Transferring feature representations across tasks and domains

Ekaterina Iakovleva

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Mathématiques et Informatique
Unité de recherche : Laboratoire Jean Kuntzmann

Transfert de représentations de caractéristiques entre les tâches et les domaines

Transferring feature representations across tasks and domains

Présentée par :

Ekaterina IAKOVLEVA

Direction de thèse :

Karteek ALAHARI Chargé de recherche HDR, INRIA CENTRE GRENOBLE-RHONE-ALPES	Directeur de thèse

Rapporteurs :

JOOST VAN DE WEIJER
Chercheur, Universitat Autónoma de Barcelona
DAVID PICARD
Directeur de recherche, ECOLE NATIONALE DES PONTS ET CHAUSSEES

Thèse soutenue publiquement le **7 décembre 2022**, devant le jury composé de :

JOOST VAN DE WEIJER Chercheur, Universitat Autónoma de Barcelona	Rapporteur
DAVID PICARD Directeur de recherche, ECOLE NATIONALE DES PONTS ET CHAUSSEES	Rapporteur
MATTHIEU CORD Professeur des Universités, SORBONNE UNIVERSITE	Examinateur
MASSIH-REZA AMINI Professeur des Universités, UNIVERSITE GRENOBLE ALPES	Président

Invités :

DIANE LARLUS
Ingénieur docteur, NAVER LABS EUROPE
JAKOB VERBEEK
Ingénieur HDR, META AI RESEARCH

Abstract

Year after year advances in deep learning allow to solve a rapidly increasing range of challenging tasks, as well as to set new, even more ambitious goals. Such a success, however, comes at a price of increasing requirements for all aspects of learning: large-scale models, which tend to perform best, require large quantities of data, memory, computational resources and time to be properly trained. This cannot always be achieved in practice, especially on smaller datasets, which promotes exploration of the ways to transfer knowledge, i.e. re-purposing existing state-of-the-art models to solving new tasks.

This problem of transferring knowledge between tasks comes with its own challenges caused by different factors, such as the type of knowledge which needs to be transferred, or availability of data. In this thesis we focus on two setups from this category: few-shot learning and multi-domain learning. Both problems share the motivation to learn a model that would be able to generalise to solving the same type of task, e.g. image classification, on a number of different domains.

Our first contribution explores probabilistic modeling for few-shot classification, where the model aims to solve a wide range of classification tasks, each accompanied with a handful of labeled examples. Limited supervision leads to high uncertainty about the predictions, which can be naturally tackled by probabilistic framework. We treat the task-specific classifier as a latent variable, and propose a novel amortised variational inference scheme which uses a single network to predict parameters of the distribution both for the prior and for the approximated posterior of the latent variable in the considered graphical model. The prior is conditioned on the support set of the task, while the approximated posterior is conditioned on the union of the support and query sets. Minimisation of the distance between these two distributions provides additional guidance from the support set during training, allowing us to exploit the disparity between the two sets of data. We evaluate our model on several few-shot classification benchmarks, and show that it can achieve competitive results on all of them. We also demonstrate the benefits of modeling uncertainty by showing that a sampled ensemble of classifiers slightly improves the performance compared to the inferred classifier mean. This result that cannot be achieved by models relying on Monte Carlo approximations, which, according to our experiments, tend to underestimate the true variance.

Our second contribution proposes a novel type of adaptation modules for multi-domain classification, which considers a fixed set of classification tasks on a limited number of domains. We adopt the common approach of using a pre-trained feature extractor as a base network, and adjusting it to novel domains using domain-specific adapters applied to each convolutional layer in the base. For each output channel, our modulation adapter separately scales each kernel in the convolutional weight tensor with its own output-specific scalar. This results in a set of independent re-weightings of the input feature maps, which makes the resulting feature adaptation more flexible compared to previous approaches. To reduce the parameter budget, we factorise our modulation adapter as a product of the two smaller matrices. We evaluate our model on the two common multi-domain classification benchmarks, and show that both the full and the factorised versions achieve state-of-the-art results. In addition to that, we provide results of our model on a span of parameter budgets, which is one of the advantages of our approach. For each parameter budget, modulation adapters outperform the competitors which generally offer only a single budget setting.

Acknowledgements

I would like to thank my husband Roman, who has been my biggest support during the PhD. This journey would not have been possible without you. I am grateful for all the patience and care you have been treating me with every single day, for always finding the best advice in work-related discussions, and for being the best companion in everything else.

I would also like to express my gratitude towards my PhD advisors, Jakob Verbeek and Karteek Alahari. I am grateful for the time and dedication you have invested in out projects, for your invaluable insights and all the pieces of advice you have shared with me during our work. I deeply appreciate your patience and support throughout these years.

I want to thank my colleagues in THOTH for the cheerful coffee breaks and lunches we have shared. I am also very grateful to my office roommates for being there to complain about the Reviewer 3, and to share the burden of the failed experiments. Special thanks to Nathalie Gillot for always taking care of each PhD student, and for your superpowers when dealing with bureaucracy.

Finally, I would like to thank my family and my friends for all the love and support. I have been able to feel it despite the thousands of kilometers between us.

Table of contents

List of figures

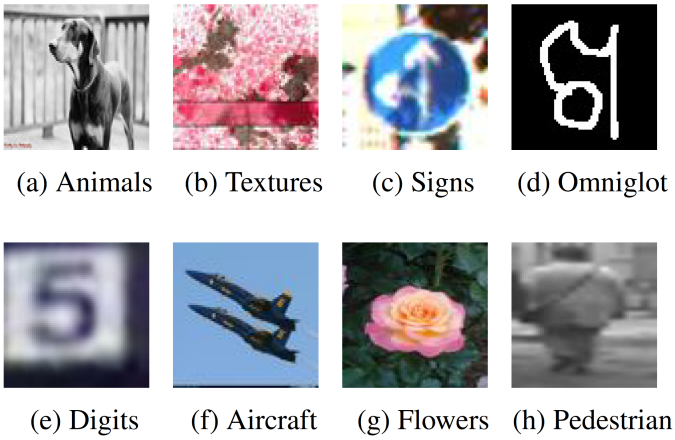
List of tables

Chapter 1

Introduction

Data has become one of the most valuable assets in the last few decades. Learning how to extract knowledge from data provided the humanity with opportunities to solve tasks that had not been successfully solved before, like learning to play Go (Silver et al., 2016) or transferring image style (Gatys et al., 2016), and to gain better insights into different areas of life, such as healthcare (Poplin et al., 2018), biology (Xu et al., 2020) or material physics (Schmidt et al., 2019). There are many scientific disciplines that use data as a primary subject of research, such as statistics, data analysis and data mining, which can be grouped into the category *data science*. Machine learning is among the first disciplines in this field that benefited from the growing amount of data. Its primary goal is to leverage the experience in order to learn an inference model generalisable to unseen data. Complexity of machine learning methods goes hand in hand with complexity of data itself, forming a positive feedback loop: the more advanced the methods get, the more data and the more complex data structures they tend to rely on, e.g. images accompanied with data from LiDAR (Geiger et al., 2012) or text descriptions (Lin et al., 2014), and vice versa.

Classical machine learning is a term that represents early methods of data processing and inference, e.g. linear and logistic regressions, k-Nearest Neighbour (k-NN), Support Vector Machine (SVM), Decision Tree and many others (Bishop, 2006; Murphy, 2012). It focuses both on supervised problems, such as classification and regression, and on unsupervised ones, like clustering and anomaly detection. A common aspect of classical machine learning is the sensitivity of the models to the quality of input data: noisiness and collinearity of features are the known factors which negatively affect the model's performance. Another challenge is the *curse of dimensionality* (Hughes, 1968; Bellman, 1957) caused by the growing amount of features that describe the data, which leads to exponentially growing requirements for the amount of training examples. As a result, proper feature selection and engineering must be applied prior to feeding data into a model.



(a) Animals (b) Textures (c) Signs (d) Omniglot

(e) Digits (f) Aircraft (g) Flowers (h) Pedestrian

Fig. 1.1 Examples of images from different domains. Illustration taken from (Guo et al., 2019a).

Deep learning and neural network development allow to deal with some of the challenges in machine learning discussed above. Automatic feature learning performed by the networks alleviates reliance on heuristics for feature engineering, ensuring direct optimisation of data representations for the task at hand. Over the last 50 years, advances in deep learning not only improved performance on the existing tasks, ranging from image classification (Krizhevsky et al., 2012; He et al., 2016a) to speech recognition (Baevski et al., 2021) and natural language processing (Devlin et al., 2018), but also offered solutions to more challenging problems, such as text-to-image generation (Ramesh et al., 2021), prediction of protein folding (Jumper et al., 2021), self-driving (Mullapudi et al., 2018; Bansal et al., 2019), etc. While neural networks scale more efficiently with the amount of training data than models in classical machine learning, their significant drawback is poor performance on smaller datasets. This is the case due to over-parameterisation of neural networks, as it allows to remember the observed examples instead of learning from them. Large number of parameters also requires significant amount of memory to train and store the models, creating another challenge for deep learning. At the same time, abundance of training examples and memory cannot always be guaranteed, which provides motivation for parameter re-use and transfer of extracted knowledge from one task to another.

1.1 Transfer of knowledge

The core idea behind knowledge transfer is to learn how to leverage models that have been trained on some task in order to learn another task. The most common example of knowledge transfer is pre-training of a deep neural network on a large (un)labelled dataset, e.g. on

	DA	TT	MTL	CL	FSL	MDL
Task type is shared	✓			✓	✓	✓
Domain is shared		✓	✓			
All tasks / domains are available simultaneously	✓		✓			✓
Observed tasks / domains can be revisited	✓	✓	✓		✓	✓
Performance on all task is equally important		✓	✓	✓	✓	✓
Task / parameter budget is limited		✓				✓

Table 1.1 Differences between the tasks that involve transfer of knowledge. *DA*: domain adaptaion, *TT*: task transfer, *MTL*: multi-task learning, *CL*: continual learning, *FSL*: few-shot learning, *MDL*: multi-domain learning.

ImageNet (Russakovsky et al., 2015), and then finetuning it on a smaller target dataset. Beyond pre-training and finetuning, there is a wide range of learning paradigms that involve some form of knowledge transfer. They differ from each other in the relationships and interactions that are formed between the tasks, as well as in the number of tasks that are considered in the first place, or the amount of data that is available during training.

There are several task attributes that can be used in various combinations to define a transfer learning problem. In this thesis, we distinguish the transfer paradigms by considering the following set of questions:

- *Is the task type shared across all tasks?* An example of two different task types would be object classification and image segmentation, as shown in Figure 1.2. At the same time car classification and flower classification tasks are considered to be of the same type.

- *Is the data domain shared across all tasks?* Here, the domain is defined not only by the data distribution, but also by the corresponding label space, as in Figure 1.1. For example, in few-shot learning each task can be generated by subsampling a small number of classes from the original dataset, which creates multiple associated task-specific domains. An example of the shared domain would be learning to solve different tasks on the same dataset, as in Figure 1.2.

- *Are the domains and/or tasks available simultaneously?* This question defines whether joint training is possible, or whether the tasks should be considered in isolation from each other.

- *Is it possible to revisit the task which has already been trained on?* The ability to finetune the trained model can significantly improve the model's performance when the tasks are not available at once, and vice versa.

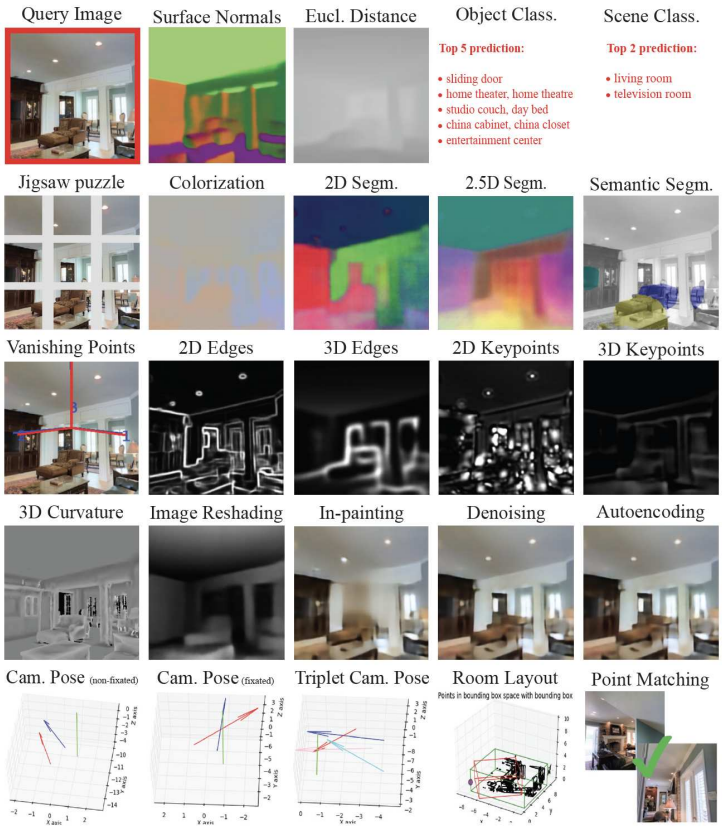


Fig. 1.2 Examples of different tasks on the same domain. Illustration taken from (Zamir et al., 2018).

- *Are the tasks equally important in terms of their performance?* The positive answer to this question poses an additional challenge of preserving the extracted knowledge from one task while switching to learning on another one.
- *Is there any type of restriction imposed on the model?* An example would be a limited parameter budget available for the model, or the limited amount of tasks and/or domains that can be used for training.

Table 1.1 provides a comparison of the most popular transfer learning problems using their representation via the chosen attributes. Below, we take a closer look at the mentioned transfer setups: we briefly discuss the motivation for each of them, and outline the corresponding challenges that need to be addressed.

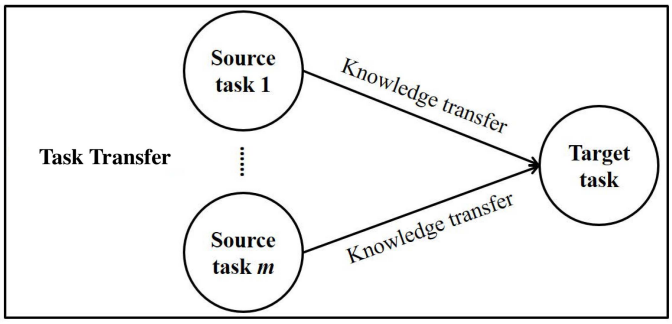


Fig. 1.3 Task transfer paradigm. Illustration taken from (Zhang & Yang, 2021).

1.1.1 Domain adaptation

In domain adaptation (Glorot et al., 2011; Ganin & Lempitsky, 2015), the task – or at least its type – remains the same during the train and the test phases, but the datasets come from different data distributions, unlike standard supervised machine learning. This might be caused by multiple reasons, e.g. by changes in the process that generates data, varying conditions and sources of data acquisition, modifications of the data processing methods. The common ground in all of these cases is the discrepancy between the domains in the train and the test datasets, which are referred to as the *source* and the *target domains*. Although the number of domains under consideration is not necessarily limited to two, it is usually assumed to be a small number.

Domain adaptation problems can be further split into several categories defined by the relationships between the source and the target domains. These relations depend not only on the corresponding data distributions, but also on the associated label spaces, ranging from the setup where all labels are shared, to the setup where either of the domains have additional private categories (Farahani et al., 2021). The standard pipeline with training on the source set and testing on the target set results in performance drop, so the main goal in domain adaptation is to learn how to generalize a model trained on one domain to another domain by directly addressing the domain shift and aligning the data distributions.

1.1.2 Task transfer

In contrast to domain adaptation, task transfer (Zamir et al., 2018) fixes the data domain while considering a collection of tasks on it, e.g. image segmentation, colorization, point matching etc. The goal is to maximise the overall performance on all of these tasks combined, while imposing a *supervision budget*, i.e. the maximum number of tasks that can be trained

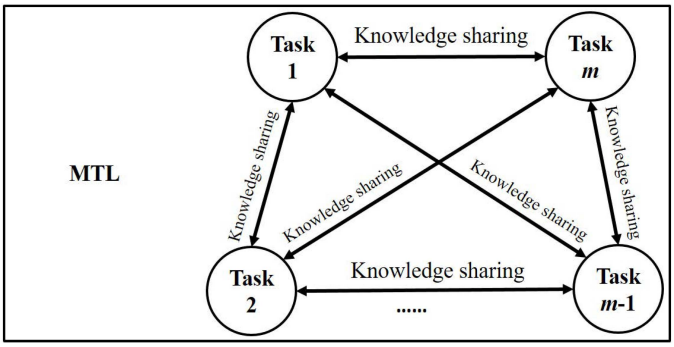


Fig. 1.4 Multi-task learning paradigm. Illustration taken from (Zhang & Yang, 2021).

from scratch. These tasks are denoted as the *source tasks*, and the knowledge obtained from learning on them is then transferred to the set of *target tasks*, as shown in Figure 1.3.

The core motivation of task transfer is to reduce the supervision requirements in deep learning by utilizing relatedness and complementarity of the tasks. While it is possible to apply the standard supervised learning quite efficiently, separately for each task in the source and the target sets, such approach would require a lot of labeled data which might be difficult or expensive to obtain, especially for dense tasks like semantic segmentation or depth estimation. The learning pipeline in transfer learning consists of several parts, including supervised learning on the source set of tasks, and learning transfer functions which adapt the trained models to solving novel tasks from the target set. An important notion introduced in task transfer is transferability between various pairs of tasks, which is measured in order to structurise the space of deep learning tasks and to create a taxonomy (Zamir et al., 2018).

1.1.3 Multi-task learning

Similar to task transfer, multi-task learning (MTL) (Kokkinos, 2017; Misra et al., 2016) aims to maximise the collective performance on a fixed set of tasks. There is no explicit restriction on the number of domains, however, MTL benchmarks usually include one domain simultaneously labeled for several different tasks (Lin et al., 2014; Zamir et al., 2018), as in task transfer. The principal difference between these two problems is that MTL does not impose any supervision budget, and the entire available supervision is utilised for training, meaning that all tasks are trained concurrently, as shown in Figure 1.4. MTL does not favor any task over the others, which distinguishes it from the problems that focus on a single target task, while the remaining auxiliary tasks are of no interest, and are added solely to improve learning on this main task.

Ideally, task co-training should bring a lot of advantages. For example, related tasks could benefit from similar representations, which facilitates parameter sharing and potentially reduces the memory footprint of the models. In practice, not all tasks benefit from being trained together, and additional knowledge might actually lead to deterioration in the model performance which is described by the notion of *negative transfer*. MTL directly addresses this problem by designing adaptive models that share the knowledge only within certain groups of tasks which improve from co-training. Interestingly, the ability to successfully co-train all tasks together, and the ability to avoid training on some set of tasks by pre-training the model on another set are not necessarily correlated (Standley et al., 2020), which further motivates to separate task transfer from MTL, even though the two problems share a large portion of their motivation and goals.

1.1.4 Continual learning

The key requirement set by task transfer and multi-task learning problems is constant and concurrent availability of data for all tasks. This might not always be the case in real life, e.g. due to impossibility to keep data for ethical or legal reasons in off-line learning setups, or due evolution of the data generating process over time in real-time systems. Continual learning (Li & Hoiem, 2017; Castro et al., 2018), also referred to as lifelong learning and incremental learning, tries to maximise the joint performance on a continual stream of tasks under two main constraints: a) the model has access only to one task at a time, and b) every task can only be visited once.

In general, relationships between the tasks are unconstrained: they can be of different types, i.e. image classification and segmentation, and data can come from different domains. The most considered setup, however, is the one where the type of the task remains the same, e.g. only image classification, and each task has its own data domain and label space. Equal importance of all tasks, combined with the impossibility to revisit old data makes the standard supervised learning ineffective for continual learning. With changes in the data distribution, neural networks tend to overwrite the knowledge the has been observed before, resulting in the phenomenon called *catastrophic forgetting* (McCloskey & Cohen, 1989; Ratcliff, 1990). Tackling this challenge is one of the main focuses of continual learning. The straightforward solution would be to learn a separate model for each task, but this approach is very expensive, might not always work efficiently when the amount of supervision is insufficient and lacks positive transfer between the tasks. Even though there is no explicit restriction on the parameter budget, it is commonly desired to aim for reduction in the total size of the continual learning model.

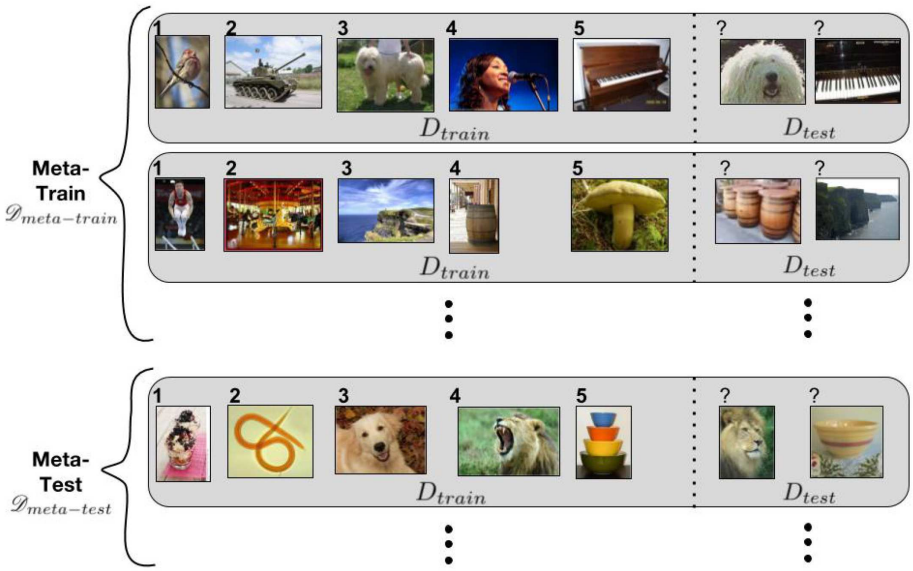


Fig. 1.5 Few-shot learning paradigm. Illustration taken from (Ravi & Larochelle, 2017).

1.1.5 Few-shot learning

Although it is not mentioned explicitly, the problems discussed above operate with the assumption that all tasks and domains are accompanied with reasonable, albeit limited amount of supervision, so that the expressive power of the chosen network architecture was not limited by scarcity of labeled data. But in practice there might be tasks where the number of training examples is very small, e.g. due to the high cost of labeling. This is the case in few-shot learning (Vinyals et al., 2016; Oreshkin et al., 2018), where the training and testing datasets consist of numerous supervised tasks, each containing just a handful of labeled samples, as shown in Figure 1.5. Although all tasks are simultaneously available, similarly to task transfer and MTL, the task distribution is different in the train and the test phases, similarly to domain adaptation. Differently from all previous setups, the test dataset consists of completely new tasks that have never been observed during training, with data coming from the same or even from unseen domains. Novel tasks are still accompanied with the same limited supervision, so the trained model should be able to generalise well from the available labeled test data.

Standard supervised learning requires training a new model from scratch for each task, which would result in quick overfitting since there is not enough training examples, especially when the neural network has a lot of parameters. Alternatively, if we opt for a smaller network in pursuit of handling overfitting, we can significantly limit the capacity and the predictive abilities of the model. Both scenarios are undesirable, which calls for a better utilisation of available supervision. Instead of relying entirely on the data distribution within a single task,

few-shot learning shifts focus towards the task distribution, and extracts information on a meta-level, which is sometimes referred to as *learning to learn*, or *meta-learning* (Finn et al., 2017; Ravi & Larochelle, 2017).

1.1.6 Multi-domain learning

While few-shot learning can be viewed as an extreme case of data insufficiency, multi-domain learning (MDL) (Rebuffi et al., 2017b; Mallya et al., 2018) considers a milder setup, with task-specific datasets being of various yet reasonable sizes. Once more, the goal is to maximise the collective performance on a pre-defined set of tasks. In MDL, all tasks are of the same type, but they differ in the data domain and the corresponding output space, similar to continual learning or domain adaptation. The pool of tasks or, equivalently, domains in MDL is usually much smaller than in few-shot learning, but at the same time the number of labeled examples per category is much higher. Conceptually, MDL is very close to MTL, since all tasks are available concurrently, and there are no shifts in data distributions between the train and the test phases.

Standard supervised learning applied individually to each domain is a reasonable way to approach MDL. Such strategy, however, is memory inefficient, as it requires to store a separate model instance for each domain, which can result in a large memory footprint considering that state-of-the-art networks require billions of parameters to be trained. MDL directly addresses the problem of uncontrollable growth of the model size when the same type of task is solved on multiple domains, and aims to reduce the total number of learnable parameters. While no specific parameter budget is imposed on the model, the core motivation of MDL is to exploit relatedness of the tasks in order to improve the overall performance while keeping the number of domain-specific parameters as small as possible.

1.1.7 Challenges and goals

We focus on the two transfer learning problems that aim to solve the same type of task on multiple domains: few-shot learning and multi-domain learning. Both problems are characterised by the scarcity of available data, which makes the model highly prone to overfitting. Limited amount of data, while being a challenge on its own, imposes additional restrictions on the possible architecture designs, since high-performing networks tend to have a lot of trainable parameters. Our goal in this thesis is to develop models for these two types of problems that would allow for high specialisation to each task and domain while minimising the number of domain-specific trainable parameters.

1.2 Contributions

This thesis is based on two main contributions towards few-shot learning, discussed in Section 1.1.5, and multi-domain learning, discussed in Section 1.1.6. As the first contribution, we propose a Bayesian framework with shared amortised variational inference for few-shot learning. The second contribution introduces a novel multiplicative adaptation module for efficient network re-use in the context of multi-domain learning. Below, we provide more details regarding each of the contributions.

- **Shared amortised variational inference for few-shot learning.** When only a few labeled examples are available per category, straightforward re-use of standard deep learning models leads to overfitting and becomes unpractical (Finn et al., 2017; Vinyals et al., 2016). The main goals of few-shot learning are to overcome the challenges imposed by scarcity of data within a single task by having access to a large pool of related tasks, and to leverage similarities between them through knowledge transfer. Overfitting, however, is not the only problem in few-shot learning. Low data regime also results in high variance of the model predictions which must be properly accounted for. Probabilistic modeling is a natural way to handle model uncertainty, and it is the reason we choose this framework to approach few-shot learning problem. With the probabilistic framework, predictions are usually made by sampling from the prior or the posterior which are defined by the considered graphical model. Previous works in this realm use either fixed or learnable but task-agnostic priors. Additionally, some models use implicit distributions obtained through meta-optimisation which can be computationally expensive when the entire network is being meta-learned.

  Our first contribution proposes a Bayesian approach, where parameters of the task-specific classifier are viewed as latent variables. We model both the prior and the posterior over these latent variables explicitly as conditional distributions, where the prior is conditioned on the train set of the task, and the posterior is conditioned on the union of the train and test sets. Since the true posterior is intractable, we use variational inference (Kingma & Welling, 2014; Rezende et al., 2014) to approximate it. We parameterise the prior and the posterior with the shared amortised inference network which maps data to distribution parameters in a single feedforward pass. We show that the proposed model, which we denote as SAMOVAR, not only achieves competitive results on the three commonly used few-shot learning benchmarks, but also preserves prediciton uncertainty, unlike the previous approach which only uses the prior conditioned on the train set.

  This work was published at the International Conference on Machine Learning in 2020.

- **Modulation adapters for multi-domain learning.** An important property of multi-domain learning (MDL), compared to other problems which involve transfer of knowledge between tasks, is its focus on the parameter budget. MDL motivates to explore various trade-offs between the model performance and the number of parameters trained per domain, favoring solutions that are both well performing and memory efficient. A reasonable approach to MDL is to share as many parameters as possible between the domains, adding only a small memory overhead per task. In practice, it is usually achieved by pre-training a feature extractor on the largest domain, and adjusting it for the remaining ones. Approaches to network adjustment in the literature vary a lot, and include binary masking of convolutional weight tensors, insertion of adaptation layers in different parts of the network and data-dependent finetuning.

