Amortized Bayesian Meta-Learning

Sachin Ravi & Alex Beatson
Department of Computer Science, Princeton University
`{sachinr,abeatson}@cs.princeton.edu`

Abstract

Meta-learning has proven to be a successful strategy in attacking problems in supervised learning and reinforcement learning that involve small amounts of data. State-of-the-art solutions involve learning an initialization and/or learning algorithm using a set of training episodes so that the meta-learner can generalize to an evaluation episode quickly. These methods perform well but often lack good quantification of uncertainty, which can be vital to real-world applications when data is lacking. We propose a meta-learning method which efficiently amortizes hierarchical variational inference across tasks, learning a prior distribution over neural network weights so that a few steps of Bayes by Backprop will produce a good task-specific approximate posterior. We show that our method produces good uncertainty estimates on contextual bandit and few-shot learning benchmarks.

1 Introduction

Deep learning has achieved success in domains that involve a large amount of labeled data [1, 2] or training samples [3, 4]. However, a key aspect of human intelligence is our ability to learn new concepts from only a few experiences. It has been hypothesized that this skill arises from accumulating prior knowledge and using it appropriately in new settings [5]. Meta-learning attempts to endow machine learning models with the same ability by training a meta-learner to perform well on a distribution of training tasks. The meta-learner is then applied to an unseen task, usually assumed to be drawn from a task distribution similar to the one used for training, with the hope that it can learn to solve the new task efficiently. Though performance on meta-learning benchmarks has greatly increased in the past few years, it is unclear how well the associated methods would perform in real-world settings, where the relationship between training and evaluation tasks could be tenuous. For success in the wild, in addition to good predictive accuracy, it is also important for meta-learning models to have good predictive uncertainty - to express high confidence when a prediction is likely to be correct but display low confidence when a prediction could be unreliable.

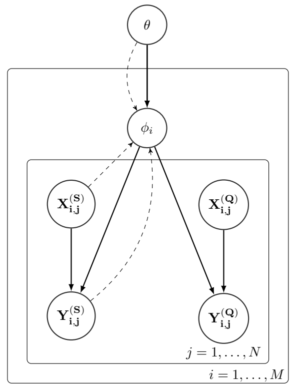
θ
ϕ_i
$\mathbf{X}^{(\mathbf{S})}_{\mathbf{i,j}}$
$\mathbf{X}^{(\mathbf{Q})}_{\mathbf{i,j}}$
$\mathbf{Y}^{(\mathbf{S})}_{\mathbf{i,j}}$
$\mathbf{Y}^{(\mathbf{Q})}_{\mathbf{i,j}}$
$j = 1, \ldots, N$
$i = 1, \ldots, M$

Figure 1: Graphical model for proposed amortized variational inference scheme. Dotted lines denote variational approximations

Bayesian methods offer a principled framework to reason about uncertainty, and approximate Bayesian methods have been used to provide deep learning models with predictive uncertainty [6, 7]. Accordingly, we consider meta-learning under a bayesian framework in order to transfer the aforementioned benefits to our setting. Specifically, we extend the work of [8], who considered hierarchical variational inference for meta-learning. In this paper, we show how the meta-learning framework of [9] can be used to efficiently amortize variational inference for the Bayesian model of [8] in order to combine the former's flexibility and scalability with the latter's uncertainty quantification.

2 Meta-Learning via Hierarchical Variational Inference

We first start by reviewing the hierarchical variational bayes formulation used in [8] for meta-learning. Assume we observe data from M episodes, where the i^{th} episode consists of data $\mathcal{D}_i$ containing N data items, meaning $\mathcal{D}_i = \{(\mathbf{X}_{i,j}, \mathbf{Y}_{i,j})\}_{j=1}^{N}$. We assume a hierarchical model with global latent variable θ and episode-specific variables ϕ_i, $i = 1, \dots M$ (see Figure 1). Hierarchical variational inference can then be used to lower bound the likelihood of the data:

$$\log\left[\prod_{i=1}^{M} p(\mathcal{D}_i)\right] = \log\left[\int p(\theta)\left[\prod_{i=1}^{M}\int p(\mathcal{D}_i|\phi_i)p(\phi_i|\theta)\,d\phi_i\right]d\theta\right]$$
$$\geq \mathbb{E}_{q(\theta;\psi)}\left[\sum_{i=1}^{M}\mathbb{E}_{q(\phi_i;\lambda_i)}\left[\log p(\mathcal{D}_i|\phi_i)\right] - \mathrm{KL}(q(\phi_i;\lambda_i)\|p(\phi_i|\theta))\right] - \mathrm{KL}(q(\theta;\psi)\|p(\theta))$$

