Journal of Computer and Knowledge Engineering, Vol. 5, No. 2, 2022. (45-56)
DOI: 10.22067/cke.2022.77529.1060

Meta-Learning for Medium-Shot Sparse Learning via Deep Kernels [*]

Research Article

Zohreh Adabi-Firuzjaee[1] Sayed Kamaledin Ghiasi-Shirazi[2]

Abstract: Few-shot learning assumes that we have a very small dataset for each task and trains a model on the set of tasks. For real-world problems, however, the amount of available data is substantially much more; we call this a medium-shot setting, where the dataset often has several hundreds of data. Despite their high accuracy, deep neural networks have a drawback as they are black-box. Learning interpretable models has become more important over time. This study aims to obtain sample-based interpretability using the attention mechanism. The main idea is reducing the task training data into a small number of support vectors using sparse kernel methods, and the model then predicts the test data of the task based on these support vectors. We propose a sparse medium-shot learning algorithm based on a metric-based Bayesian meta-learning algorithm whose output is probabilistic. Sparsity, along with uncertainty, effectively plays a key role in interpreting the model's behavior. In our experiments, we show that the proposed method provides significant interpretability by selecting a small number of support vectors and, at the same time, has a competitive accuracy compared to other less interpretable methods.

Keywords: Bayesian Meta-learning, Medium-shot Learning, Sample-based Interpretability, Sparse Kernel, Attention

1. Introduction

So far, two approaches for deep learning have received more attention. The first approach is deep learning on a large dataset, which has been more successful than other machine learning methods in image, language, and signal processing [1]. In deep learning, as it is difficult for humans to analyze a huge amount of data, one tries to train deep neural networks with it so that the information in the data could be exploited through interaction with the model. We need a massive amount of data to use deep learning, but in most real-world problems the amount of labeled data is not enough to train a deep model. The second approach is known as few-shot learning [2]. It aims to make deep learning models like humans and learn new concepts well by seeing a few examples [3].

In few-shot learning, the assumption is that the number of training data is very small. For example, in few-shot classification, the number of data for each class ranges between one and five. This assumption is contrary to the fact that in real-world problems, we easily have more data for each task, or it is even possible for the user to provide a few hundred samples. Therefore, many practical problems such as classification of medical images [4] and time series prediction are naturally in the medium-shot setting. Medium-shot learning is an extension of few-shot learning in terms of the number of data. In recent years, meta-learning methods have shown remarkable performance in solving few-shot learning problems [5]. In this paper, we consider meta-learning methods for the case of medium-shot setting.

Deep neural networks have attracted widespread attention due to their ability to obtain high accuracies in various problems. However, there is a serious debate about them related to interpretability [6]: to what extent and on what basis can we trust the response of neural networks? Because the nature of deep networks is black-box, many methods have been proposed to interpret neural networks and their decision-making [7]. In problems where the model has to make a decision, the user wants to know why the model has made this decision. The decision of the model can be described in different ways. One of these methods is that the model determines based on the data it has made its decision. Therefore, the user can determine the quality of a decision by examining the samples that the model has selected.

The medium size of the data in the medium-shot setting provides us with the possibility and opportunity of interpretation based on the evaluation of the entire training data of the task. Our goal is to train a model in such a way that it determines which data have a more important role in its decision-making, and we consider these data as support vectors. Our idea to achieve this kind of interpretability is to follow the perspective of attention in deep learning. We want to learn which data to pay more attention to. For this purpose, we present an interpretable meta-learning algorithm. We start our work with Deep Kernel Transfer (DKT), a metric-based meta-learning algorithm [8]. DKT is a Gaussian process with a deep kernel, so it combines the representational power of neural networks and the reliable uncertainty of Gaussian processes simultaneously. To implement the attention mechanism, we use sparse kernel methods and extend the DKT algorithm to the medium-shot setting. By sparsifying the expansion of the decision function, we can have sample-based interpretability with the selected data as support vectors. The resulting algorithm, Sparse DKT, reduces the data to a small number of support vectors for each task. In the Sparse DKT algorithm, only the support vectors at the test time directly influence the prediction of the test data label. The experimental results show that Sparse DKT, in addition to interpretability, has comparable accuracy to other state-of-the-art meta-learning methods, including the DKT algorithm.

The main contributions of this article are:

1. Introducing learning with the medium-shot setting and utilizing deep meta-learning algorithms for it;
2. Learning a sample-based interpretable model using the attention mechanism;
3. Applying sparse kernel methods for determining a small

[*] Manuscript received; 05 July 2022, Revised, 06 October 2022, Accepted, 09 October 2022.

[1]. PhD student, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

[2]. Corresponding author, Assistant Professor, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
Email: k.ghiasi@um.ac.ir

subset of training data as support vectors.

The remaining structure of this article is as follows: in section 2, the basic concepts about meta-learning, interpretability, attention perspective, sparse kernel, and related works are described. In section 3, the proposed algorithm is presented. The evaluation of the presented algorithm in classification will be in the section 4. In section 5, conclusion and future works are presented.

2. Preliminaries

2.1. Meta-learning

Meta-learning is one of the areas that has received attention in recent years [9-34]. In classic learning, in order to learn a task, the model is trained on the task data in such a way that it has a good generalization of the new data. The objective of meta-learning, also known as learning to learn, is to go to a higher level and understand how to solve tasks rather than just learning a single task (Figure 1). Humans face with different issues over time and develop better ways to deal with new ones by drawing on their experiences. Similar to humans, we should train the model on a set of tasks from the same distribution sequentially in meta-learning. By completing each task, we acquire metadata that the model can use to learn a new, unseen task more effectively and quickly.

A. Meta-learning setup

In meta-learning, as shown in Figure 2, instead of one task, we have a set of tasks, $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^{T}$, which are from the same distribution. According to Figure 2, for each task, indexed by τ, we have the data $\mathcal{D}_\tau = \{X, y\}$, which can be divided into two parts, the train/support set, D_τ^{tr}, and the test/query set, $\mathcal{D}_\tau^{ts}$. The test data that is used for meta-test is denoted by the asterisk symbol as $\mathcal{D}_* = \{\mathcal{D}_*^{tr}, \mathcal{D}_*^{ts}\}$.

B. Few-shot learning

Few-shot learning refers to tasks with a few training data. For example, in the few-shot classification represented as N way - K shot, N is the number of classes in the task, and K (usually considered 1 or 5) training samples are available for each class (Figure 2 shows 3 way- 2 shot classification). Few-shot learning aims to make deep neural networks capable of learning a new concept by observing a small number of training samples. The small amount of training data makes it infeasible to train the deep neural network, but the meta-learning approach has achieved significant improvements in few-shot learning. Deep meta-learning learns a model that can solve a new task despite the small training data. Medium-shot learning is a generalization of few-shot learning, so we employ the meta-learning framework.

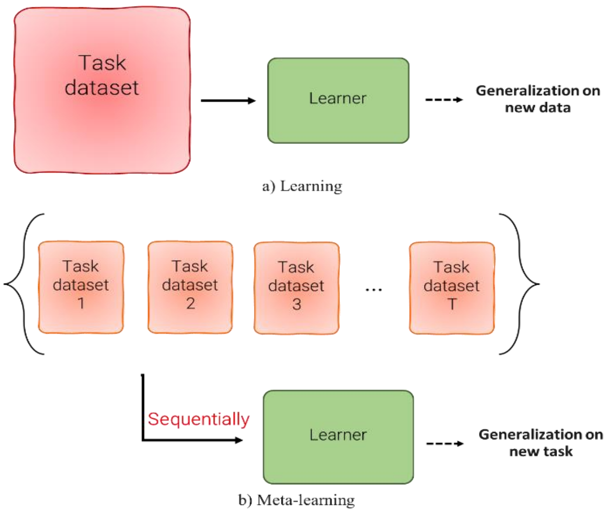


Figure 1. Difference between a) learning and b) meta-learning. In learning, training on a task data is done to generalize new data from the same dataset. In meta-learning, we train the model on a set of tasks sequentially. By learning to learn, we can solve the new task more efficiently and quickly.

