Approximate Bayesian Methods for Sequential Few-Shot Problems

Pau Ching Yap

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

March 2023

I, Pau Ching Yap, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Neural networks are known to suffer from catastrophic forgetting when trained on sequential datasets. While there have been numerous attempts to solve this problem in large-scale supervised classification, little has been done to overcome catastrophic forgetting in few-shot classification problems. We demonstrate that the popular gradient-based model-agnostic meta-learning (MAML) algorithm indeed suffers from catastrophic forgetting.

In this thesis, we introduce the Bayesian online meta-learning framework to tackle the catastrophic forgetting issue in sequential few-shot classification problems. Our framework utilises Bayesian online learning and meta-learning along with Laplace approximation and variational inference to achieve this goal. The experimental evaluations demonstrate that our framework can effectively attain this objective in comparison to various baselines. As an additional utility, we demonstrate empirically that our framework is capable of meta-learning on sequentially arriving few-shot tasks from a stationary task distribution.

Laplace approximation entails Hessian computation for its Gaussian precision matrix. We extend the Kronecker-factored Hessian approximation method in the large-scale classification setting to the gradient-based meta-learning setting. The experiments illustrate the importance of this extension for a principled framework when dealing with a long sequence of few-shot problems.

The final part of this thesis enhances the Bayesian online meta-learning framework for automation and flexibility in handling a greater deal of sequential few-shot problems. We utilise the long short-term memory networks (LSTMs) to automate the meta-learning quick adaptation. The enhancement considers

separating the neural network structure of a model, allowing the framework to cope with different types of few-shot problems. We also incorporate a generative classifier into the enhancement to act as a pointer that informs the model about the few-shot problems it encounters.

Impact Statement

The work presented in this thesis exhibits various benefits both within and outside academia. In the academic research field of artificial intelligence and machine learning, our work has presented a new research direction in continual learning for a sequence of few-shot problems. Continual learning and few-shot learning are popular research directions that have attracted great attention within the academic field of research. This thesis combines meta-learning for few-shot classification along with continual learning to address catastrophic forgetting. Our work can be extended in future research to handle few-shot problems in other settings such as unsupervised learning and reinforcement learning.

Artificial general intelligence has significantly gained public interest and attention beyond the academic field. There are numerous industrial attempts to create robots with general intelligence that can engage in human tasks. The knowledge and experiences of a human being are continually accumulated based on the events encountered. We naturally demand a robot with general intelligence to possess the capability of knowledge accumulation in order to integrate into the daily routines of human beings.

A possible future extension in the industry is to develop a personal assistant robot, which is an agent tailored to integrate seamlessly into the dynamic lifestyle and hobbies of a human being. For instance, the robot could initially be trained to assist in culinary tasks, comprising a range of abilities from recipe recommendation to meal preparation guidance. Subsequently, the robot's skill set can be expanded to include gardening, which is an entirely different domain

that requires a distinct set of knowledge and experience. The versatility to transition between culinary assistance and gardening support is crucial, since human hobbies and lifestyle choices are prone to frequent changes. We strive to create an agent that can deal with a wider range of problems, thus getting closer to achieving artificial general intelligence. The work in Chapter 6 is funded by the GoodAI company via GoodAI grant, which demonstrates an industrial interest in our work.

Acknowledgements

I would like to express my gratitude to Professor David Barber, my primary PhD supervisor, for his relentless guidance in my research work. My PhD studies would not have been possible without his help and supervision. I am also very grateful to my secondary supervisor Dr. Brooks Paige for his effort in keeping my PhD research progress on track. I would like to express my deepest appreciation to my loved ones for the unconditional love and mental support, which assisted me through various obstacles in my studies. I would like to thank my friends and fellow PhD colleagues in the group for the spontaneous yet inspiring discussions about ongoing trends and ideas in research.

Contents

1 Introduction **25**

1.1 Contributions 29

1.2 Thesis Structure 30

2 Related Work **32**

2.1 Online Meta-Learning 33

2.1.1 Regret minimisation 33

2.1.2 Same underlying task distribution 34

2.2 Offline Meta-Learning 34

2.2.1 Probabilistic 34

2.2.2 Non-probabilistic 35

2.3 Continual Learning 35

3 Background **38**

3.1 Meta-Learning 38

3.1.1 Inner and Outer Updates 38

3.1.2 Model-Agnostic Meta-Learning 40

3.2 Bayesian Online Learning 42

3.3 Laplace Approximation 44

3.3.1 Precision Update Hyperparameter 45

3.3.2 Algorithm 46

3.4 Variational Inference 47

3.4.1 Posterior Approximation 47

3.4.2 Algorithm 47

4 Bayesian Online Meta-Learning 49

4.1 Framework Overview 49

4.2 BOML with Laplace Approximation 51

4.2.1 Derivation and Implementation 51

4.2.2 Algorithm 53

4.3 BOML with Variational Inference 53

4.3.1 Derivation and Implementation 53

4.3.2 Algorithm 55

4.4 Experiments 56

4.4.1 Setup 56

4.4.2 Triathlon 58

4.4.3 Pentathlon 60

4.5 BOML in Sequential Task Setting 63

4.5.1 Setting and Algorithm 63

4.5.2 Omniglot: Stationary Task Distribution 64

4.5.3 Results 65

4.6 Discussion 66

4.7 Ablation Studies 68

4.7.1 Varying Precision Update Hyperparameter 68

4.7.2 Analysing the Approximate Posterior Covariance 69

5 Hessian Approximation 71

5.1 Introduction 71

5.2 Background 72

5.2.1 Fully-Connected Layers 73

5.2.2 Convolution Layers 76

5.2.3 Batch Normalisation Layers 79

5.3 Hessian Approximation for BOML 80

5.3.1 Pre-Adaptation Hessian 81

5.3.2 Post-Adaptation Hessian 82
5.4 Experiments 85
5.4.1 Ignoring the Jacobian 85
5.4.2 Analysing the Cross Terms 88

6 Automating Bayesian Online Meta-Learning 92
6.1 Introduction 92
6.2 Background 94
6.2.1 Learning-to-Learn with LSTM 94
6.2.2 Generative Classifier 96
6.2.3 Badger Architecture 97
6.3 Framework 98
6.3.1 Overview 98
6.3.2 LSTM Inner Loop 99
6.3.3 Task-Pointer 100
6.3.4 Relation to Badger 101
6.4 Implementation 101
6.4.1 Training 101
6.4.2 Evaluation 103
6.5 Experiments 103
6.5.1 Setup 103
6.5.2 Component Comparison 105
6.5.3 Results 106

7 Conclusion and Discussion 110
7.1 Conclusion 110
7.2 Discussion 112
7.2.1 Advantage of BOML and BOML+ 112
7.2.2 Disadvantage and Future Research 113

Appendices 115

A Hyperparameters **115**

A.1 Triathlon and Pentathlon 115

A.2 Omniglot: Sequential Tasks 117

A.3 BOML+ 117

Bibliography **118**

List of Figures

1.1 An example of catastrophic forgetting when the knowledge domain datasets arrive sequentially for training: MNIST → SVHN → ImageNet. When ImageNet arrives for training, the model might forget the previously learned MNIST and SVHN (in dashed line – –). A drastic performance drop on MNIST and SVHN when training on the new dataset ImageNet is known as catastrophic forgetting. 25

1.2 An example of a task in the Omniglot knowledge domain. The Omniglot classes are composed of various alphabets with different characters, such as Latin character 1, Sanskrit character 14, Greek character 3, and so forth. A task is formed by sampling a specific number of classes. Few-shot learning seeks for a model that can adapt quickly to a task using very few labelled examples. The model performance is evaluated on the remaining data from the task. 26

1.3 An example of the sequential few-shot classification problems with evident dataset distributional shift: Omniglot → CIFAR-FS → *mini*ImageNet. The datasets arrive in sequential order for training. Upon completion in training, we expect the model to be able to handle tasks from all knowledge domains (connected in ——). We sample an unseen task from each knowledge domain (arrow in ——). For a sampled task from Omniglot as an example, the model undergoes quick adaptation (in red dashed arrow) using very few examples from the task and we evaluate the few-shot performance (in blue dashed arrow – –) using the remaining data from the same task. 27

1.4 An example of the intrinsic human learning behaviour of generalising to relevant tasks (in red arrow) and continually accumulate knowledge from different domains (in blue arrow). Few-shot learning handles a process of the red arrows, but it cannot manage a process of the blue arrows. Conversely, continual learning deals with a process of the blue arrows, but it fails to handle a process of the red arrows. Our sequential few-shot problems setting assimilates both learning capabilities into a single agent. 28

3.1 An illustration of the base-novel split for the Omniglot classes. Omniglot comprises 1623 characters from various alphabets and each character is a class. We can, for instance, sample 1000 characters as the base set for meta-training and the remaining classes as the novel set for meta-evaluation. 39

3.2 An example of a meta-training inner and outer loop in the **5-way 1-shot** setting for Omniglot. We sample a meta-batch of tasks $\mathcal{D}^1, \ldots, \mathcal{D}^m, \ldots$ from a base set $\mathcal{D}$ of the Omniglot knowledge domain. Each task m is split into the support set $\mathcal{D}^{m,S}$ and query set $\mathcal{D}^{m,Q}$. The support set $\mathcal{D}^{m,S}$ comprises one example from each class of the 5-way task. For each task m, the inner loop quickly adapts the meta-parameters θ into a task-specific $\tilde{\theta}^m$ using $\mathcal{D}^{m,S}$. The outer loop then aggregates the losses on every $\tilde{\theta}^m$ with query set $\mathcal{D}^{m,Q}$. The aggregated loss is utilised to update the meta-parameters θ. 40

4.1 The BOML process flow for meta-training and meta-evaluation on an example sequence (Omniglot $\rightarrow$ CIFAR-FS $\rightarrow$ *mini*ImageNet) when each dataset arrives. Consider the CIFAR-FS knowledge domain $\mathfrak{D}_2$ for instance, $\mathfrak{D}_2$ is split into the base $\mathcal{D}_2$ and novel $\widehat{\mathcal{D}}_2$ sets when it arrives. Meta-training on this knowledge domain only occurs on the base set $\mathcal{D}_2$ using the recursive formula in Equation (4.3). The arrows in purple illustrate that the updated posterior is being brought forward for the next meta-training when a new dataset arrives. We meta-evaluate the few-shot performance on the accumulated novel sets $\widehat{\mathcal{D}}_1$, $\widehat{\mathcal{D}}_2$ from the knowledge domains that arrived so far. 50

4.2 The triathlon 5-way 1-shot sequence in this experiment. 58

4.3 Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Going from left to right on the x-axis of the figure is the meta-training times of the knowledge domain datasets that arrive in sequential order. The second row in the figure, for instance, corresponds to the *mini*QuickDraw knowledge domain. The first plot in the second row is empty since *mini*QuickDraw has not arrived during the meta-training time of Omniglot. The diagonal plot (middle plot) in the second row corresponds to the meta-evaluation accuracy on the novel set of *mini*QuickDraw when meta-training occurs on the base set of *mini*QuickDraw. The off-diagonal plot (last plot) shows the meta-evaluation on the *mini*QuickDraw novel set, when meta-training occurs on the next knowledge domain CIFAR-FS. Higher accuracy values in the off-diagonals indicate less forgetting. The baseline TOE corresponds to an upper limit in the performance since it has access to all datasets encountered so far. Sequential MAML corresponds to a lower limit in the performance since MAML forgets on previous datasets by design of the algorithm. 59

4.4 The pentathlon 5-way 1-shot sequence in this experiment. 60

4.5 Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Going from left to right on the x-axis of the figure is the meta-training times of the knowledge domain datasets that arrive in sequential order. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BOMLA with $\lambda = 100$ gives good performance in the off-diagonal plots (retains performances on previously learned datasets), and has a minor performance trade-off in the diagonal plots (learns less well on new datasets). The second row in the figure, for instance, corresponds to the CIFAR-FS knowledge domain. The first plot in the second row is empty since CIFAR-FS has not arrived during the meta-training time of Omniglot. The diagonal plot (second plot) in the second row shows the meta-evaluation accuracy on the novel set of CIFAR-FS when meta-training occurs on the base set of CIFAR-FS. The off-diagonal plots (last three plots) in the second row show the meta-evaluation on the CIFAR-FS novel set, when meta-training occurs on the subsequent knowledge domains *mini*ImageNet, VGG-Flowers and Aircraft. Sequential MAML gives better performance in the diagonal plots (learns well on new datasets) but worse performance in the off-diagonal plots (forgets previously learned datasets). BOMVI is also able to retain performance on previous datasets, although it may be unable to perform as good as BOMLA due to sampling and estimator variance. 61

4.6 An example of the Omniglot task sequence for meta-training in this experiment. 64

4.7 Meta-evaluation accuracy across 3 seed runs on the novel tasks along meta-training. **Left**: compares BOMLA to the baselines, **centre**: compares BOMVI to the baselines, **right**: compares BOMLA with different λ values to BOMVI. 66

4.8 Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BOMLA with a large $\lambda = 1000$ gives better performance in the off-diagonal plots (retains performances on previously learned datasets) but worse performance in the diagonal plots (does not learn well on new datasets). A small $\lambda = 1$ gives better performance in the diagonal plots (learns well on new datasets) but worse performance in the off-diagonal plots (forgets previously learned datasets). BOMVI is also able to retain performance on previous datasets, although it may be unable to perform as good as BOMLA due to sampling and estimator variance. 68

4.9 The change in the approximate posterior variance after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolutional layers of the neural network which gets closer to the classifying layer. Each plot in the figure is the colour-encoded variance corresponding to a specific knowledge domain dataset and the meta-parameters of a specific layer in the neural network model. The variance in each layer is flattened into a two-dimensional matrix visualisation. A darker colour indicates a higher variance. The variance increases in general as the convolutional layer gets closer to the classifying layer. The variance decreases in the raw level filters (Conv 1) as the model learns along the pentathlon sequence. 69

5.1 An example of a fully-connected neural network with $L = 2$ layers and weight matrices W_1, W_2. The bias vectors are omitted in this example. The weights are vectorised as $\vartheta = [\text{vec}(W_1)^T, \text{vec}(W_2)^T]^T$. The input a_0, pre-activations h_1, h_2 and activations a_1, a_2 interact according to Equation (5.4) using activation functions f_1, f_2. 73

5.2 A two-dimensional example of a 6-by-6 activation with spatial location $k \in \mathcal{K}$ for a 3-by-3 filter. For a convolutional operation of stride 1, we have $|\mathcal{K}| = 16$ in this example. The batch size and input channels are ignored in the illustration 76

5.3 A two-dimensional example for a spatial offset δ of a 3-by-3 filter. In this example we have $|\Delta| = 9$. The input and output channels are ignored in this illustration. 77

5.4 Meta-evaluation performance comparison across 3 seed runs on the novel tasks along meta-training. **Left**: compares BOMLA of Hessian approximation in Equations (5.39) and (5.40) (with Jacobian) versus BOMLA of Hessian approximation in Equation (5.41) (without Jacobian). BOMLA that uses Hessian approximation without Jacobian (—) shows a large performance degradation compared to that with Jacobian (—). **Right**: meta-evaluation accuracy difference between BOMLA with Jacobian at $\lambda = 0.01$ and BOMLA without Jacobian at $\lambda = 1$. The difference evolves around zero, indicating that the adjustment $\lambda = 1$ gives a quick fix to the posterior approximation when the Jacobian is excluded in Hessian approximation. 87

5.5 The Fisher approximation F corresponding to the Hessian in Equation (5.43) after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolution layers and the fully-connected classifier layer of the neural network. Each plot in the figure is the colour-encoded Fisher approximation F_ℓ corresponding to a specific knowledge domain dataset and a specific layer ℓ in the neural network model. F_ℓ for the middle convolution layers (ℓ = Conv 2, 3 and 4) are cropped as the full matrices are too large to visualise. The F_ℓ matrices for the convolution layers ℓ have entry values that are close to zero. . . 89

5.6 The absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ between the full Fisher and the truncated Fisher for each layer ℓ of the neural network model after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolution layers and the fully-connected classifier layer of the neural network. Each plot in the figure is the colour-encoded absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ corresponding to a specific knowledge domain dataset and a specific layer ℓ of the neural network model. The absolute difference matrices for the middle convolution layers (ℓ = Conv 2, 3 and 4) are cropped as the full matrices are too large to visualise. The absolute difference matrices for the convolution layers ℓ have entry values that are close to zero, but the absolute difference matrices for the classifier layer have large entry values. 90

5.7 The absolute difference $\left|\widetilde{\boldsymbol{F}}_{\ell} - \widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}\right|$ between the full Fisher and the truncated Fisher for a convolution and fully-connected classifier layer ℓ of the neural network model after meta-training is completed on Omniglot. We only retain the visualisation for the first convolution layer (Conv 1) since the remaining convolution layers have the same visualisation. The absolute difference matrices for the convolution layers ℓ have entry values that are close to zero. The absolute difference matrix for the classifier layer corresponding to Omniglot has entry values in the order of tens. 91

6.1 The computational graph for the gradients of $\mathcal{L}(\phi)$ with respect to the LSTM optimiser parameters ϕ when updating ξ at time step $r-1$ and r. The gradients are allowed to flow through the solid arrows during back-propagation, but the gradient flow is prohibited along the dashed arrows. The optimisee corresponds to a model which is usually a neural network with parameters ξ. For a particular time step r, we update the optimisee parameters ξ_r by adding the output g_r acquired from the LSTM. The LSTM takes the gradients ∇_r of objective f with respect to ξ_r as inputs along with hidden states h_r. When computing the gradients of the LSTM parameters, we do not take the gradient flow from ∇_r into consideration. 95

6.2 An example architecture of a Badger agent. The shared expert policy can be accessed by all experts. The red dashed line – – illustrates the connection between the experts and the shared expert policy. The selected experts i and j in yellow colour are responsible for the incoming input. Experts i and j communicate with each other and update their internal states using the internal memories to give an output. 97

6.3 The process flow of BOML+ for training and evaluation on an example sequence (Omniglot → CIFAR-FS → *mini*ImageNet) when each dataset arrives. The arrows in purple illustrate that the updated posterior is being brought forward for the next meta-training when a new dataset arrives. The items in red are the elements newly-introduced in BOML+. This figure resembles the BOML process flow in Figure 4.1, except for the elements in red that are unique to BOML+. 98

6.4 The training processes of BOML+ when each dataset arrives. . . 102

6.5 Input and output layers $\theta_t^{(I)}$ and $\theta_t^{(O)}$ from expert t are concatenated to the model structure. 102

6.6 The evaluation process of BOML+ on the accumulated novel set pool. 103

6.7 Computational graph for one step of the LSTM adaptation on a D-dimensional meta-parameters θ. The LSTMs have shared parameters but separated hidden states. The gradients are allowed to flow through the solid arrows during back-propagation, but not the dashed arrows. The inner loop cross-entropy loss f is evaluated using the D-dimensional meta-parameters $(\theta^1, \ldots, \theta^D)^T$. The gradients $(\nabla^1, \ldots, \nabla^D)^T$ of f for a specific few-shot task with respect to the meta-parameter elements are fed into the LSTMs, and the LSTMs return the updates $(g^1, \ldots, g^D)^T$ for each element of the meta-parameters. 104

6.8 The pentathlon knowledge domain dataset sequence. 105

6.9 Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BOML+ without task-pointer in Run R4 can retain performances on previously learned datasets since it performs best in the off-diagonal plots. Most of the diagonal plots accuracies of Run R4 are as good as the others, indicating that it learns well on new datasets too. 107

List of Tables

5.1 Average reduction in the meta-evaluation accuracy when the Jacobian is excluded in Hessian approximation for BomLA with $\lambda = 100$. The average is taken over tasks from all datasets, after meta-training is completed on the entire dataset sequence. The values are reported by averaging over a total of 100 tasks along with the 95% confidence interval. The performance reduction is not apparent, since the confidence intervals along with the averages cover small reduction values that are both above and below zero. 86

6.1 Meta-evaluation accuracies for Run R4 (Boml+ without task-pointer) and Run R5 (Boml+ with task-pointer) on the datasets upon the completion of meta-training across the entire pentathlon sequence of knowledge domains. 107

6.2 Meta-evaluation accuracies for Run R1 (original Boml) and Run R5 (Boml+ with task-pointer) on various datasets upon the completion of meta-training across the entire pentathlon sequence of knowledge domains. 108

A.1 Hyperparameters for the triathlon and pentathlon experiments (same value for all datasets). 115

A.2 Hyperparameters for the triathlon and pentathlon experiments (individual datasets). 116

A.3 Hyperparameters for the Omniglot sequential tasks experiment. 117

A.4 Hyperparameters for the BOML+ experiments. 117

Chapter 1

Introduction

A key objective in artificial intelligence is to create an agent that can both accumulate knowledge over time and adapt quickly to unseen tasks using the acquired knowledge. Machine learning in general requires a large amount of data from a knowledge domain for training. The models trained under such a condition are unable to adapt to unseen tasks using very few examples, even if the tasks originate from the same knowledge domain. Such models would also struggle if multiple datasets from different knowledge domains arrive sequentially for training. The current typical machine learning methods would either train all datasets together, or forget the previously acquired knowledge if the new domain is adequately distinct from the previous ones. The latter phenomenon is known as *catastrophic forgetting* in continual learning, as illustrated in Figure 1.1.

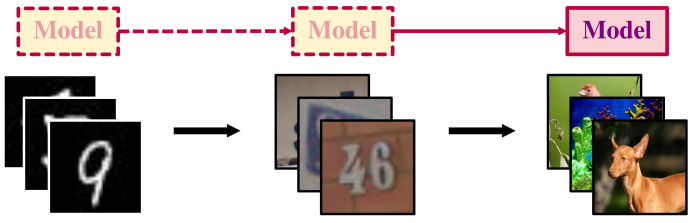

Figure 1.1: An example of catastrophic forgetting when the knowledge domain datasets arrive sequentially for training: MNIST → SVHN → ImageNet. When ImageNet arrives for training, the model might forget the previously learned MNIST and SVHN (in dashed line – –). A drastic performance drop on MNIST and SVHN when training on the new dataset ImageNet is known as catastrophic forgetting.

Continual learning (Goodfellow et al., 2013; Lee et al., 2017; Zenke et al., 2017) aims to incorporate new knowledge to an existing system as training data arrives in a sequential order. An important distinction between continual learning and online learning (Zinkevich, 2003; Shalev-Shwartz, 2007) is that continual learning handles distributional shift in the knowledge domains that arrive in sequential order, whereas online learning takes data points in an online manner from the same underlying dataset distribution for training.

Few-shot learning (Miller et al., 2000; Li et al., 2004; Lake et al., 2011) focuses on adapting to unseen tasks using very few labelled examples from a task as shown in Figure 1.2. Recent works show that meta-learning provides promising approaches to few-shot classification problems (Santoro et al., 2016; Finn et al., 2017; Ravi and Larochelle, 2017). Meta-learning or learning-to-learn (Schmidhuber, 1987; Thrun and Pratt, 1998) takes the learning process a level deeper – instead of learning from the labelled examples in the training process, meta-learning learns the example-learning process.

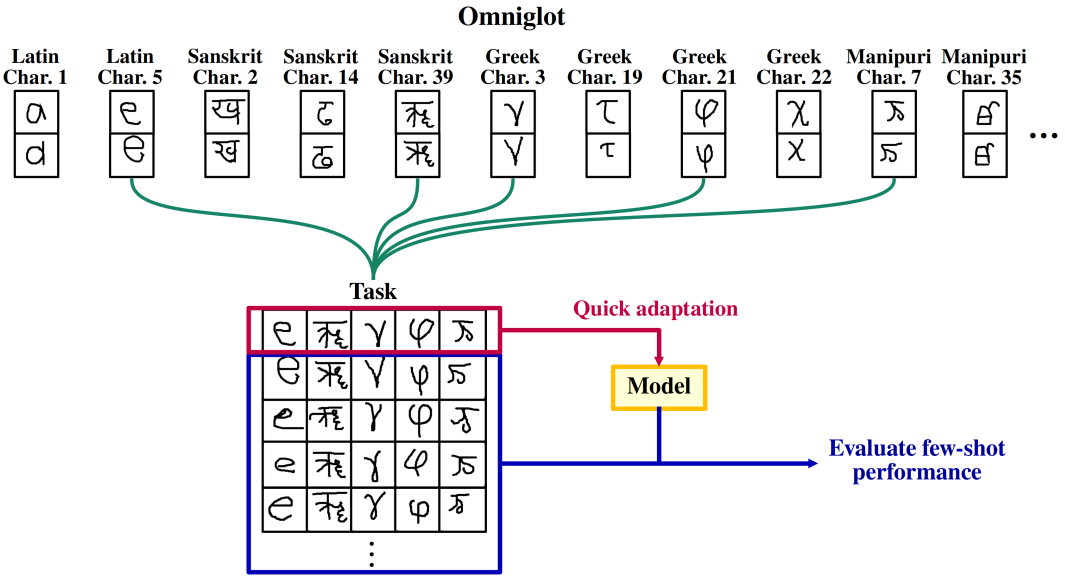

Figure 1.2: An example of a task in the Omniglot knowledge domain. The Omniglot classes are composed of various alphabets with different characters, such as Latin character 1, Sanskrit character 14, Greek character 3, and so forth. A task is formed by sampling a specific number of classes. Few-shot learning seeks for a model that can adapt quickly to a task using very few labelled examples. The model performance is evaluated on the remaining data from the task.

Despite being a promising solution to few-shot problems, meta-learning methods suffer from a limitation where a meta-learned model loses its quick adaptation ability on previous datasets as new ones arrive subsequently for training. Some popular examples of different few-shot classification problems are Omniglot (Lake et al., 2011), CIFAR-FS (Bertinetto et al., 2019) and *mini*ImageNet (Vinyals et al., 2016). A meta-learned model is restricted to few-shot classification on a specific dataset, in the sense that the training and evaluation few-shot tasks have to originate from the same distribution. The current practice to handle few-shot classification from different datasets is to meta-learn a model for each dataset separately (Snell et al., 2017; Vinyals et al., 2016; Bertinetto et al., 2019). This thesis considers meta-learning a single model for few-shot classification on multiple datasets with evident distributional shift that arrive sequentially for training as illustrated in Figure 1.3.

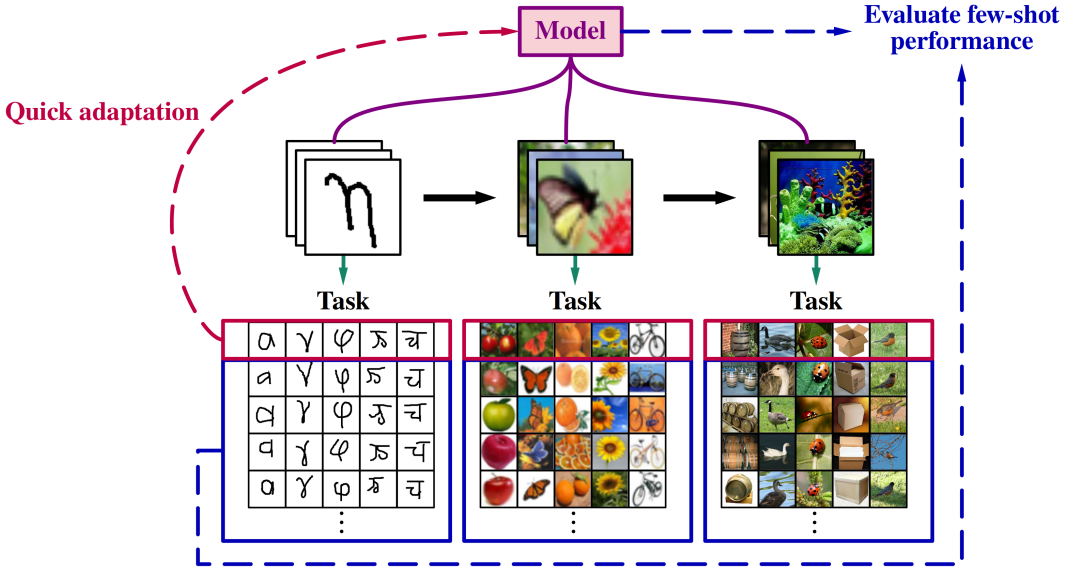

Figure 1.3: An example of the sequential few-shot classification problems with evident dataset distributional shift: Omniglot → CIFAR-FS → *mini*ImageNet. The datasets arrive in sequential order for training. Upon completion in training, we expect the model to be able to handle tasks from all knowledge domains (connected in ——). We sample an unseen task from each knowledge domain (arrow in ——). For a sampled task from Omniglot as an example, the model undergoes quick adaptation (in red dashed arrow - -) using very few examples from the task and we evaluate the few-shot performance (in blue dashed arrow - -) using the remaining data from the same task.