Here, ψ and $\lambda_1, \dots, \lambda_M$ are the variational parameters of the approximate posteriors over the global latent variables θ and the local latent variables $\phi_1, \dots, \phi_M$, respectively. Variational inference thus involves solving the following optimization problem:

$$\equiv \underset{\psi,\lambda_1\dots,\lambda_M}{\arg\min}\ \mathbb{E}_{q(\theta;\psi)}\left[\sum_{i=1}^{M} -\mathbb{E}_{q(\phi_i;\lambda_i)}\left[\log p(\mathcal{D}_i|\phi_i)\right] + \mathrm{KL}(q(\phi_i;\lambda_i)\|p(\phi_i|\theta))\right] + \mathrm{KL}(q(\theta;\psi)\|p(\theta)) \tag{1}$$

In [8], this optimization problem is solved via mini-batch gradient descent on the objective starting from random initialization for all variational parameters. They maintain distinct variational parameters λ_i for each episode i, each of which indexes a distribution over episode-specific weights $q(\phi_i; \lambda_i)$.

3 Amortized Bayesian Meta-Learning

Learning local variational parameters λ_i for a large number of episodes M becomes difficult as M grows due to the costs of storing and computing each λ_i. These problems are compounded when each ϕ_i is the weight of a deep neural network and each λ_i are variational parameters of the weight distribution. Instead of maintaining M different variational parameters λ_i indexing distributions over neural network weights ϕ_i, we compute λ_i on the fly with amortized variational inference (AVI), where a global learned model is used to predict λ_i from $\mathcal{D}_i$. A popular use of AVI is training a variational autoencoder [10], where a trained encoder network produces the variational parameters for each data point. Rather than training an encoder to predict λ_i given the episode, we show that inference can be amortized by finding a good initialization. We represent the variational parameters for each episode as the output of several steps of gradient descent from a global initialization.

Let $\mathcal{L}_{\mathcal{D}_i}(\lambda, \theta) = -\mathbb{E}_{q(\phi_i;\lambda)}\left[\log p(\mathcal{D}_i|\phi_i)\right] + \mathrm{KL}(q(\phi_i;\lambda)\|p(\phi_i|\theta))$ be the part of the objective corresponding to data $\mathcal{D}_i$. Let the procedure $SGD_K(\mathcal{D}, \lambda^{(init)}, \theta)$ represent the variational parameters produced after K steps of gradient descent on the objective $\mathcal{L}_{\mathcal{D}}(\lambda, \theta)$ with respect to λ starting at the initialization $\lambda^{(0)} = \lambda^{(init)}$ and where θ is held constant i.e.:

1. $\lambda^{(0)} = \lambda^{(init)}$
2. for $k = 0, \dots, K-1$, set
   $\lambda^{(k+1)} = \lambda^{(k)} - \alpha\nabla_{\lambda^{(k)}}\mathcal{L}_{\mathcal{D}}(\lambda^{(k)}, \theta)$

We represent the variational distribution for each dataset $q_\theta(\phi_i|D_i)$ in terms of the local variational parameters λ_i produced after K steps of gradient descent on the loss for dataset D_i, starting from the global initialization θ:

$$q_\theta(\phi_i|\mathcal{D}_i) = q(\phi_i; SGD_K(\mathcal{D}_i, \theta, \theta)).$$

Note that θ here serves as both the global initialization of local variational parameters and the parameters of the prior $p(\phi\,|\,\theta)$. We could pick a separate prior and global initialization, but we found tying the prior and initialization did not seem to have a negative affect on performance, while significantly reducing the number of total parameters necessary. With this form of the variational distribution, this turns the optimization problem of (1) into:

$$\arg\min_{\psi} \mathbb{E}_{q(\theta;\psi)} \left[\sum_{i=1}^{M} -\mathbb{E}_{q_\theta(\phi_i|\mathcal{D}_i)} \left[\log p(\mathcal{D}_i|\phi_i) \right] + \text{KL}(q_\theta(\phi_i|\mathcal{D}_i) \| p(\phi_i|\theta)) \right] + \text{KL}(q(\theta;\psi) \| p(\theta)). \tag{2}$$

Because each $q_\theta(\phi_i|D_i)$ depends on ψ via θ (the initialization for the variational parameters before performing K steps of gradient descent), we can also backpropagate through the computation of q via the gradient descent process to compute updates for ψ. This corresponds to learning a global initialization of the variational parameters such that a few steps of gradient descent will produce a good local variational distribution for any given dataset.

