Learning to Learn Dense Gaussian Processes for Few-Shot Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Gaussian processes with deep neural networks demonstrate to be a strong learner for few-shot learning since they combine the strength of deep learning and kernels while being able to well capture uncertainty. However, it remains an open problem to leverage the shared knowledge provided by related tasks. In this paper, we propose to learn Gaussian processes with dense inducing variables by meta-learning for few-shot learning. In contrast to sparse Gaussian processes, we define a set of dense inducing variables to be of a much larger size than the support set in each task, which can collect prior knowledge from experienced tasks. The dense inducing variables specify a shared Gaussian process prior over prediction functions of all tasks, which are learned in variational inference framework and offers a strong inductive bias for learning new tasks. To achieve task specific prediction functions, we propose to adapt the inducing variables to each task by efficient gradient descent. We conduct extensive experiments on common benchmark datasets for a variety of few-shot learning tasks. Our dense Gaussian processes present significantly improvements over vanilla Gaussian processes and comparable or even better performance with state-of-the-art methods.

1 Introduction

Meta learning [29, 4, 17], also referred as learning to learn, aims at acquiring knowledge from a distribution of tasks, and learn to fast resolve novel tasks sampled from the same underlying task distribution.Meta learning stimulates the rise of few-shot learning [9], and the recent advantages are mainly driven by designing learning algorithms that learn from massive tasks and acquire prior knowledge, which combined with the a small amount of labeled data, induces a model that produces reliable predictions on novel tasks. In practice, the acquisition of prior knowledge can be realized in different forms. the model agnostic meta-learning (MAML) [9] learns to adapts to new tasks by few iterations of gradient descend, which inspires many follow-up methods [3, 12, 20, 43]. The adaptation of the entire network makes it hard to be scaled to large networks, and many recent efforts focus on adapting the last classification layer only [14, 5], while assuming a universal feature extractor that is shared across all tasks. Despite the remarkable progresses, challenges remain due to the uncertainty in making predictions with very limited data, which requires models to have high robustness for the few-shot learning.

Gaussian processes (GP) [23] serve as a powerful model for the inference of functions, which enjoy appealing properties including natural uncertainty quantification and the robustness when the amount of data is limited. By combining the strong learning ability of deep neural networks, Gaussian process with deep kernels further demonstrates improved results on supervised learning [41]. To alleviate the computational cost, sparse Gaussian processes were widely developed by learning a sparse set

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

of inducing variables to effectively approximate the full dataset. In sparse Gaussian processes, the inducing variables usually in the forms of pseudo data or a subset of training data are expected to capture statistics of the whole dataset.

Gaussian processes offer desirable robustness to data scarcity by capturing uncertainty and enjoy the high expressiveness of deep kernels, without suffering from computational cost in regular learning tasks. This makes Gaussian process a well-suited learning tool for few-shot learning, which has been demonstrated in recent works [31, 21, 35]. It is shown in deep kernel transfer (DKT) [21] that a GP with a meta-learned deep kernel can deliver strong performance through a simple kernel transfer. An important assumption made by DKT is that a universal deep kernel can be obtained through training on a limited amount of tasks. However, this assumption can be too restricted to generalize to novel tasks. In practice, when solving novel tasks, the learned deep kernels can hardly guarantee to be able to explain the data well due to the scarce labelled data, and failing to fit the Gaussian process prior can cause failed predictions. It is therefore crucial to leverage shared knowledge among tasks, which remains an outstanding problem for Gaussian process few-shot learning.

In this paper, we learning to learn Gaussian processes by a dense set of inducing variables for few-shot learning. We assume a Gaussian process prior over the prediction functions of few-shot learning tasks, which is specified by a set of inducing variables learned from data. These inducing variables are dense in the sense that they have a much larger cardinality comparing to the support set of each task. Under the meta-learning setting, the dense inducing variables are learned to collect the shared knowledge from experienced tasks to improve the learning of new tasks effectively with limited data.

To achieve task-specific prediction functions, we propose to adapt the inducing variables to each task by gradient descent update. Without involving any extra model parameters, the adaptation permits online fitting to the Gaussian process prior for each individual task, and therefore allows the model to better fit any novel tasks at test time. In addition, different from sparse Gaussian processes, we learn the inducing variables in a low-dimensional deep feature space, which significantly reduces the cost of both learning and adapting the dense inducing variables.

The resultant *dense Gaussian processes* inherit the strong learning ability of deep kernels and enjoy the robustness to data scarcity and innate ability of uncertainty quantification, providing an effective few-shot learner. We validate the effectiveness of our dense Gaussian processes on a variety of few-shot learning tasks. We observe that our dense Gaussian processes achieves substantial performance improvements over vanilla deep Gaussian processes for few-shot learning and delivers state-of-the-art performance on common benchmarks for few-shot learning. Moreover, our dense Gaussian processes demonstrate strong generalizability on cross-domain few-shot learning tasks.

In summary, we make three major contributions as follows:

- We introduce *dense Gaussian processes* by learning a set of dense inducing variables for few-shot learning. The inducing variables specify a Gaussian process prior over predictive functions across tasks, which collect shared knowledge from experienced tasks that provides strong inductive bias for learning new task efficiently and effectively.
- We propose gradient descent based adaptation to establish task-specific prediction function based on Gaussian process prior. The gradient descent-based adaptation is efficient without any auxiliary network components for inferring task representations.
- We conduct extensive experiments on common benchmark datasets for few-shot classification. The proposed dense Gaussian processes achieve consistent improvements over previous Gaussian processes with deep kernels and delivers comparable or even better results than state-of-the-art methods.

