classes and for $m_2 = 100$ subsets with $l_1 = 5$ classes. Each column of pixels represents a single weights vector $\mathbf{w}_{c'}$ with 128 features (rows). Quantitative results are reported below subfigure for each class. Mean, minimum and maximum are reported across all weight examples and their features; variance is an average of variances for each feature across weight examples.

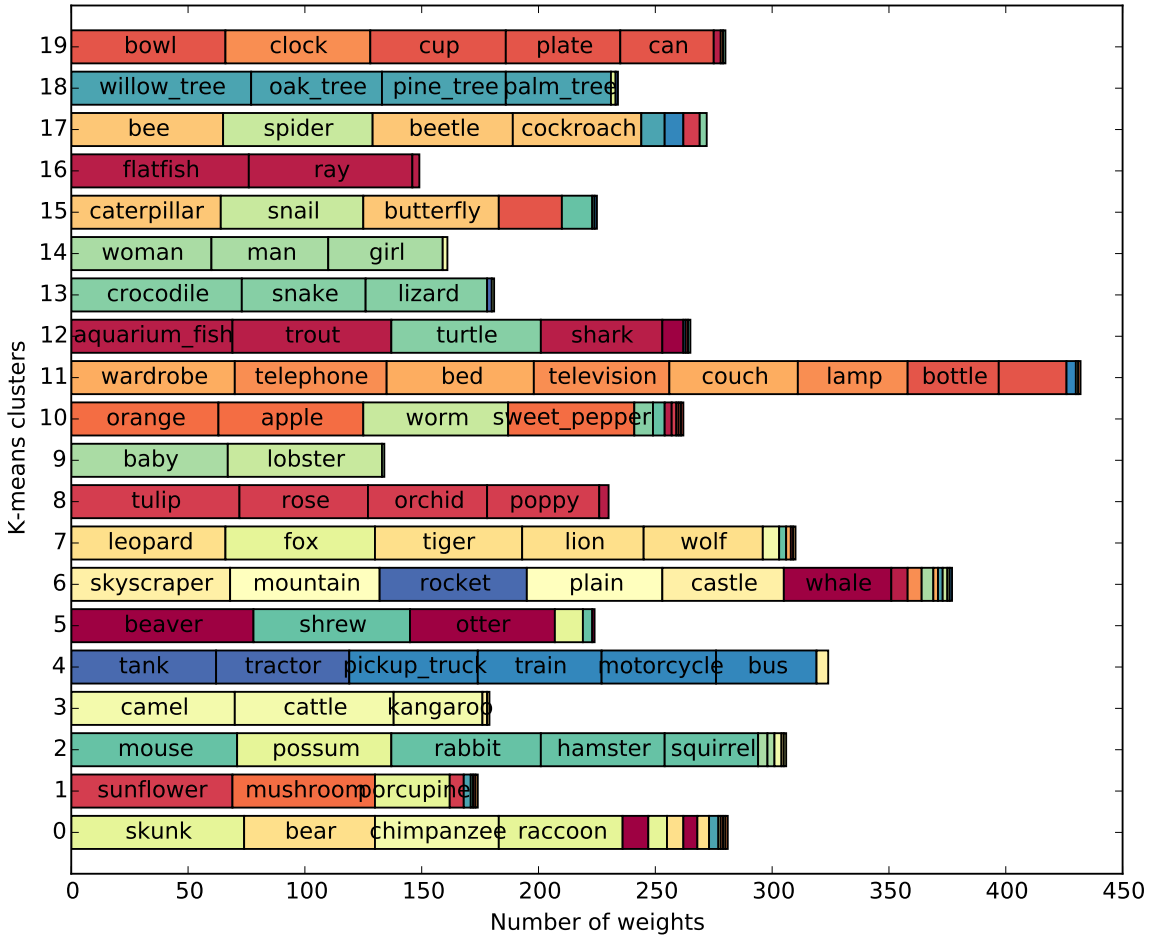


Fig. 4.3 Rows on the plot show 20 clusters obtained from k-means algorithm for the weights trained with $l = 5$ and $m = 1000$. The number of weights for each class assigned to a given cluster (row) is represented by the size of elements in the stacked bars (horizontally). Weights for classes that belong to the same superclass have the same colour.

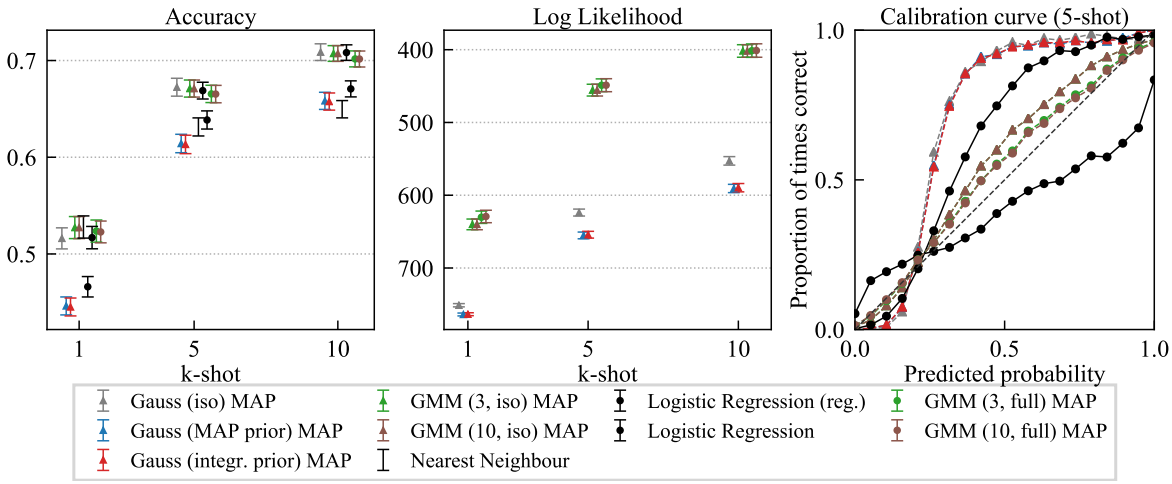


Fig. 4.4 CIFAR-100 results for k-shot learning on with priors trained on softmax weights obtained by retraining with $m = 1000$ sets each with $l = 5$ classes.