We assume a setting where $M >> N$, i.e. we have many more episodes than data points within each episode. Accordingly, we are most interested in quantifying uncertainty within a given episode and desire accurate predictive uncertainty in $q_\theta(\phi_i|D_i)$. We assume that uncertainty in the global latent variables θ should be low due to the large number of episodes, and therefore use a point estimate for the global latent variables, letting $q(\theta;\psi)$ be a dirac delta function $q(\theta) = \mathbb{1}\{\theta = \theta^*\}$. This removes the need for global variational parameters ψ and simplifies our optimization problem to:

$$\arg\min_{\theta} \left[\sum_{i=1}^{M} -\mathbb{E}_{q_\theta(\phi_i|\mathcal{D}_i)} \left[\log p(\mathcal{D}_i|\phi_i) \right] + \text{KL}(q_\theta(\phi_i|\mathcal{D}_i) \| p(\phi_i|\theta)) \right] + \text{KL}(q(\theta) \| p(\theta)), \tag{3}$$

where θ^* is the solution to the above optimization problem. Note that $KL(q(\theta)\|p(\theta))$ term can be computed even when $q(\theta) = \mathbb{1}\{\theta = \theta^*\}$, as $KL(q(\theta)||p(\theta)) = \mathbb{E}_{\theta \sim q}[-\log p(\theta)] = -\log p(\theta^*)$.

3.1 Amortized Variational Inference using only support set

In the few-shot learning problem, we must consider train and test splits for each dataset in each episode. We will call the training examples in each dataset the *support set* and the test examples in each dataset the *query set* . Thus, $\mathcal{D}_i = \mathcal{D}_i^{(S)} \cup \mathcal{D}_i^{(Q)}$, where $\mathcal{D}_i^{(S)} = \{(\mathbf{X}_{i,j}^{(S)}, \mathbf{Y}_{i,j}^{(S)})\}_{j=1}^{N}$ and $\mathcal{D}_i^{(Q)} = \{(\mathbf{X}_{i,j}^{(Q)}, \mathbf{Y}_{i,j}^{(Q)})\}_{j=1}^{N'}$, and the assumption is that during evaluation, we are only given $\mathcal{D}_i^{(S)}$ to determine our variational distribution $q(\phi_i)$ and measure the performance of the model by evaluating the variational distribution on corresponding $\mathcal{D}_i^{(Q)}$. In order to match what is done during training and evaluation, we consider a modified version of the objective of (3) that incorporates this support and query set split, giving us the following objective:

$$\arg\min_{\theta} \left[\sum_{i=1}^{M} -\mathbb{E}_{q_\theta\left(\phi_i|\mathcal{D}_i^{(S)}\right)} \left[\log p(\mathcal{D}_i|\phi_i) \right] + \text{KL}\left(q_\theta\left(\phi_i|\mathcal{D}_i^{(S)}\right) \| p(\phi_i|\theta) \right) \right] + \text{KL}(q(\theta) \| p(\theta)), \tag{4}$$

where $q_\theta\left(\phi_i|\mathcal{D}_i^{(S)}\right) = q\left(\phi_i; SGD_K\left(\mathcal{D}_i^{(S)}, \theta, \theta\right)\right)$. Note that the objective in this optimization problem still serves as a lower bound to the likelihood; conditioning on less information potentially gives us a weaker lower bound for all the training datasets, but we found empirically that the performance during evaluation was better using this type of conditioning since there is no mismatch between training vs evaluation.

4 Evaluation

We evaluate our proposed model on experiments involving contextual bandits and involving measuring uncertainty in few-shot learning benchmarks (see supplementary material for these results). We compare our method primarily against MAML, where MAML is trained by maximum likelihood estimation of the query set given a fixed number of updates on the support set, causing it to often display overconfidence in the settings we consider. For the contextual bandit task, we consider the wheel bandit problem introduced in [11]. The wheel bandit problem is a synthetic contextual bandit problem with a scalar hyperparameter that allows us to control the amount of exploration required to be successful at the problem. We refer the reader to [11] for an exact description of the problem. Thompson Sampling [12] is a classic approach to tackling the exploration-exploitation trade-off

δ	**0.5**	**0.7**	**0.9**	**0.95**	**0.99**
$\mathbf{n = 80,000}$					
Uniform	100 $\pm$ 0.08	100 $\pm$ 0.09	100 $\pm$ 0.25	100 $\pm$ 0.37	100 $\pm$ 0.78
NeuralLinear	0.95 $\pm$ 0.02	1.60 $\pm$ 0.03	4.65 $\pm$ 0.18	9.56 $\pm$ 0.36	49.63 $\pm$ 2.41
MAML	**0.20** $\pm$ 0.002	0.34 $\pm$ 0.004	1.02 $\pm$ 0.01	2.10 $\pm$ 0.03	9.81 $\pm$ 0.27
Our Model	0.22 $\pm$ 0.002	**0.29** $\pm$ 0.003	**0.66** $\pm$ 0.008	**1.03** $\pm$ 0.01	**4.66** $\pm$ 0.10
$\mathbf{n = 2,000}$					
Uniform	100 $\pm$ 0.25	100 $\pm$ 0.42	100 $\pm$ 0.79	100 $\pm$ 1.15	100 $\pm$ 1.88
MAML	1.79 $\pm$ 0.04	2.10 $\pm$ 0.04	6.08 $\pm$ 0.47	16.80 $\pm$ 1.30	55.53 $\pm$ 2.18
Our Model	**1.36** $\pm$ 0.03	**1.59** $\pm$ 0.04	**3.51** $\pm$ 0.17	**7.21** $\pm$ 0.41	**35.04** $\pm$ 1.93