Why is the sequential few-shot problems setting important? Human beings acquire knowledge continually starting from a very young age, and such a capability corresponds to continual learning in our topic of discussion. Upon learning on a specific subject, human beings tend to generalise to a different but relevant area without requiring enormous extra effort to learn the relevant area. This corresponds to few-shot learning in the machine learning terminology. For instance, a pre-school child learning about numbers might require more examples and practises when learning the numbers '1', '2' and '3' initially. The child can usually learn '4', '5' and '6' with fewer examples, after mastering the previously learned numbers. This pattern of learning is also pertinent to other knowledge domains such as object recognition.

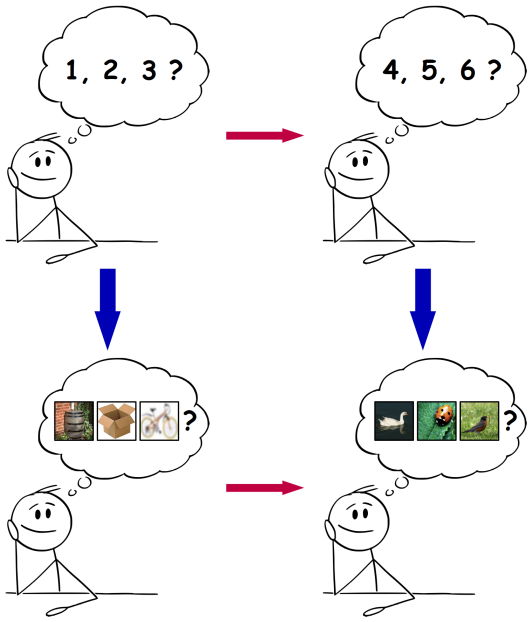

Figure 1.4: An example of the intrinsic human learning behaviour of generalising to relevant tasks (in red arrow) and continually accumulate knowledge from different domains (in blue arrow). Few-shot learning handles a process of the red arrows, but it cannot manage a process of the blue arrows. Conversely, continual learning deals with a process of the blue arrows, but it fails to handle a process of the red arrows. Our sequential few-shot problems setting assimilates both learning capabilities into a single agent.

Continual learning enables an agent to assimilate knowledge sequentially across diverse domains. However, it is unable to generalise acquired knowledge

to related tasks within a single knowledge domain, which is the crucial functionality of few-shot learning. Contrarily, few-shot learning exhibits proficiency in generalising to similar tasks within a knowledge domain, but it is incapable of acquiring knowledge over time across a multitude of domains. Human beings unknowingly employ these learning capabilities in our daily routines. We continually gather knowledge from various domains and utilise a high-level cognition from previous experiences into solving a different but related problem. We envision the possibility of creating a single agent with these skills consolidated to achieve artificial general intelligence.

1.1 Contributions

We introduce a recursive framework to train a model that is applicable to a broader scope of few-shot knowledge domains by overcoming catastrophic forgetting. Bayesian online learning (BOL) (Opper, 1998) provides a principled framework for the posterior of the model parameters, while model-agnostic meta-learning (MAML) (Finn et al., 2017) finds a good model parameter initialisation that can quickly few-shot adapt to unseen tasks. Our framework incorporates BOL and meta-learning to give a recursive formula for the posterior of the meta-parameters as new few-shot datasets arrive. Taking a MAP estimate in implementation leads to Laplace approximation, whereas using a KL-divergence leads to variational inference.

The full Hessian computation is necessary in Laplace approximation for its Gaussian precision matrix. Hessian calculations are generally infeasible due to the large size of the modern neural network architectures. Ritter et al. (2018a) apply BOL with Laplace approximation to the large-scale classification setting by approximating a Hessian with a block-diagonal Kronecker-factored Fisher approximation. We extend this Hessian approximation method to the gradient-based meta-learning setting. The extension results in a principled Bayesian online meta-learning framework with Laplace approximation.

The Bayesian online meta-learning framework in its original form suf-

fers from a restriction where the quick adaptation of the framework uses a hand-crafted gradient-based algorithm with a manually chosen learning rate. Another restriction is that the framework can only manage a single few-shot classification setting across all sequential datasets. We enhance the framework to an automated and flexible method that can handle a wider range of few-shot problems in a sequential order. Andrychowicz et al. (2016) previously use the long short-term memory networks (LSTMs) to automate the learning process for large-scale classification. We utilise LSTMs to automate the quick adaptation component of meta-learning in our framework. Each dataset in a sequence requires a different LSTM for quick adaptation. We incorporate a generative classifier (van de Ven et al., 2021) into the enhancement to act as a pointer that can inform the model on which LSTM should be responsible for a task during the evaluation period. Generative classification is originally implemented for class-incremental learning (van de Ven et al., 2021). The method is directly applicable to our enhancement if we contemplate the sequential datasets as 'classes' that arrive sequentially for class-incremental learning. We also separate the neural network structure of a model into the input, body and output layers in the enhancement. This allows the framework to deal with different types of few-shot problems.

1.2 Thesis Structure

Chapter 2 reviews prior research in the literature that is relevant to this thesis. Chapter 3 gives a background explanation of meta-learning as well as Bayesian online learning with Laplace approximation and variational inference for posterior approximation.

Chapter 4 makes the following contributions in this thesis:

- We develop the Bayesian online meta-learning (BOML) framework for sequential few-shot classification problems. Under this framework we introduce the algorithms Bayesian online meta-learning with Laplace approximation (BOMLA) and Bayesian online meta-learning with varia-

tional inference (BOMVI).

- We demonstrate that BOML can overcome catastrophic forgetting in the sequential few-shot datasets setting with apparent distributional shift in the datasets.
- We demonstrate empirically that BOML can also continually learn to few-shot classify the novel classes in the sequential meta-training few-shot tasks setting.

Chapter 5 is structured as follows:

- We review the block-diagonal Kronecker-factored Fisher approximation method in the large-scale classification setting.
- We propose an approximation to the Fisher corresponding to BOMLA that carries the desirable block-diagonal Kronecker-factored structure.
- We show empirically that the Fisher approximation in BOMLA is essential for a long sequence of few-shot problems.

Chapter 6 makes the following contributions:

- We enhance the BOML framework to BOML+ by automating the meta-learning quick adaptation component, splitting the neural network model into different parts, and incorporating a generative classifier pointer into the enhancement.
- We develop a highly parallelisable training procedure for BOML+.
- We illustrate empirically that BOML+ outperforms the original BOML framework.

Chapter 7 concludes the thesis and provides an in-depth discussion of the work in previous chapters.

Much of the work in Chapters 4 and 5 has been published in Yap et al. (2020, 2021), and Chapter 6 is part of the project funded by GoodAI company via the GoodAI grant.

Chapter 2

Related Work

Continual learning and few-shot learning have emerged as widely pursued research areas within the field of artificial intelligence. Meta-learning constitutes the foundational element of few-shot learning. It enables a model to rapidly adapt to unseen tasks even when presented with an extremely limited amount of data, which is the core motivation of few-shot learning. Meta-learning typically involves the extraction of higher-level insights that facilitates quick generalisation to new tasks in the same knowledge domain.

The novelty of this thesis lies in the formulation of the sequential few-shot problems setting, and the development of a mathematically grounded framework for training in this setting. It is an approach that synthesises the advantages of both continual learning and meta-learning to augment the learning capabilities of an agent. Despite the pioneering nature of this thesis, it is important to acknowledge that both continual learning and meta-learning are highly active fields of research. A common practice in the current research paradigm is to treat continual learning and few-shot learning as distinct research topics.

Our research integrates *offline meta-learning* with continual learning frameworks to mitigate the issue of catastrophic forgetting in sequential few-shot learning tasks. Empirical evidence from the experiment in Chapter 4.5 demonstrates that our proposed framework is also capable of managing experimental settings akin to online meta-learning. Our framework employs a *probabilistic offline meta-learning* method in the sequential few-shot problems setting, albeit

the framework is implemented via a *non-probabilistic method*. This chapter examines the existing literature in these research areas for a comprehensive review of the advancements and methodologies.

2.1 Online Meta-Learning

The work in this thesis differs from online meta-learning in terms of its problem setting and overall objective. In online meta-learning, the tasks that arrive sequentially originate from the same area of knowledge domain. The primary aim of online meta-learning is to leverage the knowledge obtained from previous tasks to reduce the number of samples required for effective training on forthcoming tasks. However it is noteworthy that the reduction in sample size for training in online meta-learning does not typically achieve the same level of sample scarcity observed in few-shot learning. Furthermore, online meta-learning predominantly focuses on optimising the efficiency of training for future tasks, with less emphasis on the maintenance of performance on previously encountered tasks. This aspect forms a critical distinction compared to our framework, as our work seeks to balance the acquisition of new knowledge with the retention of previous experience.

2.1.1 Regret minimisation

The goal in this setting is to minimise the regret function, with assumptions made on the loss function rather than the task distribution. The regret function in an online setting is defined as the difference between the model loss and the best performance attainable by some comparison class of methods. Recent works Finn et al. (2019) and Zhuang et al. (2019) belong to this category, where the aim is to compete with the best meta-learner and supersede it. These methods accumulate data as they arrive and meta-learn using all data acquired so far.

Data accumulation is undesirable since the algorithmic complexity of training increases with the amount of data accumulated, leading to longer training times as new data arrive (Finn et al., 2019; He et al., 2019). The agent

will eventually run out of memory for a long sequence of data. The BOML framework on the other hand is advantageous, as it only takes the current data and the posterior of the meta-parameters into consideration during optimisation. This gives a framework with an algorithmic complexity independent of the length of the dataset sequence.

2.1.2 Same underlying task distribution

Sequential tasks are assumed to originate from the same underlying task distribution $p(\mathcal{T})$ in this setting. Denevi et al. (2019) introduce the online-within-online (OWO) and online-within-batch (OWB) settings, where OWO encounters tasks and examples within tasks sequentially while OWB encounters tasks sequentially but examples within tasks are in batch. Our work in the sequential datasets setting is novel in overcoming few-shot catastrophic forgetting, where the goal is to few-shot classify unseen tasks drawn from a sequence of distributions $p(\mathcal{T}_1), \ldots, p(\mathcal{T}_T)$ as explained in Section 4.1. He et al. (2019), Harrison et al. (2019) and Jerfel et al. (2019) look into continual meta-learning for a non-stationary task distribution where the task boundaries are unknown to the model. Jerfel et al. (2019) consider a latent task structure to adapt to the non-stationary task distribution.

2.2 Offline Meta-Learning

Previous meta-learning works attempt to solve few-shot classification problems in an offline setting, under the assumption of having a stationary task distribution during meta-training and meta-evaluation. A single meta-learned model is aimed to few-shot classify one specific dataset with all base classes of the dataset readily available in a batch for meta-training. There are two general frameworks for the offline meta-learning setting: probabilistic and non-probabilistic frameworks.

2.2.1 Probabilistic

The MAML algorithm can be cast into a probabilistic inference problem (Finn et al., 2018) or with a hierarchical Bayesian structure (Grant et al., 2018;

Yoon et al., 2018). Yoon et al. (2018) use Stein Variational Gradient Descent (SVGD) for task-specific learning. Gordon et al. (2019) implement probabilistic inference by considering the posterior predictive distribution with amortised networks. Grant et al. (2018) discuss the use of a Laplace approximation in the task-specific inner loop to improve MAML using the curvature information. Although at first sight our work seems similar to Grant et al. (2018) due to the use of Laplace approximation, our work is clearly distinct in terms of goal and context. Grant et al. (2018) use Laplace approximation at the task-specific level, whilst we use Laplace approximation at the meta-level for the meta-parameters approximate posterior. The formulation in Grant et al. (2018) does not accumulate past experience, whereas our work enables few-shot learning on unseen tasks from multiple knowledge domains sequentially.

2.2.2 Non-probabilistic

Gradient-based meta-learning (Finn et al., 2017; Nichol et al., 2018; Rusu et al., 2019) updates the meta-parameters by accumulating the gradients of a meta-batch of task-specific inner loop updates. The meta-parameters will be used as a model initialisation for a quick adaptation on the novel tasks. Metric-based meta-learning (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017) utilises the metric distance between labelled examples. This method assumes that base and novel classes are from the same dataset distribution, and the metric distance estimations can be generalised to the novel classes upon meta-learning the base classes.

2.3 Continual Learning

Lifelong learning, continual learning, and catastrophic forgetting are terms frequently encountered in the literature within the same field of research. Lifelong learning (Thrun and Pratt, 1998) describes the ongoing ability of an agent to learn and adapt throughout the agent's lifetime. The key characteristic of lifelong learning is its ability to apply previously learned knowledge on new tasks, which may or may not be directly related to previously encountered

problems. Continual learning (Goodfellow et al., 2013) is often used interchangeably with lifelong learning, but it represents a more specific aspect of the topic. Continual learning specifically refers to a model's ability to learn from a continuous stream of data and tasks. The main challenge in continual learning is to acquire new knowledge while retaining information from previously learned tasks, without the need to store all the data from past tasks. Catastrophic forgetting (Kirkpatrick et al., 2017) is a challenge that needs to be addressed within continual learning. It describes the tendency of a model to lose information about older tasks as it learns on new tasks. Catastrophic forgetting happens because neural networks tend to overwrite weights that were important for previous tasks when they are trained on new tasks that might not originate from the same data distribution, leading to a performance degradation on the old tasks. Special measures must therefore be taken when there is a distributional shift in the sequential tasks. In terms of the conceptual scope, lifelong learning is the broadest concept which encompasses continual learning, and catastrophic forgetting is a specific issue within the topic of continual learning.

Modern continual learning works (Goodfellow et al., 2013; Lee et al., 2017; Zenke et al., 2017) focus primarily on large-scale supervised learning, in contrast to our work that looks into continual few-shot classification across sequential datasets with evident distributional shift. Wen et al. (2018) utilise few-shot learning to improve on overcoming catastrophic forgetting via logit matching on a small sample from the previous tasks.

The online learning element in our work is closely related to recent works that overcome catastrophic forgetting in large-scale supervised classification (Kirkpatrick et al., 2017; Zenke et al., 2017; Ritter et al., 2018a; Nguyen et al., 2018). In particular, our work builds on the online Laplace approximation method (Ritter et al., 2018a). Our work extends this method to the meta-learning scenario to avoid forgetting in few-shot classification problems. Nguyen et al. (2018) provide an alternative of using variational inference instead of

Laplace approximation for approximating the posterior. Our work utilises this approach and adapts the variational method to approximate the posterior of the meta-parameters by adjusting the KL-divergence objective.

Chapter 3

Background

This chapter provides a background explanation of meta-learning in the single dataset setting and reviews Bayesian online learning (BOL) for approximating the posterior of the model parameters in large-scale classification setting.

3.1 Meta-Learning

The goal of meta-learning in few-shot classification is to acquire a model that is able to perform well on an *unseen task* during evaluation after a quick adaptation using very few examples from that unseen task. We review the concept and terminology in meta-learning that differ from the large-scale machine learning.

3.1.1 Inner and Outer Updates

As meta-learning takes the typical learning process to a deeper level, a model learns the example-learning process using the **base set** and evaluates its few-shot adaptation capability on the **novel set**. Figure 3.1 shows a split of the Omniglot classes into base and novel sets. The training process in meta-learning that utilises the base set is called the **meta-training** stage, and the evaluation process that reports the few-shot performance on the novel set is known as the **meta-evaluation** stage. The base set in meta-learning resembles the training dataset in the usual large-scale machine learning, whereas the novel set resembles the testing dataset in machine learning.

Most meta-learning algorithms comprise an inner loop for example-learning and an outer loop that learns the example-learning process. Meta-learning

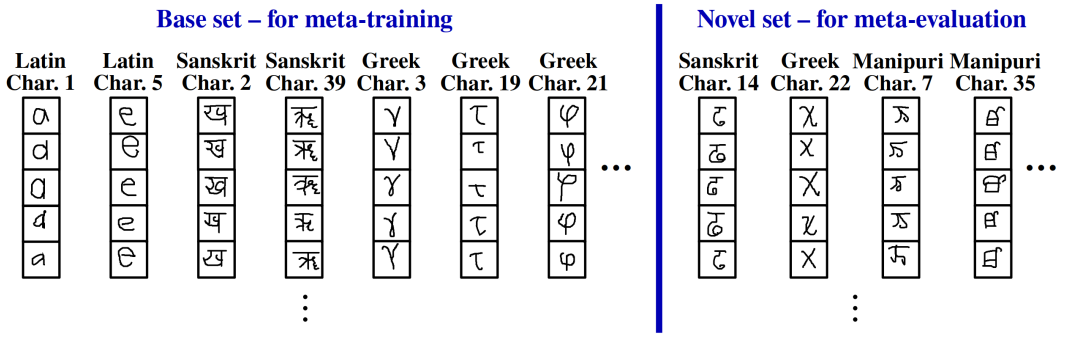

Figure 3.1: An illustration of the base-novel split for the Omniglot classes. Omniglot comprises 1623 characters from various alphabets and each character is a class. We can, for instance, sample 1000 characters as the base set for meta-training and the remaining classes as the novel set for meta-evaluation.

generally aims to find a good model parameter initialisation (called **meta-parameters**) that can quickly adapt to unseen tasks. Figure 3.2 shows the inner and outer loop of a meta-training step. Such algorithms often require sampling a meta-batch of tasks at each iteration. A few-shot task, thereby known as **task**, from a stationary task distribution $p(\mathcal{T})$ in the classification setting is formed by sampling a subset of classes from the pool of base set or novel set during meta-training or meta-evaluation respectively. An N-way K-shot task, refers to sampling N classes and using K examples per class for few-shot quick adaptation.

An offline meta-learning algorithm learns a model only for a specific dataset $\mathfrak{D}$ from a single knowledge domain, which is divided into the set of base classes $\mathcal{D}$ and novel classes $\widehat{\mathcal{D}}$ for meta-training and meta-evaluation respectively. Upon completing meta-training on $\mathcal{D}$, the goal is to perform well on an unseen task $\widehat{\mathcal{D}}^*$ sampled from the novel set $\widehat{\mathcal{D}}$ after a quick adaptation on a small subset $\widehat{\mathcal{D}}^{*,S}$ (known as the **support set**) of $\widehat{\mathcal{D}}^*$. The performance of this unseen task is evaluated on the **query set** $\widehat{\mathcal{D}}^{*,Q}$, where $\widehat{\mathcal{D}}^{*,Q} = \widehat{\mathcal{D}}^* \backslash \widehat{\mathcal{D}}^{*,S}$. Since $\widehat{\mathcal{D}}$ is not accessible during meta-training, this support-query split is mimicked on the base set $\mathcal{D}$ for meta-training as illustrated in Figure 3.2.

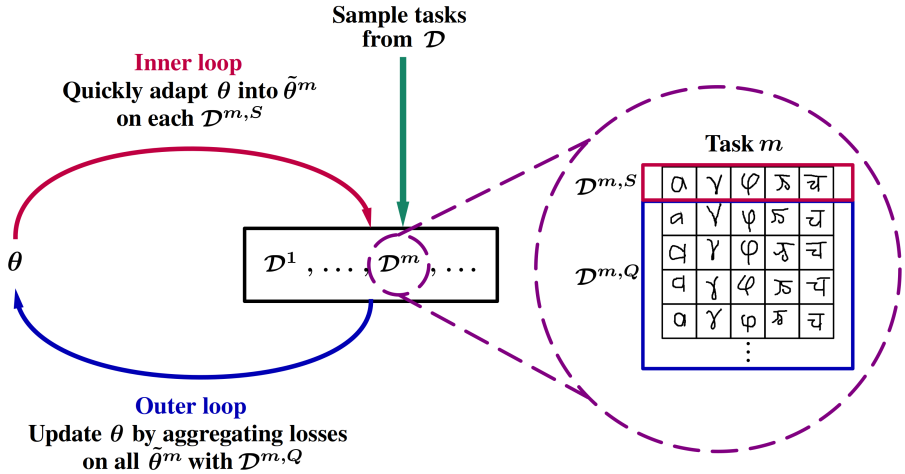

Figure 3.2: An example of a meta-training inner and outer loop in the **5-way 1-shot** setting for Omniglot. We sample a meta-batch of tasks $\mathcal{D}^1, \ldots, \mathcal{D}^m, \ldots$ from a base set $\mathcal{D}$ of the Omniglot knowledge domain. Each task m is split into the support set $\mathcal{D}^{m,S}$ and query set $\mathcal{D}^{m,Q}$. The support set $\mathcal{D}^{m,S}$ comprises one example from each class of the 5-way task. For each task m, the inner loop quickly adapts the meta-parameters θ into a task-specific $\tilde{\theta}^m$ using $\mathcal{D}^{m,S}$. The outer loop then aggregates the losses on every $\tilde{\theta}^m$ with query set $\mathcal{D}^{m,Q}$. The aggregated loss is utilised to update the meta-parameters θ.

3.1.2 Model-Agnostic Meta-Learning

Each meta-training step of the well-known meta-learning algorithm MAML (Finn et al., 2017) aims to find meta-parameters θ that can act as a good model parameter initialisation for quick adaptation to unseen tasks. Each iteration of the MAML algorithm samples M tasks from the base class set $\mathcal{D}$ and runs a few steps of stochastic gradient descent (SGD) for an inner loop task-specific learning. The number of tasks sampled per iteration is known as the **meta-batch size**. For task m, the inner loop outputs the task-specific parameters $\tilde{\theta}^m$ from a k-step SGD quick adaptation on the objective $\mathcal{L}(\theta, \mathcal{D}^{m,S})$ with the support set $\mathcal{D}^{m,S}$ and initialised at θ:

$$\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}^{m,S})), \tag{3.1}$$

where $m = 1, \ldots, M$. The outer loop gathers all task-specific adaptations to update the meta-parameters θ using the loss $\mathcal{L}(\tilde{\theta}^m, \mathcal{D}^{m,Q})$ on the query set

$\mathcal{D}^{m,Q}$. The overall MAML optimisation objective is

$$\arg\min_{\theta} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(SGD_k(\mathcal{L}(\theta, \mathcal{D}^{m,S})), \mathcal{D}^{m,Q}). \tag{3.2}$$

Algorithm 1 gives the pseudo-code of the MAML algorithm in meta-training. Lines 4 – 7 correspond to the inner loop for M tasks, and line 8 denotes the outer loop update of the meta-parameters θ. The outer loop update computes the derivative with respect to the meta-parameters θ via the task-adapted parameters $\tilde{\theta}^m$ for $m = 1, \ldots, M$. Automatic differentiation simplifies the process of differentiating through the inner loop updates, although internally this requires constructing a computational graph whose complexity increases with the number of inner update steps k. It is therefore essential to limit the number of steps k in the inner loop to ensure a manageable computational cost.

Algorithm 1 MAML meta-training

1: **Require:** base set $\mathcal{D}$, learning rate α, number of meta-training iterations J, meta-batch size M
2: **Initialise:** θ
3: **for** $i = 1, \ldots, J$ **do**
4: **for** $m = 1$ **to** M **do**
5: Sample task $\mathcal{D}^m = \mathcal{D}^{m,S} \cup \mathcal{D}^{m,Q}$
6: Inner update $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}^{m,S}))$
7: **end for**
8: Outer update $\theta \leftarrow \theta - \nabla_\theta \frac{1}{M} \sum_m \mathcal{L}(\tilde{\theta}^m, \mathcal{D}^{m,Q})$
9: **end for**

Algorithm 2 is the pseudo-code of MAML meta-evaluation on an unseen task sampled from a novel set.

Algorithm 2 MAML meta-evaluation

1: **Require:** novel set $\widehat{\mathcal{D}}$, learning rate β, meta-trained θ, number of adaptation steps K
2: Sample task $\widehat{\mathcal{D}}^* = \widehat{\mathcal{D}}^{*,S} \cup \widehat{\mathcal{D}}^{*,Q}$
3: **for** $i = 1, \ldots, K$ **do**
4: Few-shot quick adaptation $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}(\theta, \widehat{\mathcal{D}}^{*,S})$
5: **end for**
6: Report performance on $\widehat{\mathcal{D}}^{*,Q}$

Like most offline meta-learning algorithms, MAML assumes a stationary task distribution during meta-training and meta-evaluation. Under this assumption, a meta-learned model is only applicable to a specific dataset distribution. When the model encounters a sequence of knowledge domains with datasets of apparent distributional shift, it loses the few-shot adaptation ability on previous domains as new ones arrive for meta-training. Our work aims to meta-learn a single model for few-shot learning on multiple knowledge domains that arrive sequentially for meta-training. In the few-shot classification setting, we achieve this goal by incorporating meta-learning into the BOL framework to give the *Bayesian online meta-learning* (BOML) framework that considers the posterior of the meta-parameters.

3.2 Bayesian Online Learning

We briefly explain the use of BOL in approximating the posterior of the model parameters to overcome catastrophic forgetting in the large-scale supervised classification setting. Opper (1998) introduces BOL in the simplest setting of online learning where the data points of a dataset $D_t = \{(x_1, y_1), ..., (x_t, y_t)\}$ arrive in a sequential order for training. There is nothing that prohibits the implementation of BOL to a more general setting where the datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_t$ arrive in sequential order instead. Each dataset might originate from different underlying dataset distributions. For easier generalisation to our work later in this thesis, we explain BOL in the setting of sequential datasets arrival rather than online data points.

Upon the arrival of the new knowledge domain dataset $\mathfrak{D}_{t+1}$, we consider the posterior $p(\vartheta|\mathfrak{D}_{1:t+1})$ of the model parameters ϑ of a neural network. We emphasise the notational difference between the **meta-parameters** θ and **model parameters** ϑ throughout this thesis. Using Bayes' rule on the posterior gives the recursive formula

$$p(\vartheta|\mathfrak{D}_{1:t+1}) = p(\vartheta|\mathfrak{D}_{t+1}, \mathfrak{D}_{1:t}) = \frac{p(\mathfrak{D}_{t+1}|\vartheta)\, p(\vartheta|\mathfrak{D}_{1:t})}{\int p(\mathfrak{D}_{t+1}|\Theta)\, p(\Theta|\mathfrak{D}_{1:t})\, d\Theta}, \tag{3.3}$$

where Equation (3.3) follows from the assumption that each dataset $\mathfrak{D}_i$ is independent given ϑ. The likelihood $p(\mathfrak{D}_{t+1}|\vartheta)$ is computed based on the newly-arrived dataset $\mathfrak{D}_{t+1}$ only. The prior $p(\vartheta|\mathfrak{D}_{1:t})$ can also be viewed as the previous posterior due to the recursion in Equation (3.3).

The challenge in utilising Equation (3.3) is that the prior $p(\vartheta|\mathfrak{D}_{1:t})$ depends on all the previous datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_t$. Moreover the normalised posterior $p(\vartheta|\mathfrak{D}_{1:t+1})$ is usually intractable due to the relatively large structure of modern neural networks. The key idea of BOL is to approximate the exact posterior $p(\vartheta|\mathfrak{D}_{1:t})$ with a trainable parametric distribution $q(\vartheta|\phi_t)$, where the parameter ϕ_t is trained using $\mathfrak{D}_{1:t}$. Opper (1998) splits the BOL algorithm into the **update** and **projection** steps.

Update step: This step uses the approximate posterior $q(\vartheta|\phi_t)$ obtained from the previous step for an update in the form of Equation (3.3):

$$p(\vartheta|\mathfrak{D}_{t+1}, \phi_t) = \frac{p(\mathfrak{D}_{t+1}|\vartheta)\, q(\vartheta|\phi_t)}{\int p(\mathfrak{D}_{t+1}|\Theta)\, q(\Theta|\phi_t)\, d\Theta}. \tag{3.4}$$

The new posterior $p(\vartheta|\mathfrak{D}_{t+1}, \phi_t)$ from the update step might not belong to the same parametric distribution family as $q(\vartheta|\phi_t)$. In this case, the new posterior has to be projected into the same family to obtain $q(\vartheta|\phi_{t+1})$ in the following step.