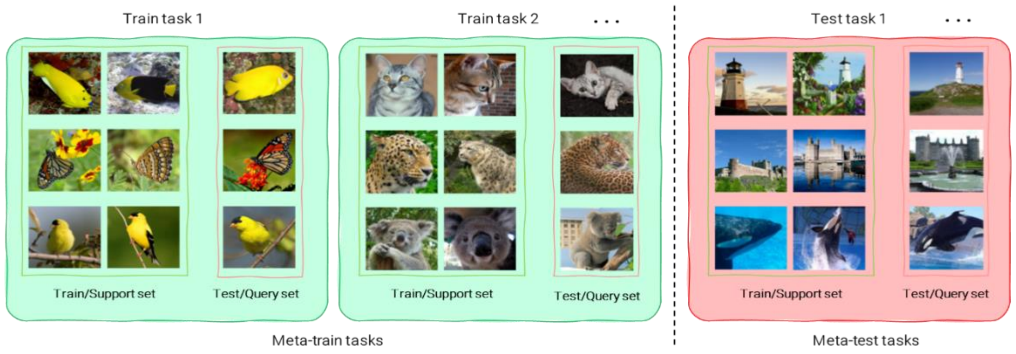


Figure 2. An example of a meta-learning setup for few-shot learning. The set of tasks $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^{T}$ is divided into two parts, meta-train and meta-test. The data of each task has train and test sets, D_τ^{tr} and D_τ^{ts} respectively

2.2. Interpretability in Deep Neural Networks

In deep learning, there are two main classes of approaches to explain the prediction of a model: feature-based and sample-based. In the feature-based approach, features from the input image that have a greater impact on the model's prediction are identified [35, 36]. The idea of [36] in few-shot learning has been applied in [37] to provide interpretable feature-based meta-learning.

In the sample-based approach, the data that have the most impact on the network's decision-making for test data are identified as samples to interpret its prediction (Figure 3) [38, 39]. ProtoAttend [40] trains a network that compares the input data with training data to predict it based on the attention mechanism and learn an attention weight that demonstrates the degree of similarity between them. To interpret the model's decision for the input data, the data whose weight is not zero affect the model's prediction and are selected as prototypes. Because there are a lot of data in deep learning and it is difficult to compare them all, a subset of the data is typically chosen as a candidate set, and attention weight is only learned for the candidate set. In contrast, the number of data is not large in medium-shot setting, and since we can evaluate all the training data, sample-based interpretability is possible. In order to achieve this, we proceed according to the attention point of view.

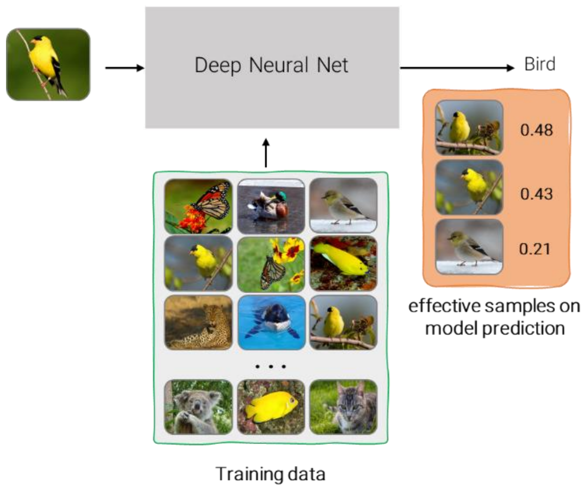


Figure 3. In sample-based interpretability, the training data that the model used to determine the label for the input data are specified.

A. Sample-based interpretability through Attention

Using the attention perspective, we can learn a model with sample-based interpretability [41]. Simply it means to compare the input data with the training data and give greater weight to the training data that is more similar to the input data when determining its label. To compare the data properly, we need to learn a metric space in which similar data are placed close together, and dissimilar data are far apart. This method is used in the metric-based meta-learning algorithms presented for few-shot learning [14–16]. In these papers, since the number of training samples is small, there is no need for sample-based interpretability, and the main objective is to increase accuracy. Since we have more data in medium-shot learning, sample-based interpretability becomes important; in some applications, explaining the model's behavior with a small number of samples makes it easier for humans to understand and evaluate the model.

B. Attention and kernel methods

The attention mechanism and kernel methods are closely related [42–45]. It can be said that the idea of attention in deep learning is derived from kernel methods [42]. Kernel methods have a kernel function $k(\boldsymbol{x},\boldsymbol{x}')$ that determines the degree of similarity [46]. Linear kernel, polynomial, RBF (Radial Basis Function), and exponential are the well-known kernel functions. Learning the kernel function corresponds to learning its parameters, e.g., in the RBF kernel

$$k(\boldsymbol{x},\boldsymbol{x}') = s * \exp\{-\tfrac{1}{l}\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2\} \qquad (1)$$

the parameters $\boldsymbol{\phi} = \{l, s\}$ are learned during training.

In deep kernel learning or DKL [47–50], we first use a deep neural network to obtain data representations, then apply a kernel function to them. The new deep kernel is

$$k(\boldsymbol{x},\boldsymbol{x}') = \tilde{k}_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), f_{\boldsymbol{\theta}}(\boldsymbol{x}')) \qquad (2)$$

where $\tilde{k}_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{x}')$ is the kernel function with parameter $\boldsymbol{\phi}$ and $f_{\boldsymbol{\theta}}$ is a deep neural network. DKL involves jointly learning kernel and network parameters. For example, optimization of the parameters in the regression of $\{X,\boldsymbol{y}\}_{n=1}^{N}$ with noise variance σ^2 is based on the log marginal likelihood,

$$\log p(\boldsymbol{y}|X) =$$
$$\frac{1}{2}\{-\boldsymbol{y}^{\top}[K+\sigma^2 I]^{-1}\boldsymbol{y} - \log|K+\sigma^2 I| + \mathrm{N}\log(2\pi)\} \qquad (3)$$

where K is the kernel matrix on the training data.

2.3. Deep Kernel Transfer

Deep Kernel Transfer or DKT falls into the category of metric-based meta-learning [8]. This class of algorithms tries to learn a metric space to compare representations based on a distance measure [14–16]. DKT is a combination of MAML (Model-Agnostic Meta-Learning) and DKL for few-shot learning. MAML [21] is based on the idea of [13] without using an additional model as a meta-learner, learns a meta-parameter as an initialization for the parameters of the network. The meta-parameter adapts quickly to the data of the new task without overfitting due to a few training data.

The computational graph of the MAML is shown in Figure 4a Using SGD (Stochastic Gradient Descent) optimization on the task training data, the MAML algorithm obtains task-specific parameter $\boldsymbol{\phi}_{\tau}$ from the meta-parameter $\boldsymbol{\theta}$. The inner loop (adaptation loop) of the MAML has a parametric form, so in the outer loop, we encounter the second gradient of $\boldsymbol{\theta}$ with respect to the optimization path in the inner loop.