2 Methodology

In this section, we start with a brief review to the few shot learning, and the key ingredients that build the foundation of the proposed method. We then introduce in detail learning to learn dense inducing variables for Gaussian process few-shot learning.

2.1 Background

Few-shot learning. The common practice of training few-shot learners is to formulate it as a *episodic training*. In each episode, the training is realized by simulating a few-shot learning task, where the learner is provided with a set of few label data called the *support set* $\mathcal{S}$, and required to make predictions on the corresponding *query set* $\mathcal{Q}$ with unlabeled test samples. Taking few-shot image classification as an example, It is usually defined as a N-way K-shot learning problem, where K is usually a small number, e.g., $K = 5$. Typically, in each episode indexed by t, one few-shot classification task is generated by first sampling N categories from the training data, each of which contains K samples to form the support set $\mathcal{S}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N\times K}\}$. An adaptation to the few-shot classification model is then performed on $\mathcal{S}_t$, by, e.g., computing prototypes [30], or updating the network parameters [9]. After adaptation, the query set is formed by evenly sampling M samples from each category, i.e., $\mathcal{Q}_t = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{N\times M}\}$, to evaluate the updated model, and the error is propagated back to update the parameters.

Gaussian processes with deep kernels. Gaussian processes (GP) [23] place a prior distribution $p(\mathbf{f})$ over the target functions $\mathbf{f}$. Consider we have a set of n datapoints, $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$ with additive noise $\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. The GP prior over $\mathbf{f}$ can be written as,

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{K}}_{nn}), \tag{1}$$

where $\boldsymbol{\mathcal{K}}_{nn} = \boldsymbol{\mathcal{K}}(\mathbf{x}_n, \mathbf{x}_n)$ is the kernel matrix of n training datapoints, and the likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{n} p(\mathbf{y}_i|\mathbf{f}_i) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{f}_i, \sigma^2). \tag{2}$$

Given a test data point $\mathbf{x}_*$, its predictive posterior is characterized by a Gaussian distribution, $\mathbf{f}_* \sim \mathcal{N}(\mathbf{f}_*|\mu_*, \sigma_*)$, where

$$\begin{aligned} \mu_* &= {\mathbf{k}_*}^T(\boldsymbol{\mathcal{K}}_{nn} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \\ \sigma_* &= \mathbf{k}(\mathbf{x}^*, \mathbf{x}^*) - {\mathbf{k}_*}^T(\boldsymbol{\mathcal{K}}_{nn} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_* \end{aligned} \tag{3}$$

Given a deep neural network $\Phi_\theta : \mathbf{X} \to \mathbf{Z}$, which maps from input space to deep feature space, we obtain the deep kernel function [41],

$$k(\mathbf{x}_i, \mathbf{x}_j|\theta) = k(\boldsymbol{\Phi}_\theta(\mathbf{x}_i), \boldsymbol{\Phi}_\theta(\mathbf{x}_j)) = k(\mathbf{z}_i, \mathbf{z}_j). \tag{4}$$

The dimensionality of deep feature space $\mathbf{Z}$ is usually smaller than the original input space $\mathbf{X}$, especially for image data, leading to more efficient kernel learning and inference. In this way, the GP prior in our method is parameterized by the network $\boldsymbol{\Phi}$ as $p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{nn})$, where $\mathbf{K}_{nn} = \mathbf{K}(\mathbf{z}_n, \mathbf{z}_n) = \mathbf{K}(\mathbf{x}_n, \mathbf{x}_n|\boldsymbol{\Phi}_\theta)$ is the covariance matrix calculated based on deep features, and the function values $\mathbf{f}$ are defined over deep features $\mathbf{f} = \mathbf{f}(\mathbf{Z}) = \mathbf{f}(\Phi(\mathbf{X}))$.

Sparse Gaussian Processes The posterior inference in a Gaussian Process induces a cubic complexity of $\mathcal{O}(n^3)$, which scales poorly with a huge amount of datapoints n. Sparse Gaussian processes are widely studied in order to improve the scalability of standard GP, and a popular direction introduces m inducing variables $\{\tilde{\mathbf{Z}}_i\}_{i=1}^m$ to approximate the full GP marginal likelihood, where typical $m \ll n$ in favor of improved efficiency.

In sparse Gaussian processes, the optimization for the inducing variables $\{\tilde{\mathbf{Z}}_i\}_{i=1}^m$ can be achieved through variational inference [33]. And the GP prior [33] is augmented with the function values of inducing variables $\mathbf{f}_m \triangleq \mathbf{f}(\tilde{\mathbf{Z}})$ as $p(\mathbf{f}, \mathbf{f}_m) = p(\mathbf{f}|\mathbf{f}_m)p(\mathbf{f}_m)$, and the joint distribution can therefore be expressed as:

$$p(\mathbf{y}, \mathbf{f}, \mathbf{f}_m) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{f}_m)p(\mathbf{f}_m), \tag{5}$$

where it is assumed that $\mathbf{y}$ is conditionally independent of $\mathbf{f}$ given $\mathbf{f}_m$. By minimizing the distance between the true posterior $p(\mathbf{f}, \mathbf{f}_m|\mathbf{y})$ and the variational approximate posterior $q(\mathbf{f}, \mathbf{f}_m) = p(\mathbf{f}|\mathbf{f}_m)q(\mathbf{f}_m)$, where $q(\mathbf{f}_m)$ is a free variational Gaussian distribution $\mathcal{N}(\mathbf{f}_m|\boldsymbol{\mu}, \boldsymbol{\Lambda})$, the maximization of log likelihood amounts to maximizing a variational lower bound:

$$\log p(\mathbf{y}) \geq \int_{\mathbf{f},\mathbf{f}_m} q(\mathbf{f}, \mathbf{f}_m) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{f}_m)}{q(\mathbf{f}, \mathbf{f}_m)} d\mathbf{f} d\mathbf{f}_m \triangleq \mathcal{L}(\tilde{\mathbf{Z}}, \boldsymbol{\Phi}, q) \tag{6}$$