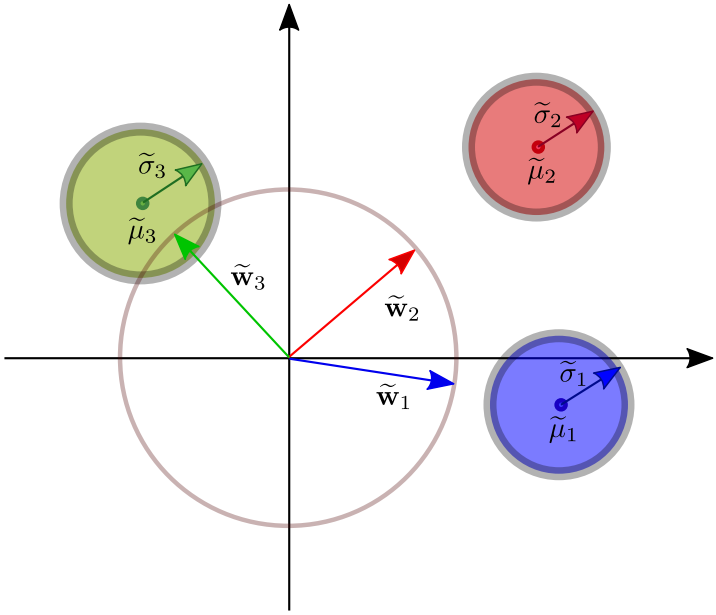


Fig. 4.5 Illustration of a 3-component GMM applied to a two dimensional problem with 3 classes of data represented by the green, red and blue circles. The weights $\widetilde{\mathbf{w}}_1$, $\widetilde{\mathbf{w}}_2$ and $\widetilde{\mathbf{w}}_3$ represent the softmax weights for each of the 3 classes.

Chapter 5

Extension 2 - Non-linear classifiers

5.1 Introduction

In the previous chapter we analysed the applicability of multi-set weight retraining to increase the number of examples of weights to train the prior over softmax weights. In this chapter we focus on improving the expressiveness of the classifier by replacing the linear softmax with a non-linear one-hidden-layer neural network.

Our motivation is that a CNN used during representational learning phase is trained to produce features that are linearly separable only for the training classes and not directly the test classes. Therefore, our hypothesis is that a classifier that is able transform the feature space into linearly separable space for the test classes should increase the classification performance. Verifying this hypothesis does not show the expected improvement from transforming the space, which we attribute to a strong tendency of the neural network to overfit the training examples. We propose next steps for further analysis that could potentially address this issue. In the rest of this chapter we describe adaptations made to the probabilistic k-shot learning framework to handle the neural network classifier and report experimental results from testing the introduced hypothesis.

The work described in this chapter is completed solely by me with the guidance from Richard E. Turner.

5.2 Probabilistic NN framework

The probabilistic Neural Network (NN) framework builds on the multi-set training approach introduced in the previous chapter, but instead of retraining the softmax

weights we retrain a one-hidden-layer neural network on top of the feature activations from the CNN. The illustration of this setup is shown in Fig. 5.1. An important parameter of this framework is the size of the hidden layers that we denote as H in the following notation.

5.2.1 Representational learning

The representational learning phase consist of two sub-phases similarly to the multi-set training. In the first sub-phase we train the original feature extractor Φ_φ on top of all the training classes $\widetilde{C}$ and fix the parameters φ up to the last hidden layer. The feature extractor is trained with a standard $\widetilde{C}$-classes softmax layer as the last layer. In the second sub-phase we use the last hidden layer activations from the large dataset $\widetilde{\mathbf{x}}$ to train a large number m of one-hidden-layer neural networks that classify subsets of the training classes with l classes in each subset. In the next section we analyse in more detail the impact and interaction between the size of a subset l and the hidden layer size H.

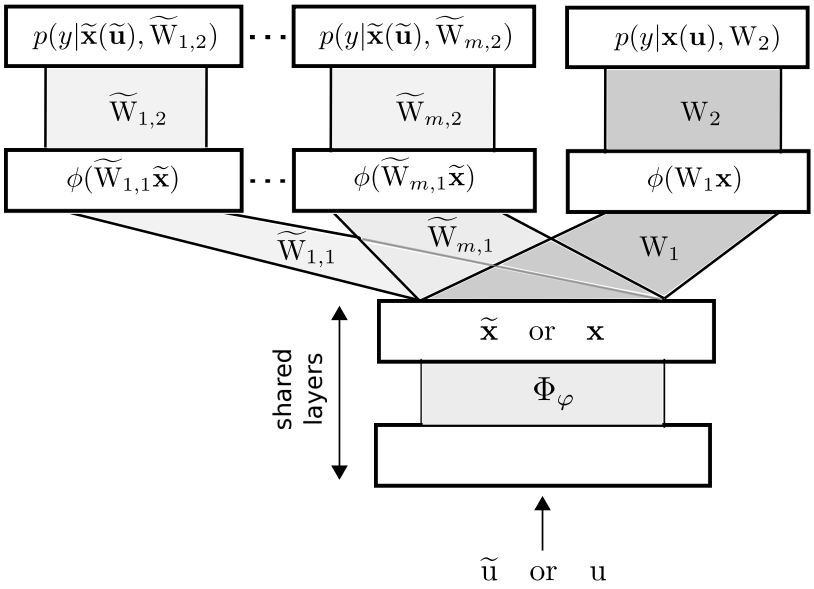


Fig. 5.1 Shared feature extractor Φ_φ trained on all training classes $\widetilde{C}$ as in the general setup, but instead of obtaining a single set of weights $\widetilde{\mathrm{W}}$ for all training classes $\widetilde{C}$, we retrain many one-hidden layer neural networks with weights $\widetilde{\mathrm{W}}_{m,1}$ and $\widetilde{\mathrm{W}}_{m,2}$ for randomly selected subsets of classes from $\widetilde{C}$.