Projection step: This step aims to obtain a projection $q(\vartheta|\phi_{t+1})$ that is as close as possible to $p(\vartheta|\mathfrak{D}_{t+1}, \phi_t)$ with minimal loss. Opper (1998) suggests a projection of minimising the KL-divergence between the new posterior and the parametric q. The posterior is typically intractable due to the enormous size of the modern neural network architectures. This leads to the requirement for a good approximation of the posterior of the model parameters. A particularly suitable candidate for this purpose is the Laplace approximation (MacKay, 1992; Ritter et al., 2018b), as it simply adds a quadratic regulariser to the training objective. Variational inference methods such as variational continual learning (Nguyen et al., 2018) is another possible method to obtain an approximation for the posterior of the model parameters. We explain both of these methods

in Chapter 3.3 and Chapter 3.4 respectively.

3.3 Laplace Approximation

Laplace approximation is one of the possible methods for projection step in BOL. We explore for a suitable parametric form to the approximate posterior $q(\vartheta|\phi_t)$. We consider finding a MAP estimate following from Equation (3.3):

$$\vartheta^*_{t+1} = \arg\max_{\vartheta} p(\vartheta|\mathfrak{D}_{1:t+1}) = \arg\max_{\vartheta} \{\log p(\mathfrak{D}_{t+1}|\vartheta) + \log p(\vartheta|\mathfrak{D}_{1:t})\}. \quad (3.5)$$

Since the posterior $p(\vartheta|\mathfrak{D}_{1:t})$ of a neural network is intractable except for small architectures, the unnormalised posterior $\tilde{p}(\vartheta|\mathfrak{D}_{1:t})$ is considered instead. Performing Taylor expansion on the logarithm of the unnormalised posterior around a mode ϑ^*_t gives

$$\log \tilde{p}(\vartheta|\mathfrak{D}_{1:t}) \approx \log \tilde{p}(\vartheta|\mathfrak{D}_{1:t})\big|_{\vartheta=\vartheta^*_t} - \frac{1}{2}(\vartheta - \vartheta^*_t)^T A_t (\vartheta - \vartheta^*_t), \quad (3.6)$$

where A_t denotes the Hessian matrix of the negative log-posterior evaluated at ϑ^*_t with entries

$$A^{ij}_t = -\frac{\partial^2}{\partial\vartheta^{(i)}\partial\vartheta^{(j)}} \log \tilde{p}(\vartheta|\mathfrak{D}_{1:t})\Big|_{\vartheta=\vartheta^*_t}. \quad (3.7)$$

The first order term of the Taylor expansion vanishes since the expansion is performed around a mode, whilst the second order term remains for consideration. The expansion in Equation (3.6) suggests using a Gaussian approximate posterior q. Given a Gaussian $q(\vartheta|\phi_t)$ with parameter $\phi_t = \{\mu_t, \Lambda_t\}$, a mean μ_{t+1} at step $t+1$ with dataset $\mathfrak{D}_{t+1}$ can be obtained by finding a mode of the approximate posterior via a standard gradient-based optimisation $\mu_{t+1} = \arg\min_\vartheta f^{\mathrm{LA}}(\vartheta, \mu_t, \Lambda_t)$ with objective:

$$f^{\mathrm{LA}}(\vartheta, \mu_t, \Lambda_t) = -\log p(\mathfrak{D}_{t+1}|\vartheta) + \frac{1}{2}(\vartheta - \mu_t)^T \Lambda_t (\vartheta - \mu_t). \quad (3.8)$$

The precision matrix is updated as

$$\Lambda_{t+1} = H_{t+1} + \Lambda_t, \tag{3.9}$$

where H_{t+1} is the Hessian matrix of the negative log-likelihood for $\mathfrak{D}_{t+1}$ evaluated at μ_{t+1} with entries

$$H_{t+1}^{ij} = -\frac{\partial^2}{\partial\vartheta^{(i)}\partial\vartheta^{(j)}} \log p(\mathfrak{D}_{t+1}|\vartheta)\Big|_{\vartheta=\mu_{t+1}}. \tag{3.10}$$

Since the full Hessian is intractable for large neural networks, we approximate it with a block-diagonal Kronecker-factored Fisher information matrix as described in Chapter 5.2.

3.3.1 Precision Update Hyperparameter

Ritter et al. (2018a) use a hyperparameter λ as a multiplier to the Hessian when updating the precision in Equation (3.9):

$$\Lambda_{t+1} = \lambda H_{t+1} + \Lambda_t. \tag{3.11}$$

In the large-scale supervised classification setting, this hyperparameter has a regularising effect on the Gaussian posterior approximation for a balance between having a good performance on a new dataset and maintaining the performance on previous datasets (Ritter et al., 2018a).

We observe that

$$\Lambda_{t+1} = \lambda(H_{t+1} + \cdots + H_1) + \Lambda_0 \tag{3.12}$$

upon recursively expanding Equation (3.11). Notably, the original update in Equation (3.9) is recovered by setting $\lambda = 1$. The precision Λ_t regularises the optimisation of ϑ at step $t+1$ associated to the posterior $p(\vartheta|\mathfrak{D}_{1:t+1})$, which is the update process from μ_t to μ_{t+1}. A large λ results in a sharply peaked Gaussian posterior and is therefore unable to learn new datasets well, but can

prevent forgetting previously learned datasets. A small λ in contrast gives a dispersed Gaussian posterior and allows better performance on new datasets by sacrificing the performance on the previous datasets. Chapter 4.7.1 discusses the effect of λ in the experiments as an ablation study.

3.3.2 Algorithm

For a neural network model, gradient-based optimisation methods such as SGD (Robbins and Monro, 1951) and Adam (Kingma and Ba, 2015) are the standard gradient-based methods in finding a mode for the Laplace approximation objective in Equation (3.8).

Algorithm 3 gives the pseudo-code of BOL with Laplace approximation. The algorithm is formed of three main elements: training on a specific dataset (lines 4 – 8), updating the Gaussian mean (line 9) and updating the Gaussian precision (lines 10 – 11). For precision update in line 11, we review the Hessian approximation method later in Chapter 5.2 that utilises the block-diagonal Kronecker-factored Fisher approximation.

Algorithm 3 Bayesian online learning with Laplace approximation

1: **Require:** sequential datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_T$, learning rate α, posterior regulariser λ, number of training epochs J, number of mini-batches M
2: **Initialise:** μ_0, Λ_0, ϑ
3: **for** $t = 1$ **to** T **do**
4: **for** $j = 1, \ldots, J$ **do** ▷ training on $\mathfrak{D}_t$ (eg: Adam or SGD)
5: **for** $m = 1, \ldots, M$ **do**
6: $\vartheta \leftarrow \vartheta - \alpha \nabla_\vartheta f^{\mathrm{LA}}(\vartheta, \mu_{t-1}, \Lambda_{t-1})$
7: **end for**
8: **end for**
9: Update mean $\mu_t \leftarrow \vartheta$ ▷ update posterior mean
10: Approximate H_t with block-diagonal Kronecker-factored Fisher F
11: Update precision $\Lambda_t \leftarrow \lambda H_t + \Lambda_{t-1}$ ▷ update posterior precision
12: **end for**

3.4 Variational Inference

3.4.1 Posterior Approximation

As we mentioned in Chapter 3.2, variational inference also provides a suitable framework for posterior approximation in the projection step of BOL. Consider approximating the posterior q by minimising the KL-divergence between the parametric q and the new posterior $p(\vartheta|\mathfrak{D}_{t+1}, \phi_t)$ in Equation (3.4), where q belongs to some pre-determined approximate posterior family $\mathcal{Q}$ with parameters ϕ_t:

$$q(\vartheta|\phi_{t+1}) = \arg\min_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\vartheta|\phi) \,\|\, p(\mathfrak{D}_{t+1}|\vartheta)\, q(\vartheta|\phi_t)) \tag{3.13}$$

$$= \arg\min_{q\in\mathcal{Q}} \big\{ -\mathbb{E}_{q(\vartheta|\phi)}[\log p(\mathfrak{D}_{t+1}|\vartheta)] + D_{\mathrm{KL}}(q(\vartheta|\phi) \,\|\, q(\vartheta|\phi_t)) \big\}. \tag{3.14}$$

The optimisation in Equation (3.14) leads to minimising the objective $f^{\mathrm{VI}}(\phi, \phi_t)$ with respect to ϕ to obtain the new parameters ϕ_{t+1}:

$$f^{\mathrm{VI}}(\phi, \phi_t) = -\mathbb{E}_{q(\vartheta|\phi)}[\log p(\mathfrak{D}_{t+1}|\vartheta)] + D_{\mathrm{KL}}(q(\vartheta|\phi) \,\|\, q(\vartheta|\phi_t)). \tag{3.15}$$

One can use a Gaussian mean-field approximate posterior

$$q(\vartheta|\phi_t) = \prod_{d=1}^{D} N(\mu_{t,d}, \sigma_{t,d}^2), \tag{3.16}$$

where $\phi_t = \{\mu_{t,d}, \sigma_{t,d}\}_{d=1}^{D}$ and $D = \dim(\vartheta)$. The first term in Equation (3.15) can be estimated via simple Monte Carlo with local reparameterisation trick (Kingma et al., 2015). The KL-divergence term in Equation (3.15) has a closed form for Gaussian distributions.

3.4.2 Algorithm

Algorithm 4 gives the pseudo-code of BOL with variational inference. The algorithm is formed of two main elements: training on a specific dataset

(lines 4 – 8) and updating the posterior parameters of the Gaussian mean-field approximation (line 9).

Algorithm 4 Bayesian online learning with variational inference

1: **Require:** sequential datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_T$, learning rate α, number of training epochs J, number of mini-batches M
2: **Initialise:** $\phi_0 = \{\mu_0, \sigma_0\}$
3: **for** $t = 1$ **to** T **do**
4: **for** $j = 1, \ldots, J$ **do** ▷ training on $\mathfrak{D}_t$ (eg: Adam or SGD)
5: **for** $m = 1$ **to** M **do**
6: $\phi \leftarrow \phi - \alpha \nabla_\phi f^{\mathrm{VI}}(\phi, \phi_{t-1})$
7: **end for**
8: **end for**
9: $\mu_t \leftarrow \mu$, $\sigma_t \leftarrow \sigma$ with $\phi = \{\mu, \sigma\}$ ▷ update posterior parameters
10: **end for**

Variational Continual Learning (VCL) (Nguyen et al., 2018) implements BOL with variational inference in large-scale machine learning settings. VCL additionally boosts the performance by retaining important information from the previous datasets in the memory. Representative data points from each previous dataset are kept in a *coreset*, and the coreset is updated when a new dataset arrives. The coreset is used in training as a memory replay mechanism, with the objective of outrunning the posterior approximation error accumulated over each dataset.

Chapter 4

Bayesian Online Meta-Learning

The central contribution of this chapter is to extend the benefits of meta-learning to the BOL scenario, thereby training models that can generalise across knowledge domains whilst dealing with parameter uncertainty in the setting of sequentially arriving datasets.

4.1 Framework Overview

In this setting, meta-training occurs sequentially on the datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_T$. Each dataset $\mathfrak{D}_i$ can be seen as a knowledge domain with an associated underlying task distribution $p(\mathcal{T}_i)$. A newly-arrived $\mathfrak{D}_{t+1}$ is separated into the base class set $\mathcal{D}_{t+1}$ and novel class set $\widehat{\mathcal{D}}_{t+1}$ for meta-training and meta-evaluation respectively, where the tasks in these two stages are drawn from the task distribution $p(\mathcal{T}_{t+1})$. Notationally, let $\mathcal{D}^S_{t+1}$ and $\mathcal{D}^Q_{t+1}$ denote the collection of support sets and query sets respectively from $\mathcal{D}_{t+1}$, so that $\mathcal{D}_{t+1} = \mathcal{D}^S_{t+1} \cup \mathcal{D}^Q_{t+1}$. Using Bayes' rule on the posterior gives the recursive formula

$$
\begin{aligned}
p(\theta|\mathcal{D}_{1:t+1}) &\propto p(\mathcal{D}^S_{t+1}, \mathcal{D}^Q_{t+1}|\theta)\, p(\theta|\mathcal{D}_{1:t}) && (4.1)\\
&= p(\mathcal{D}^Q_{t+1}|\theta, \mathcal{D}^S_{t+1})\, p(\mathcal{D}^S_{t+1}|\theta)\, p(\theta|\mathcal{D}_{1:t}) && (4.2)\\
&= \left\{ \int p(\mathcal{D}^Q_{t+1}|\tilde{\theta})\, p(\tilde{\theta}|\theta, \mathcal{D}^S_{t+1})\, d\tilde{\theta} \right\} p(\mathcal{D}^S_{t+1}|\theta)\, p(\theta|\mathcal{D}_{1:t}) && (4.3)
\end{aligned}
$$

where Equation (4.1) follows from the assumption that each dataset is independent given θ. Figure 4.1 illustrates the BOML process flow for meta-training

and meta-evaluation as datasets arrive sequentially.

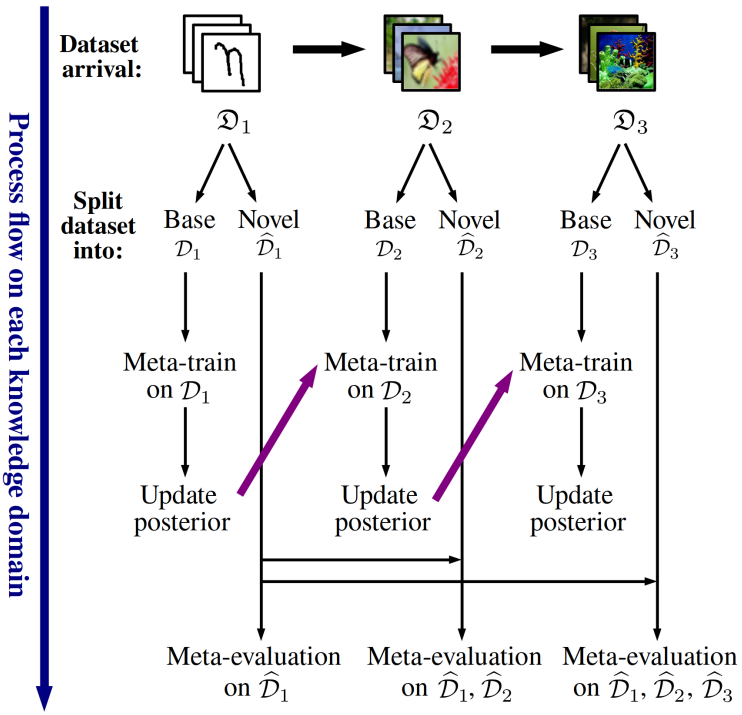

Figure 4.1: The BOML process flow for meta-training and meta-evaluation on an example sequence (Omniglot → CIFAR-FS → *mini*ImageNet) when each dataset arrives. Consider the CIFAR-FS knowledge domain $\mathfrak{D}_2$ for instance, $\mathfrak{D}_2$ is split into the base $\mathcal{D}_2$ and novel $\widehat{\mathcal{D}}_2$ sets when it arrives. Meta-training on this knowledge domain only occurs on the base set $\mathcal{D}_2$ using the recursive formula in Equation (4.3). The arrows in purple illustrate that the updated posterior is being brought forward for the next meta-training when a new dataset arrives. We meta-evaluate the few-shot performance on the accumulated novel sets $\widehat{\mathcal{D}}_1$, $\widehat{\mathcal{D}}_2$ from the knowledge domains that arrived so far.

From the meta-learning perspective, the parameters $\tilde{\theta}$ introduced in Equation (4.3) can be viewed as the task-specific parameters in MAML. There are various choices for the distribution $p(\tilde{\theta}|\theta, \mathcal{D}^S_{t+1})$ in Equation (4.3). In particular if we choose to set it as the deterministic function of taking several steps of SGD on loss $\mathcal{L}$ with the support set collection $\mathcal{D}^S_{t+1}$ and initialised at θ, we have

$$p(\tilde{\theta}|\theta, \mathcal{D}^S_{t+1}) = \delta(\tilde{\theta} - SGD_k(\mathcal{L}(\theta, \mathcal{D}^S_{t+1}))), \tag{4.4}$$

where $\delta(\cdot)$ is the Dirac delta function. This recovers the MAML inner loop with SGD quick adaptation in Equation (3.1). The recursion given by Equation (4.3)

forms the basis of our approach and the remainder of this chapter explains how we implement this.

The posterior in Equation (4.3) is typically intractable for modern neural network architectures. This leads to the requirement for a good approximate posterior. Chapters 4.2 and 4.3 demonstrate how we arrive at the algorithms Bayesian online meta-learning with Laplace approximation (BOMLA) and Bayesian online meta-learning with variational inference (BOMVI) by implementing Laplace approximation and variational inference respectively to the BOML posterior in Equation (4.3).

4.2 BOML with Laplace Approximation

4.2.1 Derivation and Implementation

As described in Chapter 3.3, the expansion in Equation (3.6) justifies the use of a Gaussian approximate posterior since the second order term corresponds to the log-probability of a Gaussian distribution. The BOML framework in Equation (4.3) with a Gaussian approximate posterior q of mean and precision $\phi_t = \{\mu_t, \Lambda_t\}$ from Laplace approximation gives a MAP estimate:

$$\theta^* = \arg\max_{\theta} \left\{ \log \bar{p}_\theta + \log p(\mathcal{D}_{t+1}^S|\theta) - \frac{1}{2}(\theta - \mu_t)^T \Lambda_t (\theta - \mu_t) \right\} \tag{4.5}$$

with

$$\bar{p}_\theta = \int p(\mathcal{D}_{t+1}^Q|\tilde{\theta}) p(\tilde{\theta}|\theta, \mathcal{D}_{t+1}^S)\, d\tilde{\theta}.$$

For an efficient optimisation, we use the deterministic $\tilde{\theta}$ in Equation (4.4) which leads to minimising the objective

$$f_{t+1}^{\text{BOMLA}}(\theta, \mu_t, \Lambda_t) = \bar{f}_\theta^{(1)} + \bar{f}_\theta^{(2)} + \frac{1}{2}(\theta - \mu_t)^T \Lambda_t (\theta - \mu_t), \tag{4.6}$$

where

$$\bar{f}_\theta^{(1)} = -\frac{1}{M} \sum_{m=1}^{M} \log p(\mathcal{D}_{t+1}^{m,Q}|\tilde{\theta}^m) \quad \text{and} \quad \bar{f}_\theta^{(2)} = -\frac{1}{M} \sum_{m=1}^{M} \log p(\mathcal{D}_{t+1}^{m,S}|\theta),$$

with $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_{t+1}^{m,S}))$ for $m = 1, \ldots, M$ and M denotes the number of tasks sampled per iteration. The first term $\bar{f}_\theta^{(1)}$ in Equation (4.6) corresponds to the MAML objective in Equation (3.2) with a cross-entropy loss, the second term $\bar{f}_\theta^{(2)}$ can be viewed as the pre-adaptation loss on the support set and the last term can be seen as a regulariser.

The new approximate posterior has mean $\mu_{t+1} = \arg\min_\theta f_{t+1}^{\text{BOMLA}}(\theta, \mu_t, \Lambda_t)$, and the precision matrix is updated as

$$\Lambda_{t+1} = \lambda(\widetilde{H}_{t+1} + H_{t+1}) + \Lambda_t, \tag{4.7}$$

where λ is the posterior-regularising hyperparameter introduced in Chapter 3.3.1, $\widetilde{H}_{t+1}$ and H_{t+1} are the Hessian matrices of the negative log-likelihood for the query and support set respectively with entries

$$\widetilde{H}_{t+1}^{ij} = \frac{1}{M} \sum_{m=1}^{M} -\frac{\partial^2}{\partial\theta^{(i)}\partial\theta^{(j)}} \log p(\mathcal{D}_{t+1}^{m,Q}|\tilde{\theta}^m))\Big|_{\theta=\mu_{t+1}}, \tag{4.8}$$

$$H_{t+1}^{ij} = \frac{1}{M} \sum_{m=1}^{M} -\frac{\partial^2}{\partial\theta^{(i)}\partial\theta^{(j)}} \log p(\mathcal{D}_{t+1}^{m,S}|\theta)\Big|_{\theta=\mu_{t+1}}. \tag{4.9}$$

The full Hessian matrices $\widetilde{H}_{t+1}$ and H_{t+1} are intractable for a neural network with larger architecture. We utilise a block-diagonal Kronecker-factored Fisher approximation method to estimate the Hessian matrices. The Hessian H_{t+1} in Equation (4.9) can be approximated in the same manner as the Hessian in Equation (3.10), and we explain the approximation of H_{t+1} in Chapter 5.3.1. The Hessian $\widetilde{H}_{t+1}$ requires a special treatment in addition to the original approximation method in the large-scale classification setting. We derive the approximation to $\widetilde{H}_{t+1}$ in Chapter 5.3.2.

We discover that the Laplace approximation method provides a well-fitted meta-training framework for BOML in Equation (4.3). Each updating step in the approximation procedure can be modified to correspond to the meta-parameters θ for few-shot classification, instead of the model parameters ϑ for large-scale supervised classification.

4.2.2 Algorithm

Algorithm 5 gives the pseudo-code of the BOMLA algorithm. The algorithm is formed of three main elements: meta-training on a specific base set (lines 4 – 11), updating the Gaussian mean (line 12) and updating the Gaussian precision (lines 13 – 14). For precision update, we approximate the Hessian using a block-diagonal Kronecker-factored Fisher approximation explained in Chapter 5.

Algorithm 5 Bayesian online meta-learning with Laplace approximation (BOMLA)

1: **Require:** sequential base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$, learning rate α, hyperparameter λ, number of meta-training iterations J, meta-batch size M
2: **Initialise:** μ_0, Λ_0, θ
3: **for** $t = 1$ **to** T **do**
4: **for** $i = 1, \ldots, J$ **do** ▷ meta-training on base set $\mathcal{D}_t$
5: **for** $m = 1$ **to** M **do**
6: Sample task $\mathcal{D}_t^m = \mathcal{D}_t^{m,S} \cup \mathcal{D}_t^{m,Q}$
7: Inner update $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_t^{m,S}))$
8: **end for**
9: Evaluate loss $f_t^{\text{BOMLA}}(\theta, \mu_{t-1}, \Lambda_{t-1})$ in Equation (4.6)
10: Outer update $\theta \leftarrow \theta - \alpha \nabla_\theta f_t^{\text{BOMLA}}(\theta, \mu_{t-1}, \Lambda_{t-1})$
11: **end for**
12: Update mean $\mu_t \leftarrow \theta$ ▷ posterior mean
13: Approximate H_t and $\widetilde{H}_t$ using Algorithm 9
14: Update precision $\Lambda_t \leftarrow \lambda\big(\widetilde{H}_t + H_t\big) + \Lambda_{t-1}$ ▷ posterior precision
15: **end for**

4.3 BOML with Variational Inference

This section demonstrates how we arrive at the BOMVI algorithm by implementing an approximate variational inference method to the BOML posterior in Equation (4.3). We proceed in a similar fashion to the BOL framework with variational inference in Chapter 3.4.

4.3.1 Derivation and Implementation

Consider approximating the posterior by minimising the KL-divergence between the approximate posterior q and the BOML posterior in Equation (4.3), where q belongs to some pre-determined approximate posterior family $\mathcal{Q}$ with

parameters ϕ_t. This gives a new approximate posterior

$$q(\theta|\phi_{t+1}) = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}\big(q(\theta|\phi) \,\big\|\, \breve{q}_{\phi_t}\big), \tag{4.10}$$

where

$$\breve{q}_{\phi_t} = \left\{ \int p(\mathcal{D}_{t+1}^{Q}|\tilde{\theta})\, p(\tilde{\theta}|\theta, \mathcal{D}_{t+1}^{S}) d\tilde{\theta} \right\} p(\mathcal{D}_{t+1}^{S}|\theta)\, q(\theta|\phi_t).$$

Similar to BOMLA, we use the deterministic $\tilde{\theta}$ in Equation (4.4). This leads to minimising the objective

$$f_{t+1}^{\text{BOMVI}}(\phi, \phi_t) = \breve{f}_{\phi}^{(1)} + \breve{f}_{\phi}^{(2)} + D_{\mathrm{KL}}(q(\theta|\phi) \| q(\theta|\phi_t)), \tag{4.11}$$

where

$$\breve{f}_{\phi}^{(1)} = -\frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{q(\theta|\phi)} \big[\log p(\mathcal{D}_{t+1}^{m,Q}|\tilde{\theta}^m) \big],$$
$$\breve{f}_{\phi}^{(2)} = -\frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{q(\theta|\phi)} \big[\log p(\mathcal{D}_{t+1}^{m,S}|\theta) \big],$$

with $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_{t+1}^{m,S}))$ for $m = 1, \ldots, M$ and M denotes the number of tasks sampled per iteration. We use a Gaussian mean-field approximation $q(\theta|\phi_t) = \prod_{d=1}^{D} N(\mu_{t,d}, \sigma_{t,d}^2)$, where $\phi_t = \{\mu_{t,d}, \sigma_{t,d}\}_{d=1}^{D}$, $D = \dim(\theta)$ and the objective in Equation (4.11) is minimised over ϕ. Using a Gaussian posterior results in a closed-form KL-divergence in Equation (4.11).

The term $\breve{f}_{\phi}^{(1)}$ in Equation (4.11) is rather cumbersome to estimate in optimisation. To compute its Monte Carlo estimate, we have to generate samples $\theta_r \sim q$ for $r = 1, \ldots, R$, and run a quick adaptation on each sampled meta-parameters θ_r before evaluating their log-likelihoods. This is computationally intensive and the estimator is prone to a large variance. Moreover, every quickly-adapted sample from θ_r contributes to the meta-learning gradients of the posterior mean and covariance, resulting in a high computational cost when taking the meta-gradients.

To solve these impediments, we introduce a slight modification to the

SGD quick adaptation $\widetilde{\theta}^m$. Instead of taking the gradients with respect to the sampled meta-parameters, we consider the gradients with respect to the *posterior mean*. A one-step SGD quick adaptation, for instance, becomes:

$$\widetilde{\theta}^m = \theta - \alpha \nabla_{\mu_t} \mathcal{L}(\mu_t, \mathcal{D}_{t+1}^{m,S}). \tag{4.12}$$

This gives $\widetilde{\theta}^m \sim N(\widetilde{\mu}_t, \text{diag}(\sigma_t^2))$ where

$$\widetilde{\mu}_t = \mu_t - \alpha \nabla_{\mu_t} \mathcal{L}(\mu_t, \mathcal{D}_{t+1}^{m,S}), \tag{4.13}$$

since $\theta \sim N(\mu_t, \text{diag}(\sigma_t^2))$. A quick adaptation with more steps works in a similar fashion. With this modification, we can calculate the Monte Carlo estimate for the term $\breve{f}_\phi^{(1)}$ in Equation (4.11) using the local reparameterisation trick as usual.

4.3.2 Algorithm

Algorithm 6 gives the pseudo-code of the BOMVI algorithm. The algorithm is formed of two main elements: meta-training on a specific base set (line 4 – 11) and updating the posterior parameters of the Gaussian mean-field approximation (line 12).

Algorithm 6 Bayesian online meta-learning with variational inference (BOMVI)

1: **Require:** sequential base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$, learning rate α, number of meta-training iterations J, meta-batch size M
2: **Initialise:** $\phi_0 = \{\mu_0, \sigma_0\}$
3: **for** $t = 1$ **to** T **do**
4: **for** $i = 1, \ldots, J$ **do** ▷ meta-training on base set $\mathcal{D}_t$
5: **for** $m = 1$ **to** M **do**
6: Sample task $\mathcal{D}_t^m = \mathcal{D}_t^{m,S} \cup \mathcal{D}_t^{m,Q}$
7: Inner update $\widetilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_t^{m,S}))$
8: **end for**
9: Evaluate loss $f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$ in Equation (4.11)
10: Outer $\mu \leftarrow \mu - \alpha \nabla_\mu f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$, $\sigma \leftarrow \sigma - \alpha \nabla_\sigma f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$
11: **end for**
12: Update $\mu_t \leftarrow \mu$ and $\sigma_t \leftarrow \sigma$ ▷ update posterior parameters
13: **end for**

4.4 Experiments

4.4.1 Setup

Model structure: For all the experiments in this chapter, we employ the model architecture proposed by Vinyals et al. (2016) that consists of 4 modules with 64 filters of size 3×3, followed by a batch normalisation, a ReLU activation and a 2×2 max-pooling. A fully-connected layer is appended to the final module before getting the class probabilities with softmax. Tables A.1 and A.2 in Appendix A.1 record the hyperparameters used in the experiments.