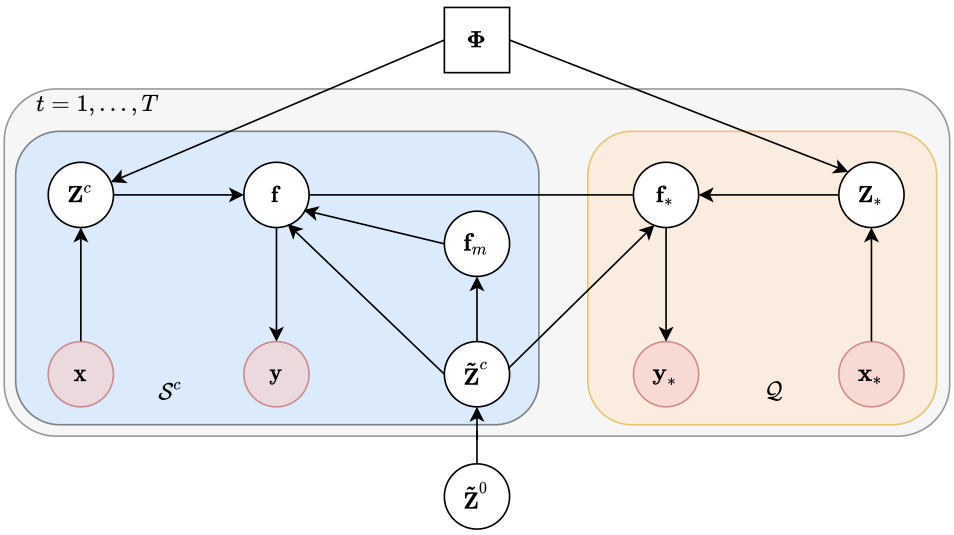

Figure 1: The graphical illustration of the proposed dense Gaussian processes for few-shot learning.

2.2 Learning to Learn Dense Gaussian Processes

In this section, we derive our dense Gaussian processes by inducing variables under the episodic training setting of meta-learning for few-shot learning.

Adapting Gaussian Process Priors For a C-way K-shot classification task, we adapt $\mathbf{f}^c \sim \mathrm{GP}^c(0, \mathbf{K}^c)$, and the samples from the posterior predictive are combined based on a multi-class likelihood such as the softmax function. We would like to learn a Gaussian process prior over prediction function, which can be adapted to a few-shot learning tasks by its support set with a few gradient descent update. The Gaussian process prior is established by a set of dense inducing variables, which is much larger than the support set in each task. The inducing variables are learned from data which serve to collect shared knowledge from experienced task to facilitate efficient learning of new tasks.

Formally, in an episode of the C-way K-shot task, the GP learner is provided with a support set $\mathcal{S}_t = \{\mathbf{x}_k^c, \mathbf{y}_k^c\}_{c=1,k=1}^{C,K}$. We adapt task-specific GP prior in the form of specific kernels $\mathbf{K}^c(\Phi^c, \tilde{\mathbf{Z}}^c)$ for each class within the task, which can be achieved through an adaptation based on Eq. 6, with both parameters initialized by the meta representation $\mathbf{\Phi}^0, \tilde{\mathbf{Z}}^0$,

$$\mathbf{\Phi}^c(\mathbf{\Phi}^0), \tilde{\mathbf{Z}}^c(\tilde{\mathbf{Z}}^0) = \arg\min_{\mathbf{\Phi}, \tilde{\mathbf{Z}}} \mathcal{L}(\mathbf{\Phi}, \tilde{\mathbf{Z}}, q). \tag{7}$$

As shown in [33], $\mathcal{L}(\mathbf{\Phi}, \tilde{\mathbf{Z}}, q)$ can be first maximized by analytically solving the optimal $q^o(\mathbf{f}_m)$,

$$q^o(\mathbf{f}_m) = N(\mathbf{f}_m | \boldsymbol{\mu}^o, \mathbf{\Lambda}^o), \quad \boldsymbol{\mu}^o = \sigma^{-2}\mathbf{K}_{mm}\mathbf{\Sigma}\mathbf{K}_{mn}\mathbf{y}, \ \mathbf{\Lambda}^o = \mathbf{K}_{mm}\mathbf{\Sigma}\mathbf{K}_{mm}, \tag{8}$$

where now $n = |\mathcal{S}| = C \times K$, thus $\mathbf{K}_{mn}$ denotes the covariance matrix between m inducing variables and n support set data, and $\mathbf{\Sigma} = (\mathbf{K}_{mm} + \sigma^{-2}\mathbf{K}_{mn}\mathbf{K}_{mn}^T)^{-1}$. The Derivation is provided in the Appendix A. Thus, the lower bound can be written as,

$$\mathcal{L}(\mathbf{\Phi}, \tilde{\mathbf{Z}}) = \log \mathcal{N}(\mathbf{y} | \mathbf{0}, \sigma^2 I + \hat{\mathbf{K}}_{nn}) - \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbf{K}_{nn} - \hat{\mathbf{K}}_{nn}), \tag{9}$$

where $\hat{\mathbf{K}}_{nn} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{nm}^T$. By maximizing (9), we can obtain an adapted GP prior for each class that parameterized by $\tilde{\mathbf{Z}}^c, \mathbf{\Phi}^c$. Ideally, the well-trained inducing variables can sparsely approximate the full dataset, and reduce the prohibitive computation complexity to $\mathcal{O}(nm^2)$.