We initialise the weights of the one-hidden-layer neural networks depending on the chosen architecture of the networks (the hidden layer size H and the number of outputs l). For the case with equal number of hidden units H and output classes l we

initialise the weights of the last layer using the identity matrix. Otherwise, we initialise all the weights using a truncated Gaussian with zero mean and one standard deviation. Furthermore, we use weight decay on the weights of the network. Similarly to the multi-set set training the regularisation is essential for the weights across subsets to be comparable.

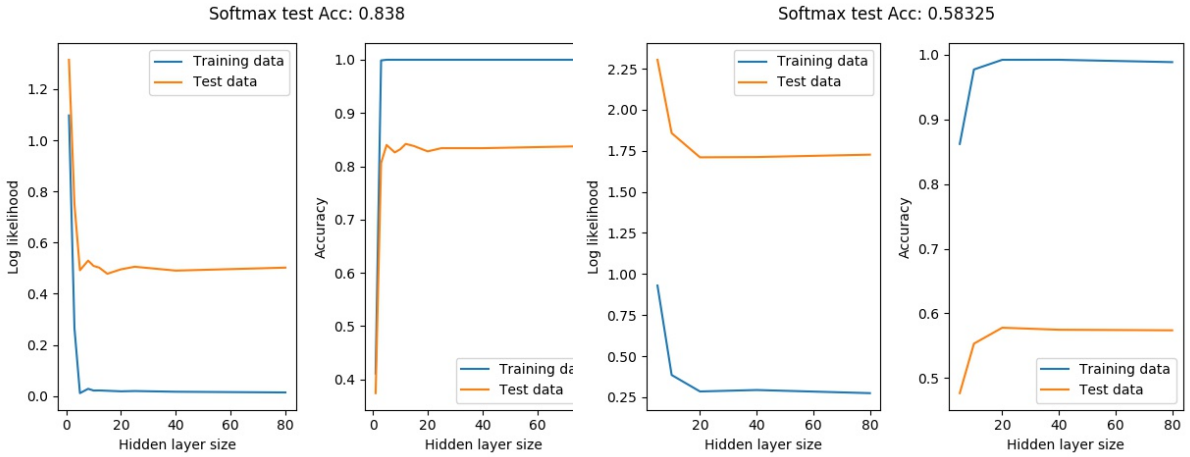
Experimental results

We first confirm that the test accuracies obtained by the NNs on the training classes during multi-set retraining are consistently similar or slightly better than that achieved by the softmax. This holds across different subsets of classes.

We further analyse the impact of the hidden layer size on the test accuracy and log likelihood for two sizes of the last softmax layer $l = \{5, 80\}$. Fig. 5.2 shows that in both cases the test accuracy plateaus at a specific hidden layer size. For $l = 80$ the test accuracy stops improving after around $H = 20$ and for $l = 5$ the network reaches is maximum capacity at $H = 5$. In both cases such optimal hidden layer size defines the minimum amount parameters that is require to perform accurate classification. Minimising the number of the model parameters is important for concept learning and k-shot learning where we are given only a small number of training examples to train the parameters.

However, it is also important to observe that with just a single set of weights $m = 1$ as with the subset of $l = 80$ classes, the smaller size of the hidden layer results in decrease in the number of training examples for learning the distribution over the weights of the first layer $\widetilde{\mathrm{W}}_{m,1}$ in the concept learning phase. The decrease in the number of examples (e.g. to 20) is particularly challenging for learning the distribution of the hidden layer weights because of their usually high dimensionality (e.g. 128-dimensional in our case). On the other hand, increasing the hidden layer size results in the increase in the number of model parameters which is unfavourable during the k-shot learning stage when we learn the model parameters from just a few examples.

Therefore, we investigate the application of the multi-set training developed in the previous chapter. Similarly to the previously analysed multi-set training setup, we use size of sets $l = 5$. In this setting the optimal size of the hidden layer is 5 as observed on Fig. 5.2. Training m neural networks allows us to obtain large number of examples for the hidden layer weights while keeping the dimensionality of the hidden layer small. In this case, the total number of model parameters is $5 \times 128 + 5 \times 5 = 665$, thus only 25 parameters more than in the single-layer softmax case considered in the general setup.



(a) Sets with $l = 5$.

(b) Sets with $l = 80$.

Fig. 5.2 Tuning of a hidden layer size in a one-hidden-layer neural network with set sizes $l \in \{5, 80\}$ on the CIFAR-100 data.

5.2.2 Concept learning

During the concept modelling stage we learn the distribution over possible weight values for each layer independently. For both layers we use the same variations of the Gaussian distribution as analysed already in the previous chapters. We also use the same procedure for tuning the hyper-parameters of the distributions. Table 5.1 and Table 5.2 show held-out negative log probabilities for the distributions over respectively the hidden layer weights and the final softmax layer weights. The reported values for the final softmax have much smaller magnitude due the small dimensionality of the weights ($H = 5$). In both bases the multivariate models provided better estimates for the held-out weights than the isotropic model.

Model	Optimised held-out neg. log prob.
Gauss (iso)	-286.2 ± 0.7
Gauss (MAP prior)	-338.0 ± 0.5
Gauss (integr. prior)	-346.9 ± 0.7

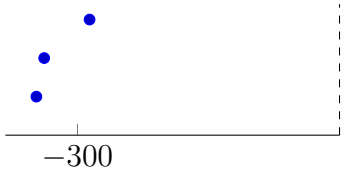


Table 5.1 Held-out negative log probabilities on random 70/10-splits of the training weights for the hidden layer weights of the NN on CIFAR-100 (lower is better) obtained by retraining with $m = 100$ sets each with $l = 5$ classes. Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

Model	Optimised held-out neg. log prob.
Gauss (iso)	-1.14 ± 0.01
Gauss (MAP prior)	-3.93 ± 0.02
Gauss (integr. prior)	-3.94 ± 0.01

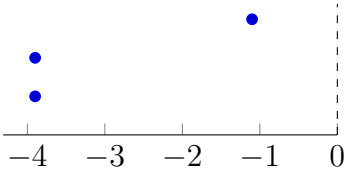


Table 5.2 Held-out negative log probabilities on random 70/10-splits of the training weights for the softmax layer weights of the NN on CIFAR-100 (lower is better) obtained by retraining with $m = 100$ sets each with $l = 5$ classes. Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

5.2.3 K-shot learning and testing

In the k-shot learning procedure we train a one-hidden-layer network using standard gradient descent optimiser (Adam) and use the priors of the weights for each of the layers trained during the concept learning phase.