The idea of DKT is to replace the inner loop computation with a Gaussian process, which has a non-parametric form. Therefore, as shown in Figure 4b, adaptation to the task is eliminated. Similar to the DKL, a Gaussian process is applied to the representations. DKT computes the marginal likelihood (3) on the data of each task and optimizes the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. By meta-learning a deep kernel on a set of tasks, we have a kernel that can be transferred to a new task without needing adaptation. By replacing the inner loop with the Gaussian process, the DKT algorithm provides a computational simplification for the MAML. Furthermore, it is regarded as a Bayesian meta-learning. In the regression and image classification in few-shot settings, DKT has achieved higher accuracy than MAML and other few-shot learning methods.

2.4. Sparse kernel methods

SVM (Support Vector Machine) is a popular sparse kernel method [51]. The Sparsity of SVM results from zeroing coefficient α for part of the data during the quadratic optimization, which determines a subset of data as support vectors. In few-shot learning, the MetaOptNet [30] has used SVM to simplify the inner loop of MAML to obtain the task-specific parameter without SGD optimization and not to encounter the second derivative in meta-parameter optimization (Figure 4c).

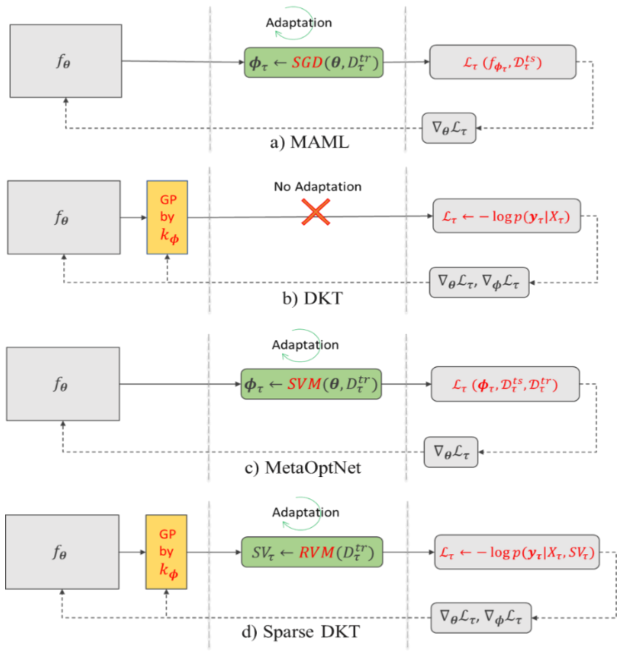

Adaptation
f_θ
$\phi_\tau \leftarrow SGD(\theta, D_\tau^{tr})$
$\mathcal{L}_\tau\,(f_{\phi_\tau}, \mathcal{D}_\tau^{ts})$
$\nabla_\theta \mathcal{L}_\tau$
a) MAML
No Adaptation
f_θ
GP by k_ϕ
$\mathcal{L}_\tau \leftarrow -\log p(\boldsymbol{y}_\tau|X_\tau)$
$\nabla_\theta \mathcal{L}_\tau, \nabla_\phi \mathcal{L}_\tau$
b) DKT
Adaptation
f_θ
$\phi_\tau \leftarrow SVM(\theta, D_\tau^{tr})$
$\mathcal{L}_\tau\,(\phi_\tau, \mathcal{D}_\tau^{ts}, \mathcal{D}_\tau^{tr})$
$\nabla_\theta \mathcal{L}_\tau$
c) MetaOptNet
Adaptation
f_θ
GP by k_ϕ
$SV_\tau \leftarrow RVM(D_\tau^{tr})$
$\mathcal{L}_\tau \leftarrow -\log p(\boldsymbol{y}_\tau|X_\tau, SV_\tau)$
$\nabla_\theta \mathcal{L}_\tau, \nabla_\phi \mathcal{L}_\tau$
d) Sparse DKT

Figure 4. Computational graph of a) MAML, b) DKT, c) MetaOptNet, and d) Sparse DKT (ours). In a), adapting to the task is equivalent to obtaining the task-specific parameter $\boldsymbol{\phi}_\tau$. In b), meta-parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ without adapting to the task are updated based on the marginal likelihood of the Gaussian process on the entire data. In c), the task-specific parameter $\boldsymbol{\phi}_\tau$ is computed by applying SVM to the training data of the task. In d), adapting to the task is equivalent to specifying the support vectors, a small subset of the task training data

The disadvantage of SVM in the MetaOptNet algorithm is that it becomes less effective in sparsifying as the data increases. Another disadvantage of SVM compared to the Gaussian process [52] is that it is not probabilistic. In contrast, the Gaussian process is not inherently sparse; the kernel matrix is calculated between the test data and all n training data at test time. Several sparse approximations have been proposed to overcome the computational and memory complexity in the Gaussian process [53, 54]. Almost all of these approximation methods specify a criterion to determine the significance of the data and greedily select a subset of the data of size $m \ll n$ to be used in the kernel matrix approximation. The main goal of methods in [55–59] is to reduce the computational complexity of the Gaussian process by assuming that there is a set of support vectors. The criteria to determine the support vectors in these methods are usually considered for adding data to this set, so the number of support vectors is defined as a fixed hyperparameter. However, since these vectors are supposed to have the most impact on the model's prediction, we are looking for support vectors to be automatically selected with a small number and high accuracy. Additionally, in the medium-shot learning, the number of data selected as support vectors should depend on the task. Therefore, in the proposed algorithm, intending to achieve sample-based interpretability using Gaussian processes, we leverage the sparse Bayesian approach, which we will explain in the following section.

3. Sparse DKT for medium-shot learning

This section presents our meta-learning algorithm, Sparse DKT, for medium-shot learning. To achieve sample-based interpretability, we need to determine the importance of data in data modeling and prediction. We measure the degree of importance with the kernel function, so we use DKT. We modify this algorithm to attain sample-based interpretability and apply attention to it in two ways: attention in adaptation and attention in prediction. Attention in adaptation is independent of the test data and is performed only on the training data. The Sparse Gaussian process is trained on the task data; In other words, it adapts to it, and the result of this adaptation is the identification of support vectors.

In contrast, attention in prediction depends on the test data but uses only support vectors from the entire training data. Due to the usage of Gaussian processes, we already have attention in prediction; that is, support vectors affect test label prediction based on how similar they are to it. We discuss the proposed algorithm for regression, but it can be easily generalized for classification.

3.1. Sparse Gaussian process as Adaptation

In the sparse Bayesian learning framework, Tipping introduces the RVM algorithm (Relevance Vector Machine) [60]. The advantage of this algorithm we adopted for our proposed algorithm is that it automatically selects the data that play the main role in data modeling when adapting to the task.

This algorithm is essentially a Gaussian process. Assume that we have data $= \{X, \boldsymbol{y}\}$, including the inputs $X = \{\boldsymbol{x}_j\}_{j=1}^{n}$ and the labels $\boldsymbol{y} = \{y_j\}_{j=1}^{n}$. Labels have Gaussian noise $\epsilon_j \sim \mathcal{N}(0, \sigma^2)$ added to latent function $f(\boldsymbol{x})$ according to $y(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) + \epsilon_j$. The prior knowledge on the function $f(\boldsymbol{x})$ is a Gaussian process $\mathcal{GP}(\mu, k_{\boldsymbol{\phi}})$ with mean μ and kernel function $k_{\boldsymbol{\phi}}$. The mean is usually considered zero.