Given a query sample $\mathbf{x}_*$, the posterior predictive distribution for class n, $q(\mathbf{f}_*^c | \mathbf{x}_*^c) = \mathcal{N}(\mu_*^c, (\sigma^2)_*^c)$ can be calculated as

$$\begin{aligned} \mu_*^c &= \mathbf{k}_{*m}\mathbf{\Sigma}\mathbf{K}_{mn}^c\mathbf{y} \\ (\sigma^2)_*^c &= k^c(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_{*m}^c(\mathbf{K}_{mm}^c)^{-1}(\mathbf{k}_{*m}^c)^\top + \mathbf{k}_{*m}^c\mathbf{\Sigma}(\mathbf{k}_{*m}^c)^\top, \end{aligned} \tag{10}$$

Algorithm 1 Learning to learn dense inducing variables.

Given: A description of the tasks as N-way K-shot with M query samples at each episode.
Given: Inner-loop and out-loop learning rates α and λ, number of steps I for inner-loop optimization.
Initialize $\mathbf{\Phi}$ and $\tilde{\mathbf{Z}}$
repeat
  Sample task with support set $\mathcal{S} = \{\mathbf{x}_k^c\}_{c=1,k=1}^{C,K}$ and query set $\mathcal{Q} = \{\mathbf{x}'_j\}_{j=1}^{M}$ in each task.
  Extract feature vectors for both $\mathcal{S}$ and $\mathcal{Q}$ by applying $\mathbf{z}_k^c = \mathbf{\Phi}(\mathbf{x}_k^c)$ and $\mathbf{z}'_j = \mathbf{\Phi}(\mathbf{x}'_j)$, respectively.
  for Class $n = 1 : N$ **do**
    for Inner iterations $i = 1 : I$ **do**
      Perform inner-loop adaptation and tuned the per-class inducing variables $\tilde{\mathbf{Z}}^{c,i}$ for a class-adapted GP prior by applying (14).
    end for
  end for
  With $\tilde{\mathbf{Z}}^{c,I}$ obtained, calculate the predictive posterior using (11), and perform the outer-loop adaptation by applying (15).
until Converge
Return $\mathbf{\Phi}$ and $\tilde{\mathbf{Z}}^0$.

and the multi-class posterior predictive can be expressed as,

$$p(\mathbf{y}_*^c = 1|\mathbf{x}_*) = \frac{\exp(\mathbf{f}_*^c \cdot \tau)}{\sum_{c'} \exp(\mathbf{f}_*^{c'} \cdot \tau)}, \quad \text{with } \mathbf{f}_*^c \sim q(\mathbf{f}_*^c|\mathbf{x}_*), \tag{11}$$

where τ is an learned temperature for MLE. Then, the loss of empirical risk on query set, act as the outer-loop criteria, can be then be calculated as

$$\mathcal{L}_{\text{Outer}}(\tilde{\mathbf{Z}}^0, \mathbf{\Phi}^0) = \sum_{(\mathbf{x}_*,\mathbf{y}_*)\in\mathcal{Q}} \log p(\mathbf{y}_*|\mathbf{x}_*). \tag{12}$$

In this way, the GP prior adapted to the task t by $\mathcal{S}_t$ is further supervised by the query set $\mathcal{Q}_t$. This in turn incorporates more knowledge from each task to consolidate task-shared meta knowledge into $\mathbf{\Phi}^0, \tilde{\mathbf{Z}}^0$, which enables new task to learned more efficiently.

Dense Inducing Variables The aforementioned few-shot episodic training deep kernel allows an inner-loop adaptation to the model, and achieves better prior fitting the novel dataset. However, performing adaptation to the entire deep kernel involves update the deep neural network parameterizing the kernel, which introduces dramatically high computation cost and latency. Meanwhile, leveraging explicitly the meta knowledge learned from massive tasks, e.g., with memory that allows adaptive information retrieval [45], or data hallucination [42] that expand the support of current tasks have demonstrated impressive results on few-shot learning. Inspired by the idea of explicit knowledge representation in few-shot learning, and data approximation through inducing variables in sparse Gaussian process, we induce *dense inducing variables*, which are learn as a rich set of inducing variables that carry the meta-knowledge of the entire task distribution in the latent deep feature space, and permits efficient model adaptation to novel tasks.

Specifically, instead of acting as a sparse approximation to the data of any particular task, the dense inducing variable here specify a shared Gaussian prior over prediction functions of all tasks, are introduce to support the entire tasks distribution, from which potential infinite number of tasks can be sampled from. To achieve effective modeling of the cross task GP prior, we allow the dense inducing variables to contain far more elements than a single task, i.e., $m \gg |\mathcal{S}|$.