  Our second contribution belongs to the group of works that modify pre-trained and fixed base feature extractor with learnable domain-specific adaptation units. Prior works in this group either leave pre-trained layers unchanged, placing domain-specific adapters between the pre-trained convolutions, or they affect only a subset of parameters in the pre-trained convolutional layer, which limits the potential for adaptation. We propose a novel adaptation unit, named MAD, which, unlike previous continuous adapters, affects the entire convolutional weight tensor in a multiplicative manner while having fewer parameters than the layer itself. We also propose to factorise the adapter as a product of two matrices with a smaller intermediate dimension to further reduce the parameter budget. Both the full and the factorised versions of our model achieve state-of-the-art results on the two most common multi-domain benchmarks. In ablation studies we also show the benefit of multiplicative adaptation compared to additive.

1.3 Thesis outline

The rest of the thesis has the following structure:

- In Chapter 2, we describe the two tasks that are the main focus of this thesis: few-shot learning and multi-domain learning. For each of these tasks we make a detailed overview of the major directions of research that have been considered in the literature, as well as review the works that explore the mentioned frameworks.

- The focus of Chapter 3, which presents the first contribution of the thesis, is the few-shot learning problem. We propose a Bayesian approach to the task within the probabilistic framework, where both the prior and the posterior are conditioned on the data, and both distributions are modeled with a shared amortised inference network.

- In Chapter 4, which presents the second contribution of the thesis, we focus on the multi-domain learning problem. We propose a novel type of adaptation modules that adjust pre-trained feature extractors to novel domains via modulation of filters in the convolutional layers.
- Chapter 5 contains the summary of the presented contributions, as well as the discussion of the possible ideas for future work.

Chapter 2

Background

In this chapter, we focus on the areas of transfer learning that are most relevant to the contributions of this thesis. In Section 2.1, which is related to the contributions in Chapter 3, we discuss major research directions in few-shot learning, and provide some examples of work in them. Section 2.2, which is related to the contributions in Chapter 4, covers the research that has been done in multi-domain learning.

2.1 Few-shot learning and meta-learning

One of the most fascinating features of the human brain is its ability to leverage previous experience to generalise while learning new tasks, as opposed to standard deep learning models which are usually highly specialised and restricted to solving specific problems. Inspired by the human ability to adapt to new tasks while having access to just a handful of labeled examples, meta-learning, often described as "learning to learn", has attracted a lot of attention recently. Meta-learning has a number of applications, including reinforcement learning (Finn et al., 2017; Mishra et al., 2018) and few-shot learning (Snell et al., 2017; Oreshkin et al., 2018). In this thesis, we focus on the few-shot image classification problem where a model is trained to solve multiple classification tasks while observing a small number (e.g. five or less) of training samples per category in each task. Unlike standard image classification where the goal is to train the parameters θ of a model M which maps an image to the label, meta-learning aims to learn the parameters Θ of the algorithm A which maps the labeled small dataset to the parameters θ of the model M that would generalise well on the task associated with this dataset.

More formally, for each few-shot image classification task there is a distribution over classification tasks $p(\mathcal{T})$ which contains images from $C \cup \tilde{C}$ classes ($C \cap \tilde{C} = \emptyset$). During the meta-training phase, each task τ is sampled from classes C and consists of the pair

$\mathbf{D}^{\tau}_{\text{meta-train}} = \{\mathbf{D}^{\tau}_{\text{train}}, \mathbf{D}^{\tau}_{\text{test}}\}$, where $\mathbf{D}^{\tau}_{\text{train}}$ is the *train* set (also referred to as the *support* set), and $\mathbf{D}^{\tau}_{\text{test}}$ is the *test* set (also referred to as the *query* set). Both sets have the same structure and consist of the pairs $\{(X^{\tau,\text{train}}_{k,n}, Y^{\tau,\text{train}}_{k,n})\}^{K,N}_{k,n=1}$ and $\{(X^{\tau,\text{test}}_{m,n}, Y^{\tau,\text{test}}_{m,n})\}^{M,N}_{m,n=1}$, respectively. Here, $X^{\tau,\text{train}}_{k,n}$ and $X^{\tau,\text{test}}_{m,n}$ are the images, $Y^{\tau,\text{train}}_{k,n}$ and $Y^{\tau,\text{test}}_{m,n}$ are the corresponding labels, N is the total number of classes in the task τ, K and M are the number of train and test samples per category. Originally, K and N used to be small numbers common for all tasks $\mathcal{T}$, and together they formed a *K-shot N-way episode* τ. However recently, a more general setup has been considered where both K and and N may vary across the episodes (Triantafillou et al., 2020). During meta-testing, the trained algorithm is evaluated on the $\tilde{\mathbf{D}}^{\tilde{\tau}}_{\text{meta-test}}$ which has the same structure as $\mathbf{D}^{\tau}_{\text{meta-train}}$, but the tasks $\tilde{\tau}$ are now sampled from the remaining classes $\tilde{C}$.

Approaches to few-shot learning that have been proposed in the literature vary a lot, and include learning of metric spaces, meta-optimisation, probabilistic modeling and transductive learning. Below, we discuss major directions of research on this topic and provide a few examples that make use of these approaches.

2.1.1 Metric learning

Metric learning is one of the most well known approaches that are used for image classification (Weinberger et al., 2006; Guillaumin et al., 2009). The idea is to learn a parameterised distance metric so that examples that belong to the same category are as close to each other as possible, while the opposite is true for those from different categories. The approach can also be viewed as representation learning with a non-parametric classifier induced by the chosen metric and the class assignment rule. In few-shot learning terms, this is a mapping of the train data $\mathbf{D}^{\tau}_{\text{train}} = \{\mathbf{X}^{\tau}_{\text{train}}, \mathbf{Y}^{\tau}_{\text{train}}\}$ into a task-specific classifier $c_{\tau}(\cdot)$ which is able to classify test samples $\mathbf{X}^{\tau}_{\text{test}}$. Models that fall into this category differ roughly in three aspects: 1) how the feature representation is computed, 2) the distance function, and 3) the decision rule that assigns the label depending on the computed distance.

In *Matching Networks*, Vinyals et al. (2016) define the task-specific classifier $c_{\tau}(\cdot)$ as a linear combination of the train labels Y^{τ}_{train} with attention mechanism $a(\cdot,\cdot)$ as weights:

$$Y^{\tau,\text{test}} = \sum_{i=1}^{KN} a(X^{\tau,\text{test}}, X^{\tau,\text{train}}_{i}) Y^{\tau,\text{train}}_{i} \tag{2.1}$$

Depending on the form of a, the classifier can have different interpretations and, thus, different properties. For example, if a is a kernel, (2.1) becomes a kernel density estimator. It can be a k-nearest neighbors classifier, if a is non-zero only for k training samples $X^{\tau,\text{train}}_{i}$ that are closest to $X^{\tau,\text{test}}$ according to some distance metric. In Matching Networks (Vinyals

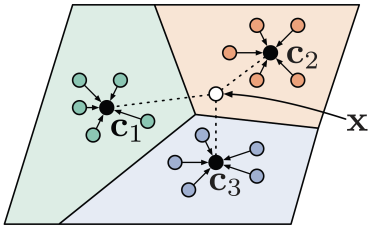


Fig. 2.1 For each class n, its prototype $\mathbf{c}_n$ is obtained by averaging embeddings of the samples from the support set that belong to the same class. Illustration taken from (Snell et al., 2017).

et al., 2016), the attention mechanism is the softmax over the cosine distance $d_{\cos}$:

$$a(X^{\tau,\text{test}}, X_i^{\tau,\text{train}}) = \frac{\exp\left(d_{\cos}\left(f(X^{\tau,\text{test}}), g(X_i^{\tau,\text{train}})\right)\right)}{\sum_{j=1}^{KN}\exp\left(d_{\cos}\left(f(X^{\tau,\text{test}}), g(X_j^{\tau,\text{train}})\right)\right)}. \tag{2.2}$$

Here f and g are embedding functions parameterised by neural networks. For better generalisation, the authors propose to make the embeddings task-specific by conditioning on $\mathbf{D}_{\text{train}}^{\tau}$. In particular, for $g(X^{\tau,\text{train}}, \mathbf{D}_{\text{train}}^{\tau})$ they use a bidirectional Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997) with $\mathbf{D}_{\text{train}}^{\tau}$ considered as a sequence. As for $f(X^{\tau,\text{test}}, \mathbf{D}_{\text{train}}^{\tau})$, they use LSTM with read-attention over $\mathbf{D}_{\text{train}}^{\tau}$, with K unrolling steps.

In *Prototypical Networks*, Snell et al. (2017) learn a non-linear mapping f such that the embedded examples from the same category n form a cluster around a task-specific representation of that category $\mathbf{c}_n^{\tau}$, as shown in Figure 2.1. The latter is referred to as a *prototype*, and it is computed as the average of the embedded examples of the same class from the train data $\mathbf{D}_{\text{train}}^{\tau}$ of the task τ:

$$\mathbf{c}_n^{\tau} = \frac{1}{K}\sum_{i=1}^{K} f(X_i^{\tau,\text{train}})[Y_i^{\tau,\text{train}} = n]. \tag{2.3}$$

Proximity of the embedded test sample $X^{\tau,\text{test}}$ to the class prototypes $\{\mathbf{c}_n^{\tau}\}_{n=1}^{N}$ is measured by the predefined distance function d, and class probabilities are then assigned by computing softmax over these distances:

$$p\left(Y^{\tau,\text{test}} = n | X^{\tau,\text{test}}\right) = \frac{\exp\left(-d(f(X^{\tau,\text{test}}), \mathbf{c}_n^{\tau})\right)}{\sum_{j=1}^{N}\exp\left(-d(f(X^{\tau,\text{test}}), \mathbf{c}_j^{\tau})\right)}. \tag{2.4}$$

The authors show that for distance functions that belong to the class of regular Bregman divergences (Bregman, 1967), Prototypical Networks perform mixture density estimation on

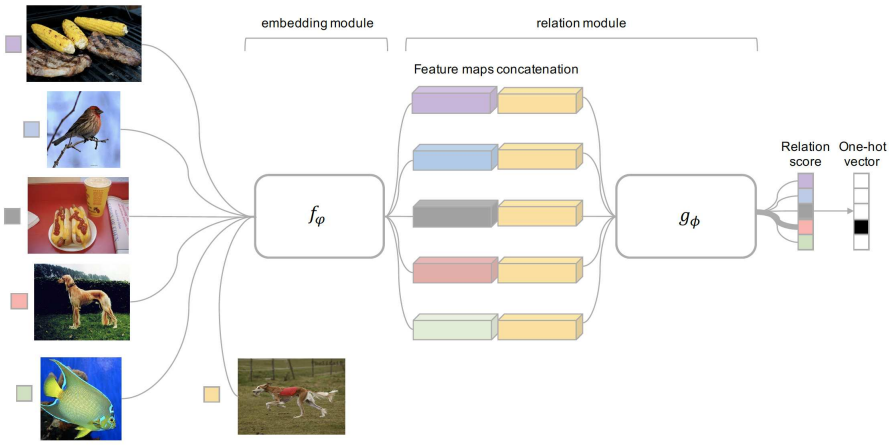


Fig. 2.2 Visualisation of Relation Network architecture for few-shot learning. It consists of two main blocks: feature extractor and relation module which predicts similarity between embedded samples and class representations. Illustration taken from (Sung et al., 2018).

the support set with an exponential family density. In particular, they use Euclidean distance which corresponds to spherical Gaussian densities. In one-shot setup ($K = 1$), Prototypical Networks are equivalent to Matching Networks, since the prototype is computed over a single example. This is no longer the case when $K > 1$.

Unlike previous approaches, *Relation Network* (Sung et al., 2018) directly learns a non-linear comparator g named "relation module" which determines whether samples embedded with a feature extractor f belong to the same categories. For each class n and task τ, feature embeddings of the training examples $\{f(X_{i,n}^{\tau,\text{train}})\}_{i=1}^{K}$ are summed to form a class representation $\mathbf{c}_n^\tau$. After that, the embedded test sample $f(X^{\tau,\text{test}})$ is concatenated with $\mathbf{c}_n^\tau$, and the resulting vector is put through the relation module g to obtain a relation score $r \in [0, 1]$ which measures similarity between $f(X^{\tau,\text{test}})$ and $\mathbf{c}_n^\tau$. This score is regressed to the true label, which is equal to 1 when the test sample and the class representation belong to the same class, and 0 otherwise. The model is trained by minimising the mean square error.

While being an important baseline for few-shot learning, Prototypical Networks have one major drawback: the feature embedding is fixed after being trained, meaning that no adaptation to newer tasks is possible during deployment. Oreshkin et al. (2018) build their approach *TADAM* upon this model, and propose to condition the feature extractor $f(\mathbf{X}, \mathbf{\Gamma}^\tau)$ on the train set $\mathbf{D}_{\text{train}}^\tau$ of the task τ in order to make the metric space task-specific. For that, they propose to use the FiLM conditioning layer (Perez et al., 2018) where a feature-wise affine transformation with learnable parameters $\mathbf{\Gamma}^\tau = \{\boldsymbol{\gamma}^\tau, \boldsymbol{\beta}^\tau\}$ is applied after each

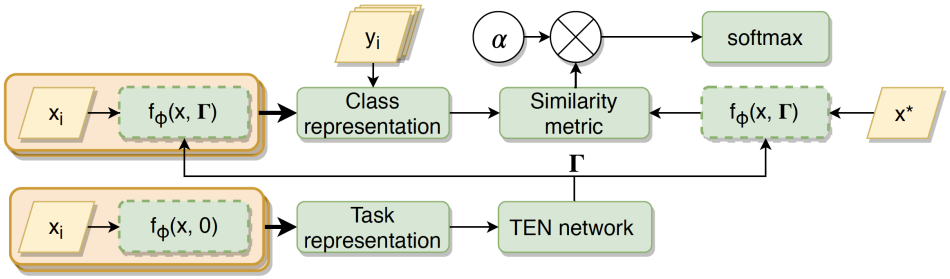


Fig. 2.3 Few-shot learning architecture in TADAM. Feature embeddings of the support set are averaged and put through the TEN module to obtain parameters of the FiLM layers. Illustration taken from (Oreshkin et al., 2018).

convolutional layer of the feature extractor f:

$$h_{l+1} = \boldsymbol{\gamma}_l^\tau \odot h_l + \boldsymbol{\beta}_l^\tau . \tag{2.5}$$

These coefficients are task-specific, and they are predicted by a separate *task embedding network* (TEN) which takes as input the average of the prototypes $\frac{1}{N}\sum_{n=1}^{N} \mathbf{c}_n^\tau$ produced by the unaltered feature extractor $f(\mathbf{X}, 0)$. This makes the latter dynamic, depending on whether parameters $\boldsymbol{\Gamma}^\tau$ of the transformation are being given to the network. The base network $f(\mathbf{X}, 0)$ is in fact task-agnostic: it is fixed after being pre-trained on a standard classification task on the entire meta-train set $\mathbf{D}_{\text{meta-train}}$, and only TEN is being few-shot learned through episodic training. In addition to that, the authors suggest to scale the distance metric from (2.4) with a non-negative temperature α, which can be either learned or cross-validated:

$$p\left(Y^{\tau,\text{test}} = n | X^{\tau,\text{test}}\right) = \frac{\exp\left(-\alpha d(f(X^{\tau,\text{test}}), \mathbf{c}_n^\tau)\right)}{\sum_{j=1}^{N} \exp\left(-\alpha d(f(X^{\tau,\text{test}}), \mathbf{c}_j^\tau)\right)}. \tag{2.6}$$

The authors show that, depending on the value, parameter α either minimises the overlap of the clusters by pushing test embeddings closer to the corresponding prototypes, or performs assignment correction by pushing apart the test embedding and the prototypes of the incorrect categories. The full workflow of TADAM is shown in Figure 2.3.

Qiao et al. (2018) also propose to pre-train a feature embedding network f_θ with parameters θ on a standard classification task and then fix it. However, they replace the last fully connected layer, which represents the linear classifier in the pre-training task, with the set of category-specific vectors $\mathbf{w}^\tau = \{\mathbf{w}_n^\tau\}_{n=1}^N$. As shown in Figure 2.4, for each category n the corresponding weight vector $\mathbf{w}_n^\tau$ is the output of the category-agnostic amortised inference

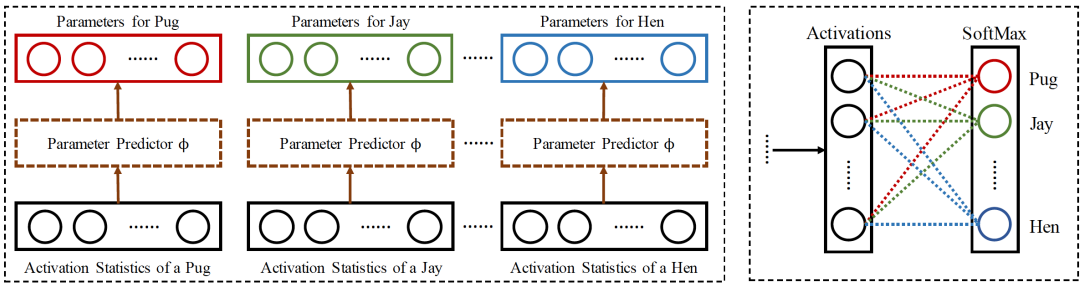


Fig. 2.4 For each class n, corresponding weight vector $\mathbf{w}_n^\tau$ in the classifier is obtained from class representation $\mathbf{s}_n^\tau$ using inference network g_ϕ. Illustration taken from (Qiao et al., 2018).

network g_ϕ with parameters ϕ that takes the category representation s_n^τ as input:

$$\mathbf{w}_n^\tau = g_\phi(\mathbf{s}_n^\tau). \tag{2.7}$$

For task τ, with probability $1 - p_{\text{mean}}$ the statistic s_n^τ is sampled uniformly from $\mathcal{A}_n$, and with probability p_{mean} vector $\overline{\mathbf{a}}_n$ is used as $\mathbf{s}_n^\tau$, where $\mathcal{A}_n = f_\theta(\mathbf{X}_n)$ is the set of all activations obtained from the penultimate layer of the pre-trained feature extractor f_θ fed with all meta-train samples $\mathbf{X}_n$ from the category n, and $\overline{\mathbf{a}}_n$ is the mean of these activations. The minimisation loss then looks as follows:

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{s}} \left(-\mathbf{w}_{\mathbf{Y}_{\text{test}}^\tau}^\tau f_\theta(\mathbf{X}_{\text{test}}^\tau) + \log \sum_{n=1}^{N} e^{\mathbf{w}_n^\tau f_\theta(\mathbf{X}_{\text{test}}^\tau)} \right) + \lambda ||\phi|| \tag{2.8}$$

For inference on novel categories $\{m\}_{m=1}^M$ during meta-test, the authors suggest to replace the sampled category representation $\mathbf{s}_m^{\tilde{\tau}}$ with the set of all category-specific activations $\tilde{\mathcal{A}}_m = f_\theta(\tilde{\mathbf{X}}_{\text{train},m})$ obtained by putting the entire meta-test training set $\tilde{\mathbf{X}}_{\text{train},m}$ from the category m through the feature extractor. These embeddings are put through the amortised network g_ϕ to obtain a set of category-specific classification vectors $\tilde{\mathbf{W}}_m = \{\tilde{\mathbf{w}}_m\} = g_\phi(\tilde{\mathcal{A}}_m)$. Then, for each meta-test test sample $\tilde{X}^{\text{test}}$, the corresponding logit $\tilde{l}_n$ in the softmax function is the maximum response out of all responses obtained using $\tilde{\mathbf{W}}_n$:

$$\tilde{l}_n = \max \tilde{\mathbf{W}}_n f_\theta(\tilde{X}^{\text{test}}). \tag{2.9}$$

CNAPS (Requeima et al., 2019) combines inference of FiLM parameters ψ_f^τ used to adapt feature representations $f_\theta(\mathbf{x}; \psi_f^\tau)$ to the task τ, as in TADAM, with learning a function approximator that infers task- and category-specific parameters of the linear classifier $\psi_w^\tau = \{\psi_{w,n}^\tau\}_{n=1}^N$ from data representations $\{f_\theta(\mathbf{D}_{\text{train},n}^\tau, \psi_f^\tau)\}_{n=1}^N$, as in (Qiao et al., 2018).

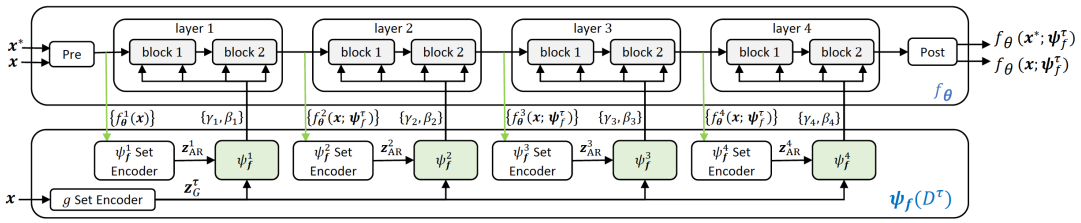


Fig. 2.5 Few-shot learning architecture with autoregressive feature adaptation in CNAPS. Layers of the feature extractor f_θ are interleaved with the task encoding networks ψ_f. For each layer f_θ^i, the corresponding network ψ_f^i produces its FiLM parameters from the features computed by the preceeding layer f_θ^{i-1}. Illustration taken from (Requeima et al., 2019).

Parameters θ of the shared task-agnostic feature extractor f_θ are similarly pre-trained on some large image classification dataset, and then fixed. While Qiao et al. (2018) uses the entire meta-train set $\mathbf{D}_{\text{meta-train}}$ to compute the category representation $\mathbf{s}_n^\tau$, CNAPS remains within the paradigm of K-shot N-way episodic training, and conditions the category representation on the training data $\mathbf{D}_{\text{train}}^\tau$ of the task at hand. Let K_n^τ be the number of train samples for category n in the task τ. To aggregate these representations in a flexible and permutation invariant manner, and to allow the model to work with arbitrary K_n^τ, the authors compute the mean embedding, i.e. the prototype, and provide it as input to the task-agnostic network to infer parameters $\psi_{w,n}^\tau$:

$$\psi_{w,n}^\tau = \psi_w \left(\frac{1}{K_n^\tau} \sum_{X \in \mathbf{D}_{\text{train},n}^\tau} f_\theta(X, \psi_f^\tau) \right). \tag{2.10}$$

As for adaptation of the feature extractor f_θ, the authors propose autoregressive inference of FiLM parameters, unlike TADAM which predicts all FiLM parameters at once, after a single forward pass of $\mathbf{D}_{\text{train}}^\tau$ through the network f_θ. In CNAPS, the feature extractor is divided into several parts denoted as "layers" f_θ^i, and each layer i is accompanied by a separate shared inference network ψ_f^i which produces corresponding FiLM parameters. Network ψ_f^i takes as input a global task representation $\mathbf{z}_G^\tau$ produced by a global set encoder of $\mathbf{D}_{\text{train}}^\tau$, concatenated with the local task representation $\mathbf{z}_{AR}^\tau$ obtained by a local set encoder of preceding feature maps $f_\theta^{i-1}(\mathbf{D}_{\text{train}}^\tau, \psi_f^{\tau,i-1})$. Visualisation of the autoregressive feature extractor is shown in Figure 2.5. Test samples $\mathbf{X}_{\text{test}}^\tau$ are put through the adapted feature extractor $f_\theta(\mathbf{x}; \psi_f^\tau)$, and the inferred classifier ψ_w^τ is applied to them to produce the softmax logits.

Bateni et al. (2020) build their model upon CNAPS, and refer to it as *Simple CNAPS*. While considering the same architecture design for feature extractor as in Figure 2.5, the authors argue that autoregressive adaptation is not necessary, and propose to condition

prediction of FiLM parameters solely on $\mathbf{z}_G^\tau$, the global set encoding of the training data $\mathbf{D}_{\text{train}}^\tau$. Differently from CNAPS, Simple CNAPS does not use a linear classifier, and, as a result, does not perform inference of its parameters. Instead, the authors use a well-known metric-based classifier, similar to (2.4), but replace the Eucledian distance metric with the Mahalanobis distance:

$$d(X, \mathbf{c}_n^\tau) = \frac{1}{2}(X - \mathbf{c}_n^\tau)^T (\mathbf{Q}_n^\tau)^{-1} (X - \mathbf{c}_n^\tau). \tag{2.11}$$

Here, $\mathbf{Q}_\mathbf{n}^\tau$ is a task- and category-specific covariance matrix that depends on the covariance $\mathbf{\Sigma}_n^\tau$ of the category n within the task τ, as well as on the total covariance $\mathbf{\Sigma}^\tau$ of the task and the hyper parameters λ_n^τ and β:

$$\mathbf{Q}_n^\tau = \lambda_n^\tau \mathbf{\Sigma}_n^\tau + (1 - \lambda_n^\tau)\mathbf{\Sigma}^\tau + \beta I. \tag{2.12}$$

2.1.2 Meta-optimisation

A significant drawback of the distance-based meta-learning is its limited model adaptation to unseen tasks. Since the task-specific classifiers are determined by the similarity metric used in a particular few-shot learning model, the only set of parameters that is trained directly in this group of works is the feature extractor, and it stays unchanged during meta-testing. At the opposite end of the spectrum, however, is full network finetuning on each task, both during meta-training and meta-testing. Pre-training the network, common in few-shot learning, can thus be viewed as learning a good network *initialisation*. There is no guarantee though that this initialisation will actually be a good starting point for further parameter tuning. *Meta-optimisation* aims to directly optimise model performance with respect to network parameter initialisation, such that rapid adaptation through a small number of parameter updates would lead to good performance on a new task.

Ravi & Larochelle (2017) observed that the gradient descent update rule resembles the update of the cell state in an LSTM (Hochreiter & Schmidhuber, 1997). The latter is defined as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{2.13}$$

where f_t and i_t are the forget and input gates at step t respectively, c_{t-1} is the cell state at step $t-1$, and $\tilde{c}_t$ is the candidate cell state at step t. Assuming $c_t = \theta_t$ and $c_{t-1} = \theta_{t-1}$ are the parameters of the model at steps t and $t-1$, $\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t$ is the gradient of the optimisation loss at step t, $i_t = \alpha_t$ is the learning rate at step t, and $f_t = 1$, (2.13) can indeed be viewed as:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t, \tag{2.14}$$

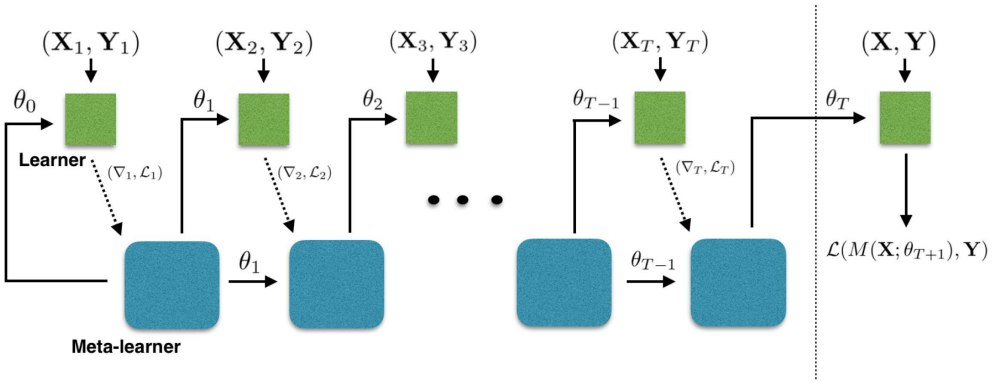


Fig. 2.6 Computational graph in Meta LSTM. Steps of the gradient descent are viewed as updates of the cell state in LSTM. A task-specific learner is updated using a series of T batches $\{(\mathbf{X}_{\text{train},i}; \mathbf{Y}_{\text{train},i})\}_{i=1}^{T}$ from the train set, with each update t being predicted by a meta-learner from the gradient of the loss on the previous batch $t-1$. After T updates, performance of the learner is evaluated on the test set $(\mathbf{X}_{\text{test}}; \mathbf{Y}_{\text{test}})$, and the gradient of this loss is used to train the meta-learner. Illustration taken from (Ravi & Larochelle, 2017).

which is the definition of the gradient descent update. The meta-training procedure of the model, which was denoted as *Meta LSTM*, is shown in Figure 2.6. To make the learning rate adaptive, as well as to have a controllable regularizer of the learner parameters, both i_t and f_t are learned as the following functions:

$$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right) \tag{2.15}$$

$$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right) \tag{2.16}$$

An interesting property of this meta-learner is that by learning $c_0 = \theta_0$, it is actually learning a good initialisation point for parameters θ, such that a small number T of LSTM state updates would result in a model $M(\mathbf{X}; \theta_T)$ that generalises well on the current task.