Size of the hidden layer H of the network strongly influences the number of model parameters that need to be trained during the k-shot learning. The number of parameters in the network with 128-dimensional activations includes $128 \times H$ parameters for the hidden layer and $H \times l$ parameters for the last softmax layer. In total, we learn $H \times (128 + l)$ model parameters. Therefore, we want to use a small H to reduce the total number of parameters in the model.

In the following two subsections we investigate the performance of the probabilistic neural network framework in the k-shot learning setting and then move the analysis to the large data setting.

K-shot learning setting

Fig. 5.3 shows that the neural networks with the Gaussian priors from the concept learning stage achieve similar performance to that of the Gaussian model from the multi-set training framework (see Fig. 4.4), which is treated as a baseline in this setting. Their performance is also on pair with the logistic regression without regularisation. This could suggest that the neural networks were not able find more linearly separable space for the test classes. However, in the k-shot learning setting this could be attributed to the too low number of training examples given the complexity of the model. Therefore, in the next subsection we analyse the performance of the neural network framework in the large data setting.

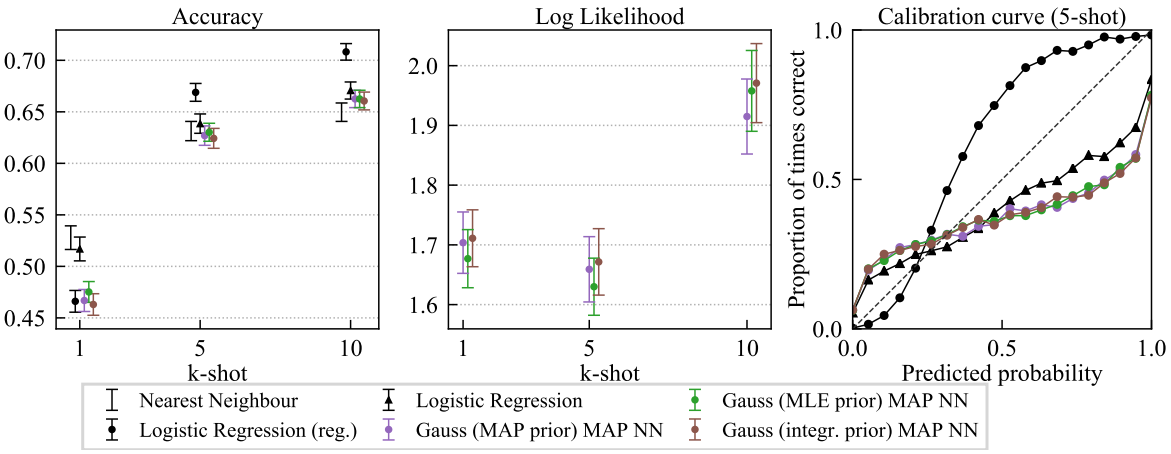


Fig. 5.3 K-shot learning 5-way classification results for the probabilistic neural network framework with $H = 5$, $l = 5$ and $m = 100$ on CIFAR-100 dataset averaged over 20 splits and 5 restarts with different training images.

Finally, the calibration curves show overly confident predictions of the neural networks. This suggests overfitting to the training examples, which most likely decreases the model performance.

Large data setting

In the large data setting the small number of training points is no longer a bottleneck for estimating the model parameters, and thus we are able to obtain the oracle performance. Furthermore, on the large training dataset the priors on the weights have smaller impact on the performance, which allows us to compare the ability to transform the feature space for both the training and test classes without other considerations.

Fig. 5.4 shows that the relative performance of the neural network models and the logistic regression stayed the same as in the low data setting. This suggests that the higher number of model parameters should not be a problem in the k-shot learning setting. However, the neural network models still show overfitting to the training data and poor calibration.

Comparing classification performance of the logistic regression on the test classes (Fig. 5.4) and training classes (Fig. 5.5) reveals a strong decrease for the test classes. This suggests potential room for improvement in terms of transforming the feature space extracted for the CNN for the test classes. However, the analysed neural network models do not appear to show such ability. We can possibly attribute this to the overfitting problem, which degrades the performance of the network in general. Alternatively, we can hypothesise that the features for the test classes do not contain information that

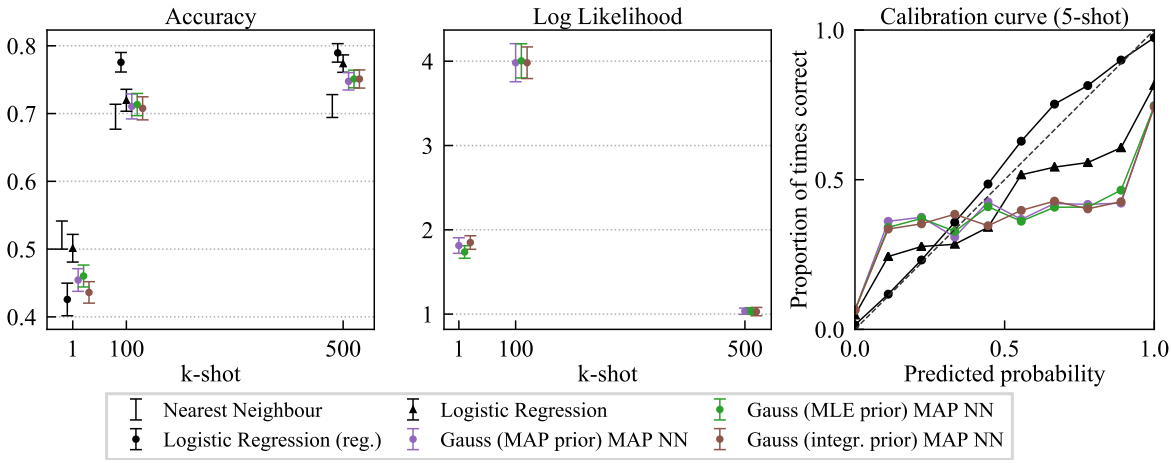


Fig. 5.4 Many-shot ($k \in \{1, 100, 500\}$) 5-way classification results for the probabilistic neural network framework with $H = 5$, $l = 5$ and $m = 100$ on CIFAR-100 dataset averaged over 20 splits of the test classes.

is required to classify the test classes accurately or that the analysed architecture of the neural network is not expressive enough to perform such transformation.