Table 1: Cumulative regret results on the wheel bandit problem with varying δ values. Results are normalized with the performance of the uniform agent (as was done in [11]) and results shown are mean and standard error for cumulative regret calculated across 50 trials

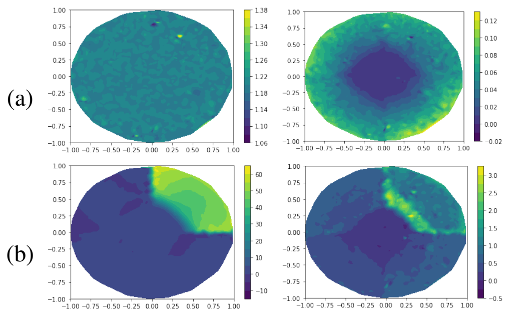

Figure 2: Visualization of arm rewards according to prior distribution of our model. (a) expectation and standard-deviation of low-reward arm (computed by sampling weights from the prior) evaluated on points on unit circle. (b) expectation and standard-deviation of one of the high-reward arms computed in same way as for low-reward arm.

involved in bandit problems that requires a posterior distribution over reward functions. At each time step an action is chosen by sampling a model from the posterior and acting optimally with respect to the sampled reward function. The posterior distribution over reward functions is then updated based on the observed reward for the action. When the posterior initially has high variance because of lack of data, Thompson Sampling explores more and turns to exploitation only when the posterior distribution becomes more certain about the rewards.

We use the setup described in [13] to apply meta-learning methods to the wheel bandit problem. For meta-learning methods, there is a pre-training phase in which training episodes consist of randomly generated data across δ values from wheel bandit task. Then, these methods are evaluated using Thompson sampling on problems defined by specific values of δ. Unlike the models in [11], our model and MAML have a chance to develop some sort of prior that they can utilize to get a head start. We can straightforwardly apply Thompson sampling in our model using the approximate posterior at each time step whereas for MAML we just take a greedy action at each time step given the current model parameters. The results of evaluating the meta-learning methods after the pre-training phase are shown in Table 1. We vary the number of contexts and consider $n = 80,000$ (which was used in [11]) and $n = 2,000$ (to see how the models perform under fewer time steps). We can see that as δ increases and more exploration is required to be successful at the problem, our model has a increasingly better cumulative regret when compared to MAML. Lastly, we visualize the learned prior $p(\phi \mid \theta)$ in Figure 2. We can see that the standard deviation of the central low-reward arm is small everywhere, as there is little reward variability in this arm across δ values. For the high-reward arm in the upper-right corner, we see that the standard deviation is high at the edges of the area in which this arm can give high reward. This variation in the posterior indicates the region in which we would like to target our exploration to figure out what δ value we are currently facing.

5 Conclusion

We described a method to efficiently use hierarchical variational inference to learn a meta-learning model that is scalable across many training episodes and large networks. The method corresponds to learning a prior distribution over the network weights so that a few gradient steps will produce a good approximate posterior. Through various experiments we show that our model is able to reason effectively about uncertainty in contextual bandit and few-shot learning tasks.

References

[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[2] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE, 2017.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[5] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[6] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059, 2016.

[7] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227, 2017.

[8] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *Proceedings of the 35th International Conference on Machine Learning*, pages 205–214, 2018.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[11] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[12] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[13] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[14] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

[15] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *arXiv preprint arXiv:1806.03836*, 2018.

[16] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[17] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1613–1622. JMLR. org, 2015.

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

[19] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[20] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[21] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

[22] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 2015, page 2901. NIH Public Access, 2015.

6 Supplementary Material

6.1 Related Work

Our method is very closely related to [9] and recent work proposing Bayesian variants of MAML. [14] provided the first Bayesian variant of MAML using the Laplace approximation. In concurrent work to this paper, [15] and [16] propose Bayesian variants of MAML with different approximate posteriors. [16] approximates MAP inference of the task-specific weights ϕ_i, and maintain uncertainty only in the global model θ. Our paper, however, considers tasks in which it is important to quantify uncertainty in task-specific weights - such as contextual bandits and few-shot learning. [15] focuses on uncertainty in task-specific weights, as we do. They use a point estimate for all layers except the final layer of a deep neural network, and use Stein Variational Gradient Descent to approximate the posterior over the weights in the final layer with an ensemble. This avoids placing Gaussian restrictions on the approximate posterior; however, the posterior's expressiveness is dependant on the number of particles in the ensemble, and memory and computation requirements scale linearly and quadratically in the size of the ensemble, respectively.