Finn et al. (2017) later showed that learning a good initialisation of the model parameters θ_0 by meta-learning the gradient descent alone, while using constant learning rate and regularizer provides better and more stable results. In their approach *MAML*, the authors optimise the performance of the model $M(\mathbf{X}_{\text{test}}^{\tau_i}; \theta_T^{\tau_i})$ with respect to θ_0, where $\theta_T^{\tau_i}$ represents the parameters after T steps of the gradient descent with the constant learning rate α on the train data $\mathbf{D}_{\text{train}}^{\tau_i} = \{\mathbf{X}_{\text{train}}^{\tau_i}, \mathbf{Y}_{\text{train}}^{\tau_i}\}$ of the task τ_i. For simplicity, let $T = 1$. Then the

meta-objective is:

$$\begin{aligned}\min_{\theta_0} \sum_{\tau_i \sim p(\mathcal{T})} & \mathcal{L}\left[M(\mathbf{X}_{\text{test}}^{\tau_i}; \theta_1^{\tau_i}), \mathbf{Y}_{\text{test}}^{\tau_i}\right] \\ &= \min_{\theta_0} \sum_{\tau_i \sim p(\mathcal{T})} \mathcal{L}\left[M\left(\mathbf{X}_{\text{test}}^{\tau_i}; \theta_0 - \alpha \nabla_{\theta_0} \mathcal{L}\left[M\left(\mathbf{X}_{\text{train}}^{\tau_i}; \theta_0^{\tau_i}\right), \mathbf{Y}_{\text{train}}^{\tau_i}\right]\right), \mathbf{Y}_{\text{test}}^{\tau_i}\right].\end{aligned} \tag{2.17}$$

The procedure of the local, task-specific gradient update of θ_0 on the training data $D_{\text{train}}^{\tau_i}$ of the task τ_i with the constant learning rate α is sometimes referred to as the *inner loop* of the meta-learning algorithm:

$$\theta_1^{\tau_i} = \theta_0 - \alpha \nabla_{\theta_0} \mathcal{L}\left[M\left(\mathbf{X}_{\text{train}}^{\tau_i}; \theta_0^{\tau_i}\right), \mathbf{Y}_{\text{train}}^{\tau_i}\right], \tag{2.18}$$

and it is followed by the global meta-gradient update of θ_0 on the test data $D_{\text{test}}^{\tau_i}$ of the task τ_i with the constant learning rate β, which is sometimes referred to as the *outer loop*:

$$\theta_0 = \theta_0 - \beta \sum_{\tau_i \sim p(\mathcal{T})} \nabla_{\theta_0} \mathcal{L}\left[M\left(\mathbf{X}_{\text{test}}^{\tau_i}; \theta_1^{\tau_i}\right), \mathbf{Y}_{\text{test}}^{\tau_i}\right]. \tag{2.19}$$

Since meta-optimisation of MAML requires computing gradients of the gradients, an additional backward pass through M is necessary in order to compute the Hessian-vector products. The authors also considered the first-order approximation of the model, where the second derivatives are omitted, and this approach turned out to perform nearly as well.

In *Reptile* (Nichol et al., 2018), the authors further investigate the first-order optimisation approach to meta-learning, where the goal is to learn a good initialisation of the neural network parameters while treating the gradients as constants and ignoring the gradients of higher order. Reptile is closely related to the first-order version of MAML, but it does not require splitting the meta-train data $\mathbf{D}_{\text{meta-train}}^{\tau_i} = \{\mathbf{X}_{\text{meta-train}}^{\tau_i}, \mathbf{Y}_{\text{meta-train}}^{\tau_i}\}$ into the train $\mathbf{D}_{\text{train}}^{\tau_i}$ and the test $\mathbf{D}_{\text{test}}^{\tau_i}$ sets for gradient updates. While using a similar "inner-outer loop" structure of the algorithm, the authors propose to treat $\theta_T^{\tau_i} - \theta_0$ as a gradient, and update the initialisation in this direction instead of computing the gradient in the outer loop. The inner loop here is still defined by (2.18), but the outer loop now looks as follows:

$$\theta_0 = \theta_0 + \beta \frac{1}{I} \sum_{i=1}^{I} \theta_1^{\tau_i} - \theta_0. \tag{2.20}$$

When the number of the gradient updates $T = 1$, the algorithm corresponds to stochastic gradient descent on the expected loss. However, when $T > 1$, as in Reptile, such corre-

spondence no longer takes place, and the difference comes from the terms related to the derivatives of the loss of second and higher orders.

All of the aforementioned approaches finetune and meta-learn initialisation for the entire neural network, which scales poorly with the size of the network, and is much more prone to overfitting as data is scarce on each task. *MTL* (Sun et al., 2019) parameterises the weight tensor in the convolutional layers of the network as a product of the base weight tensor $\mathbb{R}^{M\times N\times K\times K}$ and a tensor scaler $\mathbb{R}^{M\times N\times 1\times 1}$. The same representation is used for the bias weight vector. The base tensors are first pre-trained with the standard classification task on the meta-train set $\mathbf{D}_{\text{meta-train}}$, which will later stay fixed. After that, the newly introduced scaler weight tensors are learned via the stochastic gradient descent on the expected loss, while the task-specific linear classifiers are meta-trained using MAML. In addition to that, the authors propose to keep track of the classes that yield the worst performance on the task and use them to sample the "hard" tasks.

In *R2-D2*, Bertinetto et al. (2019) propose to replace the gradient descent or LSTM updates in the inner loop of meta-optimisation models, referred to as the *base learners*, with standard machine learning algorithms that involve closed-form solutions, such as ridge regression. The idea is to meta-learn in the outer loop the feature embedding network f and hyperparameters ρ of the base learner, so that the ridge regression applied to the last fully connected layer W in f worked well on new tasks. In this case W serves as a linear predictor which takes as input the feature embedding $f_{-1}(\mathbf{X})$ produced by the preceding layers. As in previous approaches, such formulation of the problem requires backpropagation through the base learner. For each task τ, R2-D2 applies standard automatic differentiation to the closed form solution of the ridge regression with a regularisation coefficient $\lambda \in \rho$ to obtain a task-specific classifier W^τ:

$$\begin{aligned} W^\tau &= \arg\min ||f_{-1}(\mathbf{X}^\tau_{\text{train}})W - \mathbf{Y}^\tau_{\text{train}}||^2 + \lambda||W||^2 \\ &= \left(f_{-1}(\mathbf{X}^\tau_{\text{train}})^T f_{-1}(\mathbf{X}^\tau_{\text{train}}) + \lambda I\right)^{-1} f_{-1}(\mathbf{X}^\tau_{\text{train}})^T \mathbf{Y}^\tau_{\text{train}}. \end{aligned} \tag{2.21}$$

To prevent a quadratic growth with the embedding size of the matrix $f_{-1}(\mathbf{X}^\tau_{\text{train}})^T f_{-1}(\mathbf{X}^\tau_{\text{train}})$ to be inverted, the authors use the Woodbury matrix identity (Petersen & Pedersen, 2008). In addition to that, the linear classifier is affinely transformed with hyperparameters $\alpha, \beta \in \rho$:

$$\mathbf{Y}^\tau_{\text{test}} = \alpha f_{-1}(\mathbf{X}^\tau_{\text{test}})W^\tau + \beta. \tag{2.22}$$

In a similar vein, Lee et al. (2019) consider the support vector machine (SVM) as a base learner in their *MetaOptNet*.

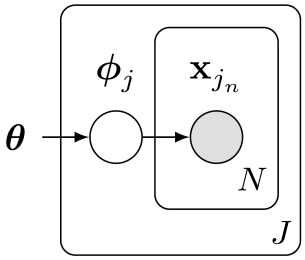


Fig. 2.7 The probabilistic graphical model associated with MAML. Here $\mathbf{x}_{j_n}$ corresponds to $\mathbf{D}^{\tau}$ in out notation, and ϕ_j corresponds to ϕ^{τ}. Illustration taken from (Grant et al., 2018).

2.1.3 Probabilistic modeling

One of the prominent features of few-shot learning compared to standard deep learning is that the model predictions heavily depend not only on the model used for training but also on the training data $\mathbf{D}^{\tau_i}_{\text{train}}$ available for tasks $\tau_i \in \mathcal{T}$, since each of them is represented by just a handful of labeled examples. Scarcity of data introduces high uncertainty about the obtained solutions, but none of the approaches discussed so far measure it directly. This makes probabilistic modeling a convenient framework to reason about the predictive functions and to evaluate the confidence in the results. Another benefit of probabilistic inference is its potential to make the models more robust and to prevent overfitting which is not uncommon in a low data regime, as in few-shot learning (Mishra et al., 2018).

Grant et al. (2018) propose a hierarchical probabilistic model for meta-learning, and suggest to view MAML as empirical Bayes. Assuming ϕ^{τ} are parameters that should be specific to task $\tau \in \mathcal{T}$ while simultaneously affecting parameters $\phi^{\tau'}$ for other tasks $\tau' \in \mathcal{T}$, the mutual dependence of these parameters can be enforced through their individual dependence on the shared task-agnostic parameters θ. The corresponding probabilistic graphical model is shown in Figure 2.7. Then, the point estimate for θ can be obtained by maximising the following marginal likelihood of test data $\mathbf{D}^{\tau}_{\text{test}}$:

$$p(\mathbf{D}^{\tau}_{\text{test}}|\theta) = \int p(\mathbf{D}^{\tau}_{\text{test}}|\phi^{\tau})p(\phi^{\tau}|\theta)\mathrm{d}\phi^{\tau}. \tag{2.23}$$

In the context of MAML, the local ϕ^{τ} are modelled as the updated parameters obtained after one or a few steps of the inner loop gradient descent on $\mathbf{D}^{\tau}_{\text{train}}$, and the global θ are modelled as the initialisation from which the optimisation starts. This fast adaptation procedure can be viewed as an estimator $\hat{\phi}^{\tau}$ of the true parameters $\phi^{\tau} \sim p(\phi^{\tau}|\theta)$:

$$\hat{\phi}^{\tau} = \theta - \alpha\nabla_{\theta}\left[-\log p(\mathbf{D}^{\tau}_{\text{train}}|\theta)\right], \tag{2.24}$$

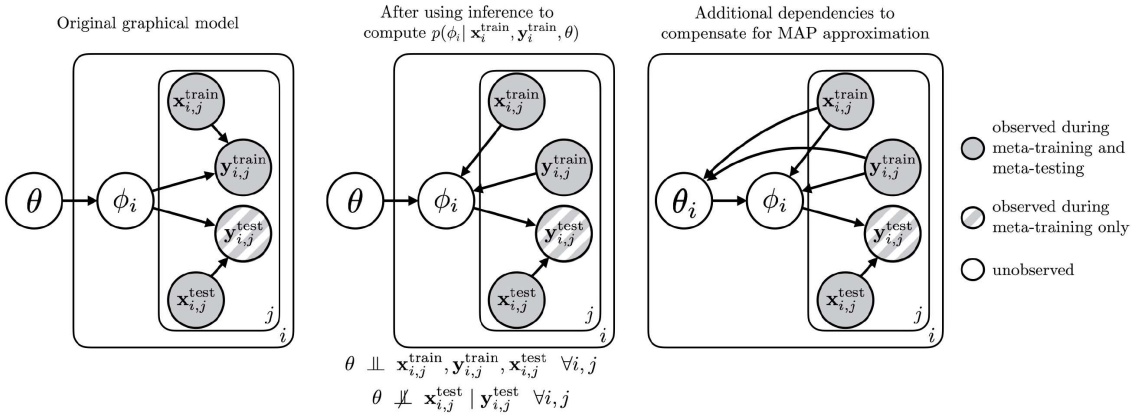


Fig. 2.8 Inference performed over ϕ_i transforms the original graphical model (left), which corresponds to MAML, into the graphical model in the center. In PLATIPUS (right), additional dependency of the prior on the training data is introduced. Here $(\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$ corresponds to $\mathbf{D}^\tau$ in out notation, and ϕ_i, θ_i correspond to ϕ^τ and θ^τ, respectively. Illustration taken from (Finn et al., 2018).

which can be used for maximisation of the approximated marginal likelihood (2.23) since the integral is intractable:

$$\begin{aligned} -\log p(\mathbf{D}_{\text{test}}^\tau|\theta) &\approx -\log p(\mathbf{D}_{\text{test}}^\tau|\hat{\phi}^\tau) \\ &= -\log p(\mathbf{D}_{\text{test}}^\tau|\theta - \alpha\nabla_\theta\left[-\log p\left(\mathbf{D}_{\text{train}}^\tau|\theta\right)\right]). \end{aligned} \tag{2.25}$$

This is the exact optimisation objective used in the original MAML paper, which makes it an empirical Bayes applied to the hierarchical probabilistic model in Figure 2.7. The authors argue that the truncated gradient descent provides an estimator $\hat{\phi}^\tau$ which is a value of the mode of an implicit posterior $p(\phi^\tau|\mathbf{D}_{\text{test}}, \theta)$. This posterior is the result of treating the empirical loss as a negative log-likelihood $p(\mathbf{D}_{\text{test}}|\phi^\tau)$, as well as treating the regularisation loss combined with early stopping as a prior $p(\phi^\tau|\theta)$:

$$p(\phi^\tau|\mathbf{D}_{\text{test}}, \theta) \propto p(\mathbf{D}_{\text{test}}|\phi^\tau)p(\phi^\tau|\theta) \tag{2.26}$$

The authors also propose an improved version of MAML, referred to as *LLAMA*, which replaces the point estimate in (2.23) with the Laplace approximation of the negative log posterior:

$$-\log p(\mathbf{D}_{\text{test}}^\tau|\theta) \approx -\log p(\mathbf{D}_{\text{test}}^\tau|\hat{\phi}^\tau) - \log p(\hat{\phi}^\tau|\theta) + \frac{1}{2}\log\det(\mathbf{H}^\tau), \tag{2.27}$$

where $\det(\mathbf{H}^\tau)$ is a determinant of the Hessian matrix of its second-order partial derivatives.

In *PLATIPUS*, Finn et al. (2018) propose to use variational inference to model the uncertainty in few-shot learning. Similarly to LLAMA, they consider a generative model $p(\theta, \phi^\tau) = p(\phi^\tau|\theta)p(\theta)$ shown in Figure 2.8 (left). They also use the same MAP approximation $\hat{\phi}^\tau$ via truncated gradient descent from (2.24) as the inferred approximate posterior $q(\phi^\tau|\theta, \mathbf{Y}^\tau_{\text{train}}, \mathbf{X}^\tau_{\text{train}}) \approx \delta(\phi^\tau = \hat{\phi}^\tau)$, where $\delta(\cdot)$ is the Dirac delta. This creates a factor over parameters ϕ^τ, and turns the original graphical model into the one shown in Figure 2.8 (middle), where the train data $\mathbf{D}^\tau_{\text{train}} = \{\mathbf{Y}^\tau_{\text{train}}, \mathbf{X}^\tau_{\text{train}}\}$ and the global parameters θ become conditionally independent. To model the global parameters θ, the authors use a learnable Gaussian prior $p(\theta) = \mathcal{N}(\mu_\theta, \sigma^2_\theta)$, where the covariance σ^2_θ is diagonal. The approximate posterior $q_\psi(\theta|\mathbf{Y}^\tau_{\text{test}}, \mathbf{X}^\tau_{\text{test}})$ is then modelled with the following inference network with parameters ψ:

$$q_\psi(\theta|\mathbf{Y}^\tau_{\text{test}}, \mathbf{X}^\tau_{\text{test}}) = \mathcal{N}\left(\mu_\theta + \gamma_q \nabla \log p(\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \mu_\theta); \mathbf{v}_q\right), \tag{2.28}$$

Where $\mathbf{v}_q$ is the trainable variance. The optimisation objective is the variational lower bound of the approximate marginal log-likelihood:

$$\begin{aligned}\log p(&\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \mathbf{D}^\tau_{\text{train}}) \\ &\geq \mathbb{E}_{\theta\sim q_\psi}\left[\log p(\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \hat{\phi}^\tau) + \log p(\theta)\right] + \mathcal{H}\left(q_\psi(\theta|\mathbf{X}^\tau_{\text{test}}, \mathbf{y}^\tau_{\text{test}})\right),\end{aligned} \tag{2.29}$$

where $\mathbb{E}_{\theta\sim q_\psi}$ is an expectation over the approximate posterior q_ψ, and $\mathcal{H}$ is its entropy. The authors admit that the proposed approximation of the marginal likelihood via the point estimation $\hat{\phi}^\tau$ is crude, and propose to to compensate for that by making the prior over θ more expressive. This is achieved by making it task-specific and conditioning it on the training data $\mathbf{D}^\tau_{\text{train}}$:

$$p(\theta^\tau|\mathbf{Y}^\tau_{\text{train}}, \mathbf{X}^\tau_{\text{train}}) = \mathcal{N}\left(\mu_\theta + \gamma_q \nabla \log p(\mathbf{Y}^\tau_{\text{train}}|\mathbf{X}^\tau_{\text{train}}, \mu_\theta); \mathbf{v}_q\right). \tag{2.30}$$

Kim et al. (2018a) rely on the Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016) to perform non-parametric variational inference. The choice of SVGD is motivated by its ability to draw samples from the target distribution without requiring it to be tractable, unlike traditional variational inference, while having higher convergence rate compared to MCMC. The authors consider Empirical Bayes applied to hierarchical probabilistic model, similar to (2.23). Unlike Grant et al. (2018), however, they do not use a single point estimator $\hat{\phi}^\tau$ obtained via truncated gradient descent to approximate the intractable integral. Instead, the learnable initialisation θ_0 is replaced with an ensemble of M learnable initialisations Θ_0 viewed as M initial particles in SVGD. After that, SVGD is used to obtain M samples

from the posterior $p(\phi^\tau|\mathbf{D}^\tau_{\text{train}}, \Theta_0)$ in order to compute the Monte Carlo approximation of the marginal likelihood:

$$p(\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \mathbf{D}^\tau_{\text{train}}, \Theta_0) \approx \frac{1}{M}\sum_{m=1}^{M} p(\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \phi^\tau_m), \text{ where } \phi^\tau_m \sim p(\phi^\tau|\mathbf{D}^\tau_{\text{train}}, \Theta_0). \quad (2.31)$$

Algorithm-wise, this method, referred to as *Bayesian Fast Adaptation (BFA)*, has the same inner-outer loop structure as MAML, but the gradient descent updates in the inner loop are replaced with SVGD updates, and there are M model instances that are being optimised both in the inner and the outer loops, instead of one. BFA can be interpreted as Bayesian ensemble with interaction between the individual models induced by SVGD, and it is equivalent to MAML when $M = 1$, since SVGD reduces to gradient descent if there is only one particle. The authors argue that, despite Bayesian inference, the model is prone to overfitting due to performing empirical risk minimisation. For this reason they propose a novel *Chaser Loss* defined by dissimilarity between the approximate task posterior $p(\phi^\tau|\mathbf{D}^\tau_{\text{train}}, \Theta_0)$ and the true task posterior $p(\phi^\tau|\mathbf{D}^\tau_{\text{train}} \cup \mathbf{D}^\tau_{\text{test}}, \Theta_0)$, which is measured by the distance d between the sets of corresponding samples, and the final model is denoted as *BMAML*:

$$\begin{aligned}\mathcal{L}_{\text{BMAML}}(\Theta_0) &= \sum_{\tau\in\mathcal{T}} d(\phi^\tau_{\text{train}}, \text{stopgrad}(\phi^\tau_{\text{train + test}})) \\ &= \sum_{\tau\in\mathcal{T}}\sum_{m=1}^{M} ||\phi^\tau_{\text{train},m} - \text{stopgrad}(\phi^\tau_{\text{train+test},m})||^2_2.\end{aligned} \quad (2.32)$$

All probabilistic models discussed so far are derived from MAML, and they directly model network parameters as random variables, which limits their potential for probabilistic inference and makes it computationally heavy. Similarly to the authors of PLATIPUS, Rusu et al. (2019) replace the point estimation of the initialisation θ with the data dependent distribution over it, as in (2.30). In their model *LEO*, the dimensionality problem is addressed by replacing the direct modeling of network parameters θ with learning an amortised latent encoding $\mathbf{z}$ of the data. The model embeds the training data $\mathbf{D}^\tau_{\text{train}}$ into a low-dimensional probabilistic latent space first, followed by meta-optimisation of the sampled embedding "initalisation" $\mathbf{z}^\tau_0$ in this space and subsequent decoding of the adapted latent representation $\mathbf{z}^\tau$ into a distribution over task-specific classifier weights $\mathbf{w}^\tau$. Both the latent code $\mathbf{z}^\tau$ and the linear softmax classifier $\mathbf{w}^\tau$ are modelled as the sets of the class-specific vectors $\{\mathbf{z}^\tau_n\}^N_{n=1}$ and $\{\mathbf{w}^\tau_n\}^N_{n=1}$, respectively. More specifically, the training samples $\mathbf{X}^\tau_{\text{train},n}$ from the class n go through a feature encoder, followed by Relation Net (Sung et al., 2018) which produces parameters $\mu^\tau_{z,n}$ and $\sigma^\tau_{z,n}$ of the Gaussian distribution over the initialisation of the class-specific latent code $z^\tau_{n,0} \sim q(z^\tau_{n,0}|\mathbf{D}^\tau_{\text{train},n}) = \mathcal{N}(\mu^\tau_{z,n}, diag(\sigma^\tau_{z,n}))$. After that, a sampled latent

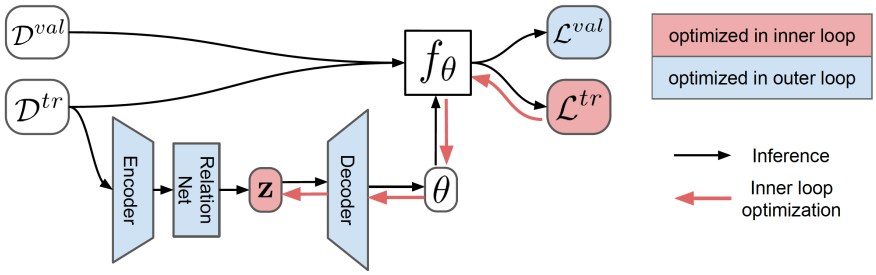


Fig. 2.9 Few-shot learning architecture in LEO. Illustration taken from (Rusu et al., 2019).

representation $z^\tau_{n,0}$ is decoded into the parameters $\mu^\tau_{w,n}$ and $\sigma^\tau_{w,n}$ of the Gaussian distribution over the class-specific classifier weight vector $\mathbf{w}^\tau_n \sim q(\mathbf{w}^\tau_n|\mathbf{z}^\tau_{n,0}) = \mathcal{N}(\mu^\tau_{w,n}, diag(\sigma^\tau_{w,n}))$. A classifier $\mathbf{w}^\tau$, composed of the sampled classifier vectors $\mathbf{w}^\tau_n$ stacked column-wise, is then used to compute the loss on the train data $\mathbf{D}^\tau_{\text{train}}$ for inner-loop optimisation of the latent representations $\mathbf{z}^\tau = \{\mathbf{z}^\tau_n\}^N_{n=1}$:

$$\mathcal{L}_{\text{train}}(\mathbf{w}^\tau) = -\mathbf{w}_{Y^\tau_{\text{train}}} \cdot \mathbf{X}^\tau_{\text{train}} + \log\left(\sum_{n=1}^{N} e^{\mathbf{w}^\tau_n \cdot \mathbf{X}^\tau_{\text{train}}}\right), \tag{2.33}$$

$$\mathbf{z}^\tau_n = \mathbf{z}^\tau_{n,0} - \alpha \nabla_{\mathbf{z}^\tau_{n,0}} \mathcal{L}^\tau_{\text{train}}. \tag{2.34}$$

In the outer loop, which updates the parameters $\theta = \{\theta_e, \theta_r, \theta_d\}$ of the encoder, relation and decoder networks, the loss (2.33) is computed with updated weights $\mathbf{w}^\tau_n \sim q(\mathbf{w}^\tau_n|\mathbf{z}^\tau_n)$ on the test data $\mathbf{D}^\tau_{\text{test}}$, and it is combined with aditional terms for regularisation of the latent space:

$$\mathcal{L}^\tau_{\text{LEO}}(\theta) = \mathcal{L}_{\text{test}}(\mathbf{w}^\tau) + \beta D_{\text{KL}}\left(q(\mathbf{z}^\tau_{n,0}|\mathbf{D}^\tau_{\text{train},n})||p(\mathbf{z}^\tau_{n,0})\right) + \gamma||\text{stopgrad}(\mathbf{z}^\tau_n) - \mathbf{z}^\tau_{n,0}||^2_2 + R, \tag{2.35}$$

where $p(\mathbf{z}^\tau_{n,0}) = \mathcal{N}(0, \mathcal{I})$, and $R = \lambda_1\left(||\phi_e||^2_2 + ||\phi_r||^2_2 + ||\phi_d||^2_2\right) + \lambda_2||\mathcal{C}_d - \mathcal{I}||_2$, assuming linear encoder, relation and decoder networks, and assuming $\mathcal{C}_d$ is the correlation matrix between rows of ϕ_d. The full architecture of LEO is shown in Figure 2.9.

Gordon et al. (2019) use the same probabilistic model as in Figure 2.7 and Figure 2.8 (left). In their approach named *VERSA*, global parameters θ represent the parameters of the shared feature extractor obtained through point estimation, and local parameters $\phi^\tau \equiv \psi^\tau = \{\mathbf{w}^\tau, \mathbf{b}^\tau\}$ represent the task-specific classifier and the bias vector inferred by the probabilistic model. The goal is to learn an approximate predictive distribution further approximated with

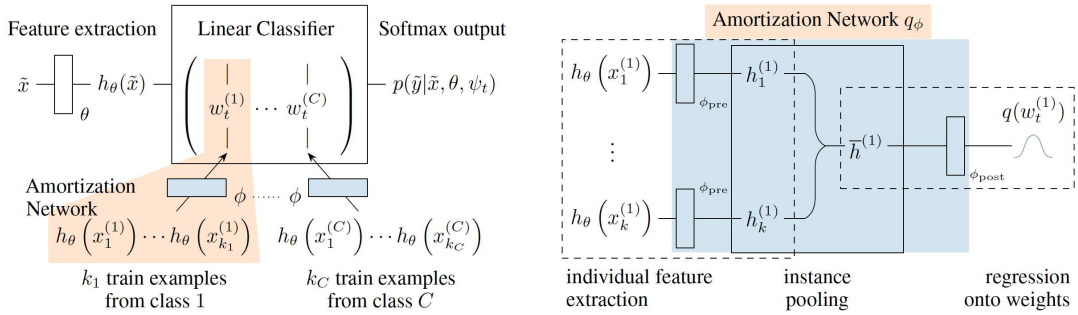


Fig. 2.10 Few-shot learning architecture in VERSA. For each class n, the corresponding weight vector in the task-specific linear classifier is obtained by putting the class prototype $\overline{\mathbf{h}}^{(n)}$, computed on the support set of the task, through the amortised network. Illustration taken from (Gordon et al., 2019).