5.3 Conclusion

While the probabilistic neural network framework presents an interesting area for experiments, it did not provide intended results. We attribute this mostly to the tendency for strong overfitting to the training examples observed in the employed networks. We see multiple potential improvements that can be explored to address this problem. First, a stronger prior over network weights is expected to reduce the overfitting. The strength and calibration of the prior could be increased by training the prior distribution on higher number of examples for the weights during the representational learning phase (in this work we trained $m = 100$ neural networks). Alternatively, interesting area of exploration would involve employing a more expressive prior in the form of a joint distribution over weights for all the layers in the network.

Finally, by analysing the impact of the size of the hidden layer, we found the potential to achieve improvement in the probabilistic framework by modifying the size of the network layers. We explore this in more detail in the next chapter.

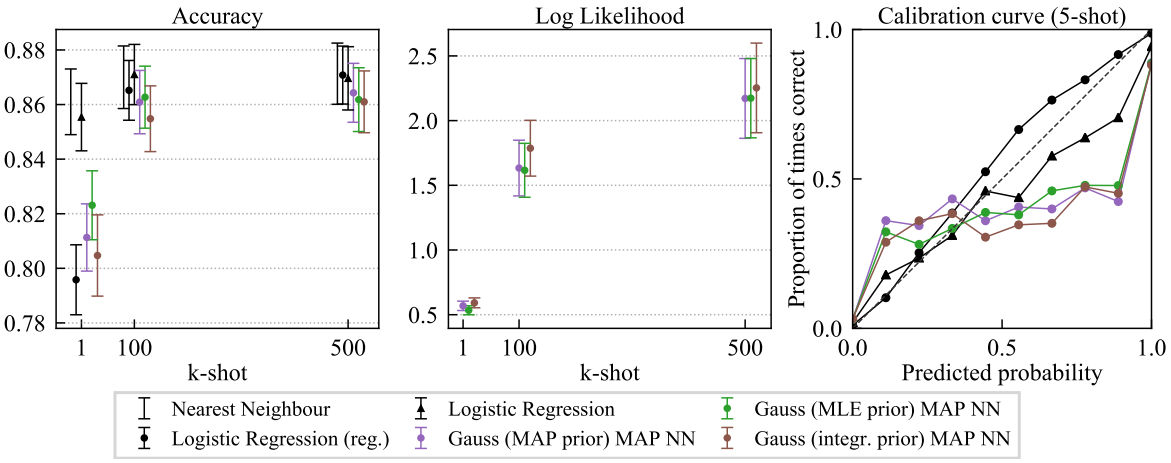


Fig. 5.5 Many-shot ($k \in \{1, 100, 500\}$) 5-way classification results for the probabilistic neural network framework with $H = 5$, $l = 5$ and $m = 100$ on CIFAR-100 dataset averaged over 20 splits of the training classes.

Chapter 6

Extension 3 - Bottleneck layer

6.1 Motivation

In this chapter we investigate the impact of the dimensionality of the embedding feature space obtained from a CNN on the k-shot learning performance. The dimensionality of the feature space will be modified by, amongst others, an additional bottleneck layer at the top of the previously analysed CNN. While powerful feature extractors obtain rich feature representations that generalise well to unseen tasks, the dimensionality of such features is often very high. In the setting of the probabilistic k-shot learning framework, where we need to learn model parameters from just a few examples, such high dimensionality of the problem poses a challenging task.

In this chapter we analyse the trade-off between the expressivity of the highly-dimensional rich feature representations and robust estimation of model parameters during concept learning and k-shot learning using low-dimensional features. Our analysis are performed by varying the size of the feature space extracted during the representational learning phase and evaluating the resulting models based on the final k-shot learning performance. We conclude that in the analysed setup, a rich feature representation has in fact more impact on the k-shot learning performance than a robust estimation of the prior for the model weights.

The work described in this chapter is completed solely by me with the guidance from Richard E. Turner.

6.2 Representational learning

We start our analysis by comparing the test accuracies of CNN feature extractors trained on the large training dataset $\widetilde{\mathcal{D}}$ w.r.t. the size of last hidden layer. We compare five network architectures: the architecture from the general setup (see Table 3.1) and its four modifications. The first three architectures are constructed by adding a new fully-connected hidden layer (bottleneck layer) of size 20, 40 and 60 after the 128-dimensional hidden layer from the general setup. The fourth architecture is created by replacing the 128-dimensional hidden layer from the general setup with a 256-dimensional hidden layer.

Table 6.1 shows 80-way test accuracies on the training classes for each of the CNN architectures. The results show a decrease in accuracy for smaller feature spaces, but the decrease is not drastic and thus provides a promising avenue for further experiments. Additional inspection of the PCA explained variance reinforces this claim (see Fig. 6.1).

Bottleneck dim.	20	40	60	128	256
Test acc.	56.9%	57.5%	58.3%	59.0%	60.%

Table 6.1 CNN 80-way validation accuracy on training classes w.r.t the size of the bottleneck layer (last hidden layer).