We can rewrite the latent function $\boldsymbol{f}$ in the parametric form $\boldsymbol{f} = K\boldsymbol{w}$ in the equation $\boldsymbol{y} = \boldsymbol{f} + \boldsymbol{\epsilon}$. K is the covariance matrix based on the kernel function $k_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{x}')$. In the Gaussian process, the weight $\boldsymbol{w}$ has a Gaussian distribution $\mathcal{N}(0, \alpha_0^{-1} I)$, where α_0 is a hyperparameter. In RVM, Gaussian distribution $p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}(0, A^{-1})$ is considered for weights, where $A = diag(\boldsymbol{\alpha})$ is a diagonal covariance matrix. As a result, RVM is a Gaussian process with kernel function:

$$c(\boldsymbol{x},\boldsymbol{x}') = \sum_{i=1}^{n} \frac{1}{\alpha_i} k_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{x}_i) k_{\boldsymbol{\phi}}(\boldsymbol{x}',\boldsymbol{x}_i) \tag{4}$$

where $k_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{x}_i)$ is equal to the kernel function that is defined based on the training data $\boldsymbol{x}_i$. $c(\boldsymbol{x},\boldsymbol{x}')$ is an expansion of the product of values of the kernel function $k_{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{x}_i)$ in which all data contribute. The kernel function of the data that will be included in the expansion is determined by coefficient α_i. When α_i goes to infinity, the kernel function corresponding to $\boldsymbol{x}_i$ data is removed; as a result, the expansion $c(\boldsymbol{x},\boldsymbol{x}')$ becomes sparse. The covariance matrix of RVM can be expressed as $C = KA^{-1}K^{\top}$.

The next step is that based on Bayes Equation 5, and having likelihood $p(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{f},\sigma^2 I)$,

$$p(\boldsymbol{w}|\boldsymbol{\alpha},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{W})p(w|\alpha)}{p(y)} \tag{5}$$

obtain the posterior distribution of the weight,

$$p(\boldsymbol{w}|\boldsymbol{\alpha},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu},\Sigma)$$

$$\boldsymbol{\mu} = \sigma^{-2}\Sigma K^{\top}\boldsymbol{y}$$

$$\Sigma = (A + \sigma^{-2}K^{\top}K)^{-1}. \tag{6}$$

RVM training is similar to Gaussian process training; We optimize the logarithm of the marginal likelihood (7) with respect to the hyperparameters $\boldsymbol{\alpha}$ and σ^2.

$$p(\boldsymbol{y}) = \mathcal{N}(0, C + \sigma^2 I)$$

$$\log p(\boldsymbol{y}) =$$

$$-1/2\{\boldsymbol{y}^{\top}[C + \sigma^2 I]^{-1}\boldsymbol{y} + log|C + \sigma^2 I| + nlog2\pi\} \tag{7}$$

By deriving the Equation 7 with respect to $\boldsymbol{\alpha}$ and σ^2 and setting them equal to zero, optimization equations are obtained as follows:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2}$$

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}$$

$$(\sigma^{-2})^{new} = \frac{||\boldsymbol{y} - K\boldsymbol{\mu}||^2}{n - \Sigma_j \gamma_j} \tag{8}$$

where Σ_{ii} is the i-th diagonal component of the covariance matrix Σ in (6). $\gamma_i \in [0,1]$ indicates how much the data contributed to the determination of w_i. To get $\boldsymbol{\alpha}$ and σ^2, we can use an iterative algorithm. During training, many α_i become infinite, which causes variance and mean corresponding to their weights to be zero. When weight w_i becomes zero, the kernel function at $\boldsymbol{x}_i$ does not contribute to describing the data so that it can be removed from the model. The data that have non-zero weight are considered as support vectors. Another method to train RVM is to use the Expectation-Maximization algorithm [61]. In this study, we use the sequential algorithm proposed in the [62] (The authors of [62] published their code in MATLAB, and we re-implemented it with Python. http://www.miketipping.com/sparsebayes.htm). In this algorithm, the set of support vectors is initially empty, and important data are added to this set sequentially. The computational cost of RVM is significantly decreased by using this addition method, which is better for learning in the medium-shot setting.

3.2. Sparse DKT algorithm

The Sparse DKT algorithm using RVM as the inner loop, on the one hand, is a simplification for the MAML; on the other hand, it adds interpretability to the DKT. According to Figure 4, the difference between DKT and Sparse DKT is the addition of the adaptation loop. Unlike MetaOptNet, in Sparse DKT, the parameters of the kernel function are part of the meta-parameters and are updated by loss of each task. Sparse DKT Pseudocode is given in Algorithm 1. In meta-training, what is important for us from utilizing the RVM algorithm as the inner loop of Sparse DKT is to obtain $\boldsymbol{\alpha}$. We are interested in learning which data are most important in describing the whole data and consequently in the model's prediction. The Sparse DKT algorithm selects the data whose α coefficient is not infinite as task support vectors. In the outer loop, they are used in the optimization with RVM marginal likelihood (7).

Algorithm 1. Sparse Deep Kernel Transfer (Sparse DKT)

```
Require: $\mathcal{M} = \{\mathcal{D}_\tau\}_{\tau=1}^{T}$ meta-train tasks
Require: $\boldsymbol{\phi}$ kernel hyperparameters, $\boldsymbol{\theta}$ neural network weights
Require: $\beta_1, \beta_2$ step size
1:   while not done do
2:     Sample $\mathcal{D}_\tau$ from $\mathcal{M}$
3:     SV= RVM($\mathcal{D}_\tau$) //Obtain support vectors of $\mathcal{D}_\tau$ with RVM
4:     //Use marginal likelihood to update parameters
5:     $\mathcal{L}_\tau = -log\ p(\boldsymbol{y}|X,\boldsymbol{\phi},\boldsymbol{\theta})$   //Eq (7)
6:     $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \beta_1\nabla_{\boldsymbol{\phi}}\mathcal{L}_\tau$ , $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_2\nabla_{\boldsymbol{\theta}}\mathcal{L}_\tau$
7:   end while
8:   function RVM($\mathcal{D}$)
9:     //Automatically select support vectors
       //of the dataset $\mathcal{D}$
10:    Initialize $\boldsymbol{\alpha}$ and $\sigma^2$
11:    while not converged:
12:      Update $\boldsymbol{\mu}$ and $\Sigma$     //Eq (6)
13:      Update $\boldsymbol{\alpha}$ and $\sigma^2$    // (8)
14:    return support vectors from $\mathcal{D}$ for finite $\alpha_i$ values
15:  end function
```

At the meta-test time, for the test task with data $\mathcal{D}_*^{tr} = \{X, \boldsymbol{y}\}$ and $\mathcal{D}_*^{ts}$, the support vectors of the task are first selected from the training data $\mathcal{D}_*^{tr}$ by running RVM. In addition to the support vectors, the mean and covariance of the posterior weight distribution are also obtained, which we use in the RVM prediction distribution,

$$p(y_*|X,\boldsymbol{y},\boldsymbol{x}_*) = \mathcal{N}(\mu_*,\sigma_*^2)$$

$$\mu_* = \boldsymbol{k}_*\boldsymbol{\mu}, \qquad \boldsymbol{\mu} = \sigma^{-2}\Sigma\, K_{mn}y$$

$$\sigma_*^2 = \sigma^2 + \boldsymbol{k}_*\Sigma\,\boldsymbol{k}_*, \qquad \Sigma = (A + \sigma^{-2}K_{mn}K_{nm})^{-1} \tag{9}$$

where $\boldsymbol{k}_*$ is the covariance between $\boldsymbol{x}_* \in \mathcal{D}_*^{ts}$ and m support vectors. K_{mn} is the covariance between support vectors and training data.