L Monte Carlo samples:

$$\begin{aligned} q(\mathbf{Y}^\tau_{\text{test}}|\mathbf{X}^\tau_{\text{test}}, \mathbf{D}^\tau_{\text{train}}, \theta) &= \int p(\mathbf{Y}^\tau_{\text{test}}|\psi^\tau, \theta) q(\psi^\tau|\mathbf{D}^\tau_{\text{train}}, \theta) d\psi^\tau \\ &\approx \frac{1}{L}\sum_{l=1}^{L} p(\mathbf{Y}^\tau_{\text{test}}|\psi^\tau_l, \theta), \quad \text{where } \psi^\tau_l \sim q(\psi^\tau|\mathbf{D}^\tau_{\text{train}}, \theta). \end{aligned} \tag{2.36}$$

Similarly to (Qiao et al., 2018) and LEO (Rusu et al., 2019), VERSA performs context independent amortised inference of the task-specific linear classifier $\mathbf{w}^\tau = \{\mathbf{w}^\tau_n\}_{n=1}^N$ composed of the class-specific vectors $\mathbf{w}^\tau_n$:

$$q_\phi(\mathbf{w}^\tau|\mathbf{D}^\tau_{\text{train}}, \theta) = \prod_{n=1}^{N} q_\phi\left(\mathbf{w}^\tau_n|\mathbf{D}^\tau_{\text{train},n}, \theta\right). \tag{2.37}$$

First, both the train samples $\mathbf{X}^\tau_{\text{train}}$ and the test samples $\mathbf{X}^\tau_{\text{test}}$ go through the feature extractor h_θ parameterised with θ. The prototype, i.e. the average embedding $\overline{\mathbf{h}}^{(n)}$, is then computed for each class n from $\mathbf{D}^\tau_{\text{train}}$. The amortisation network g_ϕ with parameters ϕ takes each of these prototypes as inputs, and infers parameters of the corresponding Gaussian distributions over $\mathbf{w}^\tau_n$ and b^τ_n with diagonal covariance. The classifier $\mathbf{w}^\tau$ and the bias vector $\mathbf{b}^\tau$ are sampled from these distributions using the local reparameterisation trick (Kingma et al., 2015), and they are further applied to the embedded test samples $h(\mathbf{X}^\tau_{\text{test}})$ to produce the softmax output of the linear classification. The latter is used for the cross-entropy loss viewed as the likelihood $p(\mathbf{Y}^\tau_{\text{test}}|\psi^\tau, \theta)$ in (2.36). The computational flow is shown in Figure 2.10. Due to multiple amortisation, VERSA is able to handle tasks with arbitrary number of training samples per class, i.e. "shots", and with arbitrary number of classes per task, i.e. "ways",

which makes the model very flexible and easily scalable without changing the parameter budget. Another advantage of the design is sole reliance on the feed-forward passes, which enables fast inference at test time.

In Section 2.1 we discussed the most common directions of research in few-shot learning. Despite the differences, many of them share the core incentive to develop an efficient way of obtaining the task-specific parameters, e.g. a task-specific classifier for few-shot classification. Our work in Chapter 3 is inspired by the approach to use probabilistic modeling in order to infer weights of the task-specific classifier. Motivated by the lack of works that would exploit the disparity between conditioning on the support set only, and conditioning on the support and query sets combined, we propose a novel probabilistic inference scheme that aims to minimise the distance between the distributions with these two types of conditioning.

2.2 Multi-domain learning

One of the biggest drawbacks of deep learning today is the requirement to train a new model for each new task and domain, even when similar tasks have already been observed in the past. *Multi-domain learning* (MDL) addresses this problem by training a universal model which is able to solve a number of related tasks (e.g. image classification) on a number of different data domains. MDL is partially related to the field of research which aims to create a re-usable, *universal* (Bilen & Vedaldi, 2017) data representation capable of handling different problems. However, this is not the direct goal of MDL. In general, multi-domain learning can be described as searching for the trade-off between re-using fixed problem-agnostic feature representations, and learning highly specialised problem-specific modules to adjust those representations to the current task, all this while reducing the total number of parameters.

A common approach to MDL consists of two steps: a) pre-training and fixing some feature extractor on a large dataset, e.g. ImageNet (Russakovsky et al., 2015), and b) adapting it in some way with an additional set of learnable parameters specific to the domain at hand. Various approaches to multi-domain learning have been proposed in the literature, which can be roughly categorised into three families: masking methods, adaptation modules and everything else. Below, we describe these groups in more details.

2.2.1 Masking methods

One way to adapt a pre-trained but fixed feature extractor to solving related tasks on new domains is to partially re-use its convolutional filters by means of feature selection. Meth-

ods which belong to this family choose relevant features for each domain d by applying corresponding learnable masks $\boldsymbol{\alpha}^d$ to the weights in the pre-trained convolutional filters $\mathbf{f}$. These masks are binary (i.e. $\alpha^d_{mnkl} \in \{0, 1\}$), which is a significant advantage in terms of per-task memory overhead, since they require as little as 1 extra bit per mask parameter. This is especially valuable when masking is applied element-wise. Generally, neural network weights are 32-bit float numbers, so binary masks would then result in $1/32$ (3.12%) memory overhead relative to the size of the base feature extractor (Mallya et al., 2018).

This is the case for *Piggyback* (Mallya et al., 2018), which is one of the first models that introduced binary masks for MDL. For each domain d, and for each shared convolutional layer with the weight tensor $\mathbf{f} \in \mathbb{R}^{M\times N\times K\times K}$, a real-valued tensor $\tilde{\boldsymbol{\alpha}}^d$ of the same shape is learned. This tensor is first put through a hard binary thresholding function, also known as a *straight-through estimator*, with hyperparameter τ as a threshold:

$$\alpha^d_{mnkl} = \begin{cases} 1, & \text{if } \tilde{\alpha}^d_{mnkl} > \tau, \\ 0, & \text{otherwise} \end{cases} \tag{2.38}$$

The obtained binary mask $\boldsymbol{\alpha}^d$ is then used for elementwise multiplication with the weight tensor $\mathbf{f}$, resulting in a new domain-specific convolutional weight tensor $\mathbf{g}^d$ with the weights

$$[\mathbf{g}]^d_{mnkl} = \alpha^d_{mnkl} \cdot [\mathbf{f}]_{mnkl}, \tag{2.39}$$

where $[\mathbf{g}]^d_{mnkl}$ denotes an element of the convolution $\mathbf{g}^d$. Since the hard thresholding function is non-differentiable, the authors suggest to update the real-valued tensor $\tilde{\boldsymbol{\alpha}}^d$ using gradients computed for the masks $\boldsymbol{\alpha}^d$, which can be viewed as noisy estimation of the true gradients of $\boldsymbol{\alpha}^d$. It is argued that such handling of backpropagation can serve as some form of regularisation (Courbariaux et al., 2015; Hubara et al., 2016).

WTPB (Mancini et al., 2018) generalises the previous approach by considering an affine transformation of the shared convolutional weight tensor $\mathbf{f}$. This transformation depends on the domain-specific binary mask $\boldsymbol{\alpha}^d$, as well as domain-specific real-valued coefficients $k^d_i, i \in \{0, 1, 2, 3\}$:

$$\mathbf{g}^d = k^d_0\mathbf{f} + k^d_1\mathbf{1} + k^d_2\boldsymbol{\alpha}^d + k^d_3\mathbf{f} \circ \boldsymbol{\alpha}^d. \tag{2.40}$$

Here, $\circ$ denotes an elementwise multiplication (otherwise known as Hadamard product), and $\mathbf{1}$ is a tensor of ones of the same shape as $\mathbf{f}$. The process of learning the masks is the same as in Piggyback: the binary tensor $\boldsymbol{\alpha}^d$ is obtained by applying the hard thresholding function (2.38) to the real-valued tensor $\tilde{\boldsymbol{\alpha}}^d$, and this function is replaced with the identity function for backpropagation. Since there are only four additional real-valued parameters per layer

compared to Piggyback, the final per-task memory overhead is still very small compared to the base network.

Berriel et al. (2019) argue that different domains and/or tasks may have different memory and computational complexity, with easier tasks requiring even less parameters, as opposed to more difficult tasks. Their proposed model, denoted as BA^2, is able to flexibly satisfy the budget constraints defined by the user, by means of the channel-wise feature selection applied to each shared convolution $\mathbf{f} \in \mathbb{R}^{M \times N \times K \times K}$ in the base network. More precisely, a binary control vector $\boldsymbol{\alpha}^d \in \{0,1\}^N$ is learned that masks entirely some of the N input channels in $\mathbf{f}$:

$$[\mathbf{g}]^d_{mnkl} = \alpha^d_n \cdot [\mathbf{f}]_{mnkl}. \tag{2.41}$$

This not only reduces the total number of domain-specific parameters even further, but also affects the final compuatational graph: the masked convolutional channels can be removed at inference time, resulting in lower computational complexity and storage requirements. As in the aforementioned models, the binary mask vector $\boldsymbol{\alpha}^d$ is the output of the hard thresholding function (2.38) applied to the real-valued vector $\tilde{\boldsymbol{\alpha}}^d$, with the same approach to backpropagation. To satisfy the specified budget $\beta \in [0,1]$, the authors define a constrained optimisation problem with the following generalised Lagrange function:

$$\boldsymbol{\alpha}^{d*} = \arg\min_{\boldsymbol{\alpha}^d} \left[\mathcal{L}(\boldsymbol{\alpha}^d) + \max_{\lambda \geq 0} \left(\lambda \left(\frac{1}{N} \sum_{n=1}^{N} \alpha^d_n - \beta \right) \right) \right], \tag{2.42}$$

where $\mathcal{L}(\boldsymbol{\alpha}^d)$ is the loss for domain d, and λ is the Karush-Kuhn-Tucker (KKT) multiplier.

2.2.2 Adaptation modules

Another common approach to MDL is to represent the network parameters as a combination of domain-agnostic shared layers, referred to as the *base network*, and domain-specific modules inserted into different parts of this base network, referred to as the *adapters*. This structure is usually restricted to the feature extractor, and the following layers are generally viewed as the *output heads*, which are individual for each domain and task. Adapters that have been proposed in the literature differ not only in their expressive power, i.e. the degree to which they can affect the parameters in the base network, but also in their connectivity, i.e. how they interact with the layers in this base network. Similar to the masking approach, the base network is pre-trained on some large dataset, and the weights of the feature extractor are then fixed and shared across all domains.

In DAN (Rosenfeld & Tsotsos, 2018), each convolutional layer $\mathbf{f} \in \mathbb{R}^{M \times N \times K \times K}$ in the feature extractor is adapted to a new domain by applying a trainable linear transformation

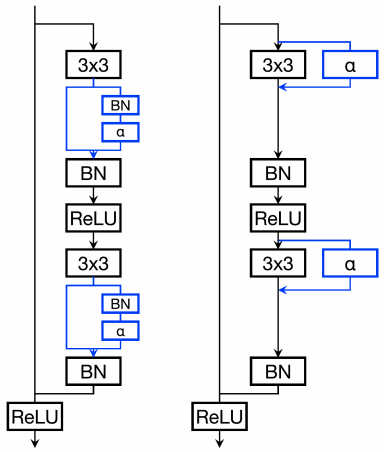


Fig. 2.11 Residual Adapters (left) vs Parallel Adapters (right). Illustration taken from (Rebuffi et al., 2018).

$\boldsymbol{\alpha}^d \in \mathbb{R}^{M \times M}$ to its weights. This is equivalent to inserting a 1×1 domain-specific convolution after each domain-agnostic $K \times K$ convolution in the shared base network:

$$[\mathbf{g}]^d_{mnkl} = \sum_{i=1}^{M} \alpha^d_{mi} \cdot [\mathbf{f}]_{inkl} \tag{2.43}$$

Adapters $\boldsymbol{\alpha}^d$ are only learned for new domains, and are not used for pre-training the feature extractor. The authors argue that such linear transformations of the convolutional layers are limited by the expressive power of the base network. This is probably not the case when these convolutions have residual connections which preserve information from the previous layers, making up for a possible loss of information in the base convolution.

Rebuffi et al. (2017a) go one step further, and add residual connections to these domain-specific 1×1 convolutions $\boldsymbol{\alpha}^d \in \mathbb{R}^{M \times M}$, referred to as *Residual Adapters (RA)*:

$$[\mathbf{g}]^d_{mnkl} = \sum_{i=1}^{M} (1 + \alpha^d_{mi}) \cdot [\mathbf{f}]_{inkl} \tag{2.44}$$

The structure of the residual block in the base ResNet feature extractor is described in Figure 2.11 (left). Unlike DAN, RA uses these sequentially connected adaptation modules during the pre-training stage, which empirically improves the model performance compared to standard pre-tratining but makes the method less flexible about the changes in the design of the adapters, e.g. when moving to a new task.

Parallel Adapters (PA) (Rebuffi et al., 2018) address this problem by proposing a different topology of the adapter's connectivity. More precisely, the sequential residual adaptation module, which follows each $K \times K$ convolution in the base network, is replaced with a

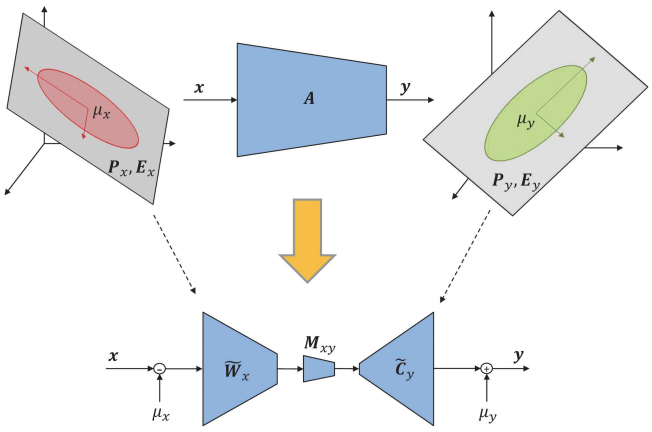


Fig. 2.12 Procedure of covariance normalisation in CovNorm. As a result, each adaptation layer $\mathbf{A}$ is approximated with a series of three transformations: projection into the input PCA space with $\tilde{\boldsymbol{W}}_x$, followed by mini-adaptation with $\mathbf{M}_{xy}$, and reconstruction into the output PCA space with $\tilde{\mathbf{W}}_y$. Illustration taken from (Li & Vasconcelos, 2019).

learnable parallel residual bypass $\boldsymbol{\alpha}^d \in \mathbb{R}^{M \times N}$ represented, again, by 1×1 convolution:

$$[\mathbf{g}]^d_{mnkl} = [\mathbf{f}]_{mnkl} + \begin{cases} \alpha^d_{mn} & \text{if } k = l = (K-1)/2 + 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.45}$$

The structure of the ResNet block with the inserted PA modules is shown in Figure 2.11 (right). The additive nature of the adaptation allows to use the PA modules in a *plug-and-play* manner, which increases their flexibility when moving to new domains and tasks, and provides applicability to the off-the-shelf networks.

All the adaptation modules discussed so far make use of 1×1 convolutions to modify the weights of each $K \times K$ convolution in the base network. This results in parameter reduction $\sim \frac{1}{K^2}$ per layer compared to full finetuning. If there are a lot of domains (i.e. the number of domains $D \gg K^2$), the total parameter budget may still be quite large. *CovNorm* (Li & Vasconcelos, 2019) aims to further reduce the number of additional parameters required for each domain d by using data-driven low rank approximation of existing adapters. The algorithm consists of multiple steps, starting with learning an adaptation module for each layer in the base network, e.g. the sequential adapter $A^d \in \mathbb{R}^{M \times M}$ from DAN. Assuming that inputs x^d and outputs y^d of each adapter ($y^d = A^d x^d$) are Gaussian random variables with means μ^d_x, μ^d_y and covariances Σ^d_x, Σ^d_y, the PCAs are computed for x^d and y^d as eigendecompositions of the corresponding covariance matrices. Then, k_x and k_y number of most significant eigenvalues are kept, together with the corresponding eigenvectors. If $\tilde{E}^d_x$ and $\tilde{E}^d_y$ are the diagonal matrices with these largest eigenvalues, and $\tilde{P}^d_x$ and $\tilde{P}^d_y$ are the

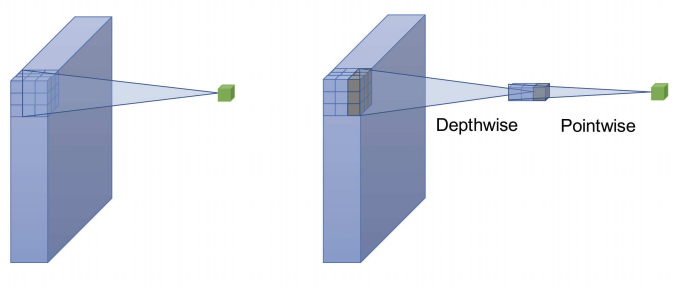


Fig. 2.13 Standard convolution (left) vs depthwise separable convolution (right). Illustration taken from (Guo et al., 2019a).

matrices of the corresponding eigenvectors, then the adapter A^d is approximated as follows:

$$y^d = \tilde{C}_y^d M_{x,y}^d \tilde{W}_x^d (x^d - \mu_x^d) + \mu_y^d, \tag{2.46}$$

where $\tilde{C}_y^d = \tilde{P}_y^d \sqrt{\tilde{E}_y^d}$ is a "coloring" matrix, $\tilde{W}_x^d = \sqrt{\tilde{E}_x^d} (\tilde{P}_x^d)^T$ is a "whitening" matrix, and $M_{x,y}^d$ is a learnable mini-adaptation layer inserted to correct mismatch between the input and output PCAs. The structure of the model is shown in Figure 2.12. The module $M_{x,y}^d$ is then finetuned on the domain d, and absorbed into either $\tilde{C}_y^d$ or $\tilde{W}_x^d$, depending on which of the numbers, k_y or k_x, is smaller. The final model has $2M \min(k_x, k_y)$ parameters in total.

2.2.3 Other approaches to multi-domain learning

Both masking methods and models with continuous adaptation modules share one core idea that the base network should be pre-trained once and then fixed, while all domain-specific parts constitute a disjoint set of parameters, which are trained separately and do not affect learning on other domains. The major advantage of this approach is that, once trained, the performance on each domain is preserved and guaranteed not to decline, which allows to use these models in the incremental learning setup where the tasks get observed sequentially, and only once. This feature also provides the great flexibility in finetuning domain-specific parameters to new data, if necessary, or even allows to choose an entirely different approach for each domain, e.g. to use Piggyback masks on one domain, and PA on the other. On the other hand, the disjoint nature of domain-specific parameters means that the correlation between the domains is used very weakly, through relatedness of each domain to the one that had been used during pre-training.

Despite the variety of approaches proposed for MDL in the literature, all of them use similar off-the-shelf convolutional networks, e.g. ResNet (He et al., 2016a) or DenseNet (Huang et al., 2017). As opposed to those, Guo et al. (2019a) suggest to replace the standard convo-

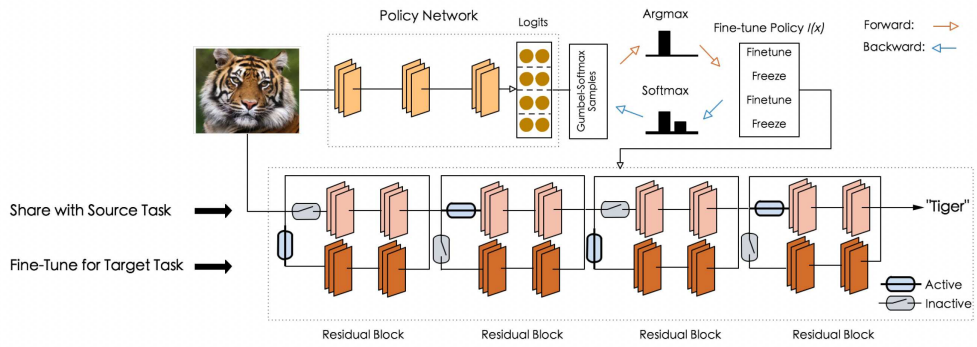


Fig. 2.14 Multi-domain learning architecture in SpotTune. For each sample, the trainable policy network infers routing decisions, defining whether the block in the feature extractor should be fintuned or re-used after pre-training. Illustration taken from from (Guo et al., 2019b).

lutional layers with the depthwise separable convolutions. The latter factorise the standard $M \times N \times K \times K$ convolutions into the sequences of the $N \times K \times K$ depthwise and $M \times N \times 1 \times 1$ pointwise convolutions, where M is the number of output channels, N is the number of input channels, and $K \times K$ is the filter resolution. The difference between the two types of convolution is illustrated in Figure 2.13. Then, it is argued that different domains might share cross-channel correlations while having different spatial correlations, so the authors consider the pointwise convolutions to be the domain-agnostic shared parameters, and the depthwise convolutions are viewed as the domain-specific learnable parameters. The resulting network is very compact, since the pointwise convolutions make up the largest portion of the depthwise separable convolutions, and sharing them across all domains greatly reduces the total parameter count. In particular, the "base" convolutional layers, represented by the pointwise convolutions, now have MN parameters instead of MNK^2, and the "adapters", represented by the depthwise separable convolutions, have NK^2 parameters.

Unlike the previous models, MTAN (Liu et al., 2019a) jointly learns domain-specific attention modules, together with the corresponsing convolutional blocks in the shared feature extractor, which they are interconnected with. In this setup, all tasks affect the base network which allows to benefit from the similarities between them. Each attention module consists of two 1×1 convolutions, and one 3×3 convolution. A soft attention mask a, produced inside each attention module and element-wise multiplied with the corresponding shared representation, performs domain-specific feature selection, thus determining the relevance of these features to the domain at hand. Inputs to all attention modules except the first are concatenations of the corresponding shared features and the domain-specific features produced by the preceding attention module.

Instead of viewing the feature extractor as a completely shared set of parameters which is augmented with a domain-specific set of parameters, SpotTune (Guo et al., 2019b) uses a single base network which is selectively finetuned in the input-dependent parts based on the learnable decision policy. The workflow of the model is illustrated in Figure 2.14. For each layer l in the base network, the decision policy $I_l(x)$ is modelled as a binary random variable sampled from the categorical distribution with parameters α_1 and α_2, and it decides whether the layer should be finetuned or re-used from the base network. Parameters of this distribution are produced by a small neural network which takes the instance x as an input, and predicts probabilities α_1 and α_2. The discrete nature of the decision policy makes the network non-differentiable, so the authors resort to the Gumbel-Softmax approach (Maddison et al., 2017) to draw samples from the categorical distribution in a differentiable manner.

In Section 2.2 we made a survey of the works in multi-domain learning. The prevailing approach to learning a model in this setup is to pre-train the feature extractor on a large data set, and adapt it to the novel domains. The task-specific overheads, i.e. the classifier, are usually out of scope and are trained from scratch, which moves the focus of the multi-domain learning problem towards adjustment of feature extraction. We take inspiration from the group of works which learn continuous adaptation units, specialised individually for each domain, and we continue to investigate possible interactions between the adapters and the convolutional weight tensors. As a result of this exploration, we propose a novel multiplicative adaptation module for multi-domain learning in Chapter 4.

Chapter 3

Meta-Learning with Shared Amortised Variational Inference

We propose a novel amortised variational inference scheme for an empirical Bayes meta-learning model, where model parameters are treated as latent variables. We learn the prior distribution over model parameters conditioned on limited training data using a variational autoencoder approach. Our framework proposes sharing the same amortised inference network between the conditional prior and variational posterior distributions over the model parameters. While the posterior leverages both the labeled support and query data, the conditional prior is based only on the labeled support data. We show that in earlier work, relying on Monte-Carlo approximation, the conditional prior collapses to a Dirac delta function. In contrast, our variational approach prevents this collapse and preserves uncertainty over the model parameters. We evaluate our approach on the miniImageNet, CIFAR-FS and FC100 datasets, and present results demonstrating its advantages over previous work.

This chapter is based on the following publication:

Iakovleva, E., Verbeek, J., Alahari, K. Meta-Learning with Shared Amortised Variational Inference. In *ICML*, 2020.

3.1 Introduction

While people have an outstanding ability to learn from just a few examples, generalisation from small sample sizes has been one of the long-standing goals of machine learning. Meta-learning, or "learning to learn" (Schmidhuber, 1999), aims to improve generalisation in small sample-size settings by leveraging the experience of having learned to solve related tasks in

the past. The core idea is to learn a meta model that, for any given task, maps a small set of training samples to a model that generalises well. Figure 1.5 illustrates the structure of the tasks in meta-learning.

A recent surge of interest in meta-learning has explored a wide spectrum of approaches. This includes nearest neighbor based methods (Guillaumin et al., 2009; Vinyals et al., 2016), nearest class-mean approaches (Dvornik et al., 2019; Mensink et al., 2012; Ren et al., 2018; Snell et al., 2017), optimisation based methods (Finn et al., 2017; Ravi & Larochelle, 2017), adversarial approaches (Zhang et al., 2018), and Bayesian models (Gordon et al., 2019; Grant et al., 2018). The Bayesian approach is particularly interesting, since it provides a coherent framework to reason about model uncertainty, not only in small sample-size settings, but also others such as incremental learning (Kochurov et al., 2018), and ensemble learning (Gal & Ghahramani, 2016). Despite its attractive properties, intractable integrals over model parameters or other latent variables, which are at the heart of the Bayesian framework, make it often necessary to turn to stochastic Monte Carlo or analytic approximations for practical implementations.

In our work, we follow the Bayesian latent variable approach, and learn a prior over the parameters of the classification model conditioned on a small training sample set for the task. We use a variational inference framework to approximate the intractable marginal likelihood function during training. The variational distribution approximates the posterior over the parameters of the classification model, given training and test data. Both the prior and posterior are parameterised as deep neural networks that take a set of labeled data points as input. By sharing the inference network across these two distributions, we leverage more data to learn these conditionals and avoid overfitting.

We compare the variational training approach with the Monte Carlo approach followed by Gordon et al. (2019) on synthetic data. We find that when using a small number of samples for stochastic back-propagation in the Monte Carlo approach, which results in faster training, the prior collapses to a Dirac delta, and the model degenerates to a deterministic parameter generating network. In contrast, our variational training approach does not suffer from this deficiency, and leads to an accurate estimation of the variance. Experiments on few-shot image classification using the miniImageNet, CIFAR-FS and FC100 datasets confirm these findings, and we observe improved accuracy using the variational approach to train the VERSA model (Gordon et al., 2019). Moreover, we use the same variational framework to train a stochastic version of the TADAM few-shot image classification model (Oreshkin et al., 2018), replacing the deterministic prototype classifier with a scaled cosine classifier with stochastic weights. Our stochastic formulation significantly improves performance over the

base architecture, and yields results competitive with the state of the art on the miniImageNet, CIFAR-FS and FC100 datasets.

3.2 Related work

Distance-based classifiers. A straightforward approach to handle small training sets is to use nearest neighbor (Weinberger et al., 2006; Guillaumin et al., 2009; Vinyals et al., 2016), or nearest prototype (Mensink et al., 2012; Snell et al., 2017; Dvornik et al., 2019; Ren et al., 2018; Oreshkin et al., 2018) classification methods. In a "meta" training phase, a metric – or, more generally, a data representation – is learned using samples from a large number of classes. At test time, the learned metric can then be used to classify samples across a set of classes not seen during training, by relying on distances to individual samples or "prototypes," i.e., per-class averages. Alternatively, it is also possible to learn a network that takes two samples as input and predicts whether they belong to the same class (Sung et al., 2018). Other work has explored the use of task-adaptive metrics, by conditioning the feature extractor on the class prototypes for the task at hand (Oreshkin et al., 2018). We show that our latent variable approach is complementary and improves the effectiveness of the latter task conditioning scheme.

Optimisation-based approaches. Deep neural networks are typically learned from large datasets using SGD. To adapt to the regime of (very) small training datasets, optimisation-based meta-learning techniques replace the vanilla SGD approach with a trainable update mechanism (Bertinetto et al., 2019; Finn et al., 2017; Ravi & Larochelle, 2017). For example, MAML (Finn et al., 2017) learns a parameter initialisation, such that a small number of SGD updates yields good performance. In addition to parameter initialisation, application of LSTM model as a means to control how gradient influences updates of the current parameters has also been explored (Ravi & Larochelle, 2017). In our work, an amortised inference network makes a single feed-forward pass through the data to estimate a distribution over the parameters, instead of making multiple passes to update the parameters.