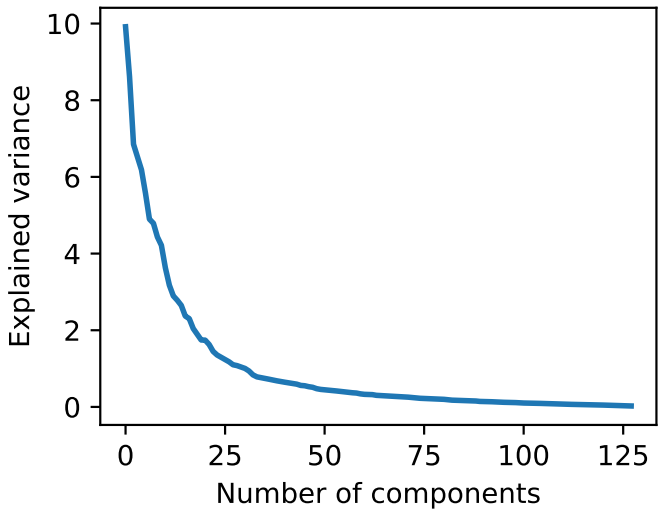


Fig. 6.1 PCA explained variance per component of the activations from the 128-dimensional hidden layer.

We proceed with analysing the quality of the features extracted for both the training classes and the test classes. We evaluate the features using the k-shot learning procedure with regularised logistic regression, which does not use the prior knowledge on the weights from the concept learning. The logistic regression is tested on a 20-way classification task with 5-shots. Table 6.2 shows that while the accuracy on the training classes slightly decreases with the decrease in the dimensionality of the bottleneck layer, the decrease for the test classes if twice as large in magnitude.

Class type	Size of bottleneck		
	128	40	20
Training classes $\widetilde{C}$	71%	68%	69%
Test classes C	42%	39%	38%

Table 6.2 5-shot accuracies on 20-way classification for CIFAR-100 using regularised logistic regression w.r.t class type and size of the bottleneck layer.

For further investigation we visualise the difference in how separable are the activations for the test examples of the test classes in both the 128-dimensional case (see Fig. 6.2) and the 40-dimensional case (see Fig. 6.3). The t-SNE visualisations confirm that the activations from the 128-dimensional layer are more separable.

6.3 Concept learning

Even though the features with lower number of dimensions have lower discriminative power for the test classes, the prior knowledge learned on the weights should be more robust because of the lower number of parameters to estimate in the distributions over weights. Table 6.3 shows that in fact after dimensionality reduction to 40-dimensions the held-out negative log probabilities of more complex models such as a GMM increase relatively to that of the Gaussian models. This suggests an improvement in the concept learning results over the baseline from the general setup (see Table 3.2).

Furthermore, Table 6.4 shows that increasing the dimensionality of the feature space to 256 dimensions decreases even more the relative performance of the more complex models. This supports the claim that the reduction in dimensionality improves the robustness of the estimated distributions over weights.

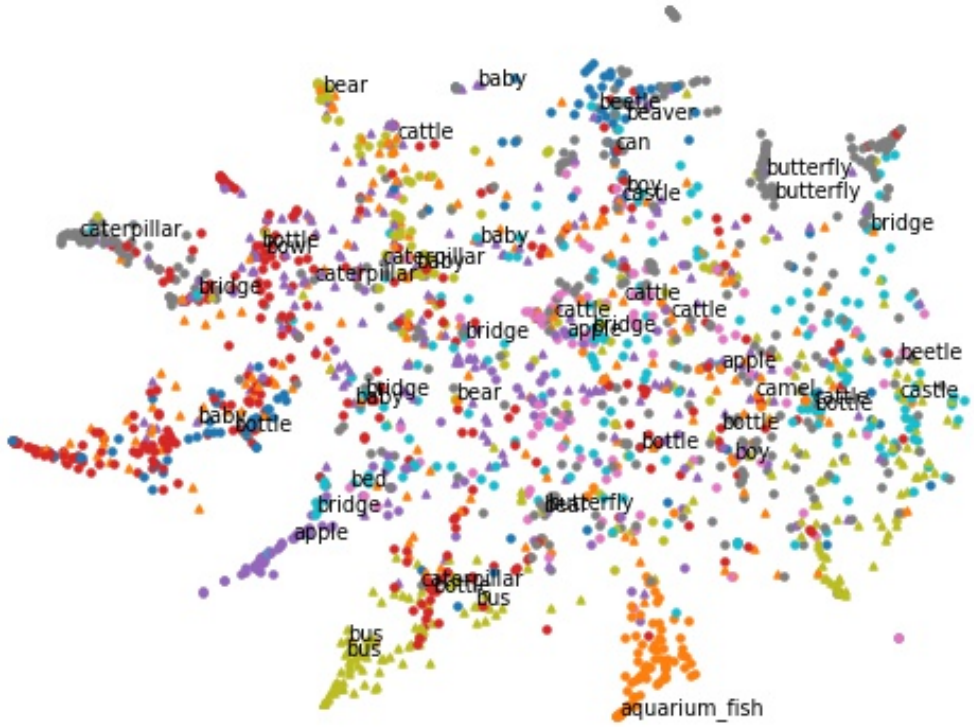


Fig. 6.2 T-SNE visualisation of activations for the test examples of the test classes from the 128-dimensional hidden layer.

6.4 K-shot learning and testing

Finally, we evaluate k-shot learning performance of the models w.r.t. the dimensionality of the last hidden layer and compare it to the baseline from the general framework (see Fig. 3.5). The comparison shows that the models with the 256-dimensional features space (Fig. 6.4) outperform both the models with the 40-dimensional space (Fig. 6.4) and the results from the general setup, irrespectively of the prior distribution. This indicates that in the analysed scenario the increase in the expressiveness of the feature representation (analysed in Table 6.2) has a stronger impact on the k-shot learning performance, and thus transfer learning, than the improvement in modelling the prior distribution using the lower dimensionality of the feature space (Table 6.3).