Datasets: The following are the datasets involved and their formation in the experiments.

1. **Omniglot**

   Omniglot (Lake et al., 2011) comprises 1623 characters from 50 alphabets and each character has 20 instances. We use 1100 characters for meta-training, 100 characters for validation and the remaining for meta-evaluation. New classes with rotations in the multiples of 90° are formed after splitting the characters as mentioned. The Omniglot dataset is also used in Chapter 4.5 for another experiment with a different setup, which we explain thereafter in that chapter.

2. ***mini*QuickDraw**

   QuickDraw (Ha and Eck, 2017) comprises 345 categories of drawings collected from the players in the game "Quick, Draw!". We generate *mini*QuickDraw by randomly sampling 1000 instances in each class of QuickDraw.

3. **CIFAR-FS**

   CIFAR-FS (Bertinetto et al., 2019) has 100 classes of objects and each class comprises 600 images. We use the same split as Bertinetto et al. (2019): 64 classes for meta-training, 16 classes for validation and 20 classes for meta-evaluation.

4. ***mini*ImageNet**

   *mini*ImageNet (Vinyals et al., 2016) takes 100 classes and 600 instances in each class from the ImageNet dataset. We use the same split as Ravi and Larochelle (2017): 64 classes for meta-training, 16 classes for validation and 20 classes for meta-evaluation.

5. **VGG-Flowers**

   VGG-Flowers (Nilsback and Zisserman, 2008) comprises 102 different types of flowers as the classes. We randomly split 66 classes for meta-training, 16 classes for validation and 20 classes for meta-evaluation.

6. **Aircraft**

   Aircraft (Maji et al., 2013) is a fine-grained dataset consisting of 100 aircraft models as the classes and each class has 100 images. We randomly split 64 classes for meta-training, 16 classes for validation and 20 classes for meta-evaluation.

Baselines: We compare our algorithms to the following baselines.

1. **Train-On-Everything (TOE)**

   When a new dataset arrives for meta-training, we randomly re-initialise the meta-parameters and run MAML meta-training using *all datasets encountered so far*. Each inner loop update samples a specific number of tasks from each dataset available. The inner losses of the tasks from all datasets are aggregated for an outer loop update.

2. **Sequential MAML**

   Upon the arrival of a new dataset, we run MAML to meta-train *only on the newly-arrived dataset*. We should be reminded that MAML is only capable of handling few-shot tasks from the same underlying task distribution by design.

3. **Follow The Meta-Leader (FTML)**

   We introduce a slight modification to FTML (Finn et al., 2019) on its

evaluation method, as FTML is not designed for few-shot learning on unseen tasks. FTML comprises two types of data accumulation during the training phase: a task buffer and a dataset buffer. Both buffers are initially empty. The task buffer is for the sequential accumulation of tasks as they arrive, whereas the dataset buffer is for the incremental data accumulation within a given task. As a new task arrives for training, it is first added to the task buffer. The datapoints within the task are also added to the dataset buffer as they arrive in an online manner for the training of the meta-parameters. This optimisation requires sampling from the task buffer across all tasks encountered so far. Upon the arrival of all data in the current task, FTML moves on to the subsequent task for training with a re-initialised dataset buffer.

The evaluation phase involves a quick adaptation of meta-parameters to the current task, which is known as the `Update-Procedure` in FTML. The `Update-Procedure` utilises all data accumulated in the dataset buffer for quick adaptation to the current task. The evaluation performance is recorded using a held-out test set from the same task. This differs from our BOML setting that evaluates on new and unseen tasks. In our experiments, we apply `Update-Procedure` in FTML to the data from unseen tasks, rather than the data from the same training task as in the original FTML.

4.4.2 Triathlon

We implement BOMLA and BOMVI to the 5-way 1-shot **triathlon** sequences. This experiment considers the few-shot triathlon sequence as in Figure 4.2.

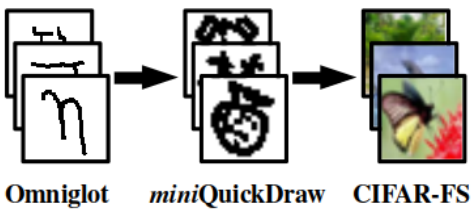

Figure 4.2: The triathlon 5-way 1-shot sequence in this experiment.

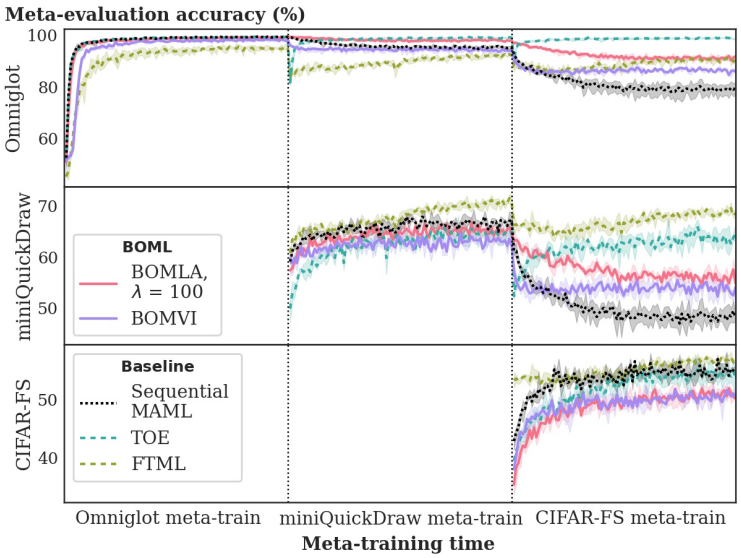

Figure 4.3: Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Going from left to right on the x-axis of the figure is the meta-training times of the knowledge domain datasets that arrive in sequential order. The second row in the figure, for instance, corresponds to the *mini*QuickDraw knowledge domain. The first plot in the second row is empty since *mini*QuickDraw has not arrived during the meta-training time of Omniglot. The diagonal plot (middle plot) in the second row corresponds to the meta-evaluation accuracy on the novel set of *mini*QuickDraw when meta-training occurs on the base set of *mini*QuickDraw. The off-diagonal plot (last plot) shows the meta-evaluation on the *mini*QuickDraw novel set, when meta-training occurs on the next knowledge domain CIFAR-FS. Higher accuracy values in the off-diagonals indicate less forgetting. The baseline TOE corresponds to an upper limit in the performance since it has access to all datasets encountered so far. Sequential MAML corresponds to a lower limit in the performance since MAML forgets on previous datasets by design of the algorithm.

The distributional shift from Omniglot to *mini*QuickDraw is less drastic, compared to the shift from *mini*QuickDraw to CIFAR-FS. Omniglot and *mini*QuickDraw are both gray-scale and the image drawings are formed of simple strokes in both datasets. The result in Figure 4.3 shows that BOMLA and BOMVI are able to prevent catastrophic forgetting in both dataset transitions. BOMLA, in particular, is able to proceed to the *mini*QuickDraw meta-training phase with **almost no forgetting on Omniglot**. In other words, the meta-level pattern of Omniglot is retained throughout the meta-training period of *mini*QuickDraw. There is a small trade-off in the performance of CIFAR-

FS as BOMLA and BOMVI avoid catastrophically forgetting Omniglot and *mini*QuickDraw.

Since this experiment focuses on 5-way classification tasks, the baseline accuracy of random guessing stands at 20%. It is noteworthy that the result in Figure 4.3 is obtained via 1-shot learning, which traditionally presents a significant challenge for achieving high accuracy in classification problems. The BOML implementations surpass the baseline accuracy of random guessing by a substantial margin, and pose a remarkable achievement in the sequential few-shot problems setting.

Sequential MAML gives a noticeable drop in the performance of Omniglot and *mini*QuickDraw when meta-training on CIFAR-FS. TOE is able to retain the few-shot performance as it has access to all previous datasets, whilst FTML gives a mixed performance. We elaborate on the result interpretation, the BOMLA-BOMVI comparison and the choice of λ along with the next experiment pentathlon, which resembles the setting of this experiment except with a more challenging dataset sequence.

4.4.3 Pentathlon

We implement BOMLA and BOMVI to the more challenging pentathlon sequence as in Figure 4.4.

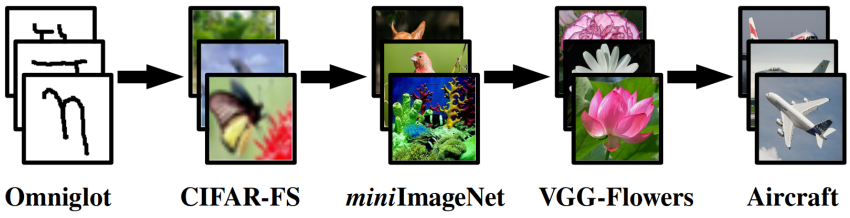

Figure 4.4: The pentathlon 5-way 1-shot sequence in this experiment.

Figure 4.5 shows that BOMLA and BOMVI are able to prevent few-shot catastrophic forgetting in the pentathlon dataset sequence. TOE is also able to retain the few-shot performance as it has access to all datasets encountered so far. Since TOE learns all datasets from random re-initialisation each time it encounters a new dataset, the meta-training time required to achieve a similarly

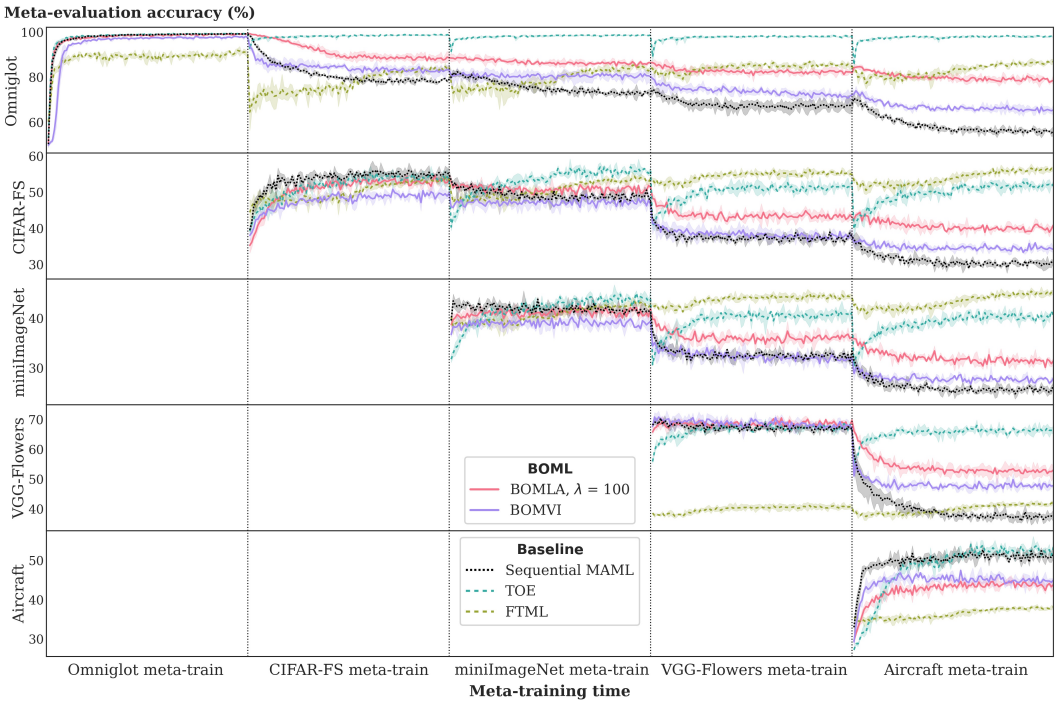

Figure 4.5: Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Going from left to right on the x-axis of the figure is the meta-training times of the knowledge domain datasets that arrive in sequential order. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BOMLA with $\lambda = 100$ gives good performance in the off-diagonal plots (retains performances on previously learned datasets), and has a minor performance trade-off in the diagonal plots (learns less well on new datasets). The second row in the figure, for instance, corresponds to the CIFAR-FS knowledge domain. The first plot in the second row is empty since CIFAR-FS has not arrived during the meta-training time of Omniglot. The diagonal plot (second plot) in the second row shows the meta-evaluation accuracy on the novel set of CIFAR-FS when meta-training occurs on the base set of CIFAR-FS. The off-diagonal plots (last three plots) in the second row show the meta-evaluation on the CIFAR-FS novel set, when meta-training occurs on the subsequent knowledge domains *mini*ImageNet, VGG-Flowers and Aircraft. Sequential MAML gives better performance in the diagonal plots (learns well on new datasets) but worse performance in the off-diagonal plots (forgets previously learned datasets). BOMVI is also able to retain performance on previous datasets, although it may be unable to perform as good as BOMLA due to sampling and estimator variance.

good meta-evaluation performance is longer compared to other runs. Sequential MAML catastrophically forgets the previously learned datasets but has the best performance on new datasets compared to other runs. FTML gives a

mixed performance on different datasets.

The baselines TOE and FTML can be memory-intensive as the dataset sequence becomes longer. They take the brute-force approach to prevent forgetting by memorising all datasets. Unlike TOE and FTML, our algorithms BomLA and BomVI only take the newly-arrived dataset and the posterior of the meta-parameters into consideration during optimisation. This gives a framework with an algorithmic complexity independent of the length of the dataset sequence.

Choosing λ: Tuning the posterior-regularising hyperparameter λ for precision update in Equation (4.7) corresponds to balancing between a smaller performance trade-off on a new dataset and less forgetting on previous datasets. We compare BomLA with different λ values and BomVI in Chapter 4.7.1 as an ablation study.

BomLA-BomVI comparison: As shown in Figure 4.5, BomLA with appropriate λ is superior to BomVI in the performance. This is due to BomLA having a better posterior approximation than BomVI. Whilst BomLA has a Gaussian approximate posterior with block-diagonal precision, BomVI uses a Gaussian mean-field approximation for the posterior. Trippe and Turner (2017) compare the performances of variational inference with different covariance structures, and discover that variational inference with block-diagonal covariance performs worse than mean-field approximation. This is because the block-diagonal covariance in variational inference prohibits variance reduction methods such as local reparameterisation trick for Monte Carlo estimation. We therefore implement BomVI using the simplest diagonal covariance structure in order to maintain the sampling and estimator variance at an acceptable level.

The variance of the Monte Carlo estimate has been proven problematic (Kingma et al., 2015; Trippe and Turner, 2017), and we addressed this issue in Chapter 4.3. As an ablation study, we analyse the change in the approximate posterior covariance in Chapter 4.7.2 whilst meta-training occurs sequentially

on datasets from different knowledge domains.

4.5 BOML in Sequential Task Setting

We demonstrate empirically that BOML can also continually learn to few-shot classify the novel classes in the sequential few-shot tasks setting, where all tasks originate from a single stationary task distribution.

4.5.1 Setting and Algorithm

This setting only involves **one dataset** $\mathfrak{D}$ with an associated underlying task distribution $p(\mathcal{T})$, where $\mathfrak{D}$ is separated into the base and novel class sets. In this setting, $\mathcal{D}_1, \ldots, \mathcal{D}_{t+1}$ denote the non-overlapping tasks formed from the base class set and they arrive sequentially for meta-training.

When a task $\mathcal{D}_t$ arrives, we break the data points of $\mathcal{D}_t$ into mini-batches. Each mini-batch is further split into the support and query sets for inner and outer loop updates. Algorithms 7 and 8 show the corresponding modifications of BOMLA and BOMVI under this setting in blue.

Algorithm 7 BOMLA for stationary task distribution

Require: sequential tasks $\mathcal{D}_1, \ldots, \mathcal{D}_T$, learning rate α, posterior regulariser λ, number of epochs J, number of mini-batches M
Initialise: μ_0, Λ_0, θ
for $t = 1$ **to** T **do**
  for $i = 1, \ldots, J$ **do** ▷ meta-training on task $\mathcal{D}_t$
    for $m = 1$ **to** M **do**
      Split the batch $\mathcal{D}_t^m = \mathcal{D}_t^{m,S} \cup \mathcal{D}_t^{m,Q}$
      Inner update $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_t^{m,S}))$
    end for
    Evaluate loss $f_t^{\text{BOMLA}}(\theta, \mu_{t-1}, \Lambda_{t-1})$ in Equation (4.6)
    Outer update $\theta \leftarrow \theta - \alpha \nabla_\theta f_t^{\text{BOMLA}}(\theta, \mu_{t-1}, \Lambda_{t-1})$
  end for
  Update mean $\mu_t \leftarrow \theta$ ▷ posterior mean
  Approximate H_t and $\widetilde{H}_t$ using Algorithm 9
  Update precision $\Lambda_t \leftarrow \lambda\big(\widetilde{H}_t + H_t\big) + \Lambda_{t-1}$ ▷ posterior precision
end for

Algorithm 8 BOMVI for stationary task distribution

Require: sequential tasks $\mathcal{D}_1, \ldots, \mathcal{D}_T$, learning rate α, number of epochs J, number of mini-batches M
Initialise: $\phi_0 = \{\mu_0, \sigma_0\}$
for $t = 1$ **to** T **do**
  for $i = 1, \ldots, J$ **do** ▷ meta-training on task $\mathcal{D}_t$
    for $m = 1$ **to** M **do**
      Split the batch $\mathcal{D}_t^m = \mathcal{D}_t^{m,S} \cup \mathcal{D}_t^{m,Q}$
      Inner update $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_t^{m,S}))$
    end for
    Evaluate loss $f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$ in Equation (4.11)
    Outer $\mu \leftarrow \mu - \alpha \nabla_\mu f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$, $\sigma \leftarrow \sigma - \alpha \nabla_\sigma f_t^{\text{BOMVI}}(\phi, \phi_{t-1})$
  end for
  Update $\mu_t \leftarrow \mu$ and $\sigma_t \leftarrow \sigma$ ▷ update posterior parameters
end for

4.5.2 Omniglot: Stationary Task Distribution

We run the sequential tasks experiment on the Omniglot dataset. To increase the difficulty level, we split the dataset based on the alphabets (super-classes) instead of the characters (classes) as in Figure 4.6. The goal of this experiment is to classify the 5-way 5-shot novel tasks sampled from the meta-evaluation alphabets. Table A.3 in Appendix A.2 shows the hyperparameters used in this experiment. We explain the model structure and alphabet splits in this experiment before proceeding to the results.

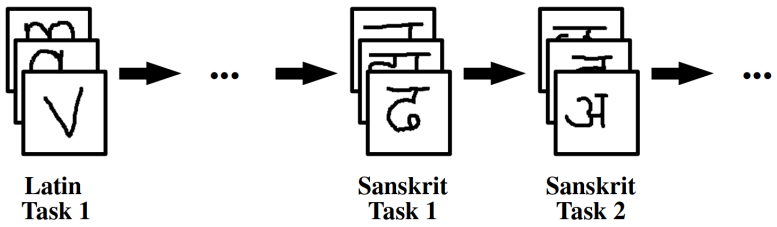

Figure 4.6: An example of the Omniglot task sequence for meta-training in this experiment.

Model structure: We use the model architecture proposed by Vinyals et al. (2016) that takes 4 modules with 64 filters of size 3×3, followed by a batch normalisation, a ReLU activation and a 2×2 max-pooling. A fully-connected layer is appended to the final module before getting the class probabilities with softmax.

Alphabet split: The Omniglot dataset comprises 50 alphabets (super-classes). Each alphabet has numerous characters (classes) and each character has 20 instances. As the meta-training alphabets arrive sequentially, we form **non-overlapping** sequential tasks from each arriving alphabet, and the tasks also do not overlap in the characters. We use 35 alphabets for meta-training, 7 alphabets for validation and 8 alphabets for meta-evaluation. The alphabet splits are as follows:

1. **35 alphabets for meta-training**

   `Alphabet_of_the_Magi, Angelic, Armenian, Atlantean, Avesta, Asomtavruli_(Georgian), Aurek-Besh, Balinese, Bengali, Braille, Burmese_(Myanmar), Early_Aramaic, Grantha, Gujarati, Gurmukhi, Hebrew, Japanese_(hiragana), Inuktitut_(Canadian_Aboriginal_Syllabics), Kannada, Keble, Japanese_(katakana), Korean, Latin, Malayalam, Manipuri, Malay_(Jawi_-_Arabic), Mongolian, Oriya, Sanskrit, Sylheti, Ojibwe_(Canadian_Aboriginal_Syllabics), Tengwar, Tifinagh, Old_Church_Slavonic_(Cyrillic), ULOG`

2. **7 alphabets for validation**

   `Anglo-Saxon_Futhorc, Arcadian, Cyrillic, Ge_ez, Glagolitic, N_Ko, Blackfoot_(Canadian_Aboriginal_Syllabics)`

3. **8 alphabets for meta-evaluation**

   `Atemayar_Qelisayer, Futurama, Greek, Mkhedruli_(Georgian), Syriac_(Estrangelo), Syriac_(Serto), Tagalog, Tibetan`

4.5.3 Results

We compare our algorithms to the baselines **TOE**, **Sequential MAML** and **FTML** similar to the triathlon and pentathlon experiments but in the sequential tasks setting. Figure 4.7 shows that BOMLA and BOMVI can accumulate

few-shot classification ability on the novel tasks over time, as the tasks arrive sequentially for meta-training. The knowledge acquired from previous meta-training tasks is carried forward in the form of a posterior, which is then used as a prior when a new task arrives for meta-training. Despite having access to all previous tasks, TOE shows no positive forward transfer in the meta-evaluation accuracy each time it encounters a new task. FTML and sequential MAML are inferior to BOMLA and BOMVI in the performance. BOMLA with $\lambda = 0.01$ gives the best performance in this experiment.

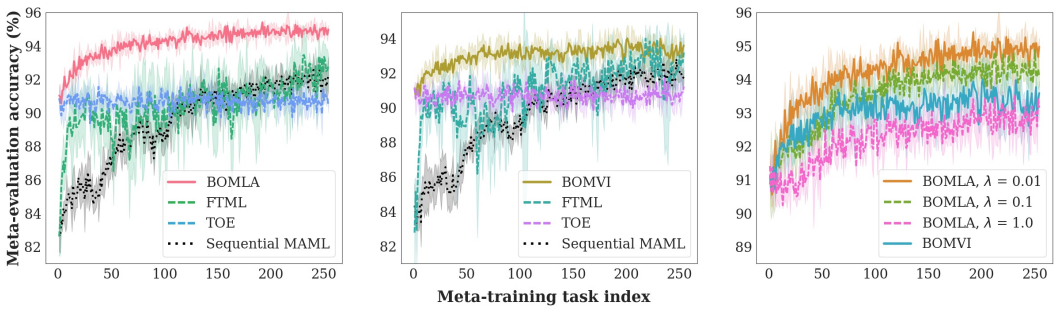

Figure 4.7: Meta-evaluation accuracy across 3 seed runs on the novel tasks along meta-training. **Left**: compares BOMLA to the baselines, **centre**: compares BOMVI to the baselines, **right**: compares BOMLA with different λ values to BOMVI.

4.6 Discussion

We notice that the optimal value of λ varies considerably across different experimental settings. The optimal λ value for the triathlon and pentathlon experiments in Chapter 4.4 is determined to be $\lambda = 100$. In contrast, the optimal λ for the Omniglot sequential task experiment in Chapter 4.5 is discovered to be $\lambda = 0.01$. The substantial distinction in the optimal λ values between these experimental settings is believed to stem from the diversity of the knowledge domains in different settings. The triathlon and pentathlon experiments consider knowledge domains from a broad spectrum, whereas the sequential task experiment is confined to tasks within the Omniglot knowledge domain. This distinction in the variation of the knowledge domains is conjectured as the primary factor contributing to the observed variation in the optimal λ values

across these experimental setups.

On a brief inspection, we observe that the arrangement order of the knowledge domain datasets does not significantly impact the performance of the BOML framework. This is further validated by the Omniglot sequential task experiment, since the task sequence is randomly generated in this setting. The sequences of both the alphabets and the tasks at the character level are arbitrarily formed.

Finn et al. (2019) discover that TOE does not explicitly learn the structure across tasks, thus unable to fully utilise the data. The TOE performance in Figure 4.7 of the Omniglot experiment is coherent with the TOE result in Finn et al. (2019). The result figures in Finn et al. (2019) show a TOE result similar to ours in the Omniglot experiment. In contrast, TOE in the triathlon and pentathlon experiments performs well as it has access to drastically more data points than TOE in the Omniglot experiment, and samples numerous tasks from all previous datasets.

In the triathlon and pentathlon experiments, sequential MAML suffers from catastrophic forgetting due to the apparent distributional shift in the datasets. The Omniglot experiment, on the other hand, has tasks originating from the same underlying distribution. As a result sequential MAML in this setting is able to accumulate few-shot ability, although it performs worse than BOMLA and BOMVI as shown in Figure 4.7 since there is only one task available at a time.

Since the original FTML is not aimed for unseen few-shot tasks and does not deal with sequential datasets setting as in the triathlon and pentathlon experiments, we have to modify FTML as described in Chapter 4.4.1. Sampling from previous tasks in the buffer is a key feature of the FTML algorithm. Certainly one can sample many tasks from the buffer to achieve perfect memory in the triathlon and pentathlon experiments, but such a baseline setup has been taken into consideration by TOE. Therefore we choose to retain the online characteristic of the original FTML in our modified implementation.

4.7 Ablation Studies

4.7.1 Varying Precision Update Hyperparameter

Tuning the BomLA hyperparameter λ for precision update in Equation (4.7) corresponds to balancing between a smaller performance trade-off on a new dataset and less forgetting on previous datasets. As shown in Figure 4.8, a larger $\lambda = 1000$ results in a more concentrated Gaussian posterior and is therefore unable to learn new datasets well, but can better retain the performances on previous datasets. A smaller $\lambda = 1$ on the other hand gives a widespread Gaussian posterior and learns better on new datasets by sacrificing the performance on the previous datasets.

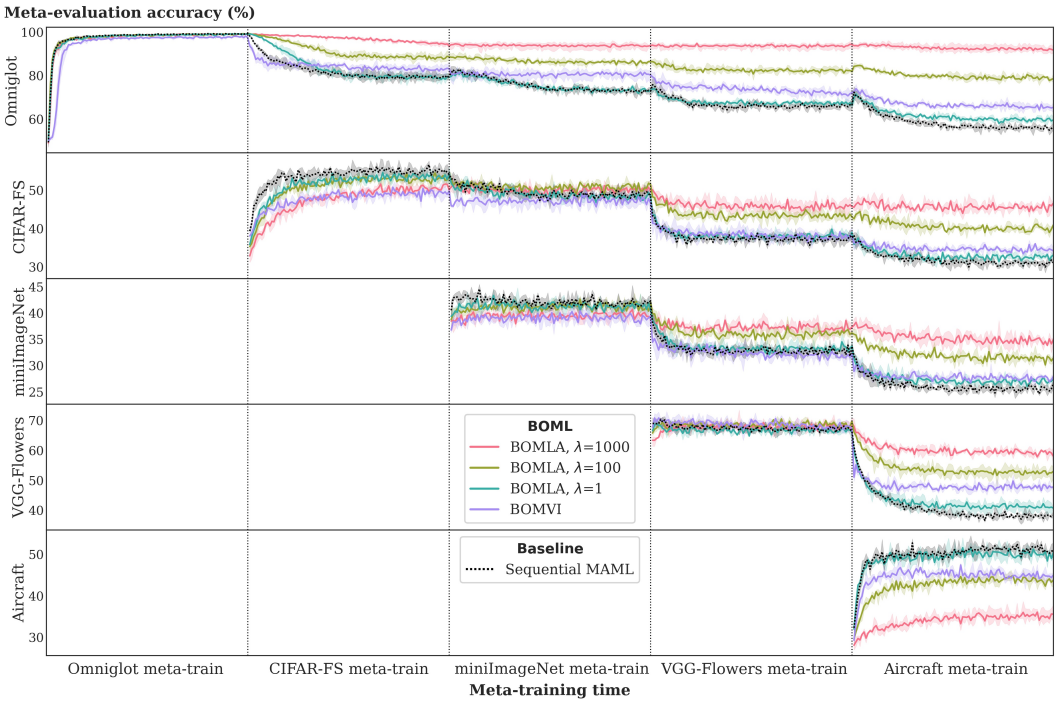

Figure 4.8: Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BomLA with a large $\lambda = 1000$ gives better performance in the off-diagonal plots (retains performances on previously learned datasets) but worse performance in the diagonal plots (does not learn well on new datasets). A small $\lambda = 1$ gives better performance in the diagonal plots (learns well on new datasets) but worse performance in the off-diagonal plots (forgets previously learned datasets). BomVI is also able to retain performance on previous datasets, although it may be unable to perform as good as BomLA due to sampling and estimator variance.