6.2 Application Details

With the objective (4) in mind, we give more details on how we implement the specific model. We begin with the distributional forms of the priors and posteriors. The formulation given above is flexible but we consider fully factorized Gaussian distributions for ease of implementation and experimentation. We let $\theta = \{\boldsymbol{\mu_\theta}, \boldsymbol{\sigma^2_\theta}\}$, where $\boldsymbol{\mu_\theta} \in \mathbb{R}^D$ and $\boldsymbol{\sigma^2_\theta} \in \mathbb{R}^D$ represent the mean and variance for each neural network weight, respectively. Then, $p(\phi_i|\theta)$ is:

$$p(\phi_i|\theta) = \mathcal{N}(\phi_i; \boldsymbol{\mu_\theta}, \boldsymbol{\sigma^2_\theta}\mathbf{I})$$

Then, $q_\theta\left(\phi_i|\mathcal{D}_i^{(S)}\right)$ is the following:

$$\{\boldsymbol{\mu_\lambda^{(K)}}, \boldsymbol{\sigma^2{}_\lambda^{(K)}}\} = SGD_K(\mathcal{D}_i^{(S)}, \theta, \theta)$$
$$q_\theta\left(\phi_i|\mathcal{D}_i^{(S)}\right) = \mathcal{N}\left(\phi_i; \boldsymbol{\mu_\lambda^{(K)}}, \boldsymbol{\sigma^2{}_\lambda^{(K)}}\right).$$

We let the prior $p(\theta)$ be:

$$p(\theta) = \mathcal{N}(\boldsymbol{\mu}; \mathbf{0}, \mathbf{I}) \cdot \prod_{l=1}^{D} \text{Gamma}(\tau_l; a_0, b_0),$$

where $\tau_l = \frac{1}{\boldsymbol{\sigma^2}_l}$ is the precision and a_0 and b_0 are the alpha and beta parameters for the gamma distribution. Note that with the defined distributions, the SGD process here corresponds to performing Bayes by Backprop [17] with the learned prior $p(\phi_i|\theta)$.

Optimization of (4) is done via mini-batch gradient descent, where we average gradients over multiple episodes at a time. The pseudo-code for training and evaluation are given in Algorithms 1 and 2 in the appendix. The KL-divergence terms are calculated analytically whereas the expectations are approximated by averaging over a number of samples from the approximate posterior, as has been done in previous work [10, 17]. The gradient computed for this approximation naively can have high variance, which can significantly harm the convergence of gradient descent [18]. Previous work has explored reducing the variance of gradients involving stochastic neural networks and we found this crucial to training the networks we use. Specifically, we use the Local Reparametrization Trick [18] for fully-connected layers and Flipout [19] for convolutional layers. Variance reduction is particularly important to the performance of our model as we perform stochastic optimization to obtain the posterior $q\left(\phi|D^{(S)}\right)$ at evaluation-time also. Lastly, note that we can easily generate multiple weight samples in the few-shot learning setting simply by replicating the data in each episode since we only have a few examples per class making up each episode.

6.3 Pseudocode

In algorithms 1 and 2 we give the pseudocode for meta-training and meta-evaluation, respectively. Note that in practice, we do not directly parameterize variance parameters but instead parameterize the standard deviation as the output of softplus function as was done in [17] so that it is always non-negative.