4. Experiments

We run classification tests using common datasets in few-shot learning in a medium-shot setting to evaluate the Sparse DKT algorithm. The number of samples has been chosen in such a way that we get out of the few-shot mode. We used PyTorch and GPyTorch [63] for the implementation of the Sparse DKT.

To compare Sparse DKT with DKT, Feature Transfer, MAML, and MetaOptNet, we have considered Omniglot, CUB-200, and miniImageNet dataset for image classification (Figure 5).

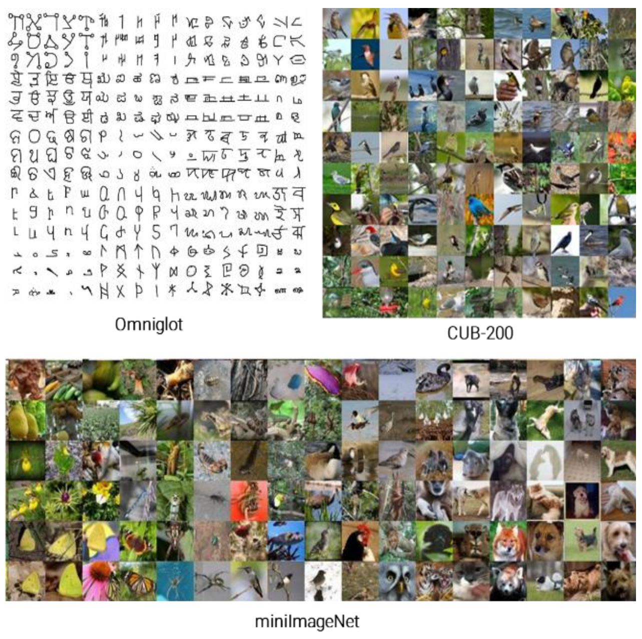


Figure 5. Images from datasets used in classification

Omniglot consists alphabet of 50 languages and has 20 hand-written samples for each character. CUB-200 contains 200 classes of different bird species. MiniImageNet has 100 classes which is a subset of ImageNet classes. Each class has 600 images. We run 2-way and 5-way classifications test. As in the DKT paper, classification is done one-versus-rest (Figure 6), i.e., for each class, we consider a binary Gaussian process model with labels {-1,1} and apply the sigmoid function to its output in order to have a probabilistic interpretation (for MetaOptNet, we also used binary SVMs for multi-class classification in experiments). The model whose output has the highest probability determines the class of the test data. We used a linear kernel in experiments and a deep neural network that has a similar architecture to the network used in the DKT paper (Figure 7).

In Feature Transfer, a network and classifier are first trained on samples for the training classes. When fine-tuning, the network parameters are fixed, and a new classifier is trained on the test classes. MAML depends on the number of gradient steps in the inner loop and has low accuracy at a few steps. Increasing the gradient steps also leads to an increase in computation and memory consumption. In order to be able to test MAML in 10 steps adaptation, we used its first order approximation [28]. Table 1 shows the result of Omniglot 5 way- 15 shot classification.

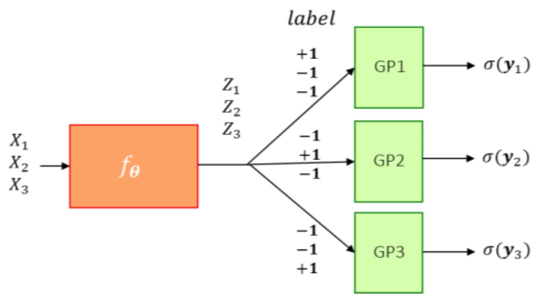


Figure 6. One-versus-rest scheme. Each model is a binary classifier for input data with labels {-1, 1}. For a probabilistic output, a sigmoid function σ is applied to it.

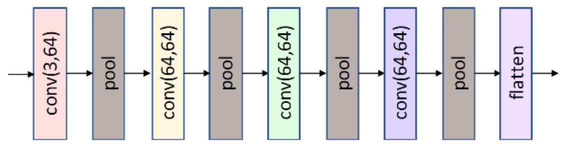


Figure 7. The CNN used as a backbone for classification. It consists of 4 convolutional layers, each consisting of a 2D convolution, a batch-norm layer, and a ReLU non-linearity.

Table 1. Average accuracy and standard deviation on Omniglot classification with average number of support vectors

Method	Omniglot 5 way - 15 shot	SVs
Feature Transfer	99.36±0.08	-
MAML	95.80±0.312	-
DKT	99.52±0.211	75
MetaOptNet	99.46±0.141	13
Sparse DKT	99.33±0.1	6

Sparse DKT is more accurate than MetaOptNet and close to DKT, while DKT uses all training data of 5 classes as support vectors for its prediction. MAML can achieve more accuracy at the cost of more adaptation steps. Table 2 shows the classification results of CUB and miniImageNet. Due to the limited resources in this section, we had to run 2-way classification. The number of task training data in CUB and miniImageNet is 50 and 125, respectively. Feature transfer overfits in the few-shot setting. However, it was able to get higher accuracy than other methods in our experiments. We believe that the accuracy of Feature Transfer decreases when the new task's classes diverge more from the training classes. We leave further investigations to future works.

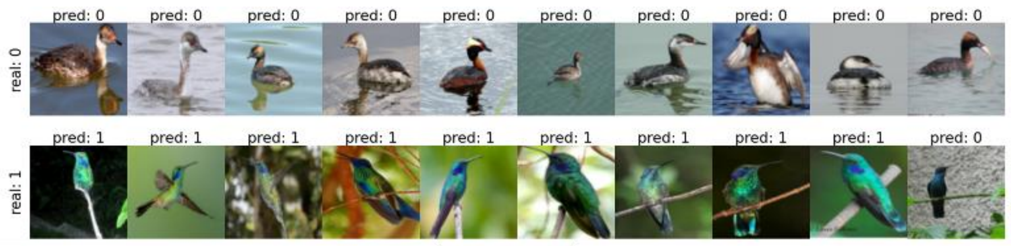
Sparse DKT is more interpretable and has higher accuracy than MetaOptNet, with a smaller number of support vectors. The efficiency of MetaOptNet in sparsity decreases with the increase of training data due to the weakness of SVM. In miniImageNet classification, the proposed method has selected 14 support vectors on average from 250 data, while MetaOptNet has selected 76 support vectors. Additionally, the experiments on these different datasets show that the number of support vectors for each application depends on intra-class and inter-class similarity. The metric space learned by the Sparse DKT to separate classes affects the number of support vectors.

In Figure 8, we have given an example of testing the trained model with the Sparse DKT and DKT on a 2 way – 50 shot classification task from the CUB meta-test dataset.

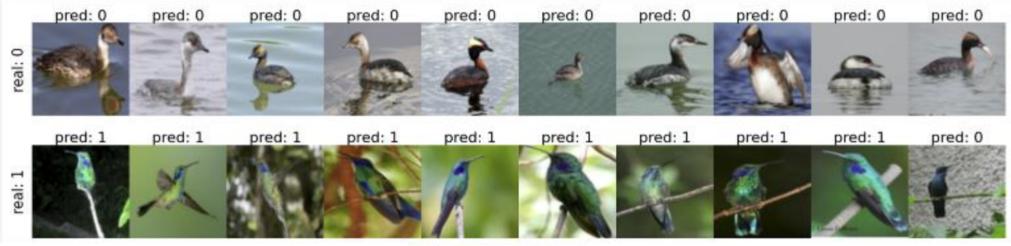
In this task, Sparse DKT has the same accuracy as DKT. In Figure 9, task training data are shown, and the data that are support vectors have been marked with a red line around the image.