In this experiment, the value $\lambda = 100$ provides the optimal balance between old and new datasets. Ideally we seek for a good performance on both old and new datasets, but in reality there is a trade-off between retaining performance on old datasets and learning well on new datasets due to posterior approximation errors.

4.7.2 Analysing the Approximate Posterior Covariance

We visualise the covariance of the meta-parameters approximate posterior from BOMVI to better understand how the uncertainty in the algorithm prevents catastrophic forgetting in few-shot classification problems. Since BOMVI uses a Gaussian mean-field approximation, we examine the variance of the meta-parameters in a neural network model.

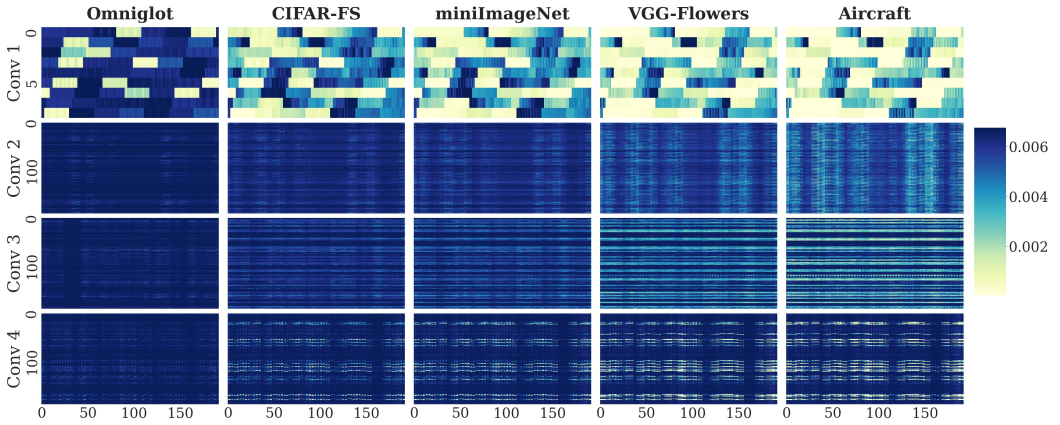

Figure 4.9: The change in the approximate posterior variance after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolutional layers of the neural network which gets closer to the classifying layer. Each plot in the figure is the colour-encoded variance corresponding to a specific knowledge domain dataset and the meta-parameters of a specific layer in the neural network model. The variance in each layer is flattened into a two-dimensional matrix visualisation. A darker colour indicates a higher variance. The variance increases in general as the convolutional layer gets closer to the classifying layer. The variance decreases in the raw level filters (Conv 1) as the model learns along the pentathlon sequence.

We follow the pentathlon sequence going from left to right in Figure 4.9:

Omniglot → CIFAR-FS → *mini*ImageNet → VGG-Flowers → Aircraft.

The Gaussian mean-field approximation becomes increasingly concentrated in general as it learns on more datasets. This is especially true for the earlier layers (Conv 1 and Conv 2), meaning that the posterior progressively becomes very confident on the meta-parameters of the raw-level filters. The variance for the layer closest to the classifier (Conv 4) remains large in general, although there are some filters with decreasing variance. As the convolutional layer gets closer to the classifying layer, a larger fine-tuning in the meta-parameters is needed (Ravi and Beatson, 2019) to cope with few-shot tasks from different knowledge domains.

The approximate posterior covariance from BomLA is too large for visualisation since it is block-diagonal. The BomLA covariance for each convolutional layer has dimension $D \times D$ where D is the number of meta-parameters in a convolutional layer. In theory, the BomLA covariance should also follow a similar pattern as the BomVI variance.

Chapter 5

Hessian Approximation

5.1 Introduction

Since the full Hessian matrices in Equation (3.10), Equation (4.8) and Equation (4.9) are intractable for large neural networks, we seek for some efficient and relatively close approximations to the Hessian matrices. Diagonal approximations (Denker and LeCun, 1991; Kirkpatrick et al., 2017) are memory and computationally efficient, but sacrifice approximation accuracy as they ignore the interaction between parameters.

We consider instead separating a Hessian matrix into blocks where different blocks are associated to different layers of a neural network. A particular diagonal block corresponds to the Hessian for a particular layer of the neural network. The block-diagonal Kronecker-factored approximation (Martens and Grosse, 2015; Grosse and Martens, 2016; Botev et al., 2017) utilises the fact that each diagonal block of the Hessian is Kronecker-factored for a single data point. This provides a better Hessian approximation as it takes the parameter interactions within a layer into consideration.

We begin this chapter by reviewing Hessian approximation in the large-scale classification setting that adopts the block-diagonal Kronecker-factored Fisher approximation method (Martens and Grosse, 2015; Grosse and Martens, 2016; Osawa et al., 2020). We then extend the block-diagonal Kronecker-factored Fisher approximation to the few-shot classification setting for our

BOML framework, and demonstrate empirically the importance of having a good Hessian approximation in the BOML framework.

5.2 Background

Consider estimating the Hessian matrix in Equation (3.10) for the large-scale classification setting. We approximate the Hessian for a single data point (x, y) using the Fisher information matrix associated to the model parameters ϑ:

$$F = \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)}\Big[\nabla_\vartheta \log p_\vartheta(y|x) \nabla_\vartheta \log p_\vartheta(y|x)^T\Big], \tag{5.1}$$

which guarantees the positive semi-definiteness of the approximation. By Bayes' rule, the joint distribution for the data point under parameter ϑ is $p_\vartheta(x, y) = p_\vartheta(y|x)p(x)$, where $p(x)$ is the distribution over inputs x, and $p_\vartheta(y|x)$ is the predictive distribution for outputs y from a model with parameters ϑ.

It is a well-known property that taking expectation of Hessian for a single data point over $p_\vartheta(x, y)$ leads to the Fisher in Equation (5.1), since

$$\begin{aligned}
&\mathbb{E}\Big[-\nabla^2_\vartheta \log p_\vartheta(y|x)\Big] \\
&\quad= \mathbb{E}\Big[-\nabla_\vartheta\Big(\frac{\nabla_\vartheta p_\vartheta(y|x)}{p_\vartheta(y|x)}\Big)\Big] &(5.2)\\
&\quad= -\int \nabla^2_\vartheta p_\vartheta(y|x)p(x)\,dx\,dy + \mathbb{E}\Big[\nabla_\vartheta \log p_\vartheta(y|x)\nabla_\vartheta \log p_\vartheta(y|x)^T\Big]. &(5.3)
\end{aligned}$$

Equation (5.3) follows from the quotient rule of differentiation. The first term of Equation (5.3) is equal to zero since $\nabla^2_\vartheta \int p_\vartheta(y|x)p(x)\,dx\,dy = 0$ under regularity conditions, which completes the proof.

We review the block-diagonal Kronecker-factored approximation to the large-scale learning Hessian in Equation (3.10) for fully-connected layers, convolutional layers and batch normalisation layers of neural networks in Chapters 5.2.1 – 5.2.3.

5.2.1 Fully-Connected Layers

The work in this section is presented by Martens and Grosse (2015). We consider a fully-connected neural network with L layers and model parameters $\vartheta = [\text{vec}(W_1)^T, \ldots, \text{vec}(W_L)^T]^T$, where W_ℓ is the weight of layer ℓ for $\ell = \{1, \ldots, L\}$ and vec denotes stacking the columns of a matrix into a vector. We denote the input of the neural network as $a_0 = x$ and the output of the neural network as a_L. As the input passes through each layer of the neural network, the pre-activation h_ℓ and activation a_ℓ for layer ℓ are

$$h_\ell = W_\ell a_{\ell-1} \quad \text{and} \quad a_\ell = f_\ell(h_\ell), \tag{5.4}$$

where f_ℓ is the activation function of layer ℓ. If a bias vector is applicable in calculating the pre-activation of a layer, we append the bias vector to the last column of the weight matrix and append a scalar one to the last element of the activation.

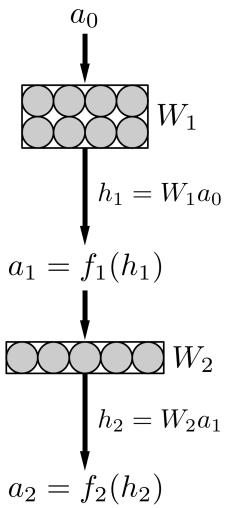

Figure 5.1: An example of a fully-connected neural network with $L = 2$ layers and weight matrices W_1, W_2. The bias vectors are omitted in this example. The weights are vectorised as $\vartheta = [\text{vec}(W_1)^T, \text{vec}(W_2)^T]^T$. The input a_0, pre-activations h_1, h_2 and activations a_1, a_2 interact according to Equation (5.4) using activation functions f_1, f_2.

Let $\mathfrak{L}(y, \text{nn}_\vartheta(x)) = -\log p_\vartheta(y|x)$ be the loss between the true label y and the output prediction of a neural network with model parameters ϑ. The

gradient of the loss with respect to ϑ can be computed by back-propagating through the neural network, starting from the final layer. Going backward for ℓ from L to 1, we have

$$\frac{\partial}{\partial W_\ell}\mathfrak{L}(y, \mathrm{nn}_\vartheta(x)) = \left(\frac{\partial}{\partial a_\ell}\mathfrak{L}(y, \mathrm{nn}_\vartheta(x)) \cdot f'_\ell(h_\ell)\right)\left(\frac{\partial h_\ell}{\partial W_\ell}\right)^T = g_\ell a_{\ell-1}^T \tag{5.5}$$

where

$$\frac{\partial h_\ell}{\partial W_\ell} = a_{\ell-1}, \quad g_\ell \coloneqq \frac{\partial}{\partial a_\ell}\mathfrak{L}(y, \mathrm{nn}_\vartheta(x)) \cdot f'_\ell(h_\ell), \quad \frac{\partial}{\partial a_{\ell-1}}\mathfrak{L}(y, \mathrm{nn}_\vartheta(x)) = W_\ell^T g_\ell.$$

The derivative of loss $\mathfrak{L}(y, \mathrm{nn}_\vartheta(x))$ with respect to the vectorised parameters $\vartheta = [\mathrm{vec}(W_1)^T \mathrm{vec}(W_2)^T \dots \mathrm{vec}(W_L)^T]^T$ can be annotated as $\partial_\vartheta \mathfrak{L} = [\mathrm{vec}(\partial_{W_1}\mathfrak{L})^T \mathrm{vec}(\partial_{W_2}\mathfrak{L})^T \dots \mathrm{vec}(\partial_{W_L}\mathfrak{L})^T]^T$, where we write $\partial_\vartheta \mathfrak{L} = \frac{\partial}{\partial \vartheta}\mathfrak{L}(y, \mathrm{nn}_\vartheta(x))$ in shorthand. The operator $\mathrm{vec}(\cdot)$ vectorises matrices into vectors by stacking the columns together.

The Fisher information matrix F in Equation (5.1) can thus be expressed block-wise, with the (i, j)-th block:

$$F_{ij} = \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)}\Big[\mathrm{vec}(\partial_{W_i}\mathfrak{L})\,\mathrm{vec}(\partial_{W_j}\mathfrak{L})^T\Big] \tag{5.6}$$

$$= \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)}\Big[\mathrm{vec}(g_i a_{i-1}^T)\,\mathrm{vec}(g_j a_{j-1}^T)^T\Big], \tag{5.7}$$

where Equation (5.7) follows from the derivation in Equation (5.5). A fruitful consequence is that Equation (5.7) can be expressed in terms of Kronecker products.

The **Kronecker product** between two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ is defined as

$$A \otimes B = \begin{bmatrix} A_{1,1}B & \cdots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{m,1}B & \cdots & A_{m,n}B \end{bmatrix}, \tag{5.8}$$

where $A_{i,j}$ is the (i, j)-th entry of A, and $(A \otimes B) \in \mathbb{R}^{mp\times nq}$. Kronecker product satisfies the following basic properties. For vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, matrices

$A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, $C \in \mathbb{R}^{n\times r}$, $D \in \mathbb{R}^{q\times s}$ and $X \in \mathbb{R}^{q\times n}$, we have:

(P1) $\operatorname{vec}(uv^T) = v \otimes u$

(P2) $(A \otimes B)^T = A^T \otimes B^T$

(P3) $(A \otimes B)(C \otimes D) = AC \otimes BD$

(P4) $(A \otimes B)\operatorname{vec}(X) = \operatorname{vec}(BXA^T)$

By setting the non-diagonal block approximations of the Fisher in Equation (5.6) to zero, we assume $F_{ij} \approx \mathbf{0}$ for $i \neq j$. Proceeding from Equation (5.7), the ℓ-th diagonal block F_ℓ of the Fisher information matrix can be approximated as

$$F_\ell = \mathbb{E}_{(x,y)\sim p_{\vartheta(x,y)}}[(a_{\ell-1} \otimes g_\ell)(a_{\ell-1} \otimes g_\ell)^T] \tag{5.9}$$

$$= \mathbb{E}_{(x,y)\sim p_{\vartheta(x,y)}}[a_{\ell-1}a_{\ell-1}^T \otimes g_\ell g_\ell^T] \tag{5.10}$$

$$\approx \mathbb{E}_{x\sim\hat{p}(x)}[a_{\ell-1}a_{\ell-1}^T] \otimes \mathbb{E}_{y\sim p_\vartheta(y|x)}[g_\ell g_\ell^T] \tag{5.11}$$

$$= A_{\ell-1} \otimes G_\ell, \tag{5.12}$$

where $A_{\ell-1} = \mathbb{E}_{x\sim\hat{p}(x)}[a_{\ell-1}a_{\ell-1}^T]$ and $G_\ell = \mathbb{E}_{y\sim p_\vartheta(y|x)}[g_\ell g_\ell^T]$. Equation (5.9) follows from Equation (5.7) with Property (P1), whereas Equation (5.10) is a direct result of Properties (P2) and (P3). Equation (5.11) follows by bringing the expectation into the Kronecker product and considering the inputs x from the empirical data distribution $\hat{p}(x)$ instead of the unknown true distribution $p(x)$. The Gaussian log-probability term can be calculated efficiently using Property (P4) without expanding the Kronecker product:

$$(A_{\ell-1} \otimes G_\ell)\operatorname{vec}(W_\ell - W_\ell^*) = \operatorname{vec}(G_\ell(W_\ell - W_\ell^*)A_{\ell-1}^T). \tag{5.13}$$

Data points often come in batches during training. For a batch of size B we have the approximations

$$A_{\ell-1} \approx \frac{1}{B}\breve{a}_{\ell-1}\breve{a}_{\ell-1}^T, \quad G_\ell \approx \frac{1}{B}\breve{g}_\ell\breve{g}_\ell^T, \tag{5.14}$$

where $\breve{a}_{\ell-1}$ and $\breve{g}_{\ell}$ are the batched version of activations and gradients with an extra final dimension of shape B. Putting the blocks together for layers $\ell = 1, \ldots, L$ gives the block-diagonal Kronecker-factored Fisher approximation

$$F \approx \begin{bmatrix} A_0 \otimes G_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & A_{L-1} \otimes G_L \end{bmatrix}. \tag{5.15}$$

5.2.2 Convolution Layers

Grosse and Martens (2016) extend the block-diagonal Kronecker-factored Fisher approximation for fully-connected layers to that for convolution layers. Consider a convolutional operation with stride 1 and padding R for convenience.

Due to the special condition of the convolutional operations, an activation $a_{i,k}$ for a specific layer with single data point should be considered in terms of spatial locations $k \in \mathcal{K}$ with index $i \in \{1, \ldots I\}$ over the number of input channels I of the weights. Figure 5.2 gives a two-dimensional illustration for the spatial locations of a 3-by-3 filter in the simple 6-by-6 activation. The batch size and input channels are ignored in the illustration.

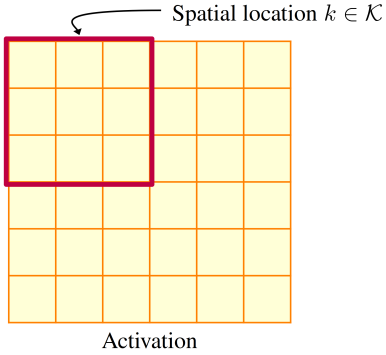

Figure 5.2: A two-dimensional example of a 6-by-6 activation with spatial location $k \in \mathcal{K}$ for a 3-by-3 filter. For a convolutional operation of stride 1, we have $|\mathcal{K}| = 16$ in this example. The batch size and input channels are ignored in the illustration

A convolutional operation uses weight $w_{c,i,\delta}$ and bias b_c, with subscript $c \in \{1, \ldots, C\}$ over the number of output channels C, and the index $\delta \in \Delta$ for spatial offset of filter from its center. The convolution operation computes

pre-activation $h_{c,k}$ as

$$h_{c,k} = \sum_{\delta \in \Delta} w_{c,i,\delta}\, a_{i,k+\delta} + b_c. \tag{5.16}$$

Figure 5.3 shows a two-dimensional illustration for the spatial offsets of a 3-by-3 filter, where the input and output channels are ignored in the example. The weight parameter associated to the spatial offset $\delta = (+1, 0)$ is $w_{c,i,(+1,0)}$ for input channel i and output channel c.

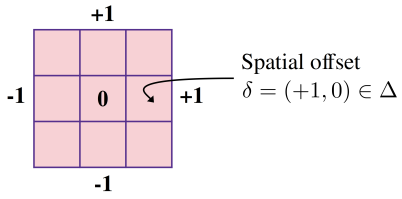

Figure 5.3: A two-dimensional example for a spatial offset δ of a 3-by-3 filter. In this example we have $|\Delta| = 9$. The input and output channels are ignored in this illustration.

For generalisation purpose, the activation of layer ℓ should be stored as a matrix of shape $(B|\mathcal{K}|, I|\Delta|)$ with batch size B. This is achieved by first collecting the activation for each spatial location $k \in \mathcal{K}$ and then flattening into an activation $\breve{\mathcal{A}}_\ell$ of shape $(B|\mathcal{K}|, I)$. We then apply the expansion operator $[\![\cdot]\!]$ to $\breve{\mathcal{A}}_\ell$ which extract patches in each spatial location within the spatial offsets of the filter, resulting in $[\![\breve{\mathcal{A}}_\ell]\!]$ of shape $(B|\mathcal{K}|, I|\Delta|)$. The activation and expanded activation in the single data point $B = 1$ case are denoted as $\mathcal{A}_\ell$ and $[\![\mathcal{A}_\ell]\!]$ respectively. The expansion operation is conducted as follows:

$$[\![\breve{\mathcal{A}}_\ell]\!]_{kB+b,i|\Delta|+\delta} = \left(\breve{\mathcal{A}}_\ell\right)_{(k+\delta)B+b,i} = a^{(b)}_{i,k+\delta}, \tag{5.17}$$

such that $k + \delta \in \mathcal{K}$, and b denotes the activation for data b in a batch of size B.

The weight parameters are also modified into shape $(C, |\Delta|, I)$ and then flatten into W_ℓ of shape $(C, I|\Delta|)$. We denote the expanded input and output of a neural network with layers $\ell = 1, \ldots, L$ as $[\![\breve{\mathcal{A}}_0]\!]$ and $[\![\breve{\mathcal{A}}_L]\!]$ respectively. As the input passes through each layer of the neural network, the pre-activation

$\mathcal{H}_\ell$ and expanded activation $[\![\breve{\mathcal{A}}_\ell]\!]$ for layer ℓ are

$$\mathcal{H}_\ell = [\![\breve{\mathcal{A}}_{\ell-1}]\!] W_\ell^T \quad \text{and} \quad [\![\breve{\mathcal{A}}_\ell]\!] = f_\ell(\mathcal{H}_\ell), \tag{5.18}$$

where f_ℓ is the activation function of layer ℓ. If a bias vector is involved in the pre-activation calculation, we append the bias vector to the last column of the weight matrix W_ℓ and append a column of 1's to the last column of the expanded activation $[\![\breve{\mathcal{A}}_\ell]\!]$.

Similar to the fully-connected layers, we define the gradient of loss $\mathfrak{L}(y, \mathrm{nn}_\vartheta(x))$ with respect to the pre-activation $\mathcal{H}_\ell$ for a mini-batch of size B as $\breve{\mathcal{G}}_\ell \coloneqq \partial_{\mathcal{H}_\ell} \mathfrak{L}$. The gradient in the single data point $B = 1$ case is denoted by $\mathcal{G}_\ell$. For convolution layers $\ell = 1, \ldots, L$, the ℓ-th diagonal block Fisher F_ℓ in Equation (5.6) is approximated by

$$F_\ell = \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)} \Big[\mathrm{vec}(\partial_{W_\ell} \mathfrak{L}) \, \mathrm{vec}(\partial_{W_\ell} \mathfrak{L})^T \Big] \tag{5.19}$$

$$\approx \mathbb{E}_{x\sim \hat{p}(x)} \Big[[\![\mathcal{A}_{\ell-1}]\!]^T [\![\mathcal{A}_{\ell-1}]\!] \Big] \otimes \mathbb{E}_{y\sim p_\vartheta(y|x)} \big[\mathcal{G}_\ell^T \mathcal{G}_\ell \big] \tag{5.20}$$

$$\approx A_{\ell-1} \otimes G_\ell, \tag{5.21}$$

where

$$A_{\ell-1} = \frac{1}{B} [\![\breve{\mathcal{A}}_{\ell-1}]\!]^T [\![\breve{\mathcal{A}}_{\ell-1}]\!] \quad \text{and} \quad G_\ell = \frac{1}{B|\mathcal{K}|} \breve{\mathcal{G}}_\ell^T \breve{\mathcal{G}}_\ell$$

are the approximation to Kronecker products in Equation (5.20) when the data is in batch. The approximation in Equation (5.20) assumes for **spatial homogeneity** on the activation and pre-activation elements in Equation (5.16) as follows:

- $\mathbb{E}_{x\sim\hat{p}(x)}[a_{i,k}] = M(i)$, $\forall i \in \{1, \ldots, I\}$, $\forall k \in \mathcal{K}$, and some function M. That is the first order statistics of the activations are independent of the spatial location.

- $\mathbb{E}_{x\sim\hat{p}(x)}[a_{i,k}\, a_{i',k'}] = \Omega(i, i', k'-k)$, $\forall i, i' \in \{1, \ldots, I\}$, $\forall k, k' \in \mathcal{K}$, and some function Ω.

- $\mathbb{E}_{y\sim p_\vartheta(y|x)}[(\partial_{h_{c,k}}\mathfrak{L})(\partial_{h_{c',k'}}\mathfrak{L})] = \Gamma(c, c', k'-k)$, $\forall c, c' \in \{1, \ldots, C\}$, $\forall k, k' \in \mathcal{K}$, and some function Γ.

  That is the second order statistics of the activations and gradients with respect to pre-activations at any spatial locations k and k' depends only on $k' - k$.

5.2.3 Batch Normalisation Layers

A batch normalisation layer often follows after a convolution layer. Suppose that the convolution layer before batch normalisation has C_ℓ number of output channels. For a mini-batch of size B, batch normalisation first normalise the inputs x_b in the batch for $b = 1, \ldots, B$ with

$$\tilde{x}_b = \frac{x_b - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{5.22}$$

where

$$\mu = \frac{1}{B}\sum_{b=1}^{B} x_b \quad \text{and} \quad \sigma^2 = \frac{1}{B}\sum_{b=1}^{B}(x_b - \mu)^2 \tag{5.23}$$

are the batch mean and variance respectively. A small ϵ is added for numerical stability. The output of the batch normalisation layer is $y_b = \gamma\tilde{x}_b + \beta$ for $b = 1, \ldots, B$, where $\gamma, \beta \in \mathbb{R}^{C_\ell}$ are the trainable weights and biases respectively.

We adopt the unit-wise Fisher approximation for batch normalisation layers (Osawa et al., 2020). Suppose that the ℓ-th layer of a neural network is a batch normalisation layer. The unit-wise method first aggregates the elements of γ and β according to the output channels $c = 1, \ldots, C_\ell$ and re-arrange the parameters into $W_\ell = [\gamma_1, \beta_1, \ldots, \gamma_{C_\ell}, \beta_{C_\ell}]^T \in \mathbb{R}^{2C_\ell}$. Rather than computing the full ℓ-th Fisher block

$$F_\ell = \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)}\left[(\partial_{W_\ell}\mathfrak{L})(\partial_{W_\ell}\mathfrak{L})^T\right] \in \mathbb{R}^{2C_\ell \times 2C_\ell}, \tag{5.24}$$

the unit-wise method instead approximates F_ℓ by blocks $F_\ell^{(c)}$ for $c = 1, \ldots, C_\ell$,

which gives

$$F_\ell \approx \begin{bmatrix} F_\ell^{(1)} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & F_\ell^{(C_\ell)} \end{bmatrix}. \tag{5.25}$$

Each block $F_\ell^{(c)}$ considers only the parameter interaction between γ_c and β_c of channel c, for $c = 1, \ldots, C_\ell$:

$$F_\ell^{(c)} = \mathbb{E}_{(x,y)\sim p_\vartheta(x,y)} \begin{bmatrix} (\partial_{\gamma_c}\mathfrak{L})^2 & (\partial_{\gamma_c}\mathfrak{L})(\partial_{\beta_c}\mathfrak{L}) \\ (\partial_{\beta_c}\mathfrak{L})(\partial_{\gamma_c}\mathfrak{L}) & (\partial_{\beta_c}\mathfrak{L})^2 \end{bmatrix}. \tag{5.26}$$

For a neural network taking inputs of batch size B, the Fisher approximation for block $F_\ell^{(c)}$ becomes

$$F_\ell^{(c)} \approx \frac{1}{B}\sum_{b=1}^{B} \begin{bmatrix} (\partial_{\gamma_c}\mathfrak{L}_b)^2 & (\partial_{\gamma_c}\mathfrak{L}_b)(\partial_{\beta_c}\mathfrak{L}_b) \\ (\partial_{\beta_c}\mathfrak{L}_b)(\partial_{\gamma_c}\mathfrak{L}_b) & (\partial_{\beta_c}\mathfrak{L}_b)^2 \end{bmatrix}, \tag{5.27}$$

where $\mathfrak{L}_b$ denotes the loss $\mathfrak{L}$ with respect to the b-th input and label.

5.3 Hessian Approximation for BOML

Recall that the Hessian matrices $\widetilde{H}_{t+1}$ in Equation (4.8) and H_{t+1} in Equation (4.9) associated to BOMLA have entries:

$$H_{t+1}^{ij} = \frac{1}{M}\sum_{m=1}^{M} -\frac{\partial^2}{\partial\theta^{(i)}\partial\theta^{(j)}} \log p(\mathcal{D}_{t+1}^{m,S}|\theta)\Big|_{\theta=\mu_{t+1}},$$
$$\widetilde{H}_{t+1}^{ij} = \frac{1}{M}\sum_{m=1}^{M} -\frac{\partial^2}{\partial\theta^{(i)}\partial\theta^{(j)}} \log p(\mathcal{D}_{t+1}^{m,Q}|\tilde{\theta}^m))\Big|_{\theta=\mu_{t+1}}.$$

We indicate H_{t+1} as the pre-adaptation Hessian for knowledge domain $\mathfrak{D}_{t+1}$, and $\widetilde{H}_{t+1}$ as the post-adaptation Hessian corresponding to the task-specific adapted parameters $\widetilde{\theta}^m$ for tasks $m = 1, \ldots, M$. We estimate the Hessian matrices H_{t+1} and $\widetilde{H}_{t+1}$ for a single data point because the cross-entropy losses in the objective Equation (4.6) are often averaged over the batch of data points in a task.

5.3.1 Pre-Adaptation Hessian

The Hessian H_{t+1} can be approximated in the same manner as the block-diagonal Kronecker-factored approximation in Chapter 5.2, since the log-likelihood in H_{t+1} is considered over the pre-adapted meta-parameters θ. Before θ is adapted via inner loop to a specific task, the approximation method for the Hessian with respect to θ is of no difference to that of the Hessian with respect to the model parameters ϑ in Chapter 5.2.