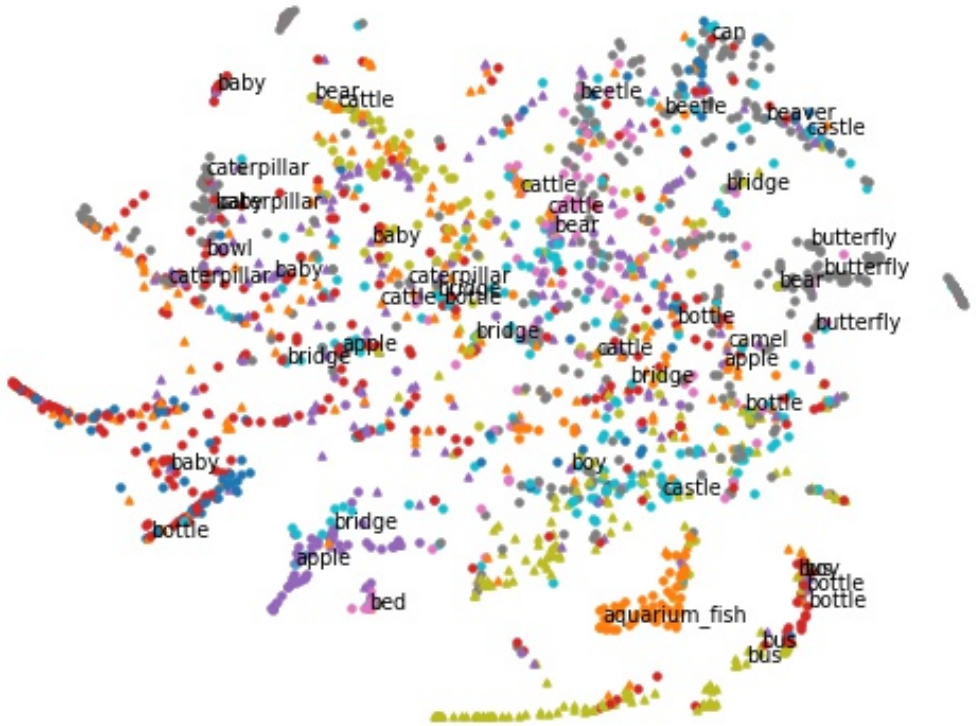


Fig. 6.3 T-SNE visualisation of activations for the test examples of the test classes from the 40-dimensional hidden layer.

6.5 Conclusions

In this chapter we analysed the trade-off between using highly-dimensional rich feature representations and low-dimensional features that result in more robust model parameter estimates. We have shown that the rich feature representations with higher dimensionality result in better k-shot learning performance of our framework improving the k-shot learning accuracies over the general setup by between 2% and 3% across different number of shots.

Model	Optimised held-out neg. log prob.
Gauss (iso)	-19.1 ± 0.2
Gauss (MAP prior)	-19.1 ± 0.2
Gauss (integr. prior)	-20.0 ± 0.1
GMM 1-means (iso) reg.	-18.9 ± 0.2
GMM 1-means (diag) reg.	-19.4 ± 0.2
GMM 1-means (full) reg.	-20.0 ± 0.1
GMM 3-means (iso) reg.	-20.1 ± 0.2
GMM 3-means (diag) reg.	-20.2 ± 0.2
GMM 3-means (full) reg.	-18.5 ± 0.2
GMM 10-means (iso) reg.	-19.0 ± 0.2
GMM 10-means (diag) reg.	-19.0 ± 0.1
GMM 10-means (full) reg.	-18.2 ± 0.1

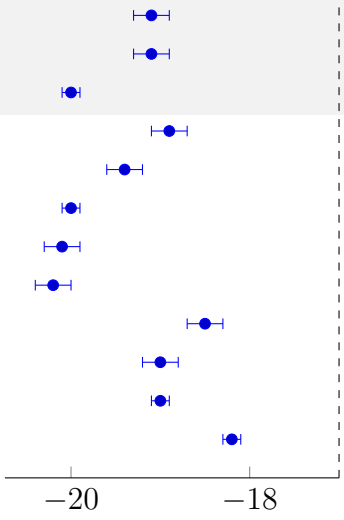


Table 6.3 Held-out negative log probabilities on random 70/10-splits of the 40-dimensional training weights on CIFAR-100 (lower is better). Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

Model	Optimised held-out neg. log prob.
Gauss (iso)	-428.9 ± 0.8
Gauss (MAP prior)	-465.4 ± 0.7
Gauss (integr. prior)	-474.2 ± 0.6
GMM 1-means (iso) reg.	-431.7 ± 0.6
GMM 1-means (diag) reg.	-441.8 ± 0.8
GMM 1-means (full) reg.	-472.4 ± 0.7
GMM 3-means (iso) reg.	-436.1 ± 0.7
GMM 3-means (diag) reg.	-442.8 ± 0.8
GMM 3-means (full) reg.	-451.6 ± 1.1
GMM 10-means (iso) reg.	-437.6 ± 0.8
GMM 10-means (diag) reg.	-439.9 ± 0.8
GMM 10-means (full) reg.	-427.3 ± 1.6

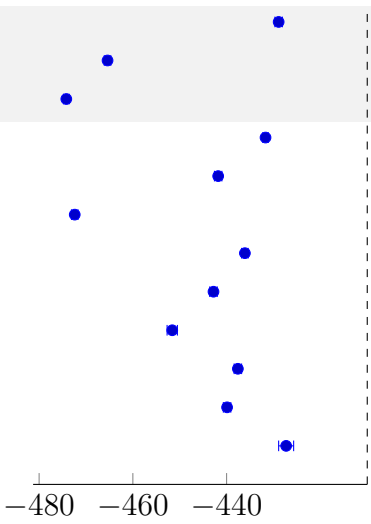


Table 6.4 Held-out negative log probabilities on random 70/10-splits of the 256-dimensional training weights on CIFAR-100 (lower is better). Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.