Table 2. Average Accuracy and Standard Deviation on CUB and miniImageNet Classification with Average number of Support Vectors

Method	CUB 2 way - 50 shot	SVs	miniImageNet 2 way - 125 shot	SVs
Feature Transfer	95.23±0.381	-	93.13±0.530	-
MAML	92.33±1.069	-	85.63±0.176	-
DKT	93.98±0.448	100	92.0±0.4	250
MetaOptNet	92.27±1.313	33	89.70±0.56	76
Sparse DKT	93.75±0.909	21	91.08±0.913	14

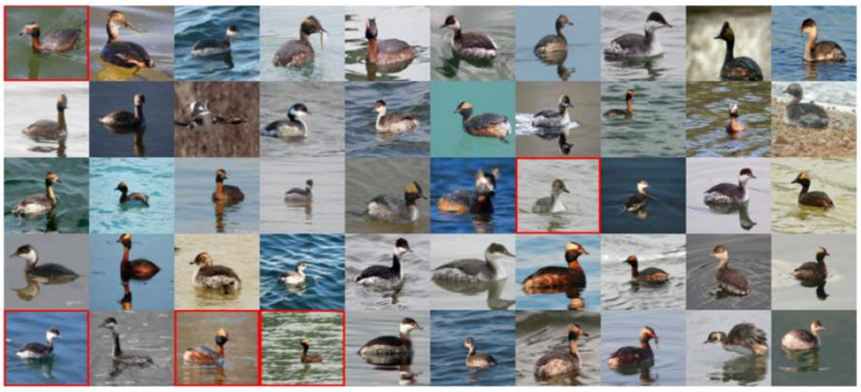


(a)

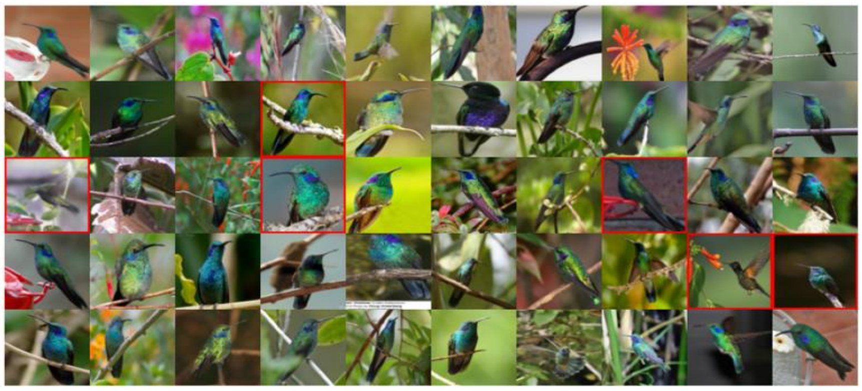


(b)

Figure 8. Comparing a) Sparse DKT and b) DKT accuracies on a CUB meta-test task. Sparse DKT has the same accuracy as DKT.



(a)



(b)

Figure 9. Sample-based interpretability of Sparse DKT in CUB 2 way – 50 shot. Support vectors of the two classes (a, b), highlighted with a red square, are the basis of the model's prediction.

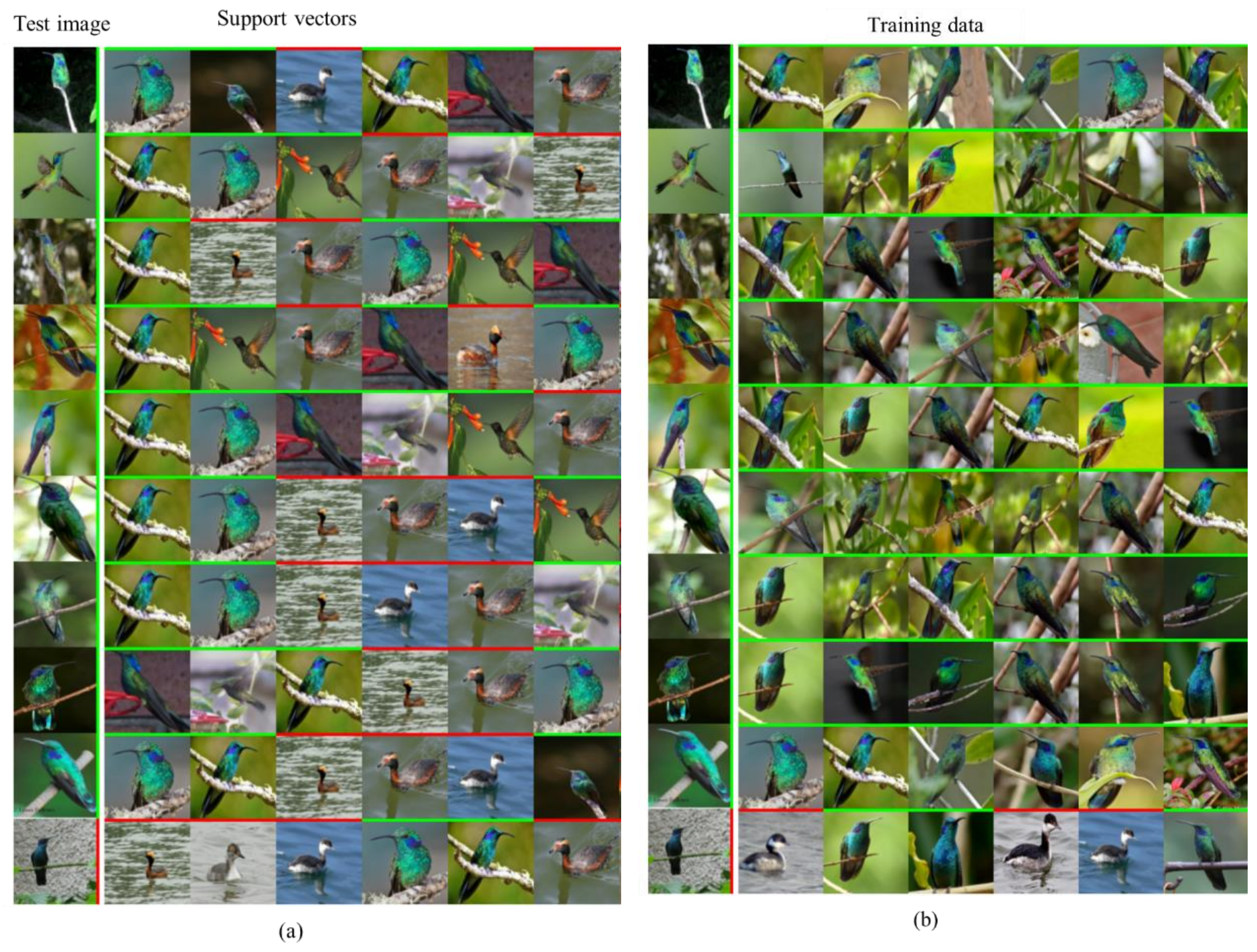


Figure 10. Comparing kernels of a) Sparse DKT and b) DKT: a) the most similar support vectors to test image, b) the most similar training data to test images. The green line above the images on the right, shows that they have the same label as the test image.