Notably, unlike in standard Sparse Gaussian processes, instead of learning the dense inducing variables as pseudo inputs in raw data space, where the dimension is usually high consider input domains like images, we propose to directly learn inducing variables in the same latent space of the feature from deep neural network. By doing so, the learning of the inducing variables effectively bypasses the heavy feature extractor, and potentially enables efficient online adaptation to the inducing variables at low cost.

With the dense inducing variables specifying a hyperprior over predictive functions across all tasks, and included as hyperparameters in the deep kernel $\mathbf{\Phi}$, we now introduce the online updating, which induces an adapted Gaussian process prior over prediction function for a specific task, through a gradient descent based adaptation to the dense inducing variables.

Specifically, given the support set of the task with support set $\mathcal{S}_t$, we perform adaptation of the Gaussian process prior by using the few labeled data, and rewrite (7) as

$$\mathcal{L}_{\text{Inner}}(\tilde{\mathbf{Z}}) = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \sigma^2 I + \hat{\mathbf{K}}_{nn}) - \frac{1}{2\sigma^2}\text{Tr}(\mathbf{K}_{nn} - \hat{\mathbf{K}}_{nn}), \tag{13}$$

and adapt the inducing variables by

$$\tilde{\mathbf{Z}}^{c,i} = \tilde{\mathbf{Z}}^{c,i-1} - \alpha \nabla_{\tilde{\mathbf{Z}}^{c,i-1}} \mathcal{L}_{\text{Inner}}(\tilde{\mathbf{Z}}^{c,i-1}), \tag{14}$$

where α specify an inner-loop learning rate, and each $\tilde{\mathbf{Z}}_n^0$ initialized with $\tilde{\mathbf{Z}}^0$. Thanks to the first-order approximation to the computationally heavy second-order derivative, the training with (14) can be remarkably accelerated.

Given the adapted inducing variables on the i-th task, the predictive posterior of function values on the query set can then be efficiently obtained by the meta updates to both the feature extractor and the dense inducing variables calculated based on (12),

$$\begin{aligned} \mathbf{\Phi} &= \mathbf{\Phi} - \lambda \nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{Outer}}(\mathbf{\Phi}, \tilde{\mathbf{Z}}^0) \\ \tilde{\mathbf{Z}}^0 &= \tilde{\mathbf{Z}}^0 - \lambda \nabla_{\tilde{\mathbf{Z}}^0} \mathcal{L}_{\text{Outer}}(\mathbf{\Phi}, \tilde{\mathbf{Z}}^0), \end{aligned} \tag{15}$$

with λ specifying the learning rate for the outer loop.

Note that by updating the deep feature extractor $\mathbf{\Phi}$ only in the outer loop, we assume a universal feature extractor, which is commonly used in meta-learning for efficiency and generalization to more powerful backbones. Meanwhile, the specification of the task specific Gaussian process prior through adapting dense inducing variables endows our model with high flexibility and strong generalization to novel tasks without scarifying efficiency, as we will demonstrate in our experiments.

In contrast to previous work that simply learns transferable Gaussian process on few-shot learning, our method adopt the episodic learning to adapting the Gaussian process prior to each task adaptation by updating the dense inducing variables, which parameterize the deep kernel jointly with the deep feature extractor $\mathbf{\Phi}$. The overall training steps of the model learning is summarized in Algorithm 1.

3 Related Work

Metric learning [30] has been explored for few-shot learning, which assumes that different related tasks would share the same metric to measure similarity. Vinyals et al. [38] proposed the matching network, which learns to map a small labeled support set and an unlabelled example to its label, obviating the need for fine-tuning to adapt to new class types. This work was originally developed for one-shot learning, and extended to a few-shot setting by Snell et al. [30]. They proposed the prototypical network to learn a metric space in which classification can be performed by computing distances to prototype representations of each class and the prototype of each class is the cluster of samples in that class. To enhance the expressivity of the prototypes, Allen et al. [1] proposed infinite mixture prototypes to adaptively represent both simple and complex data distributions for few-shot learning. Satorras et al. [28] solved few-shot learning with the prism of inference on a partially observed graphical model.

Meta-learning based on optimization has been extensively studied [25, 10, 2]. The core idea of optimization based methods is to learn an optimization procedure that is shared across tasks, which can be applied to new tasks for quick adaptation, once learned in the meta-train stage. Ravi et al. [25] proposed an LSTM based meta-learner to learn the exact optimization algorithm used to train a neural network classifier in the few-shot regime; in their methods, apart from the learned optimization algorithm, a good initialization of model parameters is also obtained after training. Finn et al. [11] proposed a model agnostic meta-learning (MAML) algorithm, which was built upon the assumption that related tasks sharing initial parameters of neural networks could be adapted to specific tasks with a few steps of gradient descent updates. It was believed that the initial weights combined with a

few more steps of gradient descent can approximate any learning algorithm, and thus gradient-based meta-learning had a number of practical benefits. Finn et al. [13] extended MAML in a probabilistic framework. Rajeswaran et al. [22] developed the implicit MAML [10] algorithm, which depended only on the solution to the inner level optimization. Zintgraf et al. [46] updated context parameters with one or several gradient steps on a task-specific loss that serves as an additional input to the model and were adapted on individual tasks.