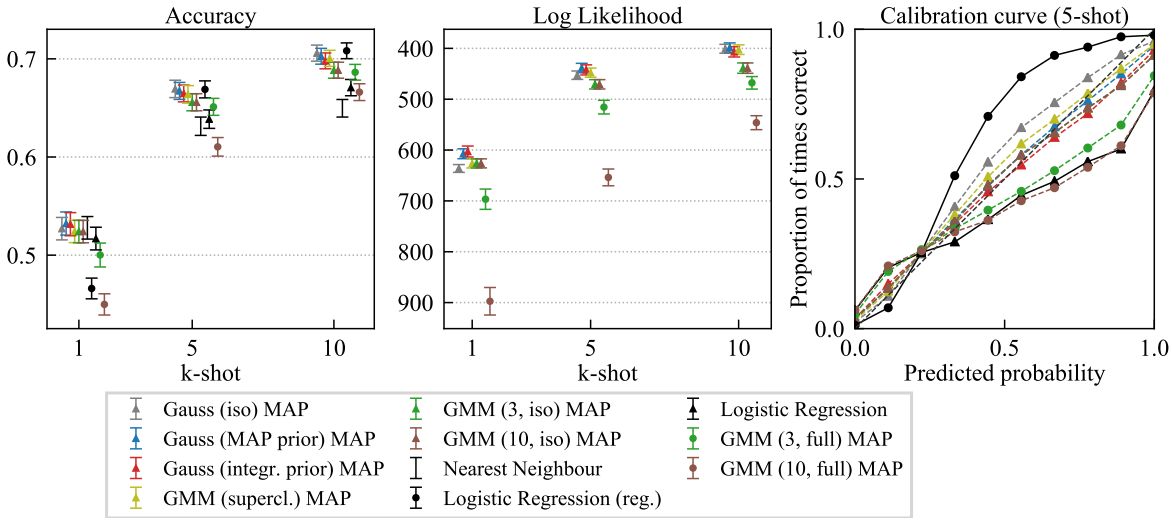


Fig. 6.4 K-shot learning results for 5-way classification in the 40-dimensional feature space.

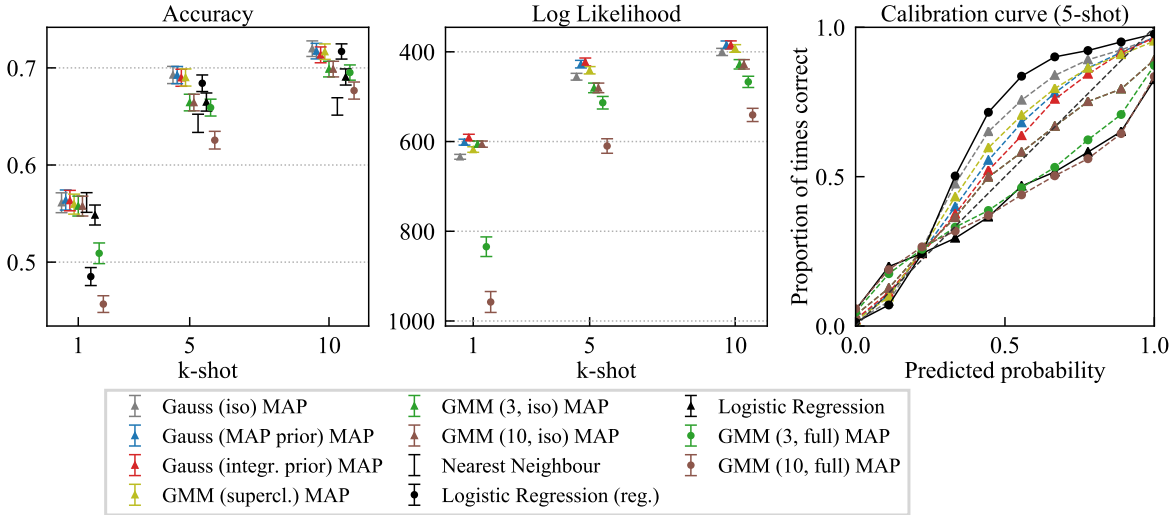


Fig. 6.5 K-shot learning results for 5-way classification in the 256-dimensional feature space.

References

B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

M. Bauer, M. Rojas-Carulla, J. B. Świątkowski, B. Schölkopf, and R. E. Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017.

P. Bloom. *How children learn the meanings of words.* The MIT Press, 2000.

J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

J. Burgess, J. R. Lloyd, and Z. Ghahramani. One-shot learning in discriminative neural networks. *NIPS Bayesian Deep Learning workshop*, 2016.

R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv e-print:1511.07289*, 2015.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 05 2015. URL http://dx.doi.org/10.1038/nature14541.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv e-print:1412.6980*, 2014.

G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. *Deep Learning workshop, International Conference of Machine Learning*, 2015.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

K. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

S. Qiao, C. Liu, W. Shen, and A. Yuille. Few-shot image recognition by predicting parameters from activations. *arXiv preprint arXiv:1706.03466*, 2017.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, volume 1, page 6, 2017.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52 (55-66):11, 2010.

J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. *arXiv e-print: 1703.05175*, 2017.

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

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016.

F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Appendix A

Supplementary material

A.1 Multi-set training with small number of examples per class

Model	Optimised held-out neg. log prob.
Gauss (iso)	-352.0 ± 0.6
Gauss (MAP prior)	-419.8 ± 0.4
Gauss (integr. prior)	-448.2 ± 0.6
GMM 1-means (iso) reg.	-351.0 ± 0.5
GMM 1-means (diag) reg.	-351.9 ± 0.5
GMM 1-means (full) reg.	-433.2 ± 0.9
GMM 3-means (iso) reg.	-353.7 ± 0.6
GMM 3-means (diag) reg.	-357.4 ± 0.6
GMM 3-means (full) reg.	-423.4 ± 1.2
GMM 10-means (iso) reg.	-354.3 ± 0.7
GMM 10-means (diag) reg.	-356.6 ± 1.0
GMM 10-means (full) reg.	-406.8 ± 1.0
GMM 20-means (full) reg.	-394.0 ± 1.0
GMM 40-means (full) reg.	-366.6 ± 1.5
GMM 70-means (full) reg.	-330.3 ± 1.6
	−400 −300

Table A.1 Held-out negative log probabilities on random 70/10-splits of the training weights on CIFAR-100 (lower is better) obtained by retraining with $m = 100$ sets $l = 5$. Values are optimised w.r.t. the hyperprior variances and averaged over 50 splits.