Latent variable models. Gradient-based estimators of parameters tend to have high variance when the sample sizes are small. It is natural to explicitly model this variance by treating parameters as latent variables in a Bayesian framework (Garnelo et al., 2018; Gordon et al., 2019; Grant et al., 2018; Kim et al., 2019; MacKay, 1991; Neal, 1995). The marginal likelihood of test labels given a training set is then obtained by integrating out these latent model parameters. This integral, required for training and prediction, is typically intractable, but it can be approximated using (amortised) variational inference (Garnelo et al., 2018; Kim et al., 2019), Monte Carlo sampling (Gordon et al., 2019), or Laplace approximation

(Grant et al., 2018). Neural processes (Garnelo et al., 2018; Kim et al., 2019) are also related to our work in their structure and their use of a shared inference network for the prior and the variational posterior. While neural processes use the task-specific latent variable as an additional input to the classification network, we explicitly model parameters of the linear classifier corresponding to each class as the latent variables. This increases interpretability of the latent space, and allows for a varying number of classes in different tasks.

Interestingly, some optimisation-based approaches can be viewed as approximate inference methods in latent variable models (Grant et al., 2018; Rusu et al., 2019). Semi-amortised inference techniques (Marino et al., 2018; Kim et al., 2018b), which combine feed-forward parameter initialisation and iterative gradient-based refinement of the approximate posterior, can be seen as a hybrid of optimisation-based and Bayesian approaches. Deterministic approaches that generate a single parameter vector for the task model, given a set of training samples (Bertinetto et al., 2016; Ha et al., 2017; Qiao et al., 2018), can be seen as a special case of the latent variable model with Dirac delta conditional distributions of the parameters.

3.3 Our meta-learning approach

We follow the common meta-learning setting of episodic training for K-shot N-way classification on the *meta-train* set with classes C (Finn et al., 2017; Gordon et al., 2019; Ravi & Larochelle, 2017). For each classification task t sampled from a distribution over tasks $p(\mathcal{T})$, the training data $D^t = \{(\boldsymbol{x}^t_{k,n}, \boldsymbol{y}^t_{k,n})\}^{K,N}_{k,n=1}$ (*support* set) consists of K pairs of samples $\boldsymbol{x}^t_{k,n}$ and their labels $\boldsymbol{y}^t_{k,n}$ from each of N classes. The meta-learner takes the KN labeled samples as input, and outputs a classifier across these N classes to classify MN unlabeled samples from the testing data $\tilde{D}^t = \{(\tilde{\boldsymbol{x}}^t_{m,n}, \tilde{\boldsymbol{y}}^t_{m,n})\}^{M,N}_{m,n=1}$ (*query* set). During the *meta-train* stage, the meta-learner iterates over T episodes where each episode corresponds to a particular task t. During the *meta-test* stage, the model is presented with new tasks where the support and query sets are sampled from the meta-test set, which consists of previously unseen classes C'. The support set is used as input to the trained meta-learner, and the classifier produced by meta-learning is used to evaluate the performance on the query set. Results are averaged over a large set of meta-test tasks.

In this section, we propose a probabilistic framework for meta-learning. In Section 3.3.1, we start with the description of the multi-task graphical model that we adopt. We then derive an amortised variational inference scheme with learnable prior for this generative model in Section 3.3.2, and propose to share the amortised networks between the prior and the approximate posterior. Finally, in Section 3.3.3 we describe the design of our model, SAMOVAR, which is trained with the proposed shared variational inference method.

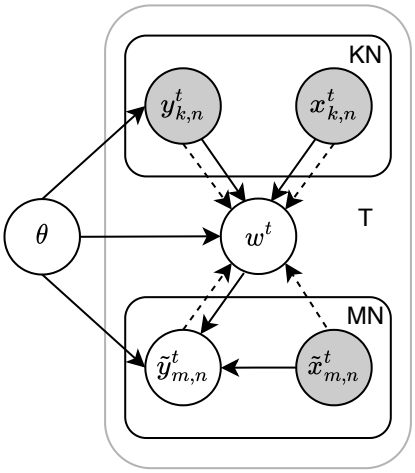


Fig. 3.1 Hierarchical graphical model. The solid lines correspond to the generative process, while the dashed lines correspond to the variational inference procedure. Shaded nodes represent observed variables, non-shaded ones correspond to latent variables.

3.3.1 Generative meta-learning model

We employ a hierarchical graphical model shown in Figure 3.1. This multi-task model includes latent parameters θ, shared across all the T tasks, and task-specific latent parameters $\{w^t\}_{t=1}^T$. The marginal likelihood of the query labels $\tilde{Y} = \{\tilde{Y}^t\}_{t=1}^T$, given the query samples $\tilde{X} = \{\tilde{X}^t\}_{t=1}^T$ and the support sets $D = \{D^t\}_{t=1}^T$, is obtained as

$$p(\tilde{Y}|\tilde{X}, D) = \int p(\theta) \prod_{t=1}^{T} \int p(\tilde{Y}^t|\tilde{X}^t, \theta, w^t) p(w^t|D^t, \theta) \mathrm{d}w^t \mathrm{d}\theta. \tag{3.1}$$

The first term, $p(\theta)$, is the prior over the global task-independent parameters θ. The second term, $p(\tilde{Y}^t|\tilde{X}^t, \theta, w^t)$, is the likelihood of query labels $\tilde{Y}^t$, given query samples $\tilde{X}^t$, task-agnostic parameters θ and task-specific parameters w^t. For example, this could be a linear classifier with weights w^t over features $f(\cdot)_\theta$ computed by a network with parameters θ from input $\tilde{X}^t$. The third term, $p(w^t|D^t, \theta)$, is the conditional distribution on the task parameters w^t given the support set D^t and global parameters θ. We parameterize this distribution with a deep neural network with parameters ϕ as $p_\phi\left(w^t|D^t, \theta\right)$.

Following Gordon et al. (2019); Grant et al. (2018); Hu et al. (2020), we consider a point estimate for θ to simplify the model. The per-task marginal likelihood is then

$$p(\tilde{Y}^t|\tilde{X}^t, D^t, \theta) = \prod_{m=1}^{M} p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, D^t, \theta), \tag{3.2}$$

$$p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, D^t, \theta) = \int p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, \theta, w^t) p_\phi(w^t|D^t, \theta) \mathrm{d}w^t, \tag{3.3}$$

To train the model, a Monte Carlo approximation of the integral in Eq. (3.3) was used in Gordon et al. (2019):

$$\mathcal{L}(\theta, \phi) = \frac{1}{TM} \sum_{t=1}^{T} \sum_{m=1}^{M} \log \frac{1}{L} \sum_{l=1}^{L} p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, \theta, w_l^t), \tag{3.4}$$

where $w_l^t \sim p_\phi(w^t|D^t, \theta)$. In our experiments in Section 3.4, we show that training with this approximation tends to severely underestimate the variance in $p_\phi(w^t|D^t, \theta)$, effectively reducing the model to a deterministic one, and defying the use of a stochastic latent variable model.

3.3.2 Shared amortised variational inference

To prevent the conditional prior $p_\phi(w^t|D^t, \theta)$ from degenerating, we use amortised variational inference (Kingma & Welling, 2014; Rezende et al., 2014) to approximate the intractable true posterior $p(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. Using the approximate posterior $q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$ parameterized by ψ, we obtain the variational evidence lower bound (ELBO) of Eq. (3.3) as

$$\begin{aligned} &\log p(\tilde{Y}^t|\tilde{X}^t, D^t, \theta) \geq \\ &\qquad \mathbb{E}_{q_\psi}\left[\log p(\tilde{Y}^t|\tilde{X}^t, \theta, w^t)\right] - \mathcal{D}_{\mathrm{KL}}\left(q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)||p_\phi(w^t|D^t, \theta)\right). \end{aligned} \tag{3.5}$$

The first term can be interpreted as a reconstruction loss, that reconstructs the labels of the query set using latent variables w^t sampled from the approximate posterior, and the second term as a regularizer that encourages the approximate posterior to remain close to the conditional prior $p_\phi(w^t|D^t, \theta)$. We approximate the reconstruction term using L Monte Carlo samples, and add a regularization coefficient β to weigh the KL term (Higgins et al., 2017). With this, our optimisation objective is:

$$\begin{aligned} \hat{\mathcal{L}}(\Theta) = \frac{1}{T} \sum_{t=1}^{T} \Bigg[&\sum_{m=1}^{M} \frac{1}{L} \sum_{l=1}^{L} \log p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, \theta, w_l^t) \\ &-\beta \mathcal{D}_{\mathrm{KL}}\left(q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)||p_\phi(w^t|D^t, \theta)\right)\Bigg], \end{aligned} \tag{3.6}$$

where $w_l^t \sim q_\psi(w|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. We maximise the ELBO w.r.t. $\Theta = \{\theta, \phi, \psi\}$ to jointly train the model parameters θ, ϕ, and the variational parameters ψ.

We use Monte Carlo sampling from the learned model to make predictions at test time as:

$$p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, D^t, \theta) \approx \frac{1}{L}\sum_{l=1}^{L} p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, \theta, w_l^t), \tag{3.7}$$

where $w_l^t \sim p_\phi(w^t|D^t, \theta)$. In this manner, we leverage the stochasticity of our model by averaging predictions over multiple realisations of w^t.

The approach presented above suggests to train separate networks to parameterise the conditional prior $p_\phi(w^t|D^t, \theta)$ and the approximate posterior $q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. However, since in both cases the conditioning data consists of labeled samples, it is possible to share the network for both distributions, and simply change the input of the network to obtain one distribution or the other. Sharing has two advantages: (i) It reduces the number of parameters to train, decreasing the memory footprint of the model and the risk of overfitting. (ii) It facilitates learning a non-degenerate prior, i.e. a prior that is not a Dirac delta.

Let us elaborate on the second point. Omitting all dependencies for brevity, KL divergence $\mathcal{D}_{\text{KL}}(q||p) = \int q(w)\left[\log q(w) - \log p(w)\right]$ in Eq. (3.5) compares the posterior $q(w)$ with the prior $p(w)$. Consider the case when the prior converges to a Dirac delta, while the posterior does not. Then, there would exist points in the support of the posterior for which $p(w) \approx 0$, therefore, the KL divergence would tend to infinity. The only alternative in this case is for the posterior to converge to the same Dirac delta. This means that, for different inputs, the inference network would produce the same deterministic latent variable w. In particular, additional conditioning data from the query set, which is available in the posterior, would leave the inference unchanged, and the network would effectively ignore and fail to incorporate new data. While in theory this is possible, we do not observe it in practice.

We coin our approach "SAMOVAR", short for Shared AMOrtised VARiational inference. Figure 3.2 illustrates the overall structure of our model.

3.3.3 Implementing SAMOVAR: architectural designs

The key properties we expect SAMOVAR to have are: (i) the ability to perform the inference in a feed-forward way (unlike gradient-based models), and (ii) the ability to handle a variable number of classes within the tasks. We build upon the work of Gordon et al. (2019); Qiao et al. (2018), to meet both these requirements. We start with VERSA (Gordon et al., 2019) where the feature extractor is followed by an amortised inference network, which returns a

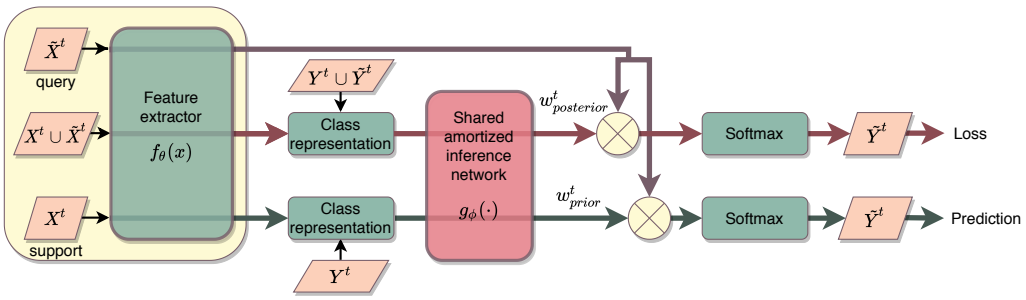


Fig. 3.2 SAMOVAR, our meta-learning model for few-shot image classification. For task t, query data $\tilde{X}^t$ and support data X^t are put through a task-agnostic feature extractor $f_\theta(x)$. The features are then averaged class-wise, and mapped by the shared amortised inference network into prior and posterior over the task-specific classifier weight vectors. Classifiers $w^t_{posterior}$ and w^t_{prior} sampled from these distributions map query features $f_\theta(\tilde{X}^t)$ to predictions on the query labels $\tilde{Y}^t$ used in training and testing, respectively.

linear classifier with stochastic weights. SAMOVAR-base, our baseline architecture built this way on VERSA, consists of the following components.

Task-independent feature extractor. We use a deep convolutional neural network (CNN), f_θ, shared across all tasks, to embed input images $\boldsymbol{x}$ in $\mathbb{R}^d$. The extracted features are the only information from the samples used in the rest of the model. The CNN architectures used for different datasets are detailed in Section 3.4.2.

Task-specific linear classifier. Given the features, we use multi-class logistic discriminant classifier, with task-specific weight matrix $w^t \in \mathbb{R}^{N\times d}$. That is, for the query samples $\tilde{\boldsymbol{x}}$ we obtain a distribution over the labels as:

$$p(\tilde{\boldsymbol{y}}^t_m|\tilde{\boldsymbol{x}}^t_m, \theta, w^t) = \text{softmax}\left(w^t f_\theta(\tilde{\boldsymbol{x}}^t_m)\right). \tag{3.8}$$

Shared amortised inference network. We use a deep permutation invariant network g_ϕ to parameterise the prior over the task-specific weight matrix w^t, given a set of labeled samples. The distribution over w^t is factorised over its rows $w^t_1, \ldots, w^t_N$ to allow for variable number of classes, and to simplify the structure of the model. For any class n, the inference network g_ϕ maps the corresponding set of support feature embeddings $\{f_\theta(\boldsymbol{x}^t_{k,n})\}^K_{k=1}$ to the parameters of a distribution over w^t_n. We use a Gaussian with diagonal covariance to model these distributions of the weight vectors, i.e.,

$$p_\phi(w^t_n|D^t, \theta) = \mathcal{N}(\boldsymbol{\mu}^t_n, \text{diag}\left(\boldsymbol{\sigma}^t_n\right)), \tag{3.9}$$

where the mean and the variance are computed by the inference network as:

$$\begin{bmatrix} \boldsymbol{\mu}_n^t \\ \boldsymbol{\sigma}_n^t \end{bmatrix} = g_\phi \left(\frac{1}{K} \sum_{k=1}^{K} f_\theta(\boldsymbol{x}_{k,n}^t) \right). \tag{3.10}$$

To achieve permutation invariance among the samples, we average the feature vectors within each class before feeding them into the inference network g_ϕ. The approximate variational posterior is obtained in the same manner, but in this case the feature average that is used as input to the inference network is computed over the union of labeled support and query samples.

To further improve the model, we employ techniques commonly used in meta-learning classification models: scaled cosine similarity, task conditioning, and auxiliary co-training.

Scaled cosine similarity. Cosine similarity based classifiers have recently been widely adopted in few-shot classification (Dvornik et al., 2019; Gidaris et al., 2019; Lee et al., 2019; Oreshkin et al., 2018; Ye et al., 2018). Here, the linear classifier is replaced with a classifier based on the cosine similarity with the weight vectors w_n^t, scaled with a temperature parameter α:

$$p(\tilde{\boldsymbol{y}}_m^t | \tilde{\boldsymbol{x}}_m^t, \theta, w_n^t) = \text{softmax}\left(\alpha \frac{f_\theta(\tilde{\boldsymbol{x}}_m^t)^\top w_n^t}{||f_\theta(\tilde{\boldsymbol{x}}_m^t)|| \cdot ||w_n^t||} \right) \tag{3.11}$$

We refer to this version of our model as SAMOVAR-SC.

Task conditioning. The limitation of the above models is that the weight vectors $\boldsymbol{w}_n^t$ depend only on the samples from class n. To leverage the full context of the task, we adopt the task embedding network (TEN) of Oreshkin et al. (2018). For each feature dimension of f_θ, TEN provides an affine transformation conditioned on the task data, similar to FiLM conditioning layers (Perez et al., 2018) and conditional batch normalization (Munkhdalai et al., 2018; Dumoulin et al., 2017). In particular, the input to TEN is the average $\boldsymbol{c}^t = \frac{1}{N} \sum_n \boldsymbol{c}_n^t$ of the per-class prototypes $\boldsymbol{c}_n^t = \frac{1}{K} \sum_k f_\theta(\boldsymbol{x}_{kn}^t)$ in the task t, and the outputs are the translation and scale parameters for all feature channels in the feature extractor layers. In SAMOVAR, we use TEN to modify both the support and query features f_θ before they enter the inference network g_ϕ. The query features that enter the linear/cosine classifiers are left unchanged.

Auxiliary co-training. Large feature extractors can benefit from auxiliary co-training to prevent overfitting, stabilise the training, and boost the performance (Oreshkin et al., 2018). We leverage this by sharing the feature extractor f_θ of the meta-learner with an auxiliary classification task across all the classes in the meta-train set, using the cross-entropy loss for a linear logistic classifier over f_θ.

3.4 Experiments

In this section we provide evaluation of the proposed model in various experimental setups. We analyze the differences between training with Monte Carlo estimation and variational inference with a controlled synthetic data experiment in Section 3.4.1. After that we present the experimental setup that we use for few-shot image classification in Section 3.4.2. Finally, we discuss the obtained results and compare our model to related work in Section 3.4.4.

3.4.1 Synthetic data experiments

We consider the same hierarchical generative process as Gordon et al. (2019), which allows for exact inference:

$$p(\psi^t) = \mathcal{N}(0, 1), \qquad p(y^t|\psi^t) = \mathcal{N}(\psi^t, \sigma_y^2), \tag{3.12}$$

where y^t is an observed variable, and ψ^t is a latent variable. We sample $T = 250$ tasks, each with $K = 5$ support observations $D^t = \{y_k^t\}_{k=1}^K$, and $M = 15$ query observations $\tilde{D}^t = \{\tilde{y}_m^t\}_{m=1}^M$. We use an inference network $q_\phi(\psi|D^t) = \mathcal{N}(\mu_q, \sigma_q^2)$ parameterised with $\phi = \{W, b\}$, where

$$\begin{bmatrix} \mu_q \\ \log \sigma_q^2 \end{bmatrix} = W \sum_{k=1}^{K} y_k^t + \boldsymbol{b}. \tag{3.13}$$

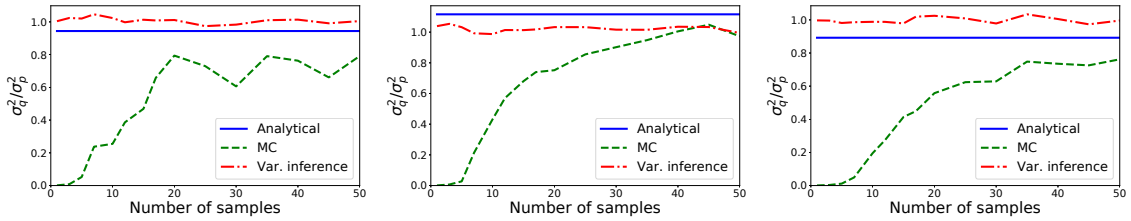
This inference network with trainable parameters ϕ is used to define the predictive distribution

$$p(\tilde{D}^t|D^t) = \int p(\tilde{D}^t|\psi) q_\phi(\psi|D^t) \, \mathrm{d}\psi. \tag{3.14}$$

Since the prior is conjugate to the Gaussian likelihood $p(y^t|\psi^t)$ in Eq. (3.12), we can analytically compute the marginal $p(\tilde{D}^t|D^t)$ in Eq. (3.14), as well as the true posterior $p(\psi|D^t)$, which are both Gaussian. We then train the inference network by optimising the logarithm of Eq. (3.14) in three different ways. Assuming T tasks with M query samples each, optimisation objectives look as follows:

1. **Exact marginal log-likelihood.** We compute the task and sample mean of the logarithm of the marginal likelihood computed analytically:

$$\mathcal{L}(\phi) = -\frac{1}{MT} \sum_{t=1}^{T} \sum_{m=1}^{M} \log \mathcal{N}(y_m^t; \mu_q(D^t), \sigma_q^2(D^t) + \sigma_y^2). \tag{3.15}$$



(a) $\sigma_y = 0.1$ (b) $\sigma_y = 0.5$ (c) $\sigma_y = 1.0$

Fig. 3.3 Ratio between the variance in ψ estimated by the trained inference network $q_\phi(\psi|D^t)$ and σ_p^2 in true posterior $p(\psi|D^t)$, for different number of samples L from the inference network during training.

2. **Monte Carlo estimation.** We draw L samples $\psi_l^t \sim q_\phi(\psi|D^t)$ from the rightmost term in Eq. (3.14) to estimate this integral using Eq. (3.12):

$$\mathcal{L}(\phi) = -\frac{1}{MT}\sum_{t=1}^{T}\sum_{m=1}^{M}\log\frac{1}{L}\sum_{l=1}^{L}\mathcal{N}(y_m^t;\psi_l^t,\sigma_y^2). \tag{3.16}$$

3. **Variational inference.** We use the inference network with a second set of parameters ϕ' as the variational posterior $q_{\psi'}$ conditioned on both $\tilde{D}^t$ and D^t. Using L samples $\psi_l^t \sim q_{\phi'}(\psi|\tilde{D}^t, D^t)$, we obtain the variational lower bound:

$$\mathcal{L}(\phi) = -\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{m=1}^{M}\frac{1}{L}\sum_{l=1}^{L}\log\mathcal{N}(y_m^t;\psi_l^t,\sigma_y^2) - \mathcal{D}_{\mathrm{KL}}(q_{\phi'}(\psi|\tilde{D}^t, D^t)||q_\phi(\psi|D^t))\right]. \tag{3.17}$$

We train these three losses with $\sigma_y \in \{0.1, 0.5, 1.0\}$. For Monte Carlo and variational methods, we use the re-parameterisation trick (Kingma & Welling, 2014; Rezende et al., 2014) in order to differentiate through sampling of ψ. We evaluate the quality of the trained inference network by sampling data D^t for a new task from the data generating process Eq. (3.12). For new data, we compare the true posterior $p(\psi|D^t)$ with the distribution $q_\phi(\psi|D^t)$ produced by the trained inference network.

Results in Figure 3.3 show that both the analytic and variational approaches recover the true posterior very well, including variational training with a single sample per parameter update. Monte Carlo training, on the other hand, requires the use of significantly larger sets of samples to produce results comparable to other two approaches. Optimisation with a small number of samples leads to significant underestimation of the target variance. This makes the Monte Carlo training approach either computationally expensive, or inaccurate in modeling the uncertainty in the latent variable.

3.4.2 Experimental setup for image classification

We now move to the experiments on the real image data. We consider three main benchmarks for few-shot learning: MiniIamgeNet, FC100 and CIFAR-FS.

MiniImageNet (Vinyals et al., 2016) consists of 100 classes selected from ILSVRC-12 (Russakovsky et al., 2015). We follow the split from Ravi & Larochelle (2017) with 64 meta-train, 16 meta-validation and 20 meta-test classes, and 600 images in each class. Following Oreshkin et al. (2018), we use a central square crop, and resize it to 84×84 pixels.

FC100 (Oreshkin et al., 2018) was derived from CIFAR-100 (Krizhevsky, 2009), which consists of 100 classes, with 600 32×32 images per class. All classes are grouped into 20 superclasses. The data is split by superclass to minimise the information overlap. There are 60 meta-train classes from 12 superclasses, 20 meta-validation, and 20 meta-test classes, each from four corresponding superclasses.

CIFAR-FS (Bertinetto et al., 2019) is another meta-learning dataset derived from CIFAR-100. It was created by a random split into 64 meta-train, 16 meta-validation and 20 meta-test classes. For each class, there are 600 images of size 32×32.

Unless explicitly mentioned, we do not use data augmentation. In cases where we do use augmentation, it is performed with random horizontal flips, random crops, and color jitter (brightness, contrast and saturation).

3.4.3 Network architectures and training specifications

We learn separate amortised inference networks to predict the mean μ and log-variance $\ln \sigma^2$ of the latent classification weight vectors w^t. Both networks have the same architecture, which depends on the feature extractor that is used. The inference networks are shared between the prior and approximate posterior distributions. In this work, we use two feature extractors: CONV-5 (Gordon et al., 2019) to compare with VERSA (Gordon et al., 2019), and ResNet-12 (Oreshkin et al., 2018) for other experiments.

CONV-5 Feature Extractor. The embedding of the image returned by the CONV-5 feature extractor is a 256-dimensional vector. Each of the inference networks for the mean and log variance of the classifier weights w^t consists of three fully connected layers with 256 input and output features, and ELU non-linearity (Clevert et al., 2016) between the layers. There are two additional inference networks that predict the mean and log variance of the classifier biases b^t. Both of them consist of two fully connected layers with 256 input and output features followed by ELU non-linearity, and a fully connected layer with 256 input and a single output feature. The design is the same as used in VERSA Gordon et al. (2019)

to ensure comparability. For a fair comparison, we follow the same experimental setup, including the network architectures, optimisation procedure, and episode sampling.

ResNet-12 Feature Extractor. With the ResNet-12 feature extractor, every image is embedded into a 512-dimensional feature vector. Each of the two inference networks consists of three fully connected layers with 512 input and output features, with skip connections and swish-1 non-linearity (Ramachandran et al., 2017) applied before addition in the first two dense layers. The cosine classifier is scaled by setting α to 25 when data augmentation is not used, and 50 otherwise. The hyperparameters were chosen through cross-validation. The task embedding network (TEN) used for task conditioning is the same as in Oreshkin et al. (2018). The main and the auxiliary tasks are trained concurrently: in episode t out of T, the auxiliary task is sampled with probability $\rho = 0.9^{\lfloor 12t/T \rfloor}$. For comparison with TADAM (Oreshkin et al., 2018) we use the same optimisation procedure, number of SGD updates, and weight decay parameters for common parts of the architecture as in the paper. For experiments with data augmentation on miniImageNet we use 40k SGD updates with momentum 0.9, and early stopping based on meta-validation performance. We set the initial learning rate to 0.1, and decrease it by a factor ten after 20k, 25k and 30k updates. On FC100 and CIFAR-FS, we use 30k SGD updates with the same momentum and initial learning rate, and the latter is decreased after 15k, 20k and 25k updates. We clip gradients at 0.1, and set separate weight decay rates for the feature extractor, TEN, fully connected layer in the auxiliary task, and inference networks. For the feature extractor and TEN the weight decay is 0.0005. For the fully connected layer in the auxiliary task the weight decay is 0.00001 on miniImageNet, and 0.0005 on FC100 and CIFAR-FS. In the 1-shot setup, the inference networks are regularised with the weight decay equal to 0.0005, regardless of the dataset. In the 5-shot setup, the weight decay parameter in the inference networks is 0.00001 on miniImageNet, and 0.00005 on FC100 and CIFAR-FS. For the 5-shot setup, mini-batches consist of two episodes, each with 32 query images. For the 1-shot setup, we sample 5 episodes per mini-batch, and 12 query images per episode. In both cases query images are sampled uniformly across classes, without any restriction on the number per class. The auxiliary 64-way classification task is trained with the batch size 64.

Choice of β. We empirically find that the regularisation coefficient $\beta = \frac{K}{Nd}$ produces good results, and it can be used as a starting point for further parameter tuning. Here d is the dimensionality of the feature vector f_θ, N is the number of classes in the task, and K is the total number of query samples in the task. On CONV-5, we set β to 0.0586 for the 5-shot setup, and we multiply it by two for the 1-shot setup. On ResNet-12, we set β to 0.0125 for both setups, and we use a value of β twice as large for the 1-shot setup without auxiliary co-training.

Table 1 Both approaches train the same meta-learning model.