It has also been investigated to explicitly design a meta-learner to learn a base-learner. Bertinetto et al. [6] explored the feasibility of incorporating fast solvers with closed-form solutions as the base learning component of a meta-learning system. Gordon et al. [15] developed meta-learning approximate probabilistic inference for prediction. The support set was used to produce the parameter distribution of the classifier, which was applied to the query set for prediction. Mishra et al. [18] proposed a generic meta-learner architecture that used a novel combination of temporal convolutions and soft attention. Memory has recently generated increasing attention in the machine learning community, which was used to augment deep neural networks [40, 16]. It has also been introduced to the meta-learning framework for few-shot learning [27, 19, 45].

Gaussian processes have recently been introduced for Few-shot Learning [36, 21, 31, 31]. Tossou et al. [36] learn a parameterized kernel operator that can be combined with a differentiable kernel algorithm during inference. Deep Kernel Transfer (DKT) [21] explore Gaussian processes for few-shot classification, which learns covariance functions parameterized by deep neural networks. More recently, Titsias et al. [35] applies Gaussian processes to meta-learning by maximizing the mutual information between the query set and a latent representation of the support set. Snell et al. [31] develop a Gaussian process classifier by combining Pólya-Gamma augmentation and the one-vs-each softmax approximation [34] in order to efficiently marginalize over functions rather than model parameters. In our work, we explore a dense set of inducing variables in Gaussian processes for few-shot classification. The resultant dense Gaussian processes enable more knowledge to be shared among related tasks and transferred to new unseen tasks for efficient and effectively learning.

4 Experiments

4.1 Datasets and Settings

We perform experiments on the widely used few-shot learning benchmarks including *mini*ImageNet, *tiered*ImageNet, CIFAR-FS , and Caltech-UCSD [39] (CUB). In *mini*ImageNet [38], there are 100 image classes from a subset of ImageNet [8], with 600 images for each class. We follow the standard practice [9] to split the training, validation, and testing sets with 64, 16, and 20 classes, respectively. *tiered*ImageNet [26] is a large subset of ImageNet that contains 608 classes with 1,300 samples in each class. Specifically, in *tiered*ImageNet, there are 351 classes from 20 categories for training, 97 classes from 6 categories for validation, and 160 classes from 8 different categories for testing. Samples for both *mini*ImageNet and *tiered*ImageNet are random cropped and resized to 84×84 for training, and standard center cropping is performed to the testing images. The 200 classes in the CUB dataset is divided into 100, 50, and 50 classes, for training, validation, and testing, respectively.

Table 1: Performance comparisons on *mini*ImageNet with 95% confidence intervals.

Methods	***mini*ImageNet**		***tiered*ImageNet**		**CIFAR-FS**	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MatchingNets [38]	43.56 $\pm$ 0.84	55.31 $\pm$ 0.73	-	-	-	-
MAML [10]	48.70 $\pm$ 1.84	63.11 $\pm$ 0.92	51.67 $\pm$ 1.81	70.30 $\pm$ 1.75	58.90 $\pm$ 1.91	71.52 $\pm$ 1.10
Reptile [20]	49.97 $\pm$ 0.32	65.99 $\pm$ 0.58	-	-	-	-
R2-D2 [6]	48.70 $\pm$ 0.60	65.50 $\pm$ 0.60	-	-	60.20 $\pm$ 1.80	70.91 $\pm$ 0.91
VERSA [14]	53.31 $\pm$ 1.80	67.30 $\pm$ 0.91	-	-	62.51 $\pm$ 1.70	75.11 $\pm$ 0.91
RelationNets [32]	50.44 $\pm$ 0.82	65.32 $\pm$ 0.70	54.48 $\pm$ 0.93	65.32 $\pm$ 0.70	55.00 $\pm$ 1.01	69.30 $\pm$ 0.80
ProtoNets [30]	47.40 $\pm$ 0.60	65.41 $\pm$ 0.52	53.31 $\pm$ 0.89	72.69 $\pm$ 0.74	55.50 $\pm$ 0.70	72.01 $\pm$ 0.60
VSM [45]	54.73 $\pm$ 1.60	68.01 $\pm$ 0.90	56.88 $\pm$ 1.71	74.65 $\pm$ 0.81	63.42 $\pm$ 1.90	77.93 $\pm$ 0.80
DKT [21]	49.73 $\pm$ 0.07	64.00 $\pm$ 0.09	-	-	-	-
OVE PG GP + Cosine (ML) [31]	50.02 $\pm$ 0.35	64.58 $\pm$ 0.31	-	-	-	-
OVE PG GP + Cosine (PL) [31]	48.00 $\pm$ 0.24	67.14 $\pm$ 0.23	-	-	-	-
Our	**56.32** $\pm$ 0.28	**72.64** $\pm$ 0.26	**58.43** $\pm$ 0.38	**76.17** $\pm$ 0.34	**64.17** $\pm$ 0.31	**78.42** $\pm$ 0.26

Table 2: Performance comparison on the CUB dataset with 95% confidence intervals.