For an N-way K-shot setting corresponding to the knowledge domain $\mathfrak{D}_{t+1}$, the support set $\mathcal{D}_{t+1}^{m,S}$ has size $B = \left|\mathcal{D}_{t+1}^{m,S}\right| = NK$, for task $m = 1, \ldots, M$. This results in the ℓ-th block Kronecker-factored approximation for H_{t+1} on a single data points as follows.

1. For a fully-connected layer ℓ:

$$\begin{aligned} F_\ell \approx & \left(\frac{1}{MB}\sum_{m=1}^{M} \breve{a}_{\ell-1}^m \big(\breve{a}_{\ell-1}^m\big)^T\right) \otimes \left(\frac{1}{MB}\sum_{m=1}^{M} \breve{g}_\ell^m \big(\breve{g}_\ell^m\big)^T\right) \\ =: & \; A_{\ell-1} \otimes G_\ell, \end{aligned} \tag{5.28}$$

   with $\breve{a}_{\ell-1}^m$ and $\breve{g}_\ell^m$ acquired as in Equation (5.14) using $\mathcal{D}_{t+1}^{m,S}$.

2. For a convolution layer ℓ:

$$\begin{aligned} F_\ell \approx & \left(\frac{1}{MB}\sum_{m=1}^{M} \left[\!\left[\breve{\mathcal{A}}_{\ell-1}^m\right]\!\right]^T \left[\!\left[\breve{\mathcal{A}}_{\ell-1}^m\right]\!\right]\right) \otimes \left(\frac{1}{MB|\mathcal{K}|}\sum_{m=1}^{M} \big(\breve{\mathcal{G}}_\ell^m\big)^T \breve{\mathcal{G}}_\ell^m\right) \\ =: & \; A_{\ell-1} \otimes G_\ell, \end{aligned} \tag{5.29}$$

   with $\left[\!\left[\breve{\mathcal{A}}_{\ell-1}^m\right]\!\right]$ and $\breve{\mathcal{G}}_\ell^m$ acquired as in Equation (5.21) using $\mathcal{D}_{t+1}^{m,S}$.

3. For a batch normalisation layer ℓ, the diagonal block $F_\ell^{(c)}$ of the F_ℓ approximation is:

$$F_\ell^{(c)} \approx \frac{1}{MB}\sum_{b=1}^{MB} \begin{bmatrix} \big(\partial_{\gamma_c}\mathfrak{L}_b^m\big)^2 & \big(\partial_{\gamma_c}\mathfrak{L}_b^m\big)\big(\partial_{\beta_c}\mathfrak{L}_b^m\big) \\ \big(\partial_{\beta_c}\mathfrak{L}_b^m\big)\big(\partial_{\gamma_c}\mathfrak{L}_b^m\big) & \big(\partial_{\beta_c}\mathfrak{L}_b^m\big)^2 \end{bmatrix} \tag{5.30}$$

as in Equation (5.27) where $\mathfrak{L}_b^m$ denotes the loss $\mathfrak{L}$ with respect to the b-th input and label from the support set $\mathcal{D}_{t+1}^{m,S}$ of task m.

5.3.2 Post-Adaptation Hessian

The Hessian $\widetilde{H}_{t+1}$ requires a special treatment in addition to the original approximation method, since its log-likelihood is considered over the *adapted parameters* $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}^{m,S}))$ for task m whilst the derivative in $\widetilde{H}_{t+1}$ is taken with respect to the meta-parameters θ.

Each (x, y) pair for the Fisher in BOMLA is associated to a task m. The Fisher information matrix $\widetilde{\boldsymbol{F}}$ corresponding to the Hessian $\widetilde{H}_{t+1}$ for a single data point is

$$\widetilde{\boldsymbol{F}} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\Big[\nabla_\theta \log p_{\tilde{\theta}^m}(y|x)\nabla_\theta \log p_{\tilde{\theta}^m}(y|x)^T\Big] \tag{5.31}$$

$$= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left[\left(\frac{\partial\tilde{\theta}^m}{\partial\theta}\right)^T \nabla_{\tilde{\theta}^m} \log p_{\tilde{\theta}^m}(y|x)\nabla_{\tilde{\theta}^m} \log p_{\tilde{\theta}^m}(y|x)^T\left(\frac{\partial\tilde{\theta}^m}{\partial\theta}\right)\right], \tag{5.32}$$

where the expectation in $\widetilde{\boldsymbol{F}}$ is taken over $(x, y) \sim p_{\tilde{\theta}^m}(y|x)p(x)$ from the query set. The additional Jacobian matrix $\frac{\partial\tilde{\theta}^m}{\partial\theta}$ breaks the Kronecker-factored structure described by Martens and Grosse (2015) for the original Fisher in Equation (5.1). It is therefore necessary to derive an adjusted approximation to the Hessian $\widetilde{H}_{t+1}$ with some further assumptions.

Finn et al. (2017) show that the first step of the quick adaptation in $\tilde{\theta}^m$ contributes the largest change to the meta-evaluation objective, and the remaining adaptation steps give a relatively small change to the objective. We can reasonably assume a one-step SGD quick adaptation for task m:

$$\tilde{\theta}^m = \theta - \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{m,S}) \tag{5.33}$$

in order to approximate the Fisher, although in other parts of the framework we use a few-step SGD as usual. We consider a cross-entropy loss for $\mathcal{L}$. By imposing the one-step SGD assumption, the (i, j)-th entry of the Jacobian term

can be interpreted as

$$\left(\nabla_\theta \tilde{\theta}^m\right)^{ij} = I^{ij} - \frac{\partial^2(-\log p(\mathcal{D}^{m,S}|\theta))}{\partial\theta^{(i)}\partial\theta^{(j)}}, \tag{5.34}$$

where I is the corresponding identity matrix.

The Hessian term in Equation (5.34) is identical to the Hessian H_{t+1} except without the summation over tasks $m = 1, \ldots, M$. We should be aware that both Hessian matrices are considering approximations for a *single data point*. As such, we can approximate the Hessian in $\nabla_\theta \tilde{\theta}^m$ for a single data point using the same approximation to H_{t+1} in Chapter 5.3.1.

The Jacobian $\nabla_\theta \tilde{\theta}^m$ is the batched version of $\frac{\partial \tilde{\theta}^m}{\partial \theta}$, since $\nabla_\theta \tilde{\theta}^m$ uses the entire support set $\mathcal{D}^{m,S}$ for computation. Breaking the expectation structure in Equation (5.32) results in the approximation

$$\widetilde{\boldsymbol{F}} \approx \frac{1}{M}\sum_{m=1}^{M} \left(I - F\right)^T \underbrace{\mathbb{E}\Big[\nabla_{\tilde{\theta}^m} \log p_{\tilde{\theta}^m}(y|x) \nabla_{\tilde{\theta}^m} \log p_{\tilde{\theta}^m}(y|x)^T\Big]}_{F^m} \left(I - F\right), \tag{5.35}$$

where

$$F = \mathbb{E}\Big[\nabla_\theta \log p_\theta(y|x) \nabla_\theta \log p_\theta(y|x)^T\Big]$$

with expectation of F taken over $(x, y) \sim p_\theta(y|x)p(x)$ from the support set. The ℓ-th diagonal block approximation of the term $\left(I - F\right)$ for layer ℓ is

$$\left(I - F\right)_\ell \approx I_\ell - A_{\ell-1} \otimes G_\ell, \tag{5.36}$$

where $A_{\ell-1}$ and G_ℓ are acquired the exact same way as in Equation (5.28) for a fully-connected layer or Equation (5.29) for a convolutional layer. If layer ℓ corresponds to batch normalisation, then F_ℓ is approximated using the block approximation in Equation (5.30).

The ℓ-th diagonal block approximation for the Fisher with respect to the task adapted parameters $\tilde{\theta}^m$ is $F_\ell^m \approx \widetilde{A}_{\ell-1}^m \otimes \widetilde{G}_\ell^m$, where $\widetilde{A}_{\ell-1}^m$ are $\widetilde{G}_\ell^m$ are acquired based on the block-diagonal Kronecker-factored Fisher approximation

described in Chapter 5.2 with batch size $B = \left|\mathcal{D}_{t+1}^{m,Q}\right|$. If layer ℓ corresponds to batch normalisation, then we use the approximation in Equation (5.27) instead.

Putting the approximations together, the ℓ-th diagonal block of $\widetilde{\boldsymbol{F}}$ in Equation (5.32) is

$$\widetilde{\boldsymbol{F}}_\ell \approx \frac{1}{M}\sum_{m=1}^{M}(I_\ell - A_{\ell-1}\otimes G_\ell)^T(\widetilde{A}_{\ell-1}^m \otimes \widetilde{G}_\ell^m)(I_\ell - A_{\ell-1}\otimes G_\ell), \tag{5.37}$$

for fully-connected or convolution layers. We expand $\widetilde{\boldsymbol{F}}_\ell$ using Property (P3) of Kronecker products to give

$$\begin{aligned}\widetilde{\boldsymbol{F}}_\ell \approx \frac{1}{M}\sum_{m=1}^{M}\Big[&\widetilde{A}_{\ell-1}^m \otimes \widetilde{G}_\ell^m - (A_{\ell-1})^T\widetilde{A}_{\ell-1}^m \otimes (G_\ell)^T\widetilde{G}_\ell^m - \widetilde{A}_{\ell-1}^m A_{\ell-1}\otimes \widetilde{G}_\ell^m G_\ell \\ &+ (A_{\ell-1})^T\widetilde{A}_{\ell-1}^m A_{\ell-1}\otimes (G_\ell)^T\widetilde{G}_\ell^m G_\ell\Big].\end{aligned} \tag{5.38}$$

Finally, moving the meta-batch averaging into the Kronecker factors gives the approximation:

$$\begin{aligned}\widetilde{\boldsymbol{F}}_\ell \approx &\widetilde{A}_{\ell-1}\otimes\widetilde{G}_\ell - (A_{\ell-1})^T\widetilde{A}_{\ell-1}\otimes(G_\ell)^T\widetilde{G}_\ell - \widetilde{A}_{\ell-1}A_{\ell-1}\otimes\widetilde{G}_\ell G_\ell \\ &+ (A_{\ell-1})^T\widetilde{A}_{\ell-1}A_{\ell-1}\otimes(G_\ell)^T\widetilde{G}_\ell G_\ell,\end{aligned} \tag{5.39}$$

where

$$\widetilde{A}_{\ell-1} = \frac{1}{M}\sum_{m=1}^{M}\widetilde{A}_{\ell-1}^m \quad \text{and} \quad \widetilde{G}_\ell = \frac{1}{M}\sum_{m=1}^{M}\widetilde{G}_\ell^m.$$

If layer ℓ is a batch normalisation layer, we have

$$\widetilde{\boldsymbol{F}}_\ell \approx \widetilde{F}_\ell - (F_\ell)^T\widetilde{F}_\ell - \widetilde{F}_\ell F_\ell + (F_\ell)^T\widetilde{F}_\ell F_\ell, \tag{5.40}$$

where $\widetilde{F}_\ell = \frac{1}{M}\sum_{m=1}^{M}F_\ell^m$.

Algorithm 9 shows the pseudo-code of block-diagonal Kronecker-factored Hessian approximation for the BOML framework. In general, the approximations to $\widetilde{\boldsymbol{F}}_\ell$ in Equations (5.39) and (5.40) are applicable to any gradient-based meta-

learning methods with a few-step gradient descent inner loop.

Algorithm 9 Block-diagonal Kronecker-factored Hessian approximation for BOML

Require: base set $\mathcal{D}_t$, number of tasks M_H, number of neural network layers L
for $m = 1$ **to** M_H **do**
  Sample task $\mathcal{D}_t^m = \mathcal{D}_t^{m,S} \cup \mathcal{D}_t^{m,Q}$
  Inner update $\tilde{\theta}^m = SGD_k(\mathcal{L}(\theta, \mathcal{D}_t^{m,S}))$
  for $\ell = 1$ **to** L **do**
    Approximate H_t with F_ℓ in Equations (5.28) – (5.30) using $\mathcal{D}_t^{m,S}$
    Compute F_ℓ^m using $\mathcal{D}_t^{m,Q}$
    Approximate $\widetilde{H}_t$ with $\widetilde{\boldsymbol{F}}_\ell$ in Equations (5.39) and (5.40)
  end for
end for
return Approximation for $\widetilde{H}_t$ and H_t

Hessian approximation for BOML might seem daunting at first sight, since we need to approximate two Hessian matrices $\widetilde{H}_{t+1}$ and H_{t+1}. At a closer inspection the approximation for Hessian H_{t+1} can be reused to approximate the Hessian $\widetilde{H}_{t+1}$. For a fully-connected or convolution layer ℓ, we keep the two pairs $\widetilde{A}_{\ell-1}, \widetilde{G}_\ell$ and $A_{\ell-1}, G_\ell$ in memory. If layer ℓ corresponds to batch normalisation, we keep $\widetilde{F}_\ell$ and F_ℓ in memory instead.

5.4 Experiments

5.4.1 Ignoring the Jacobian

We should be aware that the full approximation in Equations (5.39) and (5.40) for the Hessian $\widetilde{H}_{t+1}$ requires an extensive calculation for the cross terms. This is due to the inclusion of Jacobian $\frac{\partial \tilde{\theta}^m}{\partial \theta}$ for the Fisher $\widetilde{\boldsymbol{F}}$ in Equation (5.32).

This experiment investigates the effect of neglecting the Jacobian matrix in the Fisher $\widetilde{\boldsymbol{F}}$. Consider re-running the triathlon, pentathlon and Omniglot sequential task experiments in Chapter 4.4 by ignoring the additional Jacobian matrix $\frac{\partial \tilde{\theta}^m}{\partial \theta}$ in Equation (5.32). This results in the truncated Fisher

approximation for the ℓ-th block:

$$\widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})} \approx \widetilde{A}_{\ell-1} \otimes \widetilde{G}_{\ell} \quad \text{and} \quad \widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})} \approx \frac{1}{M} \sum_{m=1}^{M} F_{\ell}^{m} \tag{5.41}$$

where all the cross terms in Equations (5.39) and (5.40) are ignored. The former $\widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}$ approximation is for a fully-connected or convolution layer, whilst the latter approximation is for a batch normalisation layer. We compare BOMLA that uses approximation $\widetilde{\boldsymbol{F}}_{\ell}$ with **Jacobian included** versus BOMLA that uses approximation $\widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}$ with **Jacobian excluded** for each layer ℓ of the neural network model.

Dataset	Average accuracy (%) reduction
Triathlon	-1.61 ± 2.48
Pentathlon	-1.88 ± 1.94

Table 5.1: Average reduction in the meta-evaluation accuracy when the Jacobian is excluded in Hessian approximation for BOMLA with $\lambda = 100$. The average is taken over tasks from all datasets, after meta-training is completed on the entire dataset sequence. The values are reported by averaging over a total of 100 tasks along with the 95% confidence interval. The performance reduction is not apparent, since the confidence intervals along with the averages cover small reduction values that are both above and below zero.

Table 5.1 shows the mean reduction in the meta-evaluation accuracy when the Jacobian is excluded in Hessian approximation for BOMLA with precision update hyperparameter $\lambda = 100$. The mean is considered over tasks from all knowledge domains, after meta-training is completed on the entire dataset sequence. The performance decline is not apparent for the triathlon and pentathlon sequences, since these dataset sequences are relatively short with length 3 and 5 respectively.

We proceed to the Omniglot sequential task setting with a long few-shot problem sequence of length 254. We discover that omitting the Jacobian in Hessian approximation induces a significant error which deteriorates the meta-evaluation performance. Figure 5.4 (left) illustrates the performance

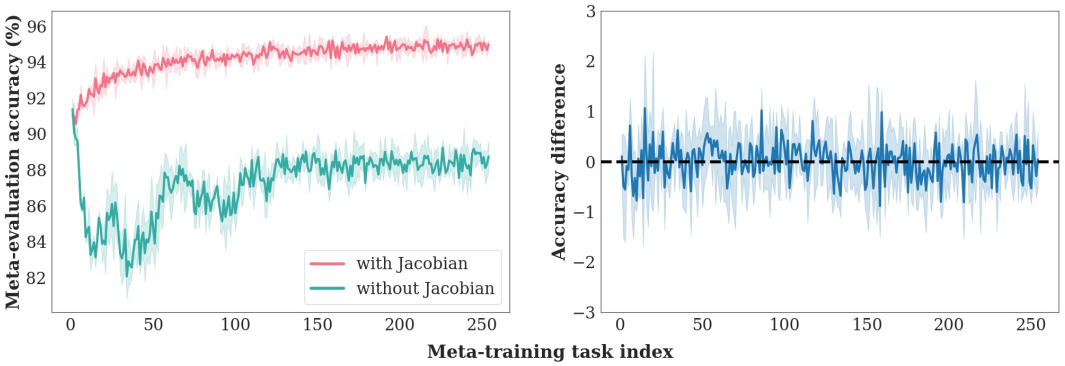

Figure 5.4: Meta-evaluation performance comparison across 3 seed runs on the novel tasks along meta-training. **Left**: compares BOMLA of Hessian approximation in Equations (5.39) and (5.40) (with Jacobian) versus BOMLA of Hessian approximation in Equation (5.41) (without Jacobian). BOMLA that uses Hessian approximation without Jacobian (——) shows a large performance degradation compared to that with Jacobian (——). **Right**: meta-evaluation accuracy difference between BOMLA with Jacobian at $\lambda = 0.01$ and BOMLA without Jacobian at $\lambda = 1$. The difference evolves around zero, indicating that the adjustment $\lambda = 1$ gives a quick fix to the posterior approximation when the Jacobian is excluded in Hessian approximation.

degradation for omitting the Jacobian in BOMLA Hessian approximation. The comparison is made with λ value fixed at $\lambda = 0.01$.

Nonetheless it is possible to adjust the approximate posterior by tuning the hyperparameter λ. We compare BOMLA without Jacobian at $\lambda = 1$ to the full BOMLA with Jacobian at $\lambda = 0.01$. Figure 5.4 (right) shows that BOMLA without Jacobian at $\lambda = 1$ and BOMLA with Jacobian at $\lambda = 0.01$ give a similar performance, since the performance difference between these two runs evolves around zero. The hyperparameter λ allows an opportunity to fix the error in the posterior approximation. When lacking in computational resource, tuning λ provides an alternative cost-effective solution for posterior approximation.

5.4.2 Analysing the Cross Terms

This section investigates the cross terms of $\widetilde{\boldsymbol{F}}_\ell$ for neural network layers $\ell = 1, \ldots, L$. We employ the pentathlon sequence for easier visualisation:

$$\text{Omniglot} \rightarrow \text{CIFAR-FS} \rightarrow \textit{mini}\text{ImageNet} \rightarrow \text{VGG-Flowers} \rightarrow \text{Aircraft}.$$

We mentioned earlier about ignoring the Jacobian in Equation (5.32), which leads to the assumption:

$$\frac{\partial \tilde{\theta}^m}{\partial \theta} \approx I, \tag{5.42}$$

where I is an identity matrix. This further brings us to the assumption where the Hessian in Equation (5.34) is approximately zero when ignoring the Jacobian:

$$\frac{\partial^2(-\log p(\mathcal{D}^{m,S}|\theta))}{\partial \theta^{(i)} \partial \theta^{(j)}} \approx \mathbf{0}. \tag{5.43}$$

To investigate whether this is a reasonable assumption, Figure 5.5 examines the Fisher F in Equation (5.35) that approximates the pre-adaptation Hessian in Equation (5.43) for a single data point. If the assumption in Equation (5.43) is reasonable, then Figure 5.5 should show that the approximation

$$F_\ell \approx 0 \tag{5.44}$$

holds for all layers ℓ of the neural network model.

Figure 5.5 illustrates the blocks of Fisher approximation F_ℓ corresponding to all neural network layers ℓ for each dataset in the pentathlon sequence. The Fisher approximation values corresponding to the fully-connected classifier layer is relatively large compared to that of the convolution layers.

The visualisation in Figure 5.5 might mislead one into believing that Equation (5.44) is a valid assumption, since the assumption holds for almost all layers of the neural network. The numerical values of F_ℓ for the classifier layer ℓ is also not as large, since the value bar only ranges to a single digit. Nonetheless we should take into consideration that the cross terms in Equations (5.39) and

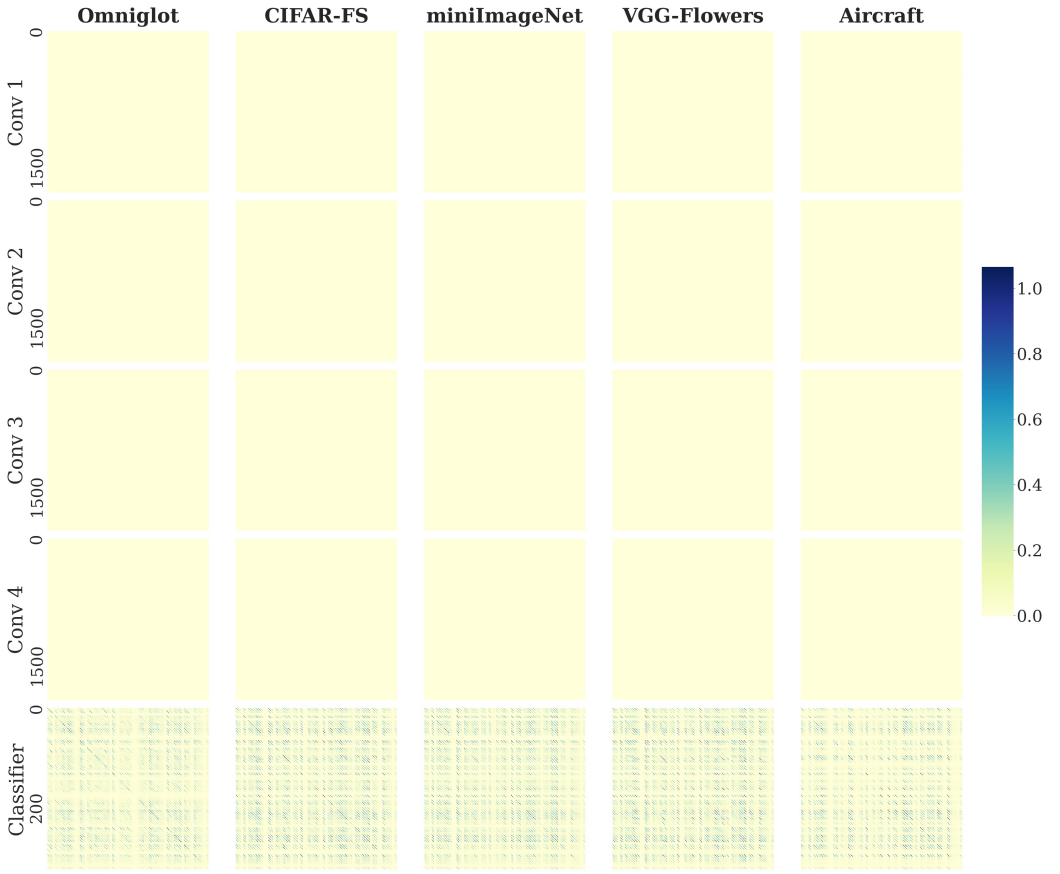

Figure 5.5: The Fisher approximation F corresponding to the Hessian in Equation (5.43) after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolution layers and the fully-connected classifier layer of the neural network. Each plot in the figure is the colour-encoded Fisher approximation F_ℓ corresponding to a specific knowledge domain dataset and a specific layer ℓ in the neural network model. F_ℓ for the middle convolution layers (ℓ = Conv 2, 3 and 4) are cropped as the full matrices are too large to visualise. The F_ℓ matrices for the convolution layers ℓ have entry values that are close to zero.

(5.40) involve multiplication which might amplify the error for excluding F in the Fisher approximation $\widetilde{\boldsymbol{F}}$.

Figure 5.5 demonstrates empirically that the assumption in Equation (5.43) is appropriate for the convolution layers, but not for the output classifier layer. Figure 5.6 further verifies this claim by showing the absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ between the full Fisher and the truncated Fisher for each layer ℓ of the neural network model. The Fisher $\widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}$ that excludes the Jacobian provides a reasonable approximation for a convolution layer ℓ, but it gives a

poor approximation for the classifier layer.

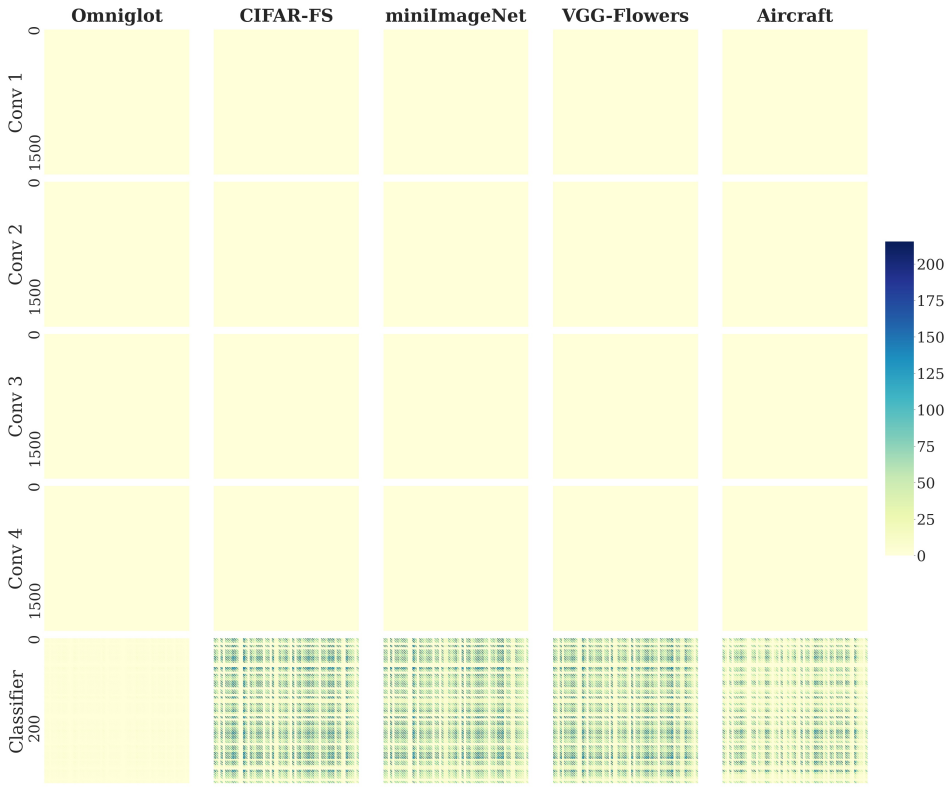

Figure 5.6: The absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ between the full Fisher and the truncated Fisher for each layer ℓ of the neural network model after meta-training is completed on each dataset. Going from left to right are the datasets of the pentathlon sequence. Going from top to bottom are the convolution layers and the fully-connected classifier layer of the neural network. Each plot in the figure is the colour-encoded absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ corresponding to a specific knowledge domain dataset and a specific layer ℓ of the neural network model. The absolute difference matrices for the middle convolution layers (ℓ = Conv 2, 3 and 4) are cropped as the full matrices are too large to visualise. The absolute difference matrices for the convolution layers ℓ have entry values that are close to zero, but the absolute difference matrices for the classifier layer have large entry values.

Since the absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ is small for almost all layers ℓ, one might believe that it is appropriate to employ the approximation $\widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}$ without Jacobian. It is however important to realise that the absolute difference for the classifier layer is in the order of hundreds over all the pentathlon datasets except for Omniglot. We take a closer look at the absolute difference $\left|\widetilde{\boldsymbol{F}}_\ell - \widetilde{\boldsymbol{F}}_\ell^{(\text{trunc})}\right|$ for the classifier layer ℓ on the Omniglot dataset in Figure 5.7.

The absolute difference for the classifier layer on the Omniglot dataset is in the order of tens, although not as large as that of the other datasets in the order of hundreds.