Algorithm 1 BBB: Meta-training

```
Input: Number of update steps $K$
Number of total episodes $M$
Inner learning rate $\alpha$
Outer learning rate $\beta$
 1: Initialize $\theta = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma^2_\theta}\}$
 2: $p(\theta) = \mathcal{N}(\boldsymbol{\mu}; \mathbf{0}, \mathbf{I}) \cdot \prod_{l=1}^{D} \text{Gamma}(\tau_l; a_0, b_0)$
 3: for $t = 0, \ldots, M$ do
 4:     $\mathcal{D}_i = \{\mathcal{D}_i^{(S)}, \mathcal{D}_i^{(Q)}\} \leftarrow$ sample random episode
 5:     $\boldsymbol{\mu^{(0)}_\lambda} \leftarrow \boldsymbol{\mu}_\theta; \boldsymbol{\sigma^2}^{(0)}_{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\sigma^2_\theta}$
 6:     for $k = 0, \ldots, K$ do
 7:         $\lambda^{(k)} \leftarrow \{\boldsymbol{\mu^{(k)}_\lambda}, \boldsymbol{\sigma^{(k)}_\lambda}\}$
 8:         $\boldsymbol{\mu^{(k+1)}_\lambda} \leftarrow \boldsymbol{\mu^{(k)}_\lambda} - \alpha \nabla_{\boldsymbol{\mu^{(k)}_\lambda}} \mathcal{L}_{\mathcal{D}_i^{(S)}}(\lambda^{(k)}, \theta)$
 9:         $\boldsymbol{\sigma^2}^{(k+1)}_{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\sigma^2}^{(k)}_{\boldsymbol{\lambda}} - \alpha \nabla_{\boldsymbol{\sigma^2}^{(k)}_{\boldsymbol{\lambda}}} \mathcal{L}_{\mathcal{D}_i^{(S)}}(\lambda^{(k)}, \theta)$
10:     end for
11:
12:     $\lambda^{(K)} \leftarrow \{\boldsymbol{\mu^{(K)}_\lambda}, \boldsymbol{\sigma^2}^{(K)}_{\boldsymbol{\lambda}}\}$
13:     $q(\theta) = \mathbb{1}\{\boldsymbol{\mu} = \boldsymbol{\mu}_\theta\} \cdot \mathbb{1}\{\boldsymbol{\sigma^2} = \boldsymbol{\sigma^2_\theta}\}$
14:     $\boldsymbol{\mu_\theta} \leftarrow \boldsymbol{\mu_\theta} - \beta \nabla_{\boldsymbol{\mu_\theta}} \left[\mathcal{L}_{\mathcal{D}_i}(\lambda^{(K)}, \theta) + \frac{1}{M}\text{KL}(q(\theta) \| p(\theta))\right]$
15:     $\boldsymbol{\sigma^2_\theta} \leftarrow \boldsymbol{\sigma^2_\theta} - \beta \nabla_{\boldsymbol{\sigma^2_\theta}} \left[\mathcal{L}_{\mathcal{D}_i}(\lambda^{(K)}, \theta) + \frac{1}{M}\text{KL}(q(\theta) \| p(\theta))\right]$
16: end for
```

Algorithm 2 BBB: Meta-evaluation

```
Input: Dataset $\mathcal{D} = \{\mathcal{D}^{(S)}, \mathcal{D}^{(Q)}\}$
Parameters $\theta = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma^2_\theta}\}$
Inner learning rate $\alpha$
 1: $\boldsymbol{\mu^{(0)}_\lambda} \leftarrow \boldsymbol{\mu}_\theta; \boldsymbol{\sigma^2}^{(0)}_{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\sigma^2_\theta}$
 2: for $k = 0, \ldots, K$ do
 3:     $\lambda^{(k)} \leftarrow \{\boldsymbol{\mu^{(k)}_\lambda}, \boldsymbol{\sigma^{(k)}_\lambda}\}$
 4:     $\boldsymbol{\mu^{(k+1)}_\lambda} \leftarrow \boldsymbol{\mu^{(k)}_\lambda} - \alpha \nabla_{\boldsymbol{\mu^{(k)}_\lambda}} \mathcal{L}_{\mathcal{D}^{(S)}}(\lambda^{(k)}, \theta)$
 5:     $\boldsymbol{\sigma^2}^{(k+1)}_{\boldsymbol{\lambda}} \leftarrow \boldsymbol{\sigma^2}^{(k)}_{\boldsymbol{\lambda}} - \alpha \nabla_{\boldsymbol{\sigma^2}^{(k)}_{\boldsymbol{\lambda}}} \mathcal{L}_{\mathcal{D}^{(S)}}(\lambda^{(k)}, \theta)$
 6: end for
 7:
 8: $q_\theta\left(\phi \mid D^{(S)}\right) = \mathcal{N}\left(\phi; \boldsymbol{\mu^{(K)}_\lambda}, \boldsymbol{\sigma^2}^{(K)}_{\boldsymbol{\lambda}}\right)$
 9: Evaluate $D^{(Q)}$ using $\mathbb{E}_{q_\theta(\phi \mid D^{(S)})}\left[p(D^{(Q)} \mid \phi)\right]$
```

6.4 Additional Results

We describe additional results involving two few-shot learning benchmarks: **CIFAR-100** and ***mini*ImageNet**, where both datasets consist of 100 classes and 600 images per class and where CIFAR-100 has images of size 32×32 and *mini*ImageNet has images of size 84×84. We split the 100 classes into separate sets of 64 classes for training, 16 classes for validation, and 20 classes for testing for both of the datasets (using the split from [20] for *mini*ImageNet, while using our own for CIFAR-100 as a commonly used split does not exist). For both benchmarks, we use the convolutional architecture used in [9], which consists of 4 convolutional layers, each with 32 filters, and a fully-connected layer mapping to the number of classes on top. For the few-shot learning

CIFAR-100	1-shot	
	5-class	10-class
MAML (ours)	51.6 $\pm$0.74	36.2 $\pm$0.46
Our Model	49.5 $\pm$0.74	35.7 $\pm$0.47

*mini*ImageNet	1-shot, 5-class
MAML (ours)	47.0 $\pm$0.59
Our Model	45.0 $\pm$0.60

Table 2: Few-shot classification accuracies with 95% confidence intervals on CIFAR-100 and *mini*ImageNet.

experiments, we found it necessary to downweight the inner KL term for better performance in our model.