Methods	**CUB** (Conv-4) 1-shot	5-shot	**CUB** (ResNet-10) 1-shot	5-shot
Feature Transfer	46.19 $\pm$ 0.64	68.40 $\pm$ 0.79	63.64 $\pm$ 0.91	81.27 $\pm$ 0.57
ABML [24]	49.57 $\pm$ 0.42	68.94 $\pm$ 0.16	-	-
Baseline ++ [7]	61.75 $\pm$ 0.95	78.51 $\pm$ 0.59	69.55 $\pm$ 0.89	85.17 $\pm$ 0.50
MatchingNet [38]	60.19 $\pm$ 1.02	75.11 $\pm$ 0.35	71.29 $\pm$ 0.87	83.47 $\pm$ 0.58
ProtoNet [30]	52.52 $\pm$ 1.90	75.93 $\pm$ 0.46	73.22 $\pm$ 0.92	85.01 $\pm$ 0.52
RelationNet [32]	62.52 $\pm$ 0.34	78.22 $\pm$ 0.07	70.47 $\pm$ 0.99	83.70 $\pm$ 0.55
MAML [10]	56.11 $\pm$ 0.69	74.84 $\pm$ 0.62	70.32 $\pm$ 0.99	80.93 $\pm$ 0.71
Bayesian MAML [44]	55.93 $\pm$ 0.71	72.87 $\pm$ 0.26	-	-
DKT [21]	62.96 $\pm$ 0.62	77.76 $\pm$ 0.62	72.27 $\pm$ 0.30	85.64 $\pm$ 0.29
OVE PG GP + Cosine (ML) [31]	63.98 $\pm$ 0.43	77.44 $\pm$ 0.18	-	-
OVE PG GP + Cosine (PL) [31]	60.11 $\pm$ 0.26	79.07 $\pm$ 0.05	-	-
Ours	**70.14** $\pm$ 0.41	**81.48** $\pm$ 0.58	**80.33** $\pm$ 0.67	**89.97** $\pm$ 0.63

Table 3: Performance comparison for cross-domain few-shot classification with 95% confidence intervals.

Methods	***mini*ImageNet** $\rightarrow$ **EMNIST** 1-shot	5-shot	**Omniglot** $\rightarrow$ **CUB** 1-shot	5-shot
Feature Transfer	64.22 $\pm$ 1.24	86.10 $\pm$ 0.84	32.77 $\pm$ 0.35	50.34 $\pm$ 0.27
ABML [24]	76.37 $\pm$ 0.29	87.96 $\pm$ 0.28	29.35 $\pm$ 0.26	45.74 $\pm$ 0.33
Baseline ++ [7]	56.84 $\pm$ 0.91	80.01 $\pm$ 0.92	39.19 $\pm$ 0.12	57.31 $\pm$ 0.11
MatchingNet [37]	75.01 $\pm$ 2.09	87.41 $\pm$ 1.79	36.98 $\pm$ 0.06	50.72 $\pm$ 0.36
ProtoNet [30]	72.04 $\pm$ 0.82	87.22 $\pm$ 1.01	33.27 $\pm$ 1.09	52.16 $\pm$ 0.17
RelationNet [32]	75.62 $\pm$ 1.00	87.84 $\pm$ 0.27	37.13 $\pm$ 0.20	51.76 $\pm$ 1.48
MAML [10]	72.68 $\pm$ 1.85	83.54 $\pm$ 1.7	34.01 $\pm$ 1.25	48.83 $\pm$ 0.62
Bayesian MAML [44]	63.94 $\pm$ 0.47	65.26 $\pm$ 0.30	33.52 $\pm$ 0.36	51.35 $\pm$ 0.16
DKT [21]	75.40 $\pm$ 1.10	90.30 $\pm$ 0.49	40.14 $\pm$ 0.18	56.40 $\pm$ 1.34
OVE PG GP (ML) [31]	68.43 $\pm$ 0.67	86.22 $\pm$ 0.20	39.66 $\pm$ 0.18	55.71 $\pm$ 0.31
OVE PG GP (PL) [31]	77.00 $\pm$ 0.50	87.52 $\pm$ 0.19	37.49 $\pm$ 0.11	57.23 $\pm$ 0.31
Ours	**78.32** $\pm$ 0.49	**90.78** $\pm$ 0.24	**44.75** $\pm$ 0.38	**61.88** $\pm$ 0.53

CIFAR-FS adopts all the 100 classes in the CIFAR-100 dataset with a training, validation, and testing split of 64, 16, and 20 classes, respectively. Each class contains 600 image samples. The image resolution for CIFAR-FS is 32×32.

4.2 Few-Shot Image Classification

The results of few-shot classification on *mini*ImageNet, *tiered*ImageNet, CIFAR-FS , and Caltech-UCSD (CUB) are reported in Tables 1 and 2. For comprehensive comparisons, we adopt two backbone networks with different scales. On all three datasets, we achieve comparable or often better performance than state-of-the-art methods. Our method yields the accuracy of 72.64 on the *mini*ImageNet dataset under the 5-way 5-shot setting, which surpasses previous best method by up to 5%. In particular, our dense Gaussian processes achieve much better performance than the recent Gaussian process based methods [21, 31]. This demonstrates the effectiveness of our method by learning dense inducing variables. For more experimental detail please refer to Appendix Section B.

4.3 Cross-Domain Few-Shot Image Classification

The efficient adaptation to the GP prior through updating the dense inducing variables enables the proposed method with improved generalization to novel tasks with unseen categories. To fully demonstrate the remarkable generalization, we perform cross-domain few-shot image classification. Following common practice, we adopt two cross-domain setting, Omniglot $\rightarrow$ EMNIST and *mini*ImageNet$\rightarrow$ CUB. The result are reported in Table 3. The proposed dense Gaussian processes demonstrate high generalizability across domains for few-shot classification and achieves much better performance than the recent Gaussian process based meothds [21, 31] in the setting of generalization from Omniglot to CUB.