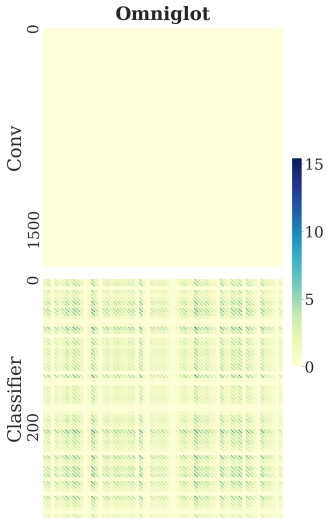

Figure 5.7: The absolute difference $\left|\widetilde{\boldsymbol{F}}_{\ell} - \widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}\right|$ between the full Fisher and the truncated Fisher for a convolution and fully-connected classifier layer ℓ of the neural network model after meta-training is completed on Omniglot. We only retain the visualisation for the first convolution layer (Conv 1) since the remaining convolution layers have the same visualisation. The absolute difference matrices for the convolution layers ℓ have entry values that are close to zero. The absolute difference matrix for the classifier layer corresponding to Omniglot has entry values in the order of tens.

The approximation error of $\widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}$ for the classifier layer ℓ is very large, despite being a reasonable approximation for the convolution layers. As illustrated in earlier experiments of Chapter 5.4.1, employing the approximation $\widetilde{\boldsymbol{F}}_{\ell}^{(\text{trunc})}$ without Jacobian induces a significant performance deterioration for a longer sequence of few-shot problems. The posterior approximation error is not apparent in the shorter triathlon and pentathlon sequences, but the error accumulates over a long sequence.

Chapter 6

Automating Bayesian Online Meta-Learning

6.1 Introduction

BOML has been shown to achieve its goal for sequential few-shot classification problems (Yap et al., 2021). The current BOML framework for few-shot classification is, however, limited in several ways. A key limitation is that the inner loop adaptation uses a hand-crafted stochastic gradient descent (SGD) method with manually picked learning rate and number of steps. The quick adaptation components from different knowledge domains do not have to communicate with each other when they encounter unseen tasks, since the quick adaptation for each knowledge domain is pre-fixed.

Another limitation of BOML is its inflexibility. BOML requires the sequential datasets to be of the same few-shot setting, since BOML only learns a single neural network model for all datasets in a sequence. For instance, all datasets of the triathlon and pentathlon sequences in Chapter 4.4 share the same 5-way 1-shot setting. As a result the neural network model in BOML insists on having the specific output dimension of 5. In order to overcome this limitation, we consider the input and output layers of the neural network model separately for each dataset, and the remaining layers are shared across all the knowledge domain datasets as in BOML.

This chapter aims to enhance the original BOML to the BOML+ framework that is highly related to the Badger architecture (Rosa et al., 2019). We automate the inner loop adaptation mechanism to replace the hand-crafted few-step SGD inner loops. We achieve this goal by learning an LSTM that outputs quick adaptation steps for each knowledge domain. This is inspired by previous works (Andrychowicz et al., 2016; Li and Malik, 2017) on learning-to-learn that automate the update process of the parameters to replace the traditional gradient descent methods.

Since each knowledge domain has its own LSTM for quick adaptation, a task-pointer mechanism is required to communicate which LSTM should be responsible for adaptation when a novel task arrives. We implement a class-incremental learning method to train a generative classifier (van de Ven et al., 2021) for identifying the domain-belonging of the novel tasks. The domain knowledge datasets are considered as 'classes' that arrive sequentially for class-incremental learning to train for a generative classifier. A class-incremental learning method is necessary in contrast to a conventional classifier, due to the requirement for a mechanism capable of learning continually from 'classes' that arrive in a sequential manner.

During meta-evaluation, the trained generative classifier informs the agent on the knowledge domain of a novel task via prediction using the support set images from the novel task. The process of assigning tasks to their respective knowledge domains closely resembles that of labelling the classes of images in a dataset. Such labelling processes can be expensive in terms of human effort. A pragmatic approach is to utilise labels during training to create a task-labelling mechanism that manages the costly labelling process during the evaluation phase. This establishes the purpose of training a generative classifier for the task-pointer mechanism.

This chapter demonstrates the enhancement of BOML to an automated framework BOML+ with greater flexibility. We develop a highly parallelisable training process for BOML+, and demonstrate the new evaluation workflow in

BOML+. The experiments show empirically that BOML+ outperforms BOML during evaluation. The work in this chapter was completed under a project funded by GoodAI via the GoodAI grant. We briefly review the learning-to-learn method (Andrychowicz et al., 2016), class-incremental learning method (van de Ven et al., 2021) and Badger architecture (Rosa et al., 2019) before proceeding to BOML+.

6.2 Background

6.2.1 Learning-to-Learn with LSTM

We briefly explain the learning-to-learn framework by Andrychowicz et al. (2016) before proceeding to our work. Large-scale machine learning often utilises some hand-crafted gradient descent methods such as SGD, Adam or Adagrad for optimisation. These methods are carefully designed with hyperparameters such as learning rate introduced for tuning. Although these methods are proven to be useful, extra time and computational cost are necessary for hyperparameter tuning to give the optimum result. As such Andrychowicz et al. (2016) introduce an automated framework in replacement of the hand-crafted algorithm for large-scale machine learning optimisation. The automated framework comprises LSTMs that take the parameter gradients as inputs and return the parameter updates as outputs. We need a method to train the LSTMs into an automated parameter-updating framework.

Consider an *optimiser* (eg. LSTM) with parameters ϕ. The goal in large-scale classification is to optimise the parameters ξ of a model (also known as *optimisee*) with respect to the objective function f. With a slight abuse of notation, the final trained optimisee parameters can be written as a function of the objective and the optimiser parameters $\xi^*(f, \phi)$. The meta-level objective with respect to the optimiser parameters is defined as $\mathcal{L}(\phi) = \mathbb{E}_f\big[f(\xi^*(f, \phi))\big]$. Figure 6.1 illustrates the computational graph for the gradients of $\mathcal{L}(\phi)$ with respect to the LSTM optimiser parameters ϕ when updating ξ at time step $r-1$ and r.

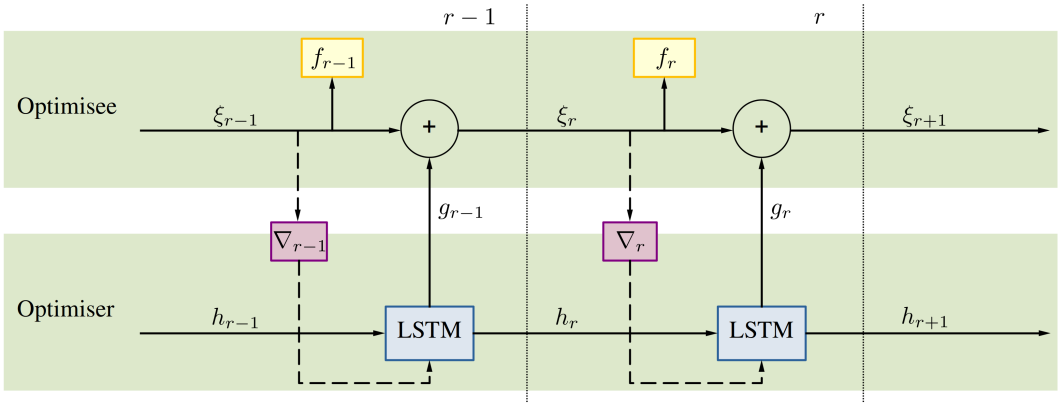

Figure 6.1: The computational graph for the gradients of $\mathcal{L}(\phi)$ with respect to the LSTM optimiser parameters ϕ when updating ξ at time step $r-1$ and r. The gradients are allowed to flow through the solid arrows during back-propagation, but the gradient flow is prohibited along the dashed arrows. The optimisee corresponds to a model which is usually a neural network with parameters ξ. For a particular time step r, we update the optimisee parameters ξ_r by adding the output g_r acquired from the LSTM. The LSTM takes the gradients ∇_r of objective f with respect to ξ_r as inputs along with hidden states h_r. When computing the gradients of the LSTM parameters, we do not take the gradient flow from ∇_r into consideration.

The goal is to train an optimiser, which is an LSTM $m(\cdot, \cdot, \phi)$ with parameters ϕ, to output update steps g_r for the optimisee parameters ξ_r at time step r. Instead of considering only the final step ξ^*, the original work accumulates weighted sum of objectives over some time horizon R:

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{r=1}^{R} w_r f(\xi_r) \right], \tag{6.1}$$

where $\xi_{r+1} = \xi_r + g_r$ and

$$\begin{bmatrix} g_r \\ h_{r+1} \end{bmatrix} = m(\nabla_r, h_r, \phi) \tag{6.2}$$

with $\nabla_r = \nabla_\xi f(\xi_r)$, hidden states h_r of the LSTM and weights w_r.

Upon completing the optimisation on $\mathcal{L}(\phi)$ with respect to ϕ, the parameter ξ is updated by $\xi_{r+1} = \xi_r + g_r$ for update step $r+1$ where g_r is provided by the trained LSTM $m\big(\nabla_r, h_r, \phi^*\big)$ with optimised parameters ϕ^*.

6.2.2 Generative Classifier

We adopt a class-incremental learning method, where classes of a dataset arrive sequentially, to train for a generative classifier (van de Ven et al., 2021). The generative classifier uses Bayes' rule $p(y|x) \propto p(x|y)p(y)$ for classification, where x is an input from dataset $\mathcal{D} = \{(x_i, \tilde{y}_i)\}_{i=1}^n$, and y is the label prediction of x.

There are two steps involved in estimating the likelihood $p(x|y)$ for classification using Bayes' rule: VAE training and likelihood estimation.

VAE training: A VAE model is learned for each class of a dataset (van de Ven et al., 2021), in which the classes arrive in sequential order for training. The encoder q_φ, decoder p_ψ and prior p_{prior} for the VAEs are:

$$q_\varphi(z|x) = N\Big(\cdot\,\Big|\mu_\varphi^{(x)}, {\sigma_\varphi^{(x)}}^2 I\Big), \quad p_\psi(x|z) = N\big(\cdot\,\big|\mu_\psi^z, I\big), \quad p_{\text{prior}}(z) = N(\cdot|\mathbf{0}, I), \tag{6.3}$$

where $\mu_\varphi^{(x)}$ and $\sigma_\varphi^{(x)}$ are the outputs of an encoder neural network of parameters φ that takes input x, and μ_ψ^z is the output of a decoder neural network of parameters ψ taking input z. The VAEs are optimised using a variational lower bound to $p(x) = \int p_\psi(x|z)p_{\text{prior}}(z)dz$:

$$LB(\varphi, \psi) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\psi(x|z)] - D_{\text{KL}}(q_\varphi(z|x)\|p_{\text{prior}}(z)). \tag{6.4}$$

Likelihood estimation: The trained encoder and decoder for each class are used to estimate the likelihood $p(x|y = c)$ for every class c:

$$p(x|y = c) = \frac{1}{S}\sum_{s=1}^{S} \frac{p_{\psi_c}\big(x|z^{(s)}\big)p_{\text{prior}}\big(z^{(s)}\big)}{q_{\varphi_c}\big(z^{(s)}|x\big)}, \tag{6.5}$$

where φ_c and ψ_c are the VAE parameters for class c, and S is the number of importance samples drawn. Finally a label prediction is returned based on which class gives the highest likelihood for a given x, as $\arg\max_y p(y|x) = \arg\max_y p(x|y)p(y)$ by Bayes' rule and the distribution $p(y)$ is assumed to be uniform over all classes.

6.2.3 Badger Architecture

The Badger architecture (Rosa et al., 2019) seeks for an agent that can adapt quickly to unseen tasks. The architecture comprises many experts, each has its own internal memory and internal state. All experts share the same expert policy that is optimised via the outer loop, aiming for a fast inner loop adaptation strategy for each expert. When an input arrives, the inner loop is triggered to update the internal states of the relevant experts together with the help of the shared expert policy in order to acquire an output. A communication mechanism is often involved to determine the experts that should be responsible for work and to pass messages between experts for the inner loop updates. Figure 6.2 shows the badger structure within an agent.

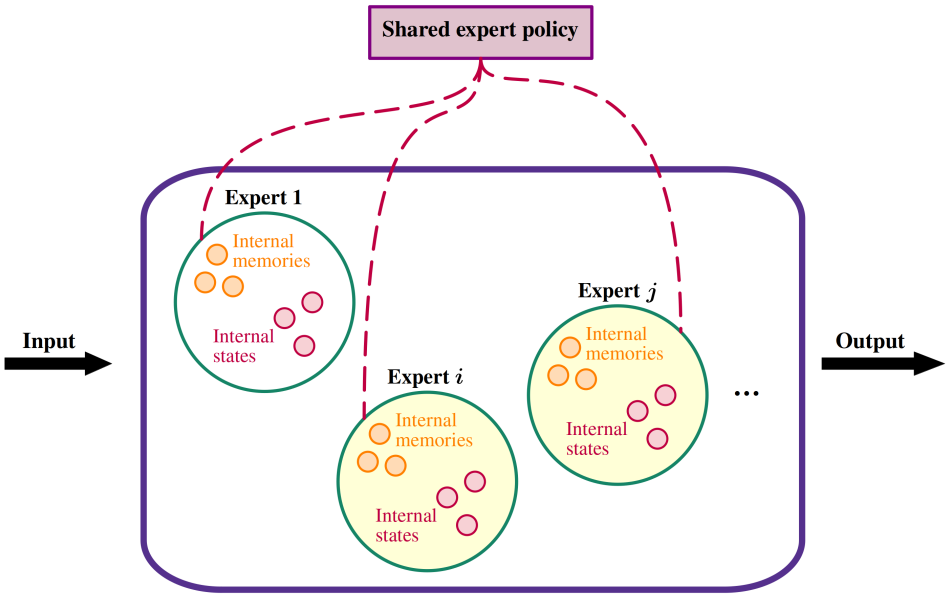

Figure 6.2: An example architecture of a Badger agent. The shared expert policy can be accessed by all experts. The red dashed line – – illustrates the connection between the experts and the shared expert policy. The selected experts i and j in yellow colour are responsible for the incoming input. Experts i and j communicate with each other and update their internal states using the internal memories to give an output.

6.3 Framework

6.3.1 Overview

This project enhances the BOML framework by automating the inner loops using LSTMs, and introducing a task-pointer mechanism to communicate the responsibilities of different LSTMs. Figure 6.3 illustrates the new components introduced in BOML+. The elements in red in Figure 6.3 are unique to BOML+ compared to the original BOML. The task-pointer in BOML+ is trained incrementally on the base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$. A new dataset-specific adaptation LSTM is trained as each knowledge domain $\mathfrak{D}_t$ for $t = 1, \ldots, T$ arrive sequentially. We explain these new components in BOML+ before progressing to the implementation of BOML+.

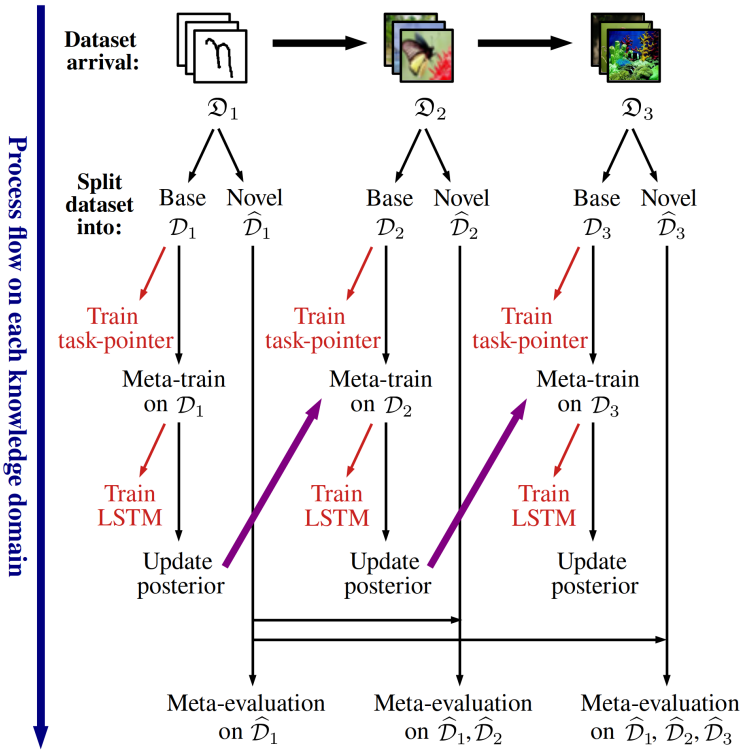

Figure 6.3: The process flow of BOML+ for training and evaluation on an example sequence (Omniglot → CIFAR-FS → *mini*ImageNet) when each dataset arrives. The arrows in purple illustrate that the updated posterior is being brought forward for the next meta-training when a new dataset arrives. The items in red are the elements newly-introduced in BOML+. This figure resembles the BOML process flow in Figure 4.1, except for the elements in red that are unique to BOML+.

6.3.2 LSTM Inner Loop

The original BOML inner loops use a traditional gradient descent method as in Equation (4.4). The SGD inner loop is a hand-crafted algorithm, in which the hyperparameters such as the learning rate and the number of adaptation steps are manually decided. Inspired by a learning-to-learn method (Andrychowicz et al., 2016), we automate the inner loops using LSTMs that output few-shot updates in replacement of the SGD updates. For each knowledge domain $\mathfrak{D}_t$, we assign an LSTM $m_t(\cdot, \cdot, \phi_t)$ for quick adaptation to any task from this knowledge domain.

Our training implementation differs from the original learning-to-learn algorithm (Andrychowicz et al., 2016), since we need the LSTMs for quick adaptation on few-shot tasks as in Equation (4.4) whereas the original framework by Andrychowicz et al. (2016) does long range parameter updates for a large-scale training. A few-shot inner loop adaptation only uses very few examples for an update on the meta-paramters θ, whilst the large-scale trained LSTM has access to large batches of examples. An inner loop often takes very few steps (possibly just one step) of gradient update, whilst the large-scale setting involves many sequential gradient update steps. As such we modify the training method of the LSTMs in Chapter 6.2.1 to fit our few-shot setting.

The procedure to train a few-shot LSTM parameters ϕ is as follows: first acquire the trained meta-parameters θ as usual using original BOML on some base set $\mathcal{D}$, then train for a few-shot LSTM with slight modifications. For few-shot tasks $m = 1, \ldots, M$, we obtain $\nabla^{(m)}$ in Equation (6.6) from the support set $\mathcal{D}^{m,S}$ of task m using the trained meta-parameters θ as initialisation, so that

$$\begin{bmatrix} g^{(m)} \\ h_1 \end{bmatrix} = m\Big(\nabla^{(m)}, h_0, \phi\Big) \tag{6.6}$$

with $\nabla^{(m)} = \nabla_\theta f(\theta, \mathcal{D}^{m,S})$. The objective corresponding to Equation (6.1) in

our few-shot case becomes

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{m=1}^{M} f\left(\tilde{\theta}^{(m)}, \mathcal{D}^{m,Q}\right) \right], \tag{6.7}$$

where $\tilde{\theta}^m = \theta + g^{(m)}$, with $\mathcal{D}^{m,Q}$ being the query set of task m, weights w_m set to unity, and f is the cross-entropy loss. Instead of accumulating over r (training time steps) in Equation (6.1), the loss in Equation (6.7) is accumulated over the few-shot tasks $m = 1, \ldots, M$. Upon completing an optimisation on $\mathcal{L}(\phi)$ with respect to ϕ, the inner loop adaptation in BOML+ for task m becomes

$$\tilde{\theta}^m = \theta + g^{(m)}, \tag{6.8}$$

where $g^{(m)}$ is provided by the trained LSTM $m\big(\nabla^{(m)}, h_0, \phi^*\big)$ with optimised parameters ϕ^*.

6.3.3 Task-Pointer

Since we have a separate LSTM for each knowledge domain, we need a mechanism to identify which LSTM should be responsible for quick adaptation when an unseen few-shot task arrives. We train a generative classifier $G(\varphi, \psi)$ using a class-incremental learning method (van de Ven et al., 2021) as described in Chapter 6.2.2, where φ and ψ are the parameters of the generative classifier's VAE encoders and decoders respectively.

The base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$ are considered as 'classes' that arrive sequentially and we learn a new VAE that is class-specific $V_t(\varphi_t, \psi_t)$ when a new base set $\mathcal{D}_t$ arrives. The final generative classifier G has encoder and decoder parameters $\varphi = \{\varphi_1, \ldots, \varphi_T\}$ and $\psi = \{\psi_1, \ldots, \psi_T\}$.

When an unseen task arrives, we use the few-shot inputs x from its support set and estimate the likelihood $p(x|y = t)$ of knowledge domain $\mathfrak{D}_t$ for $t = 1, \ldots, T$. Each few-shot input with greatest likelihood forms a vote for the knowledge domain. The knowledge domain that gets the most votes from the predictions is the final decision, and the expert associated to this knowledge

domain is triggered.

6.3.4 Relation to Badger

The BOML+ framework in this project is highly related to the GoodAI Badger architecture (Rosa et al., 2019). In the Badger context, the BOML+ meta-parameters θ correspond to the **shared expert policy**. The **outer loop** updates the meta-parameters θ as the knowledge domain datasets $\mathfrak{D}_1, \ldots, \mathfrak{D}_T$ arrive sequentially for training. The **internal memory** of the **expert** t associated to knowledge domain $\mathfrak{D}_t$ includes the LSTM $m_t(\cdot, \cdot, \phi_t)$ for $t = 1, \ldots, T$. The **inner loops** use the LSTMs for quick adaptation on few-shot tasks. The internal memory of expert t also stores the output classifying layer $\theta_t^{(O)}$ and the input convolutional layer $\theta_t^{(I)}$ of the meta-parameters. These are required if we intend to be flexible on the types of few-shot tasks that the agent can consider. The generative classifier task-pointer $G(\varphi, \psi)$ is associated to the **connectivity** of all experts in the form of a passive collaboration. In our setting, only a single expert is responsible to a knowledge domain.

6.4 Implementation

Recall from Chapter 4.1 that the datasets $\mathfrak{D}_t = \mathcal{D}_t \cup \widehat{\mathcal{D}}_t$, $t = 1, \ldots, T$ arrive sequentially for training. In our framework, training occurs sequentially on the base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$, whilst evaluation occurs cumulatively on the novel sets $\widehat{\mathcal{D}}_1, \ldots, \widehat{\mathcal{D}}_T$.

6.4.1 Training

An important advantage of the BOML+ framework is that the training process is *highly parallelisable*. Figure 6.4 illustrates the parallel training processes of all the BOML+ components. The processes are parallelised, and Process A is fully isolated from Processes B and C.

Process A is responsible for training the generative classifier task-pointer. It trains a VAE that is knowledge domain-specific $V_t(\varphi_t, \psi_t)$ on a newly arrived base set $\mathcal{D}_t$, as explained in Chapter 6.2.2. The trained VAE for each dataset is accumulated for the final generative classifier $G(\varphi, \psi)$, where $\varphi = \{\varphi_1, \ldots, \varphi_T\}$

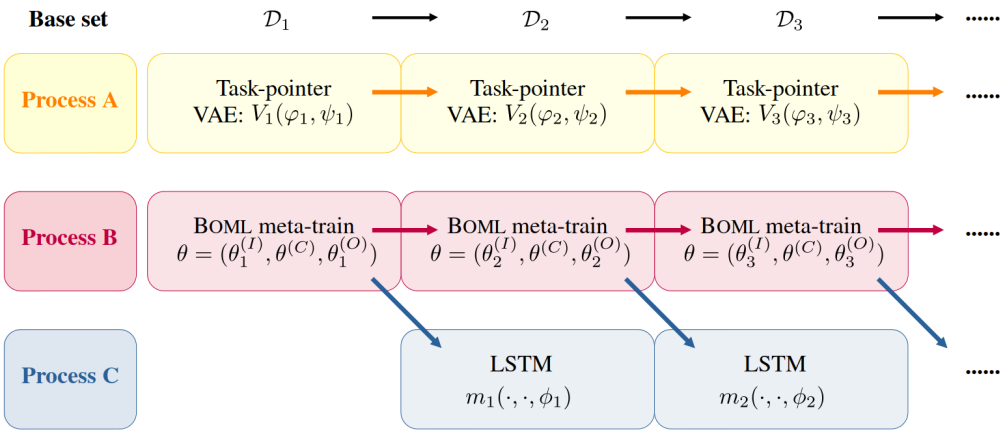

Figure 6.4: The training processes of BOML+ when each dataset arrives.

and $\psi = \{\psi_1, \ldots, \psi_T\}$.

In order to be flexible on the few-shot tasks that the agent can consider, we consider the meta-parameters θ in separate parts of input, body and output, such that $\theta = (\theta_t^{(I)}, \theta^{(C)}, \theta_t^{(O)})$. The dataset-specific meta-parameters $\theta_t^{(I)}$ and $\theta_t^{(O)}$ are kept in the internal memory of the expert in charge of knowledge domain $\mathfrak{D}_t$ as illustrated in Figure 6.5.

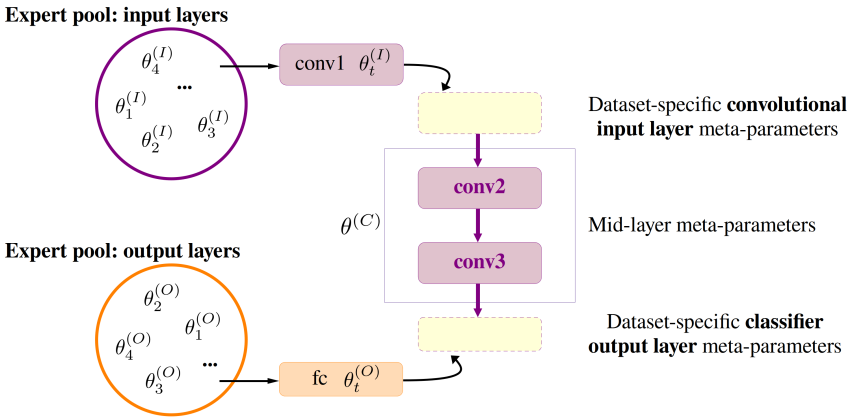

Figure 6.5: Input and output layers $\theta_t^{(I)}$ and $\theta_t^{(O)}$ from expert t are concatenated to the model structure.

The concatenated meta-parameters θ is meta-trained by Process B using the objective in Equation (4.6) for BOMLA or Equation (4.11) for BOMVI. The mid-layer meta-parameters $\theta^{(C)}$ is the **shared expert policy** in Badger terminology. It evolves in a similar fashion to the original BOML meta-parameters, as meta-training proceeds sequentially on the base sets $\mathcal{D}_1, \ldots, \mathcal{D}_T$. We pass on

the meta-parameters $\theta = (\theta_t^{(I)}, \theta^{(C)}, \theta_t^{(O)})$ trained on $\mathcal{D}_t$ to Process C, and a knowledge domain-specific LSTM $m_t(\cdot, \cdot, \phi_t)$ is trained via the method described in Chapter 6.3.2. Meanwhile Process B continues meta-training θ on the next base set $\mathcal{D}_{t+1}$.

6.4.2 Evaluation

The goal towards the end of training is to carry out few-shot learning on tasks from all novel sets $\widehat{\mathcal{D}}_1, \ldots, \widehat{\mathcal{D}}_T$. The novel sets are accumulated in a novel set pool as knowledge domains arrive in sequential order. Figure 6.6 shows the evaluation process of BOML+.

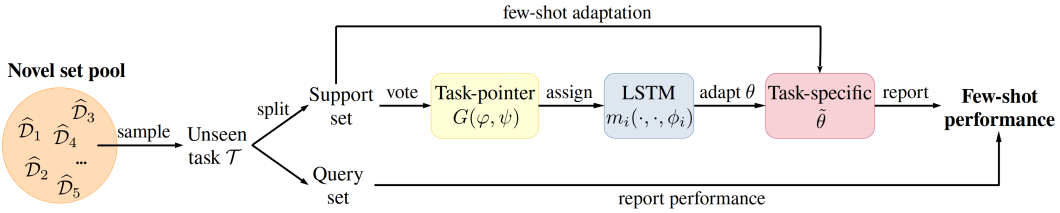

Figure 6.6: The evaluation process of BOML+ on the accumulated novel set pool.