We compared the learned kernels of Sparse DKT and DKT in Figure 10. For Sparse DKT similarity of test images to support vectors is computed. In each row, the images are sorted in the order of the most similar from left to right. The green and red lines on top of the right image, show whether the labels of the right images are the same or different from those of the test image. The vertical green line in the test image indicates that the model accurately predicted the label. The Sparse DKT kernel can detect the similarity well, even though the number of support vectors is very small.

5. Conclusion

In this study, we introduced medium-shot learning as a generalization of few-shot learning for real-world applications. Considering that interpretability in deep learning models is becoming increasingly more important, especially in sensitive scenarios, sample-based interpretability can be easily obtained by reducing the data to a small number of support vectors in medium-shot learning. We considered sparse kernel methods from an attention-based perspective to have sample-based interpretability. The proposed Sparse DKT algorithm leverages Sparse Gaussian processes in the meta-learning framework and selects the most important training data as support vectors. At the test time, it makes the predictions based on support vectors.

The impact of marginal likelihood in the trade-off between accuracy and the number of support vectors, as well as the impact of more task training data, is one of the key areas for future work. Using improved versions of RVM [64, 65] would be effective in increasing the accuracy of Sparse DKT. Since SVM in MetaOptNet is less effective in sparsifying, When data increases, we can use GLASSO [66], which also has a probabilistic solution, as an alternative to SVM in MetaOptNet. Another future work is investigating variational sparse Gaussian processes [67–70] that use variational inference for increasing the lower bound of the marginal likelihood algorithm. We can use the combination of point processes [71] with it to determine the support vectors in sparse variational Gaussian processes.

6. References

[1] Goodfellow, I., Bengio, Y., and Courville, A., Deep learning, MIT press, 2016.

[2] Wang, Y., Yao, Q., Kwok, J., and Ni, L. M., "Generalizing from a Few Examples: A Survey on Few-Shot Learning", *ACM Computing Surveys (CSUR)*, Vol. 53, No. 3, Apr. 2019, Accessed: Jan. 23, 2022. [Online]. Available: http://arxiv.org/abs/1904.05046.

[3] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B., "Human-level concept learning through probabilistic program induction", *Science*, Vol. 350, No. 6266, pp. 1332–8, doi: 10.1126/science.aab3050, Dec, 2015.

[4] Jiang, X., Ding, L., Havaei, M., Jesson, A., and Matwin, S., "Task Adaptive Metric Space for Medium-Shot Medical Image Classification", in *MICCAI*, pp. 147–155. doi: 10.1007/978-3-030-32239-7_17, 2019.

[5] Li, X., Sun, Z., Xue, J.-H., and Ma, Z., "A concise review of recent few-shot meta-learning methods", *Neurocomputing*, Vol. 456, pp. 463–468, doi: 10.1016/j.neucom.2020.05.114, Oct, 2021.

[6] Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., and Müller, K.-R., *Explainable AI: interpreting, explaining and visualizing deep learning*, Vol. 11700. Springer Nature, 2019.

[7] Zhang, Y., Tino, P., Leonardis, A., and Tang, K., "A Survey on Neural Network Interpretability", *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 5, No. 5, pp. 726–742, doi: 10.1109/TETCI.2021.3100641, Oct, 2021.

[8] Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A., "Bayesian Meta-Learning for the Few-Shot Setting via Deep Kernels", *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Oct. 2019, Accessed: Feb. 04, 2021. [Online]. Available: http://arxiv.org/abs/1910.05199.

[9] Huisman, M., van Rijn, J. N., and Plaat, A., "A survey of deep meta-learning", *Artificial Intelligence Review*, Vol. 54, No. 6, pp. 4483–4541, doi: 10.1007/s10462-021-10004-4, Aug. 2021.

[10] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T., "One-shot Learning with Memory-Augmented Neural Networks", *33rd International Conference on Machine Learning, ICML 2016*, Vol. 4, pp. 2740–2751, May 2016, Accessed: Oct. 16, 2019. [Online]. Available: http://arxiv.org/abs/1605.06065

[11] Munkhdalai, T., and Yu, H., "Meta Networks", *34th International Conference on Machine Learning, ICML*, Vol. 5, pp. 3933–3943, Mar. 2017, Accessed: Oct. 18, 2019. [Online]. Available: http://arxiv.org/abs/1703.00837

[12] Andrychowicz, M., *et al.*, "Learning to learn by gradient descent by gradient descent", *Advances in Neural Information Processing Systems*, pp. 3988–3996, Jun. 2016, Accessed: Oct. 16, 2019. [Online]. Available: http://arxiv.org/abs/1606.04474

[13] Ravi, S., and Larochelle, H., "Optimization as a model for few-shot learning", 2017.

[14] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D., "Matching Networks for One Shot Learning", *Advances in Neural Information Processing Systems*, pp. 3637–3645, Jun. 2016, Accessed: Oct. 18, 2019. [Online]. Available: http://arxiv.org/abs/1606.04080

[15] Snell, J., Swersky, K., and Zemel, R. S., "Prototypical Networks for Few-shot Learning", *Advances in Neural Information Processing Systems*, Vol. 2017-Decem, pp. 4078–4088, Mar. 2017, Accessed: Oct. 18, 2019. [Online]. Available: http://arxiv.org/abs/1703.05175

[16] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., and Hospedales, T. M., "Learning to Compare: Relation Network for Few-Shot Learning", in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208. doi: 10.1109/CVPR.2018.00131, Jun, 2018.

[17] Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E., "Meta-Learning Probabilistic Inference For Prediction", *7th In International Conference on Learning Representations, ICLR*, May 2018, Accessed: Jan. 23, 2022. [Online]. Available: http://arxiv.org/abs/1805.09921

[18] Finn, C., Xu, K., and Levine, S., "Probabilistic Model-Agnostic Meta-Learning", *Advances in Neural Information Processing Systems*, pp. 9516–9527, Jun. 2018.

[19] Garnelo, M., *et al.*, "Conditional Neural Processes", *35th International Conference on Machine Learning, ICML 2018*, Vol. 4, pp. 2738–2747, Jul. 2018, Accessed: Apr. 21, 2020. [Online]. Available: http://arxiv.org/abs/1807.01613

[20] Edwards, H., and Storkey, A., "Towards a Neural Statistician", *5th International Conference on Learning Representations, ICLR*, Accessed: Apr. 26, 2020, Jun, 2016. [Online]. Available: http://arxiv.org/abs/1606.02185

[21] Finn, C., Abbeel, P., and Levine, S., "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", *34th International Conference on Machine Learning, ICML 2017*, Vol. 3, pp. 1856–1868, Mar. 2017, Accessed: Oct. 19, 2019. [Online]. Available: http://arxiv.org/abs/1703.03400

[22] Kim, T., Yoon, J., Dia, O., Kim, S., Bengio, Y., and Ahn, S., "Bayesian Model-Agnostic Meta-Learning", *Advances in Neural Information Processing Systems*, pp. 7332–7342, Jun. 2018.

[23] Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T., "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes", *6th International Conference on Learning Representations, ICLR*, Jan. 2018.

[24] Ravi, S., and Beatson, A., "Amortized bayesian meta-learning", 2018.

[25] Raghu, A., Raghu, M., Bengio, S., and Vinyals, O., "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML", *8th International Conference on Learning Representations, ICLR 2020*, Sep. 2019.