While we focus on predictive uncertainty, we start by comparing classification accuracy of our model compared to MAML. We consider 1-shot, 5-class and 1-shot, 10-class classification on CIFAR-100 and 1-shot, 5-class classification on *mini*ImageNet, with results given in Table 2. For both datasets, we compare our model with our own re-implementation of MAML. Note that the accuracy and confidence interval for our implementation of MAML for *mini*ImageNet are smaller because we use a bigger query set for test episodes (15 examples per class vs 1 example per class) and average across more test episodes (1000 vs 600) compared to [9], respectively. Our model achieves comparable to a little worse accuracy than MAML on the considered benchmarks.

To measure the predictive uncertainty of the models, we first compute reliability diagrams [21] across many different test episodes for both models. Reliability diagrams visually measure how well calibrated the predictions of a model are by plotting the expected accuracy as a function of the confidence of the model. A well-calibrated model will have its bars align more closely with the diagonal line, as it indicates that the probability associated with a predicted class label corresponds closely with how likely the prediction is to be correct. We also show the Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) of all models, which are two quantitative ways to measure model calibration [22, 21]. ECE is a weighted average of each bin's accuracy-to-confidence difference whereas MCE is the worst-case bin's accuracy-to-confidence difference. Reliability diagrams and associated error scores are shown in Figure 3. We see that across different tasks and datasets, the reliability diagrams and error scores reflect the fact that our model is always better calibrated on evaluation episodes compared to MAML.

Another way we can measure the quality of the predictive uncertainty of a model is by measuring its confidence on out-of-distribution examples from unseen classes. This tests the model's ability to be uncertain on examples it clearly does not know how to classify. One method to visually measure this is by plotting the empirical CDF of a model's entropies on these out-of-distribution examples [7]. A model represented by a CDF curve that is towards the bottom-right is preferred, as it indicates that the probability of observing a high confidence prediction from the model is low on an out-of-distribution example. We can plot the same type of curve in our setting by considering the model's confidence on out-of-episode examples for each test episode. Empirical CDF curves for both MAML and our model are shown in Figure 4. We see that in general our model computes better uncertainty estimates than MAML, as the probability of a low entropy prediction is always smaller.

Lastly, we visualize the prior distribution $p(\phi \mid \theta)$ that has been learned in tasks involving deep convolutional networks. We show the standard deviations of randomly selected filters from the first convolutional layer to the last convolutional layer from our CIFAR-100 network trained on 1-shot, 5-class task in Figure 5. Interestingly, the standard deviation of the prior for the filters increases as we go higher up in the network. This pattern reflects the fact that across the training episodes the prior can be very confident about the lower-level filters, as they capture general, useful lower-level features and so do not need to be modified as much on a new episode. The standard deviation for the higher-level filters is higher, reflecting that fact that these filters need to be fine-tuned to the labels present in the new episode. This variation in the standard deviation represents different learning speeds across the network on a new episode, indicating which type of weights are general and which type of weights need to be quickly modified to capture the new data.