When an unseen task arrives from a novel set pool, it is split into a support set – which is a small few-shot set for quick adaptation, and the remaining as a query set for performance reporting. All images of the support set are passed to the task-pointer $G(\varphi, \psi)$ for generative prediction. The highest vote gives the final decision on the knowledge domain assignment, and its associated expert, say expert i, is triggered. The adaptation LSTM $m_i(\cdot, \cdot, \phi_i)$ from expert i runs quick adaptation using the support set on the meta-parameters $\theta = (\theta_i^{(I)}, \theta^{(C)}, \theta_i^{(O)})$, where $\theta_i^{(I)}$ and $\theta_i^{(O)}$ are from the internal memory of expert i. The adapted $\tilde{\theta}$ finally reports the few-shot performance using the query set.

6.5 Experiments

6.5.1 Setup

Few-shot classification model structure: For the experiments in this chapter, we use the model architecture that takes 3 modules with 16 filters of size 3×3, followed by a batch normalisation, a ReLU activation and a 2×2

max-pooling. A fully-connected layer is appended to the final module before getting the class probabilities with softmax. Table A.4 in Appendix A.3 records the hyperparameters used in the BOML+ experiments.

Quick adaptation LSTM structure: For the automated inner loops in each knowledge domain, we utilise the coordinate-wise LSTM structure proposed by Andrychowicz et al. (2016) using two-layer LSTMs with 20 hidden units in each layer. As illustrated in Figure 6.7, the LSTM operates in a coordinate-wise manner on the elements of the meta-parameters. Each element of the meta-parameters corresponds to a separate activation in the LSTM structure.

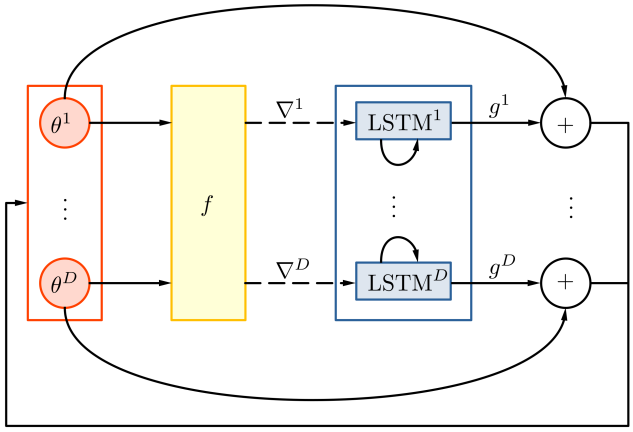

Figure 6.7: Computational graph for one step of the LSTM adaptation on a D-dimensional meta-parameters θ. The LSTMs have shared parameters but separated hidden states. The gradients are allowed to flow through the solid arrows during back-propagation, but not the dashed arrows. The inner loop cross-entropy loss f is evaluated using the D-dimensional meta-parameters $(\theta^1, \ldots, \theta^D)^T$. The gradients $(\nabla^1, \ldots, \nabla^D)^T$ of f for a specific few-shot task with respect to the meta-parameter elements are fed into the LSTMs, and the LSTMs return the updates $(g^1, \ldots, g^D)^T$ for each element of the meta-parameters.

Task-pointer structure: Each knowledge domain in the few-shot problem sequence corresponds to a single VAE in the generative classifier learned via class-incremental learning. The means and variances of the Gaussian VAE encoders and decoders are provided by neural networks of 3 fully-connected layers with 2000 hidden units. Each layer is followed by a ReLU activation. The bottleneck z dimension is 100.

Knowledge domain sequence: We investigate BOML+ using the pentathlon sequence as in Chapter 4.4.3. The pentathlon sequence is chosen over triathlon,

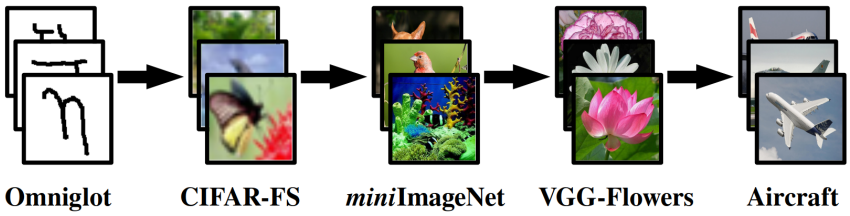

Figure 6.8: The pentathlon knowledge domain dataset sequence.

since it is a more challenging sequence and the enhancement effect of BOML+ would be more visible. The Omniglot sequential task setting in Chapter 4.5 is not suitable for BOML+, since there is no evident distributional shift in the sequential tasks and thus not possible to study the effect of introducing a new expert in BOML+ for each knowledge domain dataset. We pick the best performing BOML algorithm, that is BOMLA with a properly tuned λ, to represent BOML for the comparison with BOML+.

6.5.2 Component Comparison

There are three newly-introduced elements in BOML+: the LSTMs for automated inner loop, the dataset-specific input and output layer meta-parameters $\theta_t^{(I)}$ and $\theta_t^{(O)}$ for each knowledge domain $\mathfrak{D}_t$, and a generative classifier task-pointer. The bottom-line expectation on BOML+ described in Run R5 below is to perform at least as good as the original BOML in Run R1. We check the few-shot performance step-by-step, and compare the following combinations on the pentathlon dataset sequence.

R1 Original BOML

R2 BOML + LSTM adaptations

R3 BOML + specific $\theta_t^{(I)}$ & $\theta_t^{(O)}$

R4 BOML + specific $\theta_t^{(I)}$ & $\theta_t^{(O)}$ + LSTM adaptations

R5 BOML + specific $\theta_t^{(I)}$ & $\theta_t^{(O)}$ + LSTM adaptations + task-pointer (**BOML+**)

Run R1 implements the original BOML using the best-performing BOMLA with $\lambda = 5$ as the representative of BOML. Run R2 implements the automated LSTM quick adaptation on top of the BOML framework. Run R3 separates the input and output layer of the meta-parameters for each knowledge domain. The purpose of Runs R2 and R3 is to individually investigate the performance of the new elements in BOML+. Run R4 combines the two elements in Runs R2 and R3, resulting in BOML+ without the task-pointer element. Finally, Run R5 gives the complete BOML+ with all elements included.

6.5.3 Results

Figure 6.9 shows the few-shot performances of Runs R1 – R4. Since the task-pointer training is fully isolated from the other elements, we evaluate the effectiveness of the task-pointer in Run R5 separately after the training of all other elements completes. The diagonal plots in Figure 6.9 indicate how well the runs can learn on new datasets, and the off-diagonal plots show the capability of the runs in retaining their performances on previously learned datasets as meta-training proceeds. Table 6.1 shows that the task-pointer is capable of pointing the tasks appropriately to their respective knowledge domains.

The two elements of BOML+: a) LSTM adaptations, and b) dataset-specific input and output layer meta-parameters, drastically improve the BOML performance as shown by Run R4 (—) in Figure 6.9. An interesting observation from the result is that the two elements **individually** do not give apparent improvement to BOML. This is clearly visible from the following two comparisons: Run R1 (—) vs Run R2 (—) and Run R1 (—) vs Run R3 (—). The comparison between Runs R1 and R2 shows that using LSTM adaptations in Run R2 can retain the performance on previously learned datasets (off-diagonal plots), but it learns less well on new datasets (diagonal plots). The comparison between Runs R1 and R3, on the other hand, illustrates that using dataset-specific input and output layer meta-parameters in Run R3 can learn well on new datasets (diagonal plots), but it works less well in retaining

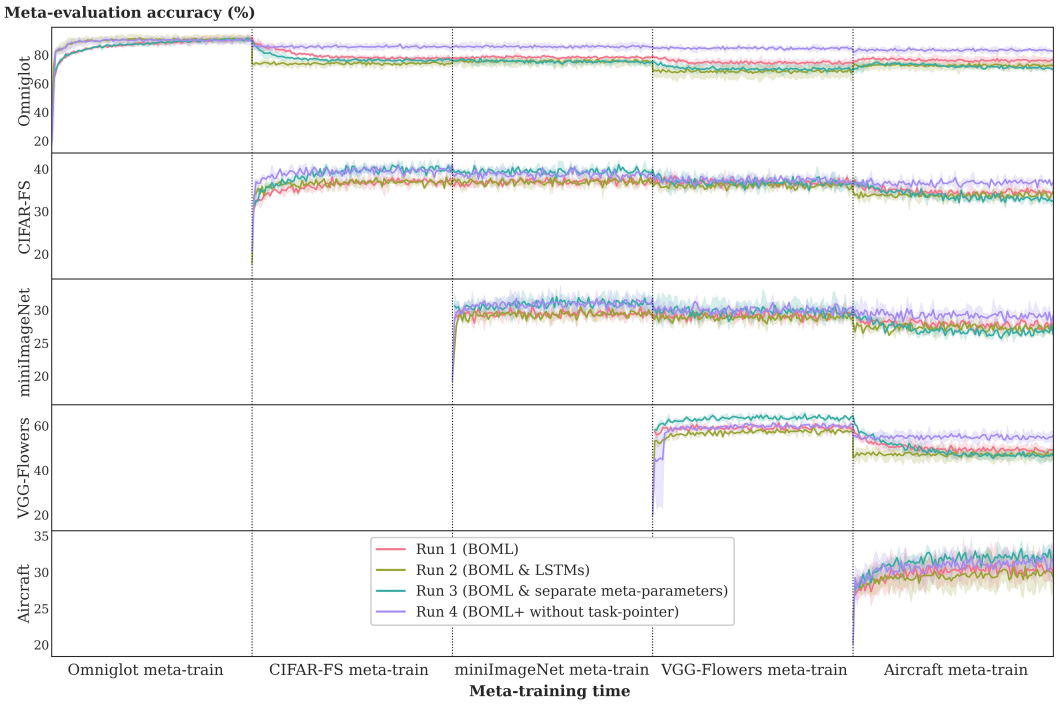

Figure 6.9: Meta-evaluation accuracy across 3 seed runs on each dataset along meta-training. Higher accuracy values indicate better results with less forgetting as we proceed to new datasets. BOML+ without task-pointer in Run R4 can retain performances on previously learned datasets since it performs best in the off-diagonal plots. Most of the diagonal plots accuracies of Run R4 are as good as the others, indicating that it learns well on new datasets too.

the performance on previously learned datasets (off-diagonal plots). When these two elements combine in Run R4 (——), it inherits the advantages of both elements, giving a method that can both learn well on new datasets and retaining performance on previously learned datasets.

Dataset	Run R4 (no task-pointer)	Run R5 (BOML+ with task-pointer)
Omniglot	85.46 ± 0.52	85.57 ± 0.52
CIFAR-FS	36.22 ± 0.60	36.31 ± 0.59
*mini*ImageNet	29.09 ± 0.45	28.81 ± 0.44
VGG-Flowers	54.49 ± 0.68	54.96 ± 0.70
Aircraft	31.93 ± 0.52	31.42 ± 0.52

Table 6.1: Meta-evaluation accuracies for Run R4 (BOML+ without task-pointer) and Run R5 (BOML+ with task-pointer) on the datasets upon the completion of meta-training across the entire pentathlon sequence of knowledge domains.

Recall that the generative classifier task-pointer exhibits a passive collaboration between experts by assigning each unseen task to its corresponding expert. We expect the trained generative classifier to correctly point each task to the expert in charge, and thus giving a performance as good as Run R4 (BOML+ without task-pointer). Since Run R4 performs best in Figure 6.9, it suffices to compare the task-pointer performance of Run R5 (BOML+) to Run R4. Table 6.1 shows that BOML+ with task-pointer performs equally as good as Run R4. This indicates that BOML+ outperforms BOML in every aspect when all newly-introduced components are implemented together in BOML+.

Dataset	Run R1 (original BOML)	Run R5 (BOML+ with task-pointer)
Omniglot	77.84 ± 1.65	85.57 ± 0.52
CIFAR-FS	33.68 ± 1.80	36.31 ± 0.59
*mini*ImageNet	28.04 ± 1.30	28.81 ± 0.44
VGG-Flowers	49.97 ± 2.22	54.96 ± 0.70
Aircraft	31.72 ± 1.93	31.42 ± 0.52

Table 6.2: Meta-evaluation accuracies for Run R1 (original BOML) and Run R5 (BOML+ with task-pointer) on various datasets upon the completion of meta-training across the entire pentathlon sequence of knowledge domains.

Table 6.2 shows the final evaluation accuracies achieved upon the completion of training across the entire sequence of knowledge domains, for a clearer comparison between BOML and its enhanced version BOML+. In this context, Run R1 represents the original BOML described in Chapter 4. We deploy the best performing BOML, which is the BOMLA algorithm, to represent BOML for baseline comparison in this experiment setting. Empirical evidence indicates that BOML+ surpasses BOML in terms of accuracy performance. Notably, BOML+ demonstrates superiority in retaining previously acquired knowledge while learning on new datasets. This improved knowledge retention capability in BOML+ can be attributed to the unique integration of distinct compartments, as previously discussed for the result in Figure 6.9. The BOML+ framework allows a dedicated internal memory for each knowledge domain. Nonetheless it is essential to acknowledge that the observed enhancement in performance

with BOML+ is relatively modest in its significance. We should thoroughly re-consider the trade-off between the increased computational demand and the marginal improvement in performance. This assessment is critical in determining the overall effectiveness and practical applicability of BOML+ in real-world scenarios.

Chapter 7

Conclusion and Discussion

7.1 Conclusion

This thesis introduced the Bayesian online meta-learning (BOML) framework with two algorithms: BOMLA and BOMVI for sequential few-shot classification problems. Our framework can overcome catastrophic forgetting in few-shot classification problems on datasets with evident distributional shift. BOML merged the BOL framework with meta-learning via Laplace approximation or variational inference. The experiments show that BOMLA and BOMVI are able to retain the few-shot classification ability when trained on sequential datasets with apparent distributional shift. This results in an ability to perform few-shot classification on multiple datasets with a single meta-learned model. BOMLA and BOMVI are also able to continually learn to few-shot classify the novel tasks, as the tasks from a stationary distribution arrive sequentially for meta-training.

BOMLA with a suitable precision-updating hyperparameter λ outperforms BOMVI in the experiments. This coincides with the fact that BOMLA has a better posterior approximation than BOMVI. The Gaussian approximate posterior of BOMLA utilises a block-diagonal precision that considers the parameter interactions within a neural network layer, whilst the Gaussian mean-field approximation of BOMVI ignores the parameter interactions in a neural network. Previous work showed that a block-diagonal covariance structure in

variational inference could not improve the performance in comparison to the mean-field approximation, due to a higher Monte Carlo estimator variance. Taking the parameter interactions into consideration is essential for a good posterior approximation, as it enables a continual learning method capable of handling more complex challenges.

We derived the necessary alterations in the Hessian approximation for BomLA, as we optimise the meta-parameters for few-shot classification instead of the usual model parameters in large-scale classification. The complete Fisher approximation for the Hessian matrices in BomLA requires extensive computation of matrix cross-terms. For a shorter sequence of knowledge domains, we showed that it is possible to simplify the Fisher calculation in BomLA without apparent performance degradation. Such a simplification is especially useful when lacking in computational resources. Nonetheless the complete Fisher approximation for BomLA is essential when dealing with a longer sequence of few-shot problems. The experiments illustrate a significant performance deterioration without using the complete BomLA Fisher approximation when handling a long sequence of few-shot classification problems.

The final part of this thesis enhanced Boml to the Boml+ framework by introducing three key elements into Boml. Firstly we introduced an automated inner loop adaptation mechanism to Boml and replaced the hand-crafted SGD quick adaptation with LSTM inner loop adaptations that are knowledge domain-specific. We enhanced the flexibility on the few-shot problem settings that the agent can consider, by including the input and output layer meta-parameters that are knowledge domain-specific into the experts' internal memories. The experiment results illustrate that these two elements together produce a framework that can both learn well on new datasets and retain performance on previously learned datasets. Finally we also introduced a generative classifier task-pointing mechanism for a passive collaboration between the experts. The task-pointer uses a small subset of examples from the arriving

unseen task to identify which expert should be responsible for adapting this task.

The enhancements in BOML+ are essential to overcome certain limitations of BOML. Each dataset in the knowledge domain sequence with evident distributional shift needs a dataset-specific quick adaptation method. When using SGD inner loop in BOML, this necessity translates to fine-tuning various hyperparameters such as the number of adaptation steps and learning rate for each knowledge domain. If each knowledge domain uses an SGD with different hyperparameters, then the BOML agent has to be informed on the knowledge domain identity when an unseen task arrives during meta-evaluation. Otherwise the agent could not identify which SGD adaptation settings should be applied. BOML+ addresses these issues by automating the SGD inner loop using LSTMs and introducing a task-pointing mechanism to identify the knowledge domain identity during meta-evaluation.

7.2 Discussion

7.2.1 Advantage of BOML and BOML+

An important reason to employ BOL in BOML and BOML+ over non-Bayesian approaches such as regret-based methods in an online setting is that BOL provides a grounded framework that suggests using the previous posterior as the prior recursively. BOL implicitly keeps a memory on previous knowledge via the posterior, in contrast to recent online meta-learning methods that explicitly accumulate previous data in a task buffer (Finn et al., 2019; Zhuang et al., 2019). Explicitly keeping a memory on previous data often triggers an important question: *how should the carried-forward data be processed in future rounds, in order to accumulate knowledge?* Finn et al. (2019) update the meta-parameters at each iteration using data sampled from the accumulated task buffer. This defeats the purpose of online learning, which by definition means to update the parameters each round using only the new data encountered.

Having to re-train on previous data to avoid forgetting also increases the

training time as the data accumulates (Finn et al., 2019; He et al., 2019). Certainly one can clamp the amount of data at some maximal limit and sample from the buffer, but the final performance of such an algorithm would be dependent on the samples being informative and of good quality which may vary across different seed runs. In contrast to memorising the datasets, having an implicit memory via the posterior in BOML and BOML+ automatically deals with the question on how to process carried-forward data and allows a better knowledge accumulation process.

BOML handles data from different knowledge domains in a genuinely sequential manner. Our framework does not require revisiting any data from previous knowledge domains when dealing with a new dataset. The previously acquired experiences from various knowledge domains are implicitly embedded in the BOML posterior of the meta-parameters. BOML+ additionally enhances the BOML framework for automation and greater flexibility. BOML+ utilises an LSTM for each knowledge domain in replacement of the BOML SGD inner loop. The BOML+ agent with a generative classifier task-pointer automatically detects the relevant LSTM for quick adaptation when a novel task arrives. We address the inflexibility of BOML by separating the input and output layers of the meta-parameters for each knowledge domain. Unlike the BOML framework, such flexibility in BOML+ enables the agent to cope with few-shot tasks of different settings. An advantage of the BOML+ framework is its highly parallelisable training process. Therefore the individual mechanisms of BOML+ can be further developed on their own without affecting the rest of the framework.

7.2.2 Disadvantage and Future Research

The enhancements in BOML+ have addressed some limitations of BOML, and the remaining are left for future development. Both BOML and BOML+ are developed for sequential few-shot classification problems. A possible future work is to extend the frameworks to a broader scope such as reinforcement learning and unsupervised learning. The current state of the BOML and BOML+ frameworks are designed to avoid catastrophic forgetting on previously learned

few-shot problems. In other words, there is no active transfer of previous knowledge when solving a new problem. For a future scope of research, we can design a framework that actively includes previous experience to aid the learning process on a new knowledge domain. BOML+ has only one expert responsible for each knowledge domain. A possible future development is to introduce a dynamic collaboration between the experts when adapting to novel tasks.

Appendix A

Hyperparameters

A.1 Triathlon and Pentathlon

Tables A.1 and A.2 are the hyperparameters used in the triathlon and pentathlon experiments.

Hyperparameter	BomLA	BomVI
Posterior regulariser λ	(various values)	-
Precision initialisation values	$10^{-4} \sim 10^{-2}$	-
Number of tasks sampled for Hessian approx.	5000	-
Covariance initialisation values	-	$\exp(-5)$
Number of Monte Carlo samples	-	20
Meta-batch size M	32	32
Number of query samples per class	15	15
Number of iterations per dataset	5000	5000
Outer loop optimiser	Adam	Adam
Outer loop learning rate	0.001	0.001
Number of tasks sampled for meta-evaluation	100	100

Table A.1: Hyperparameters for the triathlon and pentathlon experiments (same value for all datasets).

Hyperparameter	Omniglot	*mini*QuickDraw	CIFAR-FS	*mini*ImageNet	VGG-Flowers	Aircraft
Number of inner SGD steps in meta-training (k)	1	3	5	5	5	5
Inner SGD learning rate (α)	0.4	0.2	0.1	0.1	0.1	0.1
Outer learning rate decay schedule (none for BOMVI)	-	$\times$0.1 halfway	$\times$0.1 halfway	$\times$0.1 halfway	$\times$0.1 every 1000 iterations	$\times$0.1 halfway
Number of inner SGD steps in meta-evaluation	3	5	10	10	10	10

Table A.2: Hyperparameters for the triathlon and pentathlon experiments (individual datasets).

A.2 Omniglot: Sequential Tasks

Table A.3 shows the hyperparameters used in the Omniglot stationary task distribution experiment.

Hyperparameter	BOMLA	BOMVI
Posterior regulariser λ	0.01	-
Precision initialisation values	$10^{-4} \sim 10^{-2}$	-
Covariance initialisation values	-	$\exp(-10)$
Number of Monte Carlo samples	-	5
Number of mini-batches M	1	1
Number of query samples per class (meta-evaluation)	15	15
Number of epochs per task	50	50
Number of inner SGD steps in meta-training (k)	5	5
Inner SGD learning rate (α)	0.1	0.1
Outer loop optimiser	Adam	Adam
Outer loop learning rate	0.001	0.001
Number of tasks sampled for meta-evaluation	100	100
Number of inner SGD steps in meta-evaluation (k)	10	10

Table A.3: Hyperparameters for the Omniglot sequential tasks experiment.

A.3 BOML+

Hyperparameter	BOML+
Posterior regulariser λ	5
Precision initialisation values	$10^{-4} \sim 10^{-2}$
Number of tasks sampled for Hessian approx.	5000
Meta-batch size M	32
Number of query samples per class	15
Number of iterations per dataset	5000
Outer loop optimiser	Adam
Outer loop learning rate	0.001
Number of tasks sampled for meta-evaluation	100
Number of LSTM quick adaptation steps	1
Number of importance samples drawn for task-pointer S	100

Table A.4: Hyperparameters for the BOML+ experiments.

Table A.4 records the hyperparameters used in the BOML+ experiments. The dataset-specific hyperparameters when learning the meta-parameters are same as that in Table A.2 for the pentathlon experiments.

Bibliography

M. Andrychowicz, M. Denil, S. Gómez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. In *Advances in Neural Information Processing Systems 29*, 2016.

L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-Learning with Differentiable Closed-Form Solvers. In *International Conference on Learning Representations*, 2019.

A. Botev, H. Ritter, and D. Barber. Practical Gauss-Newton Optimisation for Deep Learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

G. Denevi, D. Stamos, C. Ciliberto, and M. Pontil. Online-Within-Online Meta-Learning. In *Advances in Neural Information Processing Systems 32*, 2019.

J. S. Denker and Y. LeCun. Transforming Neural-Net Output Levels to Probability Distributions. In *Advances in Neural Information Processing Systems 3*, 1991.

C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

C. Finn, K. Xu, and S. Levine. Probabilistic Model-Agnostic Meta-Learning. In *Advances in Neural Information Processing Systems 31*, 2018.

C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online Meta-Learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv preprint*, arXiv:1312.6211, 2013.

J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. Turner. Meta-Learning Probabilistic Inference for Prediction. In *International Conference on Learning Representations*, 2019.

E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths. Recasting Gradient-Based Meta-Learning as Hierarchical Bayes. In *International Conference on Learning Representations*, 2018.

R. Grosse and J. Martens. A Kronecker-Factored Approximate Fisher Matrix for Convolution Layers. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

D. Ha and D. Eck. A Neural Representation of Sketch Drawings. *arXiv preprint*, arXiv:1704.03477, 2017.

J. Harrison, A. Sharma, C. Finn, and M. Pavone. Continuous Meta-Learning without Tasks. *arXiv preprint*, arXiv:1912.08866, 2019.

X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. Teh, and R. Pascanu. Task Agnostic Continual Learning via Meta Learning. *arXiv preprint*, arXiv:1906.05201, 2019.

G. Jerfel, E. Grant, T. Griffiths, and K. A. Heller. Reconciling Meta-Learning and Continual Learning with Online Mixtures of Tasks. In *Advances in Neural Information Processing Systems 32*, 2019.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems 28*, 2015.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 2017.

G. Koch, R. Zemel, and R. Salakhutdinov. Siamese Neural Networks for One-Shot Image Recognition. In *32th International Conference on Machine Learning Deep Learning Workshop*, 2015.

B. Lake, R. Salakhutdinov, J. Gross, and J.B. Tenenbaum. One Shot Learning of Simple Visual Concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 2011.

S. Lee, J. Kim, J. Jun, J. Ha, and B. Zhang. Overcoming Catastrophic Forgetting by Incremental Moment Matching. In *Advances in Neural Information Processing Systems 30*, 2017.

F. Li, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2004.

K. Li and J. Malik. Learning to Optimize. In *International Conference on Learning Representations*, 2017.

D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 1992.

S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-Grained Visual Classification of Aircraft. *arXiv preprint*, arXiv:1306.5151, 2013.

J. Martens and R. Grosse. Optimizing Neural Networks with Kronecker-Factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from One Example Through Shared Densities on Transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational Continual Learning. In *International Conference on Learning Representations*, 2018.

A. Nichol, J. Achiam, and J. Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint*, arXiv:1803.02999, 2018.

M. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

M. Opper. A Bayesian Approach to Online Learning. In *Online Learning in Neural Networks*. Cambridge University Press, 1998.

K. Osawa, Y. Tsuji, Y. Ueno, A. Naruse, C. Foo, and R. Yokota. Scalable and Practical Natural Gradient for Large-Scale Deep Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

S. Ravi and A. Beatson. Amortized Bayesian Meta-Learning. In *International Conference on Learning Representations*, 2019.

S. Ravi and H. Larochelle. Optimization as a Model for Few-Shot Learning. In *International Conference on Learning Representations*, 2017.

H. Ritter, A. Botev, and D. Barber. Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. In *Advances in Neural Information Processing Systems 31*, 2018a.

H. Ritter, A. Botev, and D. Barber. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*, 2018b.

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 1951.

M. Rosa, O. Afanasjeva, S. Andersson, J. Davidson, N. Guttenberg, P. Hlubucek, M. Poliak, J. Vitku, and J. Feyereisl. BADGER: Learning to (Learn [Learning Algorithms] through Multi-Agent Communication). *arXiv preprint*, arXiv:1912.01513, 2019.

A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell. Meta-Learning with Latent Embedding Optimization. In *International Conference on Learning Representations*, 2019.

A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-Learning with Memory-Augmented Neural Networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

J. Schmidhuber. Evolutionary Principles in Self-Referential Learning. On Learning How to Learn: The Meta-Meta-Meta...-Hook. Diploma thesis, Institut für Informatik, Technische Universität München, 1987.

S. Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

J. Snell, K. Swersky, and R. Zemel. Prototypical Networks for Few-Shot Learning. In *Advances in Neural Information Processing Systems 30*, 2017.

S. Thrun and L. Pratt. *Learning to Learn: Introduction and Overview*. Springer, Boston, MA, 1998.

B. L. Trippe and R. E. Turner. Overpruning in Variational Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 30 – Advances in Approximate Bayesian Inference Workshop*, 2017.

G. M. van de Ven, Z. Li, and A. S. Tolias. Class-Incremental Learning With Generative Classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29*, 2016.

J. Wen, Y. Cao, and R. Huang. Few-Shot Self Reminder to Overcome Catastrophic Forgetting. *arXiv preprint*, arXiv:1812.00543, 2018.

P. Yap, H. Ritter, and D. Barber. Bayesian Online Meta-Learning with Laplace Approximation. In *International Conference on Learning Representations – Beyond Tabula Rasa in RL (BeTR-RL) Workshop*, 2020.

P. Yap, H. Ritter, and D. Barber. Addressing Catastrophic Forgetting in Few-Shot Problems. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian Model-Agnostic Meta-Learning. In *Advances in Neural Information Processing Systems 31*, 2018.

F. Zenke, B. Poole, and S. Ganguli. Continual Learning through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Z. Zhuang, Y. Wang, K. Yu, and S. Lu. No-Regret Non-Convex Online Meta-Learning. *arXiv preprint*, arXiv:1910.10196, 2019.

M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.