	5-SHOT	1-SHOT
VERSA (OUR IMPLEM.)	68.0 ± 0.2	52.5 ± 0.3
SAMOVAR-BASE	69.8 ± 0.2	52.4 ± 0.3
SAMOVAR-BASE (SEPARATE)	66.6 ± 0.2	50.8 ± 0.3

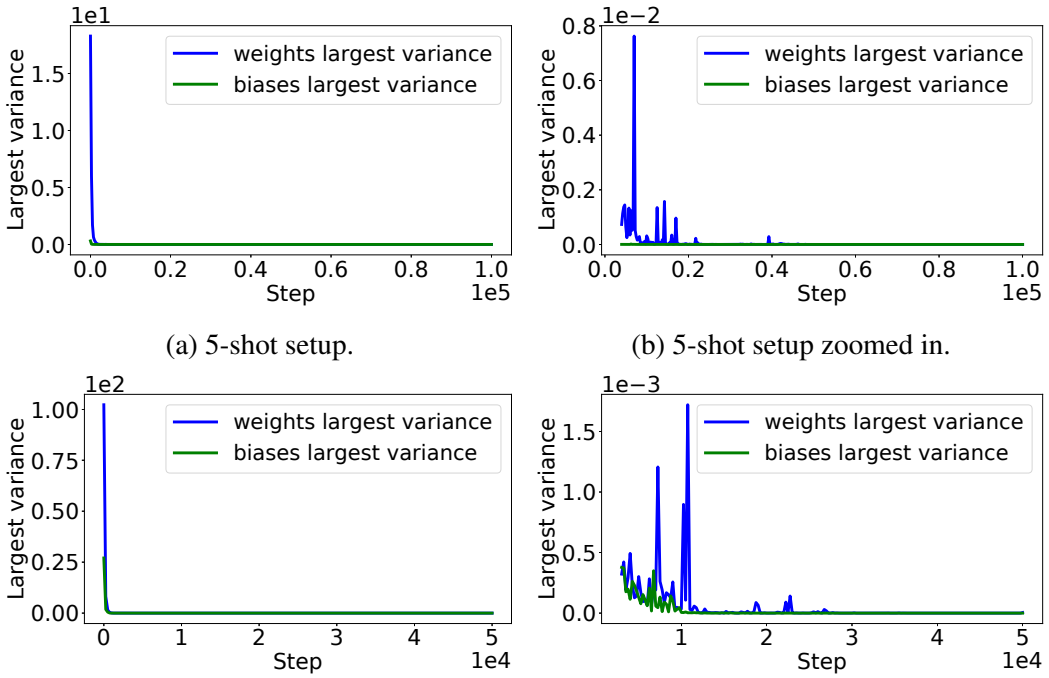
Evaluation. We evaluate classification accuracy by randomly sampling 5,000 episodes, and 15 queries per class in each test episode. We also report 95% confidence intervals computed over these 5,000 tasks. To make a prediction, we draw $d = 1,000$ samples of classifier weights for each class n from the corresponding prior, and average the resulting probabilities for the final classification.

3.4.4 Few-shot image classification results

Comparison with VERSA. In our first experiment, we compare SAMOVAR-base with VERSA (Gordon et al., 2019). Both use the same model, but differ in their training procedure. We use the code provided by Gordon et al. (2019) to implement the two approaches, making one important change: we avoid compression artefacts by storing image crops in PNG rather than JPG format, which improves results noticeably. We evaluate both models in the 1-shot ans 5-shot classification setups.

In Table 1, we report the accuracy on miniImageNet for these two models. In the 1-shot setup, both the approaches lead to similar results, while SAMOVAR yields considerably better performance in the 5-shot setup. To evaluate the effect of sharing the inference network between the prior and posterior, we run SAMOVAR-base with separate neural networks for the prior and approximate posterior, and with the reduced number of hidden units to even out the total number of parameters. From the results in the last two lines of Table 1, it can be seen that for both 1-shot and 5-shot classification sharing the inference network has a positive impact on the performance.

While training VERSA, which estimates the loss using Monte Carlo method, every 250 optimisation steps we keep track of the largest variance of the weights and biases of the predicted classifier in the batch. Figure 3.4 shows how this variance decreases with time, resulting in variance collapse. For example, the largest variance of the weights first falls below 0.001 at the step 4000 in the 5-shot setup, and at the step 3000 in the 1-shot setup. We do not observe this collapse in SAMOVAR. This is consistent with the results obtained on synthetic data.



(a) 5-shot setup.

(b) 5-shot setup zoomed in.

(c) 1-shot setup.

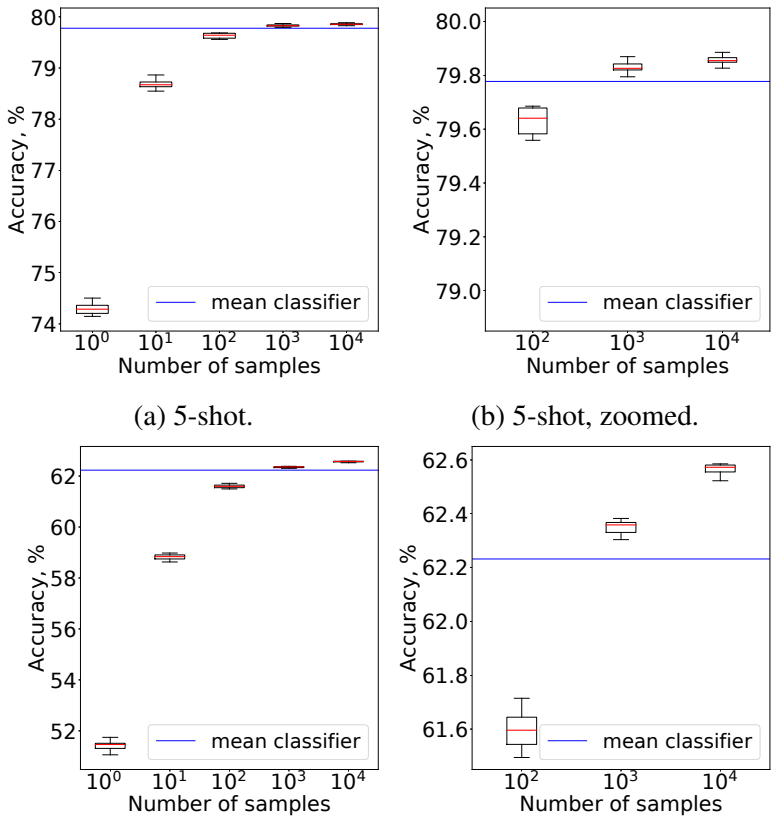
(d) 1-shot setup zoomed in.

Fig. 3.4 Largest variance in VERSA as a function of the optimisation step. Results for optimisation steps from Figure 3.4a and Figure 3.4c that follow the first encounter of variance below 0.001 are zoomed in Figure 3.4b Figure 3.4d respectively.

Table 2 Accuracy and 95% confidence intervals of TADAM and SAMOVAR on the 5-way classification task on miniImageNet. The first columns indicate the use of: cosine scaling (α), auxiliary co-training (AT), and task embedding network (TEN).

			5-SHOT		1-SHOT	
α	AT	TEN	TADAM	SAMOVAR	TADAM	SAMOVAR
			73.5 ± 0.2	75.3 ± 0.2	58.2 ± 0.3	59.3 ± 0.3
✓			74.9 ± 0.2	76.9 ± 0.2	57.4 ± 0.3	58.2 ± 0.3
	✓		74.6 ± 0.2	76.4 ± 0.2	58.7 ± 0.3	59.8 ± 0.3
		✓	72.9 ± 0.2	74.9 ± 0.2	58.2 ± 0.3	58.8 ± 0.3
✓	✓		75.7 ± 0.2	77.2 ± 0.2	57.3 ± 0.3	$\underline{60.4 \pm 0.3}$
	✓	✓	74.1 ± 0.2	$\underline{77.3 \pm 0.2}$	57.5 ± 0.3	59.5 ± 0.3
✓		✓	74.9 ± 0.2	76.8 ± 0.2	57.3 ± 0.3	58.5 ± 0.3
✓	✓	✓	75.9 ± 0.2	$\mathbf{77.5 \pm 0.2}$	57.6 ± 0.3	$\mathbf{60.7 \pm 0.3}$

Comparison with TADAM. In our second experiment, we use SAMOVAR in combination with the architecture of TADAM (Oreshkin et al., 2018). To fit our framework, we



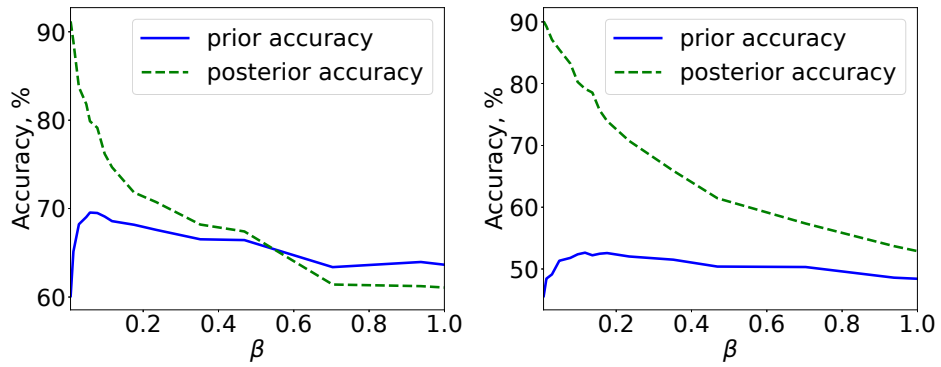
(a) 5-shot. (b) 5-shot, zoomed.

(c) 1-shot. (d) 1-shot, zoomed.

Fig. 3.5 Accuracy on miniImageNet as a function of the number of samples drawn from the learned prior over the classifier weights, compared to using the mean of the distribution.

replace the prototype classifier of TADAM with the linear classifier with latent weights. We compare TADAM and SAMOVAR with metric scaling (α), auxiliary co-training (AT) and the task embedding network (TEN) included or not. When the metric is not scaled, we use SAMOVAR-base with the linear classifier, otherwise we use SAMOVAR-SC with the scaled cosine classifier. For this ablative study we fix the random seed to generate the same series of meta-train, meta-validation and meta-test tasks for both models, and for all configurations. Results in Table 2 show that SAMOVAR provides a consistent improvement over TADAM across all tested ablations of TADAM architecture. Given that the two models differ in whether they use probabilistic inference, this comparison supports the assumption that it might be beneficial to incorporate uncertainty modeling.

Effect of sampling classifier weights. To assess the effect of the stochasticity of the model, we evaluate the predictions obtained with the mean of the distribution over the classifier weights, and with approximation of the predictive distribution Eq. (3.7) using varying number of sampled classifier weights. For both the 5-shot and 1-shot setups, we fix



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

Fig. 3.6 Mean accuracy of SAMOVAR-base classifiers sampled from the prior and posterior as a function of β. While training, we fix the random seed of the data to generate the same series of miniImageNet tasks. The evaluation is performed over 5000 random tasks.

the random seed and evaluate SAMOVAR-SC-AT-TEN on the same 1,000 random 5-way tasks. We compute accuracy 10 times for each number of samples.

Results of these experiments for 5-shot and 1-shot tasks are shown in Figure 3.5. It can be seen that for both setups the mean classification accuracy is positively correlated with the number of samples. This is expected as a larger sample size corresponds to a better estimation of the predictive posterior distribution. The dispersion of accuracy for a fixed n is slightly bigger for the 1-shot setup compared to the 5-shot setup, and in both cases it decreases as we use more samples. This difference is also expected, as the 1-shot task is much harder than the 5-shot task, so the model retains more uncertainty in the inference in the former case. The results also show that the predicted classifier mean demonstrates good results on both classification tasks, and it can be used instead of classifier samples in cases where computational budget is critical. At the same time we can see that sampling of a large number of classifiers leads to a better performance compared to the classifier mean. While on the 5-shot setup the gain from classifier sampling over using the mean is small, around 0.1% with 10K samples, on the 1-shot setup the model benefits more from the stochasticity yielding additional 0.4% accuracy with 10K samples.

Impact of β-scaling. Typically, in autoencoders the dimensionality of the latent space is smaller than that of the observed variables. This is not the case in the meta-learning classification task where the output is merely a one-hot-encoded label of the class, while the latent space is of the same size as the output of the feature extractor. In our experiments we observe that the large KL term suppresses the reconstruction term resulting in a weaker performance. In particular, there is a trade off between these parts of the objective function

$\hat{\mathcal{L}}(\Theta)$ in Eq. (3.6) which can be regulated by β-scaling of the KL term. Figure 3.6 shows the accuracy of SAMOVAR-base with CONV-5 feature extractor as a function of β. Even though in both setups there is a clear maximum, overall, the model is relatively robust to the setting of β. Let's denote the optimum β as β_{opt}. Then for the 5-shot setup the range at least from $0.83\beta_{\text{opt}}$ to $2\beta_{\text{opt}}$ produces results that are within the 1% interval from the maximum accuracy at β_{opt}. For the 1-shot setup, the same holds true for the range at least from $0.66\beta_{\text{opt}}$ to $2\beta_{\text{opt}}$. Interestingly, larger β significantly reduces the performance gap between the prior and the posterior, but results in poorer performance in both.

Comparison to the state of the art. In Table 3.3, we compare SAMOVAR to the state of the art on miniImageNet. For a fair comparison, we report results with and without data augmentation. SAMOVAR yields competitive results, notably outperforming other approaches using ResNet-12 features. The only approaches reporting better results explore techniques that are complementary to ours. Self-supervised co-training was used by Gidaris et al. (2019), which can be used as an alternative to the auxiliary 64-class classification task we used. CTM (Li et al., 2019) is a recent transductive extension of the distance-based models which identifies task-relevant features using inter- and intra-class relations. This module can also be used in conjunction with SAMOVAR, in particular, as an input to the inference network instead of the prototypes. Finally, knowledge distillation on the ensemble of 20 metric-based classifiers was used by Dvornik et al. (2019), which can be used as an alternative feature extractor in our work.

In Table 3.4, we compare to the state of the art on the FC100 dataset. We train our model using data augmentation. SAMOVAR yields the best results on the 5-shot classification task. Transductive fine-tuning (Dhillon et al., 2020) reports a higher accuracy for the 1-shot setting, but is not directly comparable due to the transductive nature of their approach. MTL HT (Sun et al., 2019) reports the best results (with large 95% confidence intervals due to the small amount of data used in their evaluation) in the 1-shot setting. It samples hard tasks after each meta-batch update by taking its m hardest classes, and makes additional updates of the optimiser on these tasks. This is complementary, and can be used in combination with our approach to further improve the results.

In Table 3.5, we compare our model to the state of the art on CIFAR-FS. Data augmentation is used during training. Similar to the aforementioned datasets, SAMOVAR yields competitive results on both tasks. On the 5-shot task, higher accuracy is reported by Dhillon et al. (2020) and Gidaris et al. (2019), while transductive SIB (Hu et al., 2020) is comparable to SAMOVAR. On the 1-shot task, SIB (Hu et al., 2020), transductive version by Dhillon et al. (2020) and Gidaris et al. (2019) report better results. Overall, the observations are consistent with those on miniImageNet.

Table 3.3 Accuracy and 95% confidence intervals of the state-of-the-art models on the 5-way task on miniImageNet. Versions of the models that use additional data during training are not included. Exception is made only if this is the sole result provided by the authors. $*$: Results obtained with data augmentation. †: Transductive methods. $\circ$: Validation set is included into training. $\triangle$: Based on a 1.25$\times$wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
MATCHING NETS(VINYALS ET AL., 2016)	CONV-4	60.0	46.6	
META LSTM(RAVI & LAROCHELLE, 2017)	CONV-4	60.6 ± 0.7	43.4 ± 0.8	600 EP. / 5×15
MAML (FINN ET AL., 2017)	CONV-4	63.1 ± 0.9	48.7 ± 1.8	600 EP. / 5 × SHOT
RELATIONNET (SUNG ET AL., 2018)	CONV-4	65.3 ± 0.7	50.4 ± 0.8	600 EP. / 5 × 15
PROTOTYPICAL NETS (SNELL ET AL., 2017)	CONV-4	65.8 ± 0.7	46.6 ± 0.8	600 EP. / 5 × 15
VERSA (GORDON ET AL., 2019)	CONV-5	67.4 ± 0.9	53.4 ± 1.8	600 EP. / 5 × SHOT
TPN (LIU ET AL., 2019B)	CONV-4†	69.9	55.5	2000 EP. / 5 × 15
SIB(HU ET AL., 2020)	CONV-4†	70.7 ± 0.4	58.0 ± 0.6	2000 EP. / 5 × 15
GIDARIS ET AL. (2019)	CONV-4	71.9 ± 0.3	54.8 ± 0.4	2000 EP. / 5 × 15
SAMOVAR-BASE (OURS)	CONV-5	69.8 ± 0.2	52.4 ± 0.3	5000 EP. / 5 × 15
QIAO ET AL. (2018)	WRN-28-10	73.7 ± 0.2	59.6 ± 0.4	1000 EP. / 5 × 15
MTL HT (SUN ET AL., 2019)	RESNET-12	75.5 ± 0.8	61.2 ± 1.8	600 EP. / 5 × SHOT
TADAM (ORESHKIN ET AL., 2018)	RESNET-12	76.7 ± 0.3	58.5 ± 0.3	5000 EP. / 100
LEO (RUSU ET AL., 2019)	WRN-28-10$^{*\circ}$	77.6 ± 0.1	61.8 ± 0.1	10000 EP. / 5 × 15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*	78.2 ± 0.5	57.7 ± 0.6	1000 EP. / 5 × 15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*†	78.4 ± 0.5	65.7 ± 0.7	1000 EP. / 5 × 15
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12$^{*\triangle}$	78.6 ± 0.5	62.6 ± 0.6	2000 EP. / 5 × 15
SIB (HU ET AL., 2020)	WRN-28-10*†	79.2 ± 0.4	70.0 ± 0.6	2000 EP. / 5 × 15
GIDARIS ET AL. (2019)	WRN-28-10*	79.9 ± 0.3	62.9 ± 0.5	2000 EP. / 5 × 15
CTM (LI ET AL., 2019)	RESNET-18*†	80.5 ± 0.1	64.1 ± 0.8	600 EP. / 5 × 15
DVORNIK ET AL. (2019)	WRN-28-10*	80.6 ± 0.4	63.1 ± 0.6	1000 EP. / 5 × 15
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12	77.5 ± 0.2	60.7 ± 0.3	5000 EP. / 100
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12*	79.5 ± 0.2	63.3 ± 0.3	5000 EP. / 5 × 15

Table 3.4 Accuracy and 95% confidence intervals of the state-of-the-art models on the 5-way task on FC100. Versions of the models that use additional data during training are not included. $*$: Results obtained with data augmentation. $\square$: Results from Lee et al. (2019). †: Transductive methods. $\triangle$: Based on a 1.25$\times$wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
PROTOTYPICAL NETS (SNELL ET AL., 2017)	RESNET-12$^{*\square\triangle}$	52.5 ± 0.6	37.5 ± 0.6	2000 EP. / 5 × 15
TADAM (ORESHKIN ET AL., 2018)	RESNET-12	56.1 ± 0.4	40.1 ± 0.4	5000 EP. / 100
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12$^{*\triangle}$	55.5 ± 0.6	41.1 ± 0.6	2000 EP. / 5 × 15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*	57.2 ± 0.6	38.3 ± 0.5	1000 EP. / 5 × 15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*†	57.6 ± 0.6	43.2 ± 0.6	1000 EP. / 5 × 15
MTL HT (SUN ET AL., 2019)	RESNET-12*	57.6 ± 0.9	45.1 ± 1.8	600 EP. / 5 × SHOT
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12*	57.9 ± 0.3	42.1 ± 0.3	5000 EP. / 5 × 15

Table 3.5 Accuracy and 95% confidence intervals of the state-of-the-art models on the 5-way task on CIFAR-FS. Versions of the models that use additional data during training are not included. All models use data augmentation. □: Results from Lee et al. (2019). †: Transductive methods. △: Based on a 1.25×wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
PROTOTYPICAL NETS (SNELL ET AL., 2017)	RESNET-12$^{\square\triangle}$	83.5 ± 0.5	72.2 ± 0.7	2000 EP. / 5×15
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12$^{\triangle}$	84.2 ± 0.5	72.0 ± 0.7	2000 EP. / 5×15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10	86.1 ± 0.5	68.7 ± 0.7	1000 EP. / 5×15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10†	85.8 ± 0.6	76.6 ± 0.7	1000 EP. / 5×15
SIB (HU ET AL., 2020)	WRN-28-10†	85.3 ± 0.4	80.0 ± 0.6	2000 EP. / 5×15
GIDARIS ET AL. (2019)	WRN-28-10	86.1 ± 0.2	73.6 ± 0.3	2000 EP. / 5×15
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12	85.3 ± 0.2	72.5 ± 0.3	5000 EP. / 5×15

3.5 Conclusion

We proposed SAMOVAR, a meta-learning model for few-shot image classification that treats classifier weight vectors as latent variables, and uses a shared amortised variational inference network for the prior and variational posterior. Through experiments on synthetic data and few-shot image classification, we show that our variational approach avoids the severe under-estimation of the variance in the classifier weights observed for training with direct Monte Carlo approximation (Gordon et al., 2019). We integrate SAMOVAR with the deterministic TADAM architecture (Oreshkin et al., 2018), and find that our stochastic formulation leads to significantly improved performance, competitive with the state of the art on the miniImageNet, CIFAR-FS and FC100 datasets.

Chapter 4

Modulation Adapters for Multi-Domain Learning

Deep convolutional networks are ubiquitous in computer vision, due to their excellent performance across different tasks for various domains. Models are, however, often trained in isolation for each task, failing to exploit relatedness between tasks and domains to learn more compact models that would generalise better in low-data regimes. Multi-domain learning aims to handle related tasks, such as image classification across multiple domains, simultaneously. Previous work on this problem explored the use of a pre-trained and fixed domain-agnostic base network, in combination with smaller learnable domain-specific adaptation modules. In this chapter, we introduce Modulation Adapters, which update convolutional filter weights of the model in a multiplicative manner for each task. Parameterising these adaptation weights in a factored manner allows us to scale the number of per-task parameters in a flexible manner, and to strike different parameter-accuracy trade-offs. We evaluate our approach on the Visual Decathlon challenge, composed of ten image classification tasks across different domains, and on the ImageNet-to-Sketch benchmark, which consists of six image classification tasks. Our approach yields excellent results, with accuracies that are comparable to or better than those of existing state-of-the-art approaches.

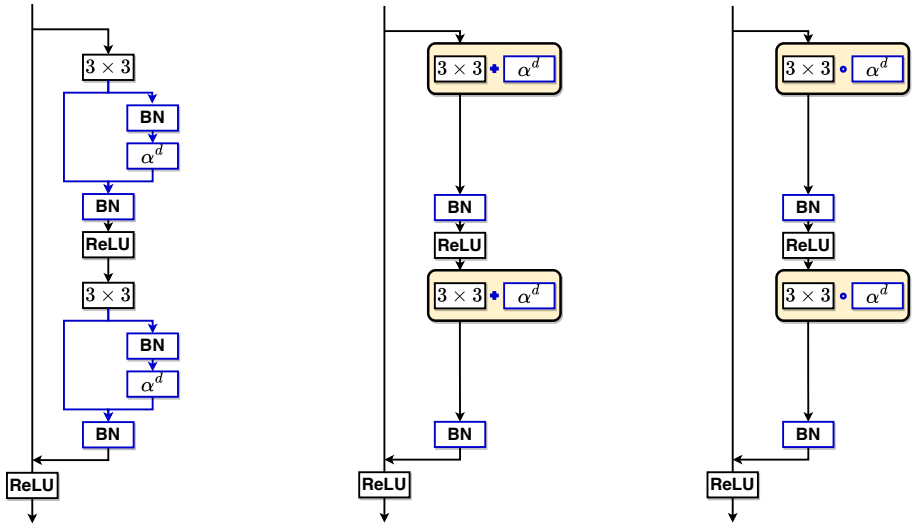
4.1 Introduction

Deep learning models are ubiquitous across many tasks and application domains (LeCun et al., 2015). Yet, one of their biggest limitations is the reliance on the large amounts of labelled data to train these models, which often have millions if not billions of parameters. Knowledge transfer is one of the main solutions to this problem, where knowledge obtained

by learning to solve one task is leveraged to learn another task in a more sample-efficient manner. A ubiquitous example of knowledge transfer is pre-training of a deep neural network on a large (un)labelled dataset, e.g., on ImageNet (Russakovsky et al., 2015), and then fine-tuning it on a smaller target dataset.

Beyond pre-training and fine-tuning, there is a wide range of learning paradigms that involve some form of knowledge transfer. In *multi-domain learning* (MDL) (Rebuffi et al., 2017a), which is the focus of this chapter, a single model is trained to solve several related tasks on a number of different domains, e.g., digit recognition and object recognition. The idea is to use knowledge from related tasks to induce regularisation across the considered domains. The common view on MDL is to transfer knowledge across domains to maximise predictive accuracy, while sharing as many parameters as possible across tasks, and adding as few task-specific parameters as possible. A common approach is to learn a base network on one of the domains, and then adapt it with a small number of parameters for tasks on other domains. A wide range of solutions to such base network adaptation has been proposed in the literature, which can be divided into two major groups of approaches. The first group applies a task-specific binary mask to either the features or the weights in each layer of the model (Berriel et al., 2019; Mallya et al., 2018; Mancini et al., 2018). While this adds a small number of parameters per task, essentially one bit per weight or feature channel, its ability to adapt to new tasks is limited for the same reason. The second group leaves the weights of the base network unchanged, but adds a number of task-specific adaptation layers, for example, a 1×1 convolution parallel to each 3×3 convolutational layer of the base network (Li & Vasconcelos, 2019; Rebuffi et al., 2017a; Rosenfeld & Tsotsos, 2018). Depending on the design of adaptation modules, this approach adds more parameters to adapt the model to the task. However, by leaving the parameters of the base network layers unchanged, task adaptation can only be achieved by added model capacity in the adaptation layers themselves, without capitalising on adaptations of all of the backbone parameters.

To overcome the limitations of existing approaches, we propose a novel type of adaptation module, which we call *Modulation Adapters*. Our approach adapts the weights of the existing layers of the base network to the task by means of non-binary scaling. More specifically, to adapt the base network to a new domain d, we learn a set of domain-specific adapters $\boldsymbol{\alpha}^d$, each modulating the fixed convolutional filters by multiplying them with a scalar α^d_{mn} that is specific to the pair (m, n) of the output and input channels to which the filter applies. See Figure 4.1c for an illustration. The latter allows for nine-fold parameter reduction compared to 3×3 convolutions that constitute the core of common base networks, such as ResNets (He et al., 2016b). To further reduce the memory footprint, we propose a factorised representation of each adapter $\boldsymbol{\alpha}^d$ as a product of two matrices with a smaller intermediate dimensionality.



(a) Residual Adapters (b) Parallel Adapters (c) Modulation Adapters

Fig. 4.1 Different adaptation units embedded into a residual block. Blocks in blue, adapters $\boldsymbol{\alpha}^d$ and batch-norm (BN) layers, contain domain-specific parameters. Blocks in black in the base network are domain-agnostic. Yellow blocks show that the 3×3 convolutions of the base network are modulated before being applied. Note that the BN blocks contribute a negligible number of parameters compared to the Modulation Adapter blocks.

In contrast to other approaches, our adapters allow to modulate all weights in the base network in a non-binary manner, while offering a scalable number of parameters.

We perform several ablative studies to assess the impact of multiplicative adaptation compared to additive, and demonstrate the benefits of the proposed approach. We evaluate our Modulation Adapters on the Visual Decathlon Challenge and the ImageNet-to-Sketch benchmark. We obtain excellent results, with accuracies that are comparable or better than those of existing state-of-the-art approaches across a wide range of parameter budgets.