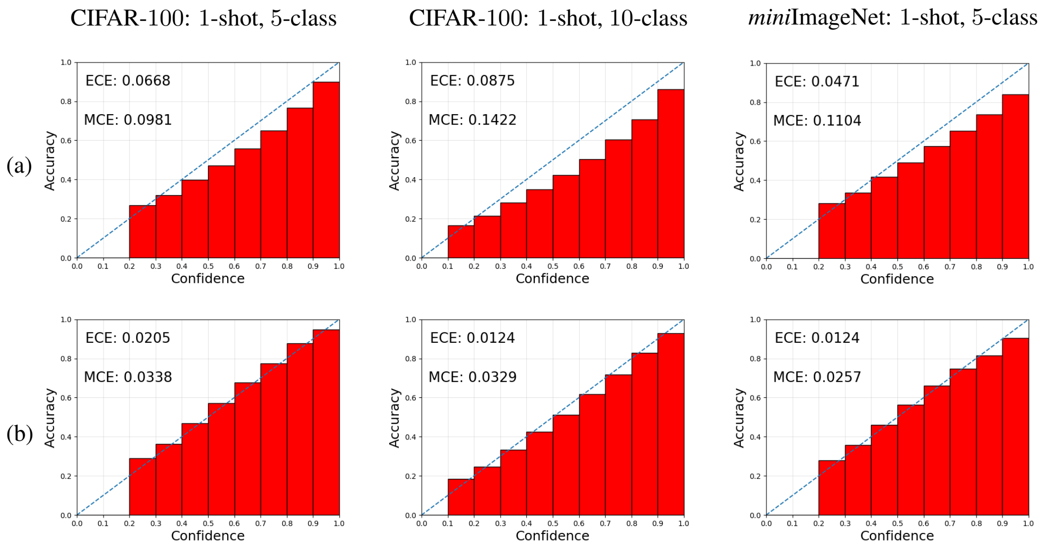

Figure 3: Reliability diagrams for MAML and our model on various tasks across datasets. Relibiality diagrams are computed by gathering predicted probabilities for query set examples across many episodes, where the same set of evaluation episodes are used for both models (a) MAML reliability diagrams (b) Reliability diagrams for our model.

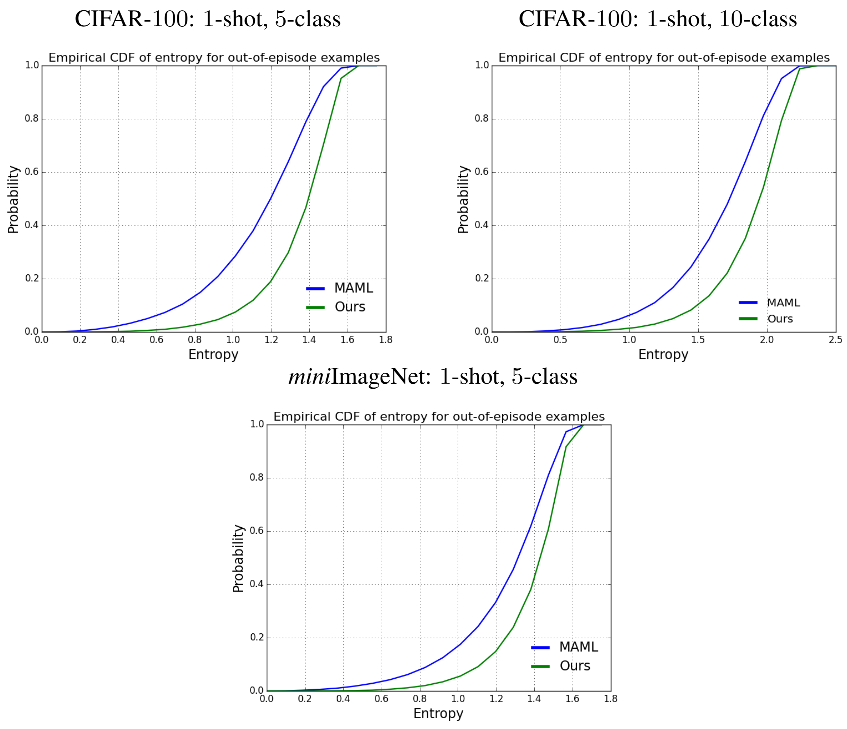

Figure 4: Comparison of empirical CDF of entropy of predictive distributions on out-of-episode examples on various tasks and datasets. Data for CDF comes from computing the entropy on out-of-episode examples across many episodes, where out-of-episode examples are generated by randomly sampling classes not belonging to the episode and randomly sampling examples from those classes. The same set of evaluation episodes are used for both models.

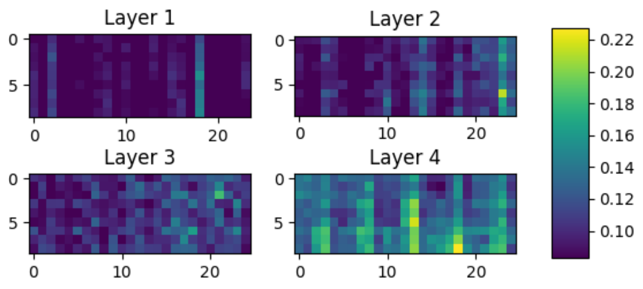

Figure 5: Standard deviation of prior for convolutional kernels across layers of network. For each image, the x-axis indexes different filters from the specific layer whereas the y-axis indexes across positions in the 3×3 kernel.

6.5 Hyperparameters

6.5.1 Contextual Bandits

	$n = 2,000$	$n = 80,000$
Number of NN Layers	2	2
Hidden Units per Layer	100	100
t_s (mini-batches per training step)	100	100
t_f (frequency of training)	20	100
Optimizer	ADAM	ADAM
Learning rate	0.001	0.001

Table 3: Hyperparameters for contextual bandit experiments. These hyperparameters were used for both MAML and our model when comparing them. Hyperparameters t_f and t_s were used as defined in [11] .