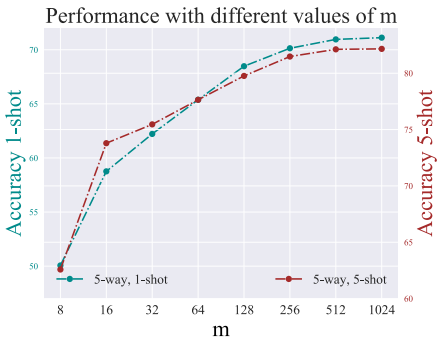

Figure 2: The performance with different sizes of inducing variables.

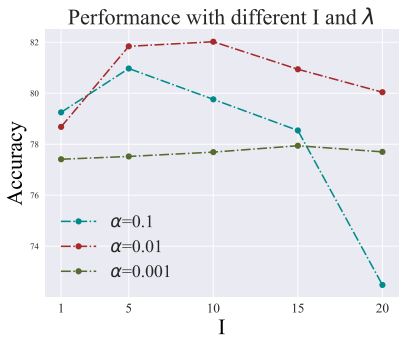

Figure 3: The performance with different numbers of adaptation steps and hyper-parameters.

4.4 Ablation Study

We perform extensive ablation studies to gain insights into the effectiveness of the proposed dense Gaussian processes for few-shot learning. All experiments are performed on the CUB dataset.

Effectiveness of dense inducing variables The proposed dense inducing variables enable shared knowledge to be collected from experienced tasks and applied to new unseen tasks. We show how the number of inducing variable effects the performance of our model. As shown in Figure 2, we report performance on both 5-way 1-shot and 5-way 5-shot experiments. We test a sequence of values of m starting from 8, which is just slightly larger than the number of support set samples in a 5-way 1-shot experiment. And the value of m is progressively increased to 1024. It is clearly demonstrated that, on both experiments with different shots, the performance grows monotonically w.r.t. the number of inducing points that are utilized. And the performance tends to saturate when $m > 256$. To maintain a balance between efficiency and performance, we choose $m = 256$ throughout all the experiments. The results in Figure 2 also validate the effectiveness of specifying the GP prior through 'dense' inducing variables, with much more elements comparing to the support set in each task.

Effectiveness of inner-loop adaptation In the proposed method, the task-specific GP prior adaptation is achieved by few-step gradient descent to the dense inducing variables. For the selection of the best configuration of the inner-loop optimization, we present comparisons performed with the number of steps I in the inner-loop gradient descent, and the inner-loop learning rate α. As shown in Figure 14, we plot the classification accuracy for 5-way 5-shot setting, and show the performance obtained by conducting 1 to 20 steps of inner-loop update with three different values of α. It is clearly shown that the performance reach the peak within 5 steps of inner-loop update, and it does not improve with even more steps of adaption, especially when combined with larger leaning rates. This indicates the efficiency of the proposed adaption of Gaussian process prior to each task. Moreover, a large α can lead to performance drop very fast when performing more than 1 step of inner-loop update. To keep the balance between accuracy and efficiency, we choose $\alpha = 0.01$ and $I = 5$ throughout all the experiments.

5 Conclusion

In this paper, we introduce learning to learn dense inducing variables for few-shot learning. By specifying a Gaussian process prior prediction functions, we introduce dense inducing variables, which are learned from data to collect shared knowledge from previous work to facilitate efficient and effective learning of new tasks. With an universal deep feature extractor learned from the training tasks, we achieve task specific Gaussian process prior by adapting the dense inducing variables through an inner-loop few-step gradient optimization. The resultant dense Gaussian processes are endowed with with strong generalization and robustness. The effectiveness of the proposed method is validated on a variety of benchmark datasets, and state-of-the-art performance across tasks are observed. Notably, the substantial performance improvements on the cross-domain few-shot learning further demonstrate the strong generalization of the proposed method.

References

[1] Kelsey R Allen, Evan Shelhamer, Hanul Shin, and Joshua B Tenenbaum. Infinite mixture prototypes for few-shot learning. In *ICML*, 2019.

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 2016.

[3] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

[4] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, volume 2, 1992.

[5] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[6] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

[7] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[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. PMLR, 2017.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[11] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *ICLR*, 2018.

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

[13] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9537–9548, 2018.

[14] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018.

[15] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

[16] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[17] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[18] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[19] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017.

[20] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[21] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 2020.

[22] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

[23] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[24] Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *ICLR*, 209.

[25] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[26] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *International Conference on Learning Representations*, 2018.

[27] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[28] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *ICLR*, 2018.

[29] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[30] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 2017.

[31] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. *International conference on learning representations*, 2021.

[32] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[33] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[34] Michalis K Titsias. One-vs-each approximation to softmax for scalable estimation of probabilities. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4168–4176, 2016.

[35] Michalis K Titsias, Sotirios Nikoloutsopoulos, and Alexandre Galashov. Information theoretic meta learning with gaussian processes. *arXiv preprint arXiv:2009.03228*, 2020.

[36] Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 2016.

[39] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. (CNS-TR-2010-001), 2010.

[40] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[41] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[42] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. In *International Conference on Learning Representations*, 2021.

[43] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[44] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *NIPS*, 2018.

[45] Xiantong Zhen, Yingjun Du, Huan Xiong, Qiang Qiu, Cees GM Snoek, and Ling Shao. Learning to learn variational semantic memory. *NeurIPS*, 2020.

[46] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *ICML*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [Yes] The proof of Theorem 1 is provided in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]