4.2 Related work

Weight and feature masking. One way to approach multi-domain learning is to adapt the convolutional layers of a base network that has been pre-trained on a large dataset. The "piggy-back" approach of Mallya et al. (2018) learns element-wise binary masks for the weights in the convolutional layers of the base network. Such parameterisation allows memory-efficient storage of domain-specific parameters: requiring to store 1 bit for each real-valued weight, i.e., a 32 fold reduction in storage for single-precision parameters. The idea of adaptation through binary masks is further generalised in WTPB (Mancini et al., 2018), where the authors use three scalars to define an affine transformation of convolutional

filter weights based on the binary masks. In BA^2 (Berriel et al., 2019), binary masks are applied across the feature channels rather than the network weights. As a result, after training masked feature maps can be removed from computational graph, reducing both the memory footprint and the computational complexity. While reduction in memory budget is an appealing property of the masking approach, the common downside is limited capability of model adaptation due to simplicity of the binary masks. In our work, we propose a method for non-binary multiplicative adaptation of the base network, which allows to span a large range of parameter budgets.

Adaptation modules. Domain-specific trainable modules which adapt a pre-trained base network have been explored in several works. Residual Adapters (Rebuffi et al., 2017a) use an intermediate residual block with a trainable 1×1 convolution after each pre-trained and fixed 3×3 convolution. Compared to the size of the base network, such type of adapters leads to nine-fold reduction in the number of additional parameters being learned for a new domain. In Parallel Adapters (Rebuffi et al., 2018), a domain-specific 1×1 convolution is used as an additive bypass to each domain-agnostic 3×3 convolution. The Parallel Adapter configuration provides more flexibility than the sequential one: it can be used on top of existing pre-trained networks since the adapters need not be present when pre-training. DAN (Rosenfeld & Tsotsos, 2018) learns linear transformations of the output filters in the base convolutions, which is equivalent to insertion of an intermediate 1×1 convolution after each base convolution. Unlike Residual Adapters, this model does not use additional skip connections and batch normalisation layers during adaptation. CovNorm (Li & Vasconcelos, 2019) learns an approximation of the sequential adaptation layers, which consists of whitening, mini-adaptation and colouring operations. For each trained adapter, the whitening and colouring matrices are obtained from PCA of the corresponding input and output activations, while the mini-adaptation layer is learned from scratch. Our method also falls into the category of models which contain the adaptation modules. Different from the other approaches, we perform modulation of the fixed convolutional filters by scaling their weights with a non-binary factor specific to each input-output channel combination.

Adapter modules have been investigated in the context of NLP (Houlsby et al., 2019) to specialize Transformer models (Vaswani et al., 2017) to specific tasks. In the context of generative image modeling, adapters have been used to adapt pre-trained GANs to new domains with few samples (Atsuhiro & Tatsuya, 2019; Esther et al., 2020). Our model is related to weight modulation approaches for GANs used in StyleGAN2 (Karras et al., 2020) and in CISP (Anokhin et al., 2021).

4.3 Method

We consider the multi-domain learning problem where a model is required to solve D classification tasks on D data domains, while minimising the total number of parameters being learned. We begin with the popular framework, e.g., Mallya et al. (2018); Mancini et al. (2018); Rebuffi et al. (2017a), of pre-training a base feature extractor on a large dataset, which is then fixed and shared across all domains. After that we learn a separate adapter for each domain, which modifies the fixed feature extractor to improve its performance on the given domain. Along with the adapters, we also learn domain-specific batch normalisation layers and a classification head which predicts domain-specific labels. The objective function is a sum of the domain-specific objective functions that correspond to individual tasks, and in our case it is the sum of corresponding cross-entropy losses.

We detail our approach and its properties in Section 4.3.1, and discuss the factored parametrisation in Section 4.3.2. Then, in Section 4.3.3, we provide a detailed comparison of our approach with respect to the most related prior works.

4.3.1 Modulation Adapters

We suppose the CNN base network has already been pre-trained. Let $\mathbf{f} \in \mathbb{R}^{M\times N\times K\times K}$ be the filter of a specific layer in the base network, where K is the spatial filter size, N is the number of input features, and M the number of output features.

To adapt the convolutional layers of the base network to a new domain $d \in \{1, \ldots, D\}$, we define a modulation adaptation unit ρ, with domain specific parameters $\boldsymbol{\alpha}^d \in \mathbb{R}^{M\times N}$ that modulate the convolutional filter in a multiplicative manner. For an input $\mathbf{x}$ with N feature maps, this unit produces an output $\mathbf{y}$ with M feature maps:

$$\mathbf{y} = \rho(\mathbf{f}, \mathbf{x}, \boldsymbol{\alpha}^d) = \left(\mathbf{f} \circ \boldsymbol{\alpha}^d\right) * \mathbf{x}, \tag{4.1}$$

where $*$ is the convolutional operator, and $\circ$ is an element-wise product with the adapter parameters $\boldsymbol{\alpha}^d \in \mathbb{R}^{M\times N}$ broadcasted across the spatial dimensions of the convolutional weight tensor $\mathbf{f} \in \mathbb{R}^{M\times N\times K\times K}$. Adaptation module ρ applied to the pre-trained filter $\mathbf{f}$ is equivalent to a new convolutional filter $\mathbf{g}^d$, with the weights obtained as:

$$[\mathbf{g}]^d_{mnkl} = \alpha^d_{mn} \cdot [\mathbf{f}]_{mnkl}. \tag{4.2}$$

Note that modulation of the filter weights for each domain can also be interpreted as applying scaling of the input features, which is specific for each output channel. Therefore, with modulation each output channel can mix the input channels in different ways, e.g., amplify

some and ignore others. The input channels are, however, processed by the same spatial filter for each input-output channel combination, leading to reduction in the number of trainable parameters relative to the size of the base network.

We learn a separate set of modulation weights for each convolutional layer in the base network, and denote our adaptation unit as a *Modulation Adapter* (MAD). Figure 4.1c shows how Modulation Adapter units are incorporated into a single residual block. While all convolutional weights are altered for a new domain, we only need to store the pre-trained filters $\mathbf{f}$ as well as the Modulation Adapters, which are learned from scratch. The latter contain MN parameters for each convolution in the base model. This is a reduction in the number of parameters by a factor K^2 (the spatial filter size), relative to size of the corresponding convolution in the base network (e.g., for 3×3 convolutions $K^2 = 3 \times 3 = 9$).

4.3.2 Factorisation of Modulation Adapters

The number of parameters in our Modulation Adapters is given by the product of the number of input channels and the number of output channels of a convolutional layer. For residual blocks the number of input and output channels is the same, and thus the number of adaptation parameters is quadratic in the number of channels. Therefore, the number of adaptation parameters is typically largest in deeper layers of the base network which tend to contain hundreds of feature channels. For example, the ResNet-26 architecture we use in our experiments has 256 channels in the deeper layers, resulting in 64k parameters in each adaptation module. An alternative, allowing for linear scaling of the number of parameters with the number of feature channels, would be to separately scale the input and the output feature channels of the convolutional layer. This scaling, however, could be absorbed in the domain specific BatchNorm layers, which has been found to be too restrictive for effective adaptation in previous work (Rebuffi et al., 2017a).

Inspired by parameter reduction demonstrated by matrix decomposition in CovNorm (Li & Vasconcelos, 2019), we adopt a similar strategy in our work. However, instead of learning an unconstrained full adapter and approximating it with a product of two (or more) matrices using some form of matrix decomposition, e.g. SVD as in ConvNorm, we propose to directly represent and learn MAD as a product of two matrices, each with a smaller intermediate dimension $I < \min(M, N)$:

$$\boldsymbol{\alpha}^d = \boldsymbol{\beta}^d \times \boldsymbol{\gamma}^d, \tag{4.3}$$

where $\boldsymbol{\beta}^d \in \mathbb{R}^{M\times I}$, $\boldsymbol{\gamma}^d \in \mathbb{R}^{I\times N}$ and $\times$ denotes matrix multiplication. Such representation limits the maximum rank of the MAD to I, and can be interpreted as a rank factorisation. In

practice, we observe that for certain base networks the full Modulation Adapters are, indeed, learned to be sparse, which we show in Section 4.5. This justifies restriction of the rank promoted by the proposed representation, while direct learning of the factors reduces the memory requirements during training.

We learn the factors $\boldsymbol{\beta}^d$ and $\boldsymbol{\gamma}^d$ from scratch, and compute their product prior to scaling the base filter bank $\mathbf{f}$. The domain-specific convolution $\mathbf{g}^d$ is obtained as:

$$[\mathbf{g}]^d_{mnkl} = \left(\sum_{i=1}^{I} \beta^d_{mi} \cdot \gamma^d_{in} \right) \cdot [\mathbf{f}]_{mnkl}. \tag{4.4}$$

In this case MAD can still be interpreted as scaling the input channels in a specific way for each output channel, but the scaling coefficients are no longer completely independent. This adapter factorisation reduces the number of parameters being trained from MN in the full adapter to $I(M + N)$ in the factorised case. The intermediate dimension I is a hyper-parameter that can be used to trade-off the number of parameters with prediction accuracy.

4.3.3 Comparison to related approaches

Some earlier work described in Section 4.2 can also be understood in terms of domain specific filters $\mathbf{g}^d$ obtained by applying the domain specific adapters $\boldsymbol{\alpha}^d$ to the fixed filters $\mathbf{f}$ of the base network in order to modify the latter. This allows us to further clarify the differences and similarities between prior work and our Modulation Adapters. In this discussion, both $\mathbf{g}^d$ and $\mathbf{f}$ have the same shapes as before, while the form of $\boldsymbol{\alpha}^d$ varies across the approaches.

Masking methods apply element-wise scaling of convolutional weights:

$$[\mathbf{g}]^d_{mnkl} = \alpha^d_{mnkl} \cdot [\mathbf{f}]_{mnkl}, \tag{4.5}$$

where the scaling coefficients α^d_{mnkl} are either binary (Mallya et al., 2018), or take two unique non-binary values (Mancini et al., 2018). Rather than masking individual weights, BA^2 (Berriel et al., 2019) masks the entire features:

$$[\mathbf{g}]^d_{mnkl} = \alpha^d_{m} \cdot [\mathbf{f}]_{mnkl}. \tag{4.6}$$

Although these adapters are compact to store, their binary nature and factored scaling of features limit task adaptation. Non-binary weights in our Modulation Adapters provide more degrees of freedom for adaptation, while our factorisation approach allows to control the parameter budget.

Linear feature combination methods (Rosenfeld & Tsotsos, 2018) learn adapters that linearly combine filter outputs:

$$[\mathbf{g}]^d_{mnkl} = \sum_{i=1}^{M} \alpha^d_{mi} \cdot [\mathbf{f}]_{inkl}, \tag{4.7}$$

which is equivalent to inserting a domain specific 1×1 convolution after the (fixed) convolution of the base network. Residual Adapters (Rebuffi et al., 2017a), depicted in Figure 4.1a, add a skip-connection on top of the linear combination of features:

$$[\mathbf{g}]^d_{mnkl} = \sum_{i=1}^{M} (1 + \alpha^d_{mi}) \cdot [\mathbf{f}]_{inkl}. \tag{4.8}$$

Linear combination approaches have a comparable parameter budget as our (non-factored) Modulation Adapters: M^2 for the former, and MN for the latter. These adapters are implemented as additional linear layers applied to the output of the corresponding unaltered convolutions. Our approach can not be viewed as a linear transformation applied to either inputs or outputs, since we perform element-wise multiplication of adapters with parameters of the convolutions.

Parallel Residual Adapters (Rebuffi et al., 2018), depicted in Figure 4.1b, adopt domain specific 1×1 convolutions in parallel to the fixed 3×3 convolutions of the base network. This is equivalent to adapting the central element of the pre-tained filters:

$$[\mathbf{g}]^d_{mnkl} = [\mathbf{f}]_{mnkl} + \begin{cases} \alpha^d_{mn} & \text{if } k = l = (K-1)/2 + 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

While the number of adapter parameters is MN as in our approach, this adapter only affects the central elements of the filters, which is restrictive and limits the space of possible adaptations.

4.4 Experimental Evaluation

We describe our experimental setup in Section 4.4.1. After that, we present ablation experiments in Section 4.4.2, followed by comparison to previous work in Section 4.4.3.

4.4.1 Experimental setup

Datasets. In our experiments we use the two most common multi-domain learning benchmarks: the Visual Decathlon Challenge (Rebuffi et al., 2017a) and ImageNet-to-Sketch (Mallya et al., 2018), which contain image classification datasets from heterogeneous visual domains, and each dataset contains different classes to recognise.

The *Visual Decathlon Challenge* consists of ten image classification datasets: ImageNet (Russakovsky et al., 2015), CIFAR-100 (Krizhevsky, 2009), Aircraft (Maji et al., 2013), Daimler pedestrian classification (Munder & Gavrila, 2006), Describable textures (Cimpoi et al., 2014), German traffic signs (Stallkamp et al., 2012), Omniglot (Lake et al., 2015), Street view house numbers (Netzer et al., 2011), UCF101 Dynamic Images (Soomro et al., 2012; Bilen & Vedaldi, 2016), and VGG-Flowers (Nilsback & Zisserman, 2008).

The second benchmark, *ImageNet-to-Sketch*, consists of six image classification datasets: ImageNet (Russakovsky et al., 2015), VGG-Flowers (Nilsback & Zisserman, 2008), Stanford Cars (Krause et al., 2013), Caltech-UCSD Birds (Welinder et al., 2010), Skteches (Eitz et al., 2012) and WikiArt (Saleh & Elgammal, 2016).

Training details. On the *Visual Decathlon Challenge*, we use ResNet-26 (Rebuffi et al., 2017a; Rosenfeld & Tsotsos, 2018; Li & Vasconcelos, 2019) and the weights of its feature extractor pre-trained on ImageNet from (Rebuffi et al., 2017a). All convolutional layers in the feature extractor consist of 3×3 filters, which are fixed after pre-training and accompanied by domain-specific Modulation Adapters. The latter are initialised with ones and learned for each domain separately. Batch normalisation layers, which follow each of these modulated convolutions, are also finetuned for each domain separately, and are initialised with values from the pre-trained base network. Finally, domain-specific fully connected layers used as classification heads are learned from scratch, individually for each task. The model is trained with SGD with momentum 0.9 for 140 epochs. Initial learning rate is set to 0.1, and decreased by a factor ten after epochs 80 and 110. We set the weight decay to 0.0035 for VGG-Flowers, 0.0025 for Aircraft and Describable textures, 0.0015 for UCF101 Dynamic Images, 0.001 for Daimler pedestrian classification and Omniglot, and 0.0005 for CIFAR-100, German traffic signs and Street View House Numbers.

For the *ImageNet-to-Sketch* benchmark, we use DenseNet-121 (He et al., 2016a) as the base network, which is commonly used in previous work (Mallya et al., 2018; Mancini et al., 2018; Berriel et al., 2019; Guo et al., 2019b). Similarly to the Visual Decathlon Challenge, we apply Modulation Adapters to all convolutions in the base network, including those with 1×1 filters. Despite the absence of parameter reduction in the latter case, we empirically find that modulation of such convolutions leads to better performance compared to finetuning them directly. Domain-specific batch normalisation layers and classification heads are initialised

	Params	ImNet	Airc	C100	DPed	DTD	GTSR	Flwr	Oglt	SVHN	UCF	Mean	Score
Modulation adapter (full)	2	60.8	66.8	79.2	97.9	56.9	99.2	86.4	90.0	97.0	52.5	78.7 ± 0.3	3828 ± 178
Scaling central element	2	60.8	39.6	74.7	97.7	53.1	98.0	76.5	85.8	94.8	44.0	72.5 ± 0.3	2252 ± 120
Additive 3×3 adapt.	2	60.8	30.5	53.6	91.5	25.3	95.1	44.1	88.8	94.5	27.2	61.1 ± 0.2	916 ± 11
Modulation adapter ($I = 36$)	1.39	60.8	65.3	81.7	98.2	59.4	99.2	84.8	89.5	96.8	52.1	78.8 ± 0.1	3798 ± 134
Modulation adapter (best)	1.52	60.8	65.3	81.7	98.3	59.4	99.3	85.1	89.6	96.8	52.5	78.9 ± 0.1	3878 ± 68
Modulation adapter (LOO)	1.38	60.8	64.9	81.7	98.2	59.1	99.2	84.4	89.5	96.8	52.1	78.7 ± 0.3	3802 ± 123

Table 4.1 Ablation experiments. Classification accuracy is reported per dataset, along with the mean, and the decathlon score. For the latter two we also report their standard deviations across the ten repetitions of the experiments.

and trained the same way as for ResNet-26, while the weights of Modulation Adapters are initialised with 0.15. The model is trained with AdamW (Loshchilov & Hutter, 2019) for 60 epochs. The initial learning rate is set to 0.001, and decreased by a factor ten after epochs 20 and 40. We set the weight decay to 0.0035 for Flowers, 0.001 for CUBS, Stanford Cars and Sketch, and 0.0005 for WikiArt.

Evaluation metrics. We report average classification accuracy across all domains, as well as domain-specific accuracies. Each model is trained 10 times with random initialisations, and the average across these runs is reported. We also use the score function introduced in (Rebuffi et al., 2017a). This score S is computed as:

$$S = \sum_{d=1}^{D} a_d \max\{0, E_d^{\max} - E_d\}^{b_d}, \tag{4.10}$$

where D is the number of domains in the benchmark, $a_d = 1000\,(E_d^{\max})^{-b_d}$ is the weight which ensures a perfect score of 1000 on a single domain, $E_d^{\max}$ and E_d are the baseline and the model's test errors, respectively, and $b_d = 2$ is the exponent that rewards more significant reductions of the classification error. The baseline error $E_d^{\max}$ is twice the error of a per-domain fully finetuned model. The score of the finetuning baseline is therefore 250 per dataset. We report the total number of trainable parameters relative to the size of the base feature extractor network, not counting the linear classification heads.

4.4.2 Ablation studies

We conduct ablation studies on the Visual Decathlon Challenge in order to assess several variants of our model, as well as the effect of scaling, selecting the parameter budget, and the complementarity with additive adapters.

Scaling central element vs. entire filter. We experiment with a version of Modulation Adapter in which we only scale the central element of the 3×3 filters, rather than the entire filter. This is similar to additive Parallel Adapters (PA) (Rebuffi et al., 2018), which use

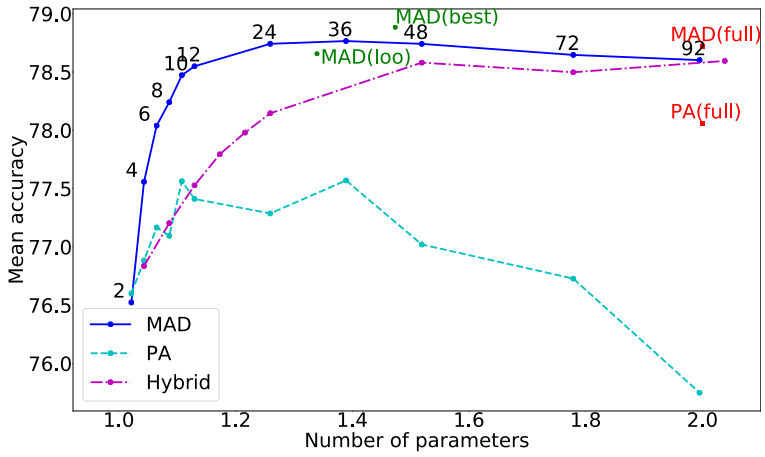


Fig. 4.2 Mean accuracy vs. parameter budget for different adapter-based methods: decomposed version of our Modulation Adapters (MAD), decomposed version of Parallel Adapters (PA, our implementation), hybrid adapters that combine MAD and PA.

1×1 filters to adapt the central elements of the 3×3 filters of the base network. With this central-element-only scaling approach, we obtain a mean accuracy of 72.5% and a decathlon score of 2252, see Table 4.1. This is significantly worse than the performance of our full model (78.7% mean accuracy, and score 3828), showing the benefit of scaling the entire 3×3 filter for each input-output channel combination. Interestingly, scaling only the central element is also worse than the result of Parallel Adapters, which reported a mean accuracy of 78.1% and a score of 3412 (see Table 4.2). Although scaling the central element is equivalent after training, our hypothesis is that optimising the scaling adapter with stochastic gradient descend is more difficult than optimising the additive adapter. Using an optimiser with adaptive learning rates, such as Adam (Kingma & Ba, 2015), may alleviate this.

Additive adaptation of all filter elements. In our second ablation, we consider a variant of our approach in which we adapt all filter elements, as in Eq. (4.2), but in an additive rather than multiplicative manner. This is similar to PA, but now the adapter updates all the filter elements rather than only the central one. This variant leads to a substantially worse mean accuracy of 61.1% and a decathlon score of 916, see line "Additive 3×3 adapt." in Table 4.1. Our interpretation of this deterioration is that strong additive adaptation of all filter weights leads to spatially uniform filters that fail to detect spatial patterns.

Decomposition of Modulation Adapters. In Figure 4.2, we consider the effect of decomposing our Modulation Adapters as a product of two smaller matrices of weights, as in Eq. (4.3). We vary the intermediate dimension I from 2 to 92, and plot the mean accuracy against the parameter budget. We note that the mean accuracy quickly increases when allowing more parameters, reaches a maximum at $I = 36$, and then slightly declines. The

mean accuracy of the full, non-decomposed model, and the model with $I = 36$ are very close at 78.7% and 78.8% respectively, see Table 4.1. The decomposed model with $I = 36$, however, yields a saving of more than 60% in the parameter budget.

While on average there is a smooth trend between the accuracy and the parameter budget, the optimal parameter budget differs per dataset. In Table 4.1 (line "Modulation Adapter (best)"), we show what the optimal accuracy would be if we were to set this budget for each dataset separately. We notice that this only leads to minor improvements of 0.1% and 0.2% on average across domains w.r.t. the $I = 36$ and the full models, which is within the standard deviation of the reported results. We further illustrate the stability of the trade-off between the accuracy and the parameter budget across the datasets with a leave-one-out (LOO) experiment. In this case, for each dataset we select the parameter budget by taking the optimal budget on all the other datasets. We then average the results across all the datasets. As in previous cases, we only find minor deviations in the mean accuracy, reaching 78.7%, which is the same as the performance of the full non-decomposed model.

Decomposition of Parallel Adapters. Similar to the decomposition used in our Modulation Adapters, we implemented a factorised version of Parallel Adapters (PA) (Rebuffi et al., 2018), where each 1×1 adapter is given as a product of two matrices with a smaller intermediate dimension I. As shown in Figure 4.2, unlike our approach, PA does not behave well with decomposition relative to the full version of PA. Moreover, decomposed PA performs consistently worse than our Modulation Adapters across all parameter budgets. For reference, we also included the "full" non-decomposed version of MAD and PA in the graph.[1]

Hybrid adapters. To study the complementarity of multiplicative and additive adaptation, we experiment with a hybrid approach that mixes our Modulation Adapters with Parallel Adapters (Rebuffi et al., 2018). We use decomposed versions of MAD and PA, since our goal is to keep the total number of parameters below the size of the full version of a single model. The available parameter budget is evenly split between the two types of adapters. The results in Figure 4.2 show that these adapters are not complementary: the hybrid approach is consistently worse than our Modulation Adapters, in particular for small parameter budgets, while providing a consistent improvement over the decomposed version of Parallel Adapters.

[1]Result for PA (full) taken from original paper: 78.1% mean accuracy and score 3412. We implemented the decomposed version of PA (as well as our models) based on the public code-base of PA (full) from Rebuffi et al. (2018). In our experiments, we obtained mean accuracy 77.7% and score 3206 for PA (full). If we take best result across these ten runs, we obtain mean accuracy 78.2% and score 3460, consistent with the original paper.

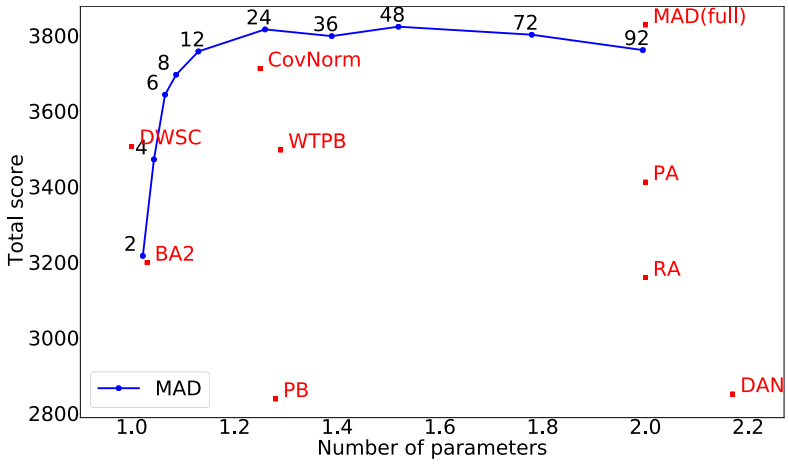


Fig. 4.3 Total score vs. parameter budget for Modulation Adapters (MAD) and the other approaches.

4.4.3 Comparison to the state of the art

Visual Decathlon Challenge. In Figure 4.3, we compare the total score of our full and decomposed Modulation Adapters to the state-of-the-art methods across a range of parameter budgets. While other methods provide a single operating point, our decomposed Modulation Adapters enable adaptation to new domains across a wide range of parameter budgets, and yield better or comparable score. In particular, CovNorm (Li & Vasconcelos, 2019) has a score 3713, and for a similar parameter budget our Modulation Adapters ($I = 24$) obtain an improved score 3816. Parallel Adapters (Rebuffi et al., 2018) report a score 3412, while at the same parameter budget our non-decomposed Modulation Adapters obtain a score 3828. With 3507, DWSC (Guo et al., 2019a) achieves a lower score than the top-performing methods, but uses a smaller parameter budget. Unlike the other methods, however, DWSC does not use the ResNet-26 base network, but an architecture based on depthwise separable convolutions, and is therefore not directly comparable. The number of parameters is still given relative to the ResNet-26 model.

In Table 4.2, we additionally report the accuracy per dataset as well as the decathlon score. Here, we also include three more baselines. The "Feature" baseline does not adapt the base network, and only learns the linear classification head per dataset. This baseline obtains poor mean accuracy (54.3%) and score (544), due to insufficient adaptation. The "BN Adapt." baseline only learns dataset-specific BatchNorm layers, but leaves all the other parameters of the base network unchanged. This adds very few parameters, but leads to a remarkable improvement with respect to the feature baseline, by scaling the output of the convolutional layers in a dataset-specific manner. After training, this approach is equivalent to only scaling

Number of images	Params	ImNet 1.3m	Airc 7k	C100 50k	DPed 30k	DTD 4k	GTSR 40k	Flwr 2k	Oglt 26k	SVHN 70k	UCF 9k	Mean	Score	Rank
Feature (Rebuffi et al., 2017a)	1	59.7	23.3	63.1	80.3	45.4	68.2	73.7	58.8	43.5	26.8	54.3	544	14.7
BN Adapt. (Bilen & Vedaldi, 2017)	∼1	59.9	43.1	78.6	92.1	51.6	95.8	74.1	84.8	94.1	43.5	71.8	1363	13.4
BA2 (Berriel et al., 2019)	1.03	56.9	49.9	78.1	95.5	55.1	99.4	86.1	88.7	96.9	50.2	75.7	3199	8.9
Finetune (Rebuffi et al., 2017a)	10	59.9	60.3	**82.1**	92.8	55.5	97.5	81.4	87.7	96.6	51.2	76.5	2500	9.4
PB (Mallya et al., 2018)	1.28	57.7	65.3	79.9	97.0	57.5	97.3	79.1	87.6	**97.2**	47.5	76.6	2838	9.1
DAN (Rosenfeld & Tsotsos, 2018)	2.17	57.7	64.1	80.1	91.3	56.5	98.5	86.1	89.7	96.8	49.4	77.0	2851	8.3
RA (Rebuffi et al., 2017a)	2	60.3	61.9	81.2	93.9	57.1	99.3	81.7	89.6	96.6	50.1	77.2	3159	7.6
WTPB (full) (Mancini et al., 2018)	1.29	60.8	52.8	82.0	96.2	58.7	99.2	**88.2**	89.2	96.8	48.6	77.2	3497	5.8
DWSC (Guo et al., 2019a)	∼1	**64.0**	61.1	81.2	97.0	55.5	99.3	85.7	89.1	96.2	49.3	77.8	3507	7.4
SpotTune (Guo et al., 2019b)	11	60.3	63.9	80.5	96.5	57.1	**99.5**	85.2	88.8	96.7	52.3	78.1	3612	6.4
PA (Rebuffi et al., 2018)	2	60.3	64.2	81.9	94.7	58.8	99.4	84.7	89.2	96.5	50.9	78.1	3412	6.5
CovNorm (Li & Vasconcelos, 2019)	1.25	60.4	**69.4**	81.3	**98.8**	**59.9**	99.1	83.4	87.7	96.6	48.9	78.6	3713	6.6
MAD (full)	2	60.8	66.8	79.2	97.9	56.9	99.2	86.4	**90.0**	97.0	**52.5**	78.7	**3828**	4.1
MAD ($I = 24$)	1.26	60.8	64.9	81.5	98.2	59.1	99.2	85.1	89.5	96.8	52.3	78.7	3816	4
MAD ($I = 36$)	1.39	60.8	65.3	81.7	98.2	59.4	99.2	84.8	89.5	96.8	52.1	**78.8**	3798	**3.9**

Table 4.2 Comparison to the state-of-the-art methods on the Visual Decathlon Challenge. Best results per metric (mean and per dataset classification accuracy, as well as score) are in bold, and the second best are underlined.

the output channels of the filters. Our approach is similar, but scales filter weights in both input and output dependent manner, leading to more adaptation and far superior results. The "Finetune" baseline finetunes all the parameters of the base network for each dataset, and does not offer any savings in the parameter budget. It is outperformed by other methods that limit the number of free parameters, allowing for better generalisation to datasets with few samples. We also include SpotTune, which improves over the Finetune baseline in prediction accuracy, but does not offer reduction in the number of parameters.

Our Modulation Adapters (MAD) not only obtain high mean accuracy and score, but also offer the best or comparable accuracy per domain. This is unlike the other methods whose performance is often less consistent across datasets. To quantify this, we rank the methods by accuracy (best first) on each domain, and then average the rank of each method across domains. On the Decathlon Challenge, MAD (I=36), MAD (I=24) and MAD (full) have average ranks 3.9, 4.0 and 4.1, respectively. These are the best three results on this benchmark. CovNorm, which is our next competitor in terms of the score and accuracy, has the average rank 6.6, which is the seventh-best result. Furthermore, MAD is significantly easier to train than CovNorm. The latter consists of a complex multi-step algorithm: (i) learning Residual Adapters (Rebuffi et al., 2017), (ii) computing PCA for input and output activations, (iii) learning additional mini-adaptation layers, and (iv) joint finetuning of all components of the final adapters. Training our model is comparable to the first step of CovNorm: we learn a single multiplicative adapter per domain in a single training run.

ImageNet-to-Sketch benchmark. In Table 4.3, we compare our Modulation Adapters with the state of the art on the ImageNet-to-Sketch benchmark. The full (non-decomposed) model outperforms all the other methods both in terms of the average accuracy and the total

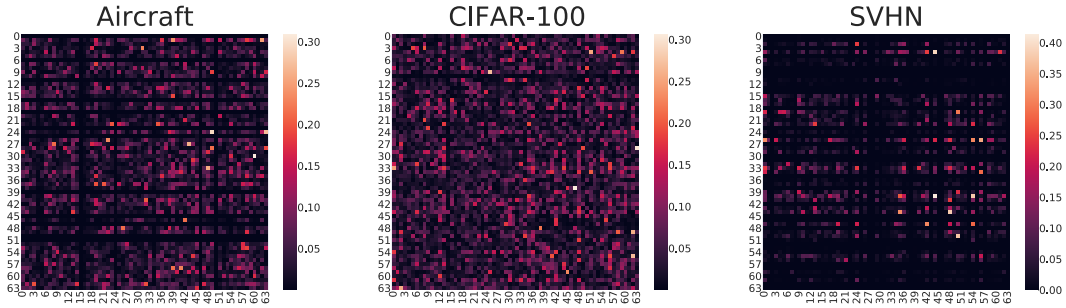
Number of images	Params	ImNet 1.3m	CUBS 6k	Stanford Cars 8k	Flowers 2k	WikiArt 42k	Sketch 16k	Mean	Score	Rank
Feature (Mallya et al., 2018)	1	**74.4**	73.5	56.8	83.4	54.9	53.1	66.0	324	6.8
PB (Mallya et al., 2018)	1.21	**74.4**	81.4	90.1	95.5	73.9	79.1	82.4	1209	5.8
BA2 (Berriel et al., 2019)	1.17	**74.4**	82.4	**92.9**	96.0	71.5	79.9	82.9	1434	4
WTPB (full) (Mancini et al., 2018)	1.21	**74.4**	81.7	91.6	**96.9**	75.7	79.8	83.4	1534	3.5
Finetune (Mallya et al., 2018)	6	**74.4**	81.9	91.4	<u>96.5</u>	<u>76.4</u>	<u>80.5</u>	83.5	1500	3
MAD (full)	4.61	**74.4**	**83.9**	<u>91.9</u>	**96.9**	**76.9**	**81.0**	**84.2**	**1668**	**1.2**
MAD (I=36)	2.33	**74.4**	<u>83.1</u>	90.6	96.4	75.3	80.2	83.3	1446	3.7
MAD (full) + WTPB (full)	1.26	**74.4**	82.7	91.4	**96.9**	76.2	80.1	<u>83.6</u>	<u>1569</u>	<u>2.7</u>

Table 4.3 Comparison to the state-of-the-art methods on the ImageNet-to-Sketch benchmark. Best results per metric (mean and per dataset classification accuracy, as well as score) are in bold, and the second best are underlined.

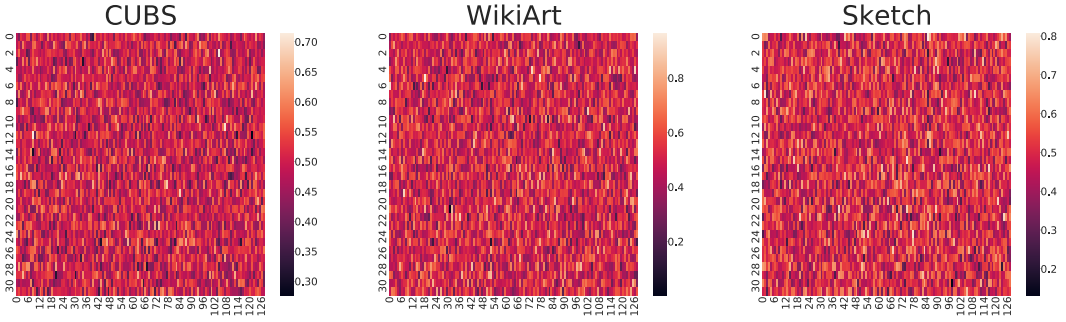
score, and is the only one to outperform finetuning the base network ("Finetune"). This result further confirms the effectiveness of the proposed multiplicative adaptation strategy. While our parameter budget is larger than those of our competitors (due to abundance of 1×1 convolutions in the DenseNet-121 base network), it is still below the budget of the fully finetuned models, which allows us to reduce the performance gap between the latter and learning only domain-specific classification heads ("Feature"). As for individual tasks, Modulation Adapters demonstrate the best performance on four datasets (CUBS, Flowers, WikiArt and Sketch), and the second-best on Stanford Cars. Results on WikiArt and Sketch are of special interest, since these datasets are quite different from ImageNet which was used for pre-training. While the latter consists of natural images, the former two contain paintings and sketches, which makes the multi-domain task on them more challenging. On ImageNet-to-Sketch, MAD (full) obtains the best average rank 1.2, while next competitors "Finetune"and WTPB obtain 3 and 3.5.

We found that factorisation of our Modulation Adapters is less effective on the ImageNet-to-Sketch dataset: using $I = 36$ intermediate dimensions for decomposition (reducing the parameter budget roughly by a factor two), we obtained an average accuracy of 83.3. The different behaviour of decomposition across the two benchmarks may be related to the relatively large number of parameters in 1×1 convolutions in the DenseNet-121 architecture, whereas in the ResNet-26 architecture most parameters reside in 3×3 convolutions. We also found that the learned adapters for the ImageNet-to-Sketch benchmark do not show the sparse structure that we observe in the adapters learned for the Visual Decathlon Challenge, see Section 4.5, making rank restriction less effective. Despite the less consistent performance, MAD (I=36) still outperforms other competitors on Sketch and CUBS, improving over full finetuning of the base network on the latter.

As an alternative strategy to reduce the parameter budget, we explored the possibility of combining our method with other approaches. In particular, we trained a hybrid model



(a) ResNet-26 modulation adapters learned on Visual Decathlon Challenge datasets.



(b) DenseNet-121 modulation adapters learned on ImageNet-to-Sketch benchmark datasets.

Fig. 4.4 Visualisation of the absolute weight values of Modulation Adapters. The vertical axis represents the output channels, while the horizontal one represents the input channels.

where WTPB (Mancini et al., 2018) is applied to 1×1 convolutions, while Modulation Adapters are applied to 3×3 convolutions ("MAD (full) + WTPB (full)"). The hybrid model improves over WTPB on three datasets, bringing it closer to individual finetuning on WikiArt and Sketch, and even outperforming it on CUBS. In terms of the average accuracy, the total score and the average rank, the hybrid model is slightly better than individual finetuning, second only to our Modulation Adapters. This shows that even more flexible models could be created from existing approaches depending on the desired parameter budget and performance requirements.

4.5 Visualisation of Modulation Adapters

Figure 4.4(a) shows the absolute weight values of Modulation Adapters (full) which adapt the same 3×3 convolutional layer at depth five in ResNet-26 on Aircraft, CIFAR-100 and SVHN datasets from the Visual Decathlon Challenge. Figure 4.4(b) does the same for 3×3

convolutional layer at depth nine in DenseNet-121 on CUBS, WikiArt and Sketch datasets from the ImageNet-to-Sketch benchmark.

Non-trivial adapters are learned, i.e., the matrices do not contain rows or columns with uniform values. In addition to that, we can observe some feature selection happening in ResNet-26: dark spots indicate zeroing of corresponding input feature maps. This is not the case for DenseNet-121 used for ImageNet-to-Sketch benchmark. The low sparsity of Modulation Adapters on the DenseNet-121 might also explain why the decomposed version is less effective on this benchmark.

4.6 Conclusion

In this paper, we address the problem of learning across multiple domains while reducing the total parameter budget. In contrast to previous works which rely on adaptation modules that are limited in their capacity, or only adapt a part of the base network parameters, we introduce Modulation Adapters, a novel type of adapters based on flexible weight modulation. We design these adapters to update the weights of the pre-trained convolutional filters by scaling their input and output channels, and provide an efficient parameterisation through decomposition for further parameter reduction. Ablative analyses as well as our model's excellent performance on the popular Visual Decathlon Challenge and ImageNet-to-Sketch benchmarks validate the benefits of our multiplicative adaptation method. At the same time other means of parameter reduction for the full Modulation Adapter could be considered to cover the setups where the learned adapters are not sparse.

Chapter 5

Conclusion

In this thesis we focus on the two problems that involve transfer of knowledge between a number of tasks with limited amount of training data: few-shot learning and multi-domain learning. Our general goal in both settings is to share as many parameters between the tasks as possible, while minimising the overall number of trainable parameters within each of them. We consider different approaches to feature adaptation, as well as different learning frameworks, which results in two models: a probabilistic generator of classifier weights for few-shot learning, and a modulation adaptor of convolutional weight tensors for multi-domain learning. In Section 5.1 we summarise the contributions of this thesis, and in Section 5.2 we discuss the ideas for future work in the considered transfer learning problems.

5.1 Summary of contributions

5.1.1 Shared amortised inference for few-shot learning

In Chapter 3, we propose a novel Bayesian inference scheme, where task- and class-specific classifier weights are modelled as latent variables with learnable prior and posterior. Probabilistic modeling of classifier weights directly addresses the model uncertainty, and provides a means to incorporate variance into the model predictions. Definition of the prior over classifier parameters through conditioning on the train set of the task, in conjunction with definition of the posterior over the same parameters through conditioning on the union of the train and test sets, makes the design easy to interpret: we want the model, which can only observe the extremely limited train set, to mimic the predictions of the model that has access to a larger set of labeled examples, forcing it to extract the knowledge more efficiently. Shared amortised inference makes the parameter budget fixed for arbitrary number of tasks,

and for arbitrary number of classes within each task, providing flexibility and allowing to tackle overfitting which is one of the main challenges in few-shot learning.

Experiments on synthetic data show that, given a small number of samples of the latent variable generated during training, variational inference promotes a more efficient recovery of the true posterior compared to Monte Carlo training, which tends to underestimate the variance. We also show that an ensemble of classifiers sampled from the predicted distribution improves the performance compared to single classifier that only uses the predicted mean and ignores the predicted variance. This result supports the initial argument for taking the model uncertainty into consideration. Evaluation on several common few-shot learning benchmarks shows that in certain setups our model, which uses the labeled train data alone to make predictions on the meta-test set, can outperform some transductive models which additionally use the unlabeled test data in order to make meta-test predictions.

5.1.2 Modulation adapters for multi-domain learning

In our second contribution, covered in Chapter 4, we propose a novel design of the trainable adaptation module for multi-domain learning. For each domain, our adapter performs independent scaling of kernels in convolutional layers of the shared feature extractor. Multiplicative adaptation preserves pre-trained spatial patterns within each of them, while re-weighting their relative importance, differently for each output feature. We factorise our modulation adapters as a product of two matrices with a smaller intermediate dimension, which not only reduces the number of domain-specific parameters, but also spans a large range of parameter budgets, unlike previous approaches which generally offer a single budget setting. Independent learning of adapters mitigates catastrophic forgetting, making the approach more flexible and potentially applicable to incremental learning.

We perform multiple ablation studies to demonstrate the advantage of the proposed design compared to other possible operations with convolutional kernels. Our experiments also show that adapter factorisation can significantly reduce the number of trainable parameters while maintaining accurate predictions. For any parameter budget, our model outperforms the competitors, thus spanning a wide range of competitive solutions with different memory requirements. Both the full and the factorised versions of modulation adapters achieve state-of-the-art results on two common multi-domain learning benchmarks, the Visual Decathlon Challenge (Rebuffi et al., 2017a) and ImageNet-to-Sketch (Mallya et al., 2018).

5.2 Future work and perspectives

5.2.1 Improved feature extraction for few-shot learning

In Chapter 3 we show the benefits of probabilistic modeling in few-shot learning. However, in our model probabilistic inference is limited to prediction of the distribution over weights of the task-specific classifier, whereas feature embedding is performed in a completely deterministic way. At the same time more complex feature extraction in the form of ensemble learning has been successfully considered in the literature (Dvornik et al., 2019), which is remotely related to infinite ensembles learned by statistical models. This brings forth motivation to incorporate probabilistic modeling into feature extraction as well.

Approaches to representation learning for few-shot learning vary a lot. Nevertheless, a considerable scope of works relies on FiLM layers (Perez et al., 2018) in order to adjust a pre-trained feature extractor to the task at hand. Parameters of these layers could be viewed as task-specific latent variables, conditioned on the data associated with the task, and inferred from the task using either a task embedding network (TEN), as in TADAM (Oreshkin et al., 2018), or more complex autoregressive task encoder, as in CNAPS (Requeima et al., 2019). FiLM parameters could be obtained then using the same shared amortised inference as for weights of the classifier in SAMOVAR.

Recently there has been an increased interest in learning universal representation (Bilen & Vedaldi, 2017; Triantafillou et al., 2021) for few-shot learning generalisation problem, where a model is required to solve the standard few-shot learning problem on a collection of datasets rather than just on one of them. This is in line with multi-domain learning, so taking inspiration from its adaptation strategies, another possible way of enhancing feature extraction for few-shot learning could be development of more sophisticated means of feature adaptation compared to FiLM. Although straightforward application of existing adapters is theoretically possible, in practice this might be hindered by insufficient amount of labeled data, with a high risk of overfitting. This risk could be reduced by using models with the smaller parameter budget, including our factorised Modulation Adapters or binary masks.

5.2.2 Domain awareness in multi-domain learning

Independent learning of domain-specific adaptation modules, which is currently the most popular approach to multi-domain learning, provides the model with a lot of flexibility, and makes it easily applicable to other transfer learning setups, such as continual learning. Combined with pre-trained and fixed feature extractor, this strategy naturally deals with catastrophic forgetting, since domain-specific modules are only affected by data from the

domain of interest. The same property, however, constitutes the biggest drawback of this approach: being trained in isolation, these modules lack any positive transfer between the domains, except for transfer from the domain that is used during the pre-training stage. This problem can be addressed by introducing context to multi-domain learning, which would be responsible for consolidation of knowledge from multiple domains on a meta-level.

There are many possible ways to implement such context. One of them could be using HyperNetworks (Ha et al., 2017) as adaptor generators, which is also related to the concept of "fast weights" (Schmidhuber, 1992) where one network predicts transformation of another network. These networks could take the output of the previous convolutional layer as input, and produce weights of the adaptor for the current convolution layer. Joint training of HyperNetworks would result in a context-dependent inference of domain-specific adaptation modules, introducing the desired domain awareness and increasing knowledge transfer while keeping adaptors domain-specific, a concept that has been shown to work in our first contribution, SAMOVAR. The factorised version of our Modulation Adaptors can potentially reduce the burden of predicting high-dimensional objects, which is usually the case with adaptors for the deeper layers of the feature extractor.

Fully joint training, however, might lead to negative transfer between some of the domains. An alternative could be definition and computation of the domain taxonomy, in a similar vein with the task taxonomy (Zamir et al., 2018) which measures transferability between different pairs of tasks. Some form of this intuition has been applied in SpotTune (Guo et al., 2019b) which learns a decision policy for selective, domain-specifc finetuning of certain layers in the pre-trained feature extractor. This method still suffers from limited transferability, in addition to providing no parameter reduction by design. Yet, the idea to develop a policy which defines which domains can be co-trained, and which should be isolated, is an interesting direction to explore.

Another possible way to introduce the context could be probabilistic modeling, more precisely – learning a prior which would depend on the set of domains at hand. One approach could be to directly model the context as a latent variable conditioned on the data from multiple domains. Another strategy could be to reverse the conditioning, e.g. to model the weights of the adapters as the latent variables, and to condition their distributions on the context computed from the data. The second approach would require designing a domain encoder that maps the set of domains to the context, which can be adopted from the practices of task encoding in few-shot learning. Even with such adaptation, there are several challenges to be tackled, e.g. how to aggregate the knowledge from the entire domain, since the are much more labeled samples within one domain in multi-domain learning compared to one task in few-shot learning.

5.2.3 Multimodal transfer learning

Representation learning is a crucial step in every deep learning task (Bengio et al., 2013; Bilen & Vedaldi, 2017). Many transfer learning models adopt the strategy of re-using a pre-traned feature extractor, which decouples the initial task into the two parts: 1) data embedding, and 2) learning problem-specific network modules built on top of the feature extractor, e.g. FiLM parameters in few-shot learning. Majority of works focus on the second part, and use feature extractors pre-trained on common benchmarks, such as ImageNet (Russakovsky et al., 2015), however, there is no guarantee that this is the best strategy.

In the last few years multimodal representation learning (Baltrušaitis et al., 2018), i.e. learning representation of data which comes from different sources, such as image, text, audio etc., has received a lot of attention. Most recently significant progress in this area has been shown by CLIP (Radford et al., 2021), a large-scale language-vision model that uses language supervision to learn image representation. One of the most impressive attributes of this model is its potential for zero-shot transfer, when a model is required to generalise to new tasks without any additional training on them. At the same time the authors acknowledge the difficulties with incorporating additional data in the few-shot learning setup, as well as admit that the model requires large validation sets to efficiently transfer the model to new tasks in the zero-shot setup, which is unrepresentative of the true zero-shot learning.

Despite the challenges, task-agnostic large-scale pre-training looks very promising, and it could potentially be adopted for other transfer learning problems. For example, it could replace the commonly used pre-training on ImageNet in multi-domain learning. Since Radford et al. (2021) use ResNet-50 as the network architecture, application of various adapters is straightforward. Another intriguing direction or research in this setup could be incorporation of the language part of the model for domain encoding, which could be further combined with the ideas discussed in Section 5.2.2.

References

Anokhin, I., Demochkin, K., Khakhulin, T., Sterkin, G., Lempitsky, V., and Korzhenkov, D. Image generators with conditionally-independent pixel synthesis. In *CVPR*, 2021.

Atsuhiro, N. and Tatsuya, H. Image generation from small datasets via batch statistics adaptation. In *ICCV*, 2019.

Baevski, A., Hsu, W.-N., Conneau, A., and Auli, M. Unsupervised speech recognition. In *NeurIPS*, 2021.

Baltrušaitis, T., Ahuja, C., and Morency, L.-P. Multimodal machine learning: A survey and taxonomy. *PAMI*, 2018.

Bansal, M., Krizhevsky, A., and Ogale, A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics Science and Systems*, 2019.

Bateni, P., Goyal, R., Masrani, V., Wood, F., and Sigal, L. Improved few-shot visual classification. In *CVPR*, 2020.

Bellman, R. *Dynamic programming*. Princeton University Press, 1957.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828, 2013.

Berriel, R., Lathuillere, S., Nabi, M., Klein, T., Oliveira-Santos, T., Sebe, N., and Ricci, E. Budget-aware adapters for multi-domain learning. In *ICCV*, 2019.

Bertinetto, L., Henriques, J., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *NeurIPS*, 2016.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

Bilen, H. and Vedaldi, A. Weakly supervised deep detection networks. In *CVPR*, 2016.

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

Bishop, C. *Pattern recognition and machine learning*. Spinger-Verlag, 2006.

Bregman, L. M. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 1967.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. End-to-end incremental learning. In *ECCV*, 2018.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *CVPR*, 2014.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016.

Courbariaux, M., Bengio, Y., and David, J.-P. BinaryConnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 2015.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understandingbert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. In *ICLR*, 2020.

Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. In *ICLR*, 2017.

Dvornik, N., Schmid, C., and Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In *ICCV*, 2019.

Eitz, M., Hays, J., and Alexa, M. How do humans sketch objects? *ACM Trans. Graphics*, 31 (4):1–10, 2012.

Esther, R., Wen-Sheng, C., Abhishek, K., and Jia-Bi, H. Few-shot adaptation of generative adversarial networks. *arXiv preprint*, 2020.

Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. A brief review of domain adaptation. *Advances in data science and information engineering*, 2021.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. *NeurIPS*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Ganin, Y. and Lempitsky, V. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D., Eslami, S., and Teh, Y. Neural processes. In *ICML workshop on theoretical foundations and applications of deep generative models*, 2018.

Gatys, L., Ecker, A., and Bethge, M. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

Geiger, A., Lenz, P., and Urtasun, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. Boosting few-shot visual learning with self-supervision. In *ICCV*, 2019.

Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical Bayes. In *ICLR*, 2018.

Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

Guo, Y., Li, Y., Wang, L., and Rosing, T. Depthwise convolution is all you need for learning multiple visual domains. In *AAAI*, 2019a.

Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., and Feris, R. SpotTune: Transfer learning through adaptive fine-tuning. In *ICCV*, 2019b.

Ha, D., Dai, A., and Le, Q. HyperNetworks. In *ICLR*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., Laroussilhe, Q. D., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.

Hu, S. X., Moreno, P., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N., and Damianou, A. Empirical bayes transductive meta-learning with synthetic gradients. In *ICLR*, 2020.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Densely connected convolutional networks. In *CVPR*, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *NeurIPS*, 2016.

Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory*, 1968.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. Attentive neural processes. In *ICLR*, 2019.

Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *NeurIPS*, 2018a.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In *ICML*, 2018b.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Kingma, D., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *NeurIPS*, 2015.

Kochurov, M., Garipov, T., Podoprikhin, D., Molchanov, D., Ashukha, A., and Vetrov, D. Bayesian incremental learning for deep neural networks. In *ICLR*, 2018.

Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *ICCV Workshops*, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

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

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 52:436–444, 2015.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

Li, H., Eigen, D., Dodge, S., Zeiler, M., and Wang, X. Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *CVPR*, 2019.

Li, Y. and Vasconcelos, N. Efficient multi-domain learning by covariance normalization. In *CVPR*, 2019.

Li, Z. and Hoiem, D. Learning without forgetting. *PAMI*, 2017.

Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. Microsoft COCO: common objects in context. In *ECCV*, 2014.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NeurIPS*, 2016.

Liu, S., Johns, E., and Davison, A. End-to-end multi-task learning with attention. In *CVPR*, 2019a.

Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S., and Yang, Y. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*, 2019b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

MacKay, D. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

Maddison, C., Mnih, A., and Teh, Y. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017.

Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint*, arXiv:1306.5151, 2013.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

Mancini, M., Ricci, E., Caputo, B., and Bulo, S. R. Adding new tasks to a single network with weight transformations using binary masks. In *ECCV Workshops*, 2018.

Marino, J., Yue, Y., and Mandt, S. Iterative amortized inference. In *ICML*, 2018.

McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier, 1989.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, 2012.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *ICLR*, 2018.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stich networks for multi-task learning. In *CVPR*, 2016.

Mullapudi, R. T., Mark, W. R., Shazeer, N., and Fatahalian, K. Hydranets: Specialized dynamic architectures for efficient inference. In *CVPR*, 2018.

Munder, S. and Gavrila, D. M. An experimental study on pedestrian classification. *PAMI*, 28 (11):1863–1868, 2006.

Munkhdalai, T., Yuan, X., Mehri, S., and Trischler, A. Rapid adaptation with conditionally shifted neurons. In *ICML*, 2018.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Neal, R. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

Oreshkin, B., López, P. R., and Lacoste, A. TADAM: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

Perez, E., Strub, F., Vries, H. D., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Petersen, K. B. and Pedersen, M. S. *The Matrix Cookbook*. Technical University of Denmark, 2008.

Poplin, R., Varadarajan, A. V., Blumer, K., Liu, Y., McConnell, M. V., Corrado, G. S., Peng, L., and Webster, D. R. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering*, 2018.

Qiao, S., Liu, C., Shen, W., and Yuille, A. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *ICML*, 2021.

Ratcliff, R. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 1990.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017a.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. icarl: Incremental classifier and representation learning. In *CVPR*, 2017b.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, 2018.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes. *NeurIPS*, 2019.

Rezende, D., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Rosenfeld, A. and Tsotsos, J. Incremental learning through deep adaptation. *PAMI*, 42(3): 651–663, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., and Bernstein, M. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Rusu, A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Saleh, B. and Elgammal, A. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *International Journal for Digital Art History*, 2016.

Schmidhuber, J. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 1992.

Schmidhuber, J. *Evolutionary Computation: Theory and Applications*, chapter A general method for incremental self-improvement and multiagent learning, pp. 81–123. Scientific Publ. Co., 1999.

Schmidt, J., Marques, M. R., Botti, S., and Marques, M. A. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 2019.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NeurIPS*, pp. 4077–4087, 2017.

Soomro, K., Zamir, A. R., and Shah, M. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *ICML*, 2020.

Sun, Q., Liu, Y., Chua, T., and Schiele, B. Meta-transfer learning for few-shot learning. In *CVPR*, 2019.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P., and Hospedales, T. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020.

Triantafillou, E., Larochelle, H., Zemel, R., and Dumoulin, V. Learning a universal template for few-shot dataset generalization. In *ICML*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. In *NeurIPS*, 2016.

Weinberger, K., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. In *NeurIPS*, 2006.

Welinder, P., S.Branson, Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-ucsd birds 200. Technical report, California Institute of Technology, 2010.

Xu, Y., Verma, D., Sheridan, R. P., Liaw, A., Ma, J., Marshall, N. M., McIntosh, J., Sherer, E. C., Svetnik, V., and Johnston, J. M. Deep dive into machine learning models for protein engineering. *Journal of chemical information and modeling*, 2020.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Learning embedding adaptation for few-shot learning. *CoRR*, 2018.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. Metagan: An adversarial approach to few-shot learning. In *NeurIPS*, pp. 2365–2374, 2018.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Trans. Knowledge and Data Engineering*, 2021. doi: 10.1109/TKDE.2021.3070203.