[26] Oh, J., Yoo, H., Kim, C., and Yun, S., "BOIL: Towards Representation Change for Few-shot Learning", 2021.

[27] Zintgraf, L., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S., "Fast context adaptation via meta-learning", in *36th International Conference on Machine Learning, ICML 2019*, Vol. 2019-June, pp. 13262–13276, 2019.

[28] Nichol, A., Achiam, J., and Schulman, J., "On First-Order Meta-Learning Algorithms", Mar. 2018, Accessed: Apr. 13, 2020. [Online]. Available: http://arxiv.org/abs/1803.02999

[29] Rajeswaran, A., Finn, C., Kakade, S., and Levine, S., "Meta-Learning with Implicit Gradients", *Advances in Neural Information Processing Systems 32(pp. 113-124)*, Sep. 2019, Accessed: Aug. 16, 2020. [Online]. Available: http://arxiv.org/abs/1909.04630

[30] Lee, K., Maji, S., Ravichandran, A., and Soatto, S., "Meta-Learning With Differentiable Convex Optimization", in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10649–10657. doi: 10.1109/CVPR.2019.01091, Jun,

2019.

[31] Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A., "Meta-learning with differentiable closed-form solvers", *7th In International Conference on Learning Representations, ICLR*, May 2018, [Online]. Available: http://arxiv.org/abs/1805.08136

[32] Gai, S., and Wang, D., "Sparse Model-Agnostic Meta-Learning Algorithm for Few-Shot Learning", in *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*, pp. 127–130, Sep. 2019. doi: 10.1109/CCHI.2019.8901909.

[33] Madan, A., and Prasad, R., "B-Small: A Bayesian Neural Network Approach to Sparse Model-Agnostic Meta-Learning", *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2730–2734, 2021.

[34] Tian, H., Liu, B., Yuan, X. -T., and Liu, Q., "Meta-learning with Network Pruning", in *Computer Vision – ECCV 2020. Lecture Notes in Computer Science*, Springer, Cham, pp. 675–700. doi: 10.1007/978-3-030-58529-7_40, 2020.

[35] Sabzevar, R. Z., Ghiasi-Shirazi, K., and Harati, A., "Prototype-based interpretation of the functionality of neurons in winner-take-all neural networks", *ArXiv*, vol. abs/2008.08750, Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.08750

[36] Chen, C., Li, O., Tao, C., Barnett, A. J., Su, J., and Rudin, C., "This Looks Like That: Deep Learning for Interpretable Image Recognition", *Advances in Neural Information Processing Systems (NeurIPS 2018))*, Vol. 32, pp. 8930–8941, Jun. 2019, [Online]. Available: http://arxiv.org/abs/1806.10574

[37] Cao, K., Brbic, M., and Leskovec, J., "Concept Learners for Few-Shot Learning", *International Conference on Learning Representations (ICLR)*, 2021.

[38] Koh, P. W., and Liang, P., "Understanding Black-box Predictions via Influence Functions", *International Conference on Machine Learning*, pp. 1885–1894, Mar. 2017, [Online]. Available: http://arxiv.org/abs/1703.04730

[39] Yeh, C. -K., Kim, J. S., Yen, I. E. H., and Ravikumar, P., "Representer Point Selection for Explaining Deep Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 31, Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.09720

[40] Arik, S. O., and Pfister, T., "ProtoAttend: Attention-Based Prototypical Learning", *Journal of Machine Learning Research*, Vol. 21, pp. 1–35, Feb. 2020, [Online]. Available: http://arxiv.org/abs/1902.06292

[41] Vaswani, A., *et al.*, "Attention is all you need", in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[42] Tsai, Y. -H. H., Bai, S., Yamada, M., Morency, L. -P., and Salakhutdinov, R. "Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

[43] Chen, Y., Zeng, Q., Ji, H., and Yang, Y., "Skyformer: Remodel Self-Attention with Gaussian Kernel and Nystrom Method", *Advances in Neural Information Processing Systems*, Vol. 34, 2021.

[44] Song, K., Jung, Y., Kim, D., and Moon, I. -C., "Implicit kernel attention", in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, No. 11, pp. 9713–9721, 2021.

[45] Choromanski, K. M., *et al.*, "Rethinking Attention with Performers", *International Conference on Learning Representations*, 2021.

[46] Schlkopf, B., Smola, A. J., and Bach, F., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018.

[47] Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P., "Deep Kernel Learning", *Artificial intelligence and statistics, AISTATS*, pp. 370–378, Nov. 2016, [Online]. Available: http://arxiv.org/abs/1511.02222

[48] Tossou, P., Dura, B., Laviolette, F., Marchand, M., and Lacoste, A., "Adaptive deep kernel learning", *arXiv preprint arXiv:1905.12131*, 2019.

[49] Salakhutdinov, R., and Hinton, G. E., "Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes", in *NIPS*, Vol. 7, pp. 1249–1256, 2007.

[50] Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P., "Manifold Gaussian processes for regression", in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345, 2016.

[51] Cortes, C., and Vapnik, V., "Support-vector networks", *Machine Learning*, Vol. 20, No. 3, pp. 273–297, doi: 10.1007/bf00994018, Sep. 1995.

[52] Rasmussen, C. E., and Williams, C. K. I., *Gaussian Processes for Machine Learning*. The MIT Press, 2006. [Online]. Available: http://www.gaussianprocess.org/gpml/

[53] Quinonero-Candela, J., and Rasmussen, C. E., "A unifying view of sparse approximate Gaussian process regression", *The Journal of Machine Learning Research*, Vol. 6, pp. 1939–1959, 2005.

[54] Liu, H., Ong, Y. -S., Shen, X., and Cai, J., "When Gaussian process meets big data: A review of scalable GPs", *IEEE Trans Neural Netw Learn Syst*, Vol. 31, No. 11, pp. 4405–4423, 2020.

[55] Smola, A., and Bartlett, P., "Sparse greedy Gaussian process regression", *Adv Neural Inf Process Syst*, Vol. 13, 2000.

[56] Seeger, M. W., Williams, C. K. I., and Lawrence, N. D., "Fast forward selection to speed up sparse Gaussian process regression", in *International Workshop on Artificial Intelligence and Statistics*, pp. 254–261, 2003.

[57] Keerthi, S. S., and Chu, W., "A matching pursuit approach to sparse Gaussian process regression", *Adv Neural Inf Process Syst*, Vol. 18, 2005.

[58] Snelson, E., and Ghahramani, Z., "Sparse Gaussian processes using pseudo-inputs", *Adv Neural Inf Process Syst*, Vol. 18, 2005.

[59] Williams, C., and Seeger, M., "Using the Nyström Method to Speed Up Kernel Machines", in *Advances in Neural Information Processing Systems*, Vol. 13, pp. 682–688, 2000.

[60] Tipping, M. E., "Sparse Bayesian Learning and the Relevance Vector Machine", *J. Mach. Learn. Res.*, Vol. 1, pp. 211–244, 2001.

[61] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[62] Tipping, M. E., and Faul, A. C., "Fast marginal likelihood maximisation for sparse Bayesian models", *International workshop on artificial intelligence and statistics*, pp. 276–283, 2003.

[63] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G., "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration", *Adv Neural Inf Process Syst*, Vol. 31, 2018.

[64] Al-Shoukairi, M., Schniter, P., and Rao, B. D., "A GAMP-based low complexity sparse Bayesian learning algorithm", *IEEE Transactions on Signal Processing*, Vol. 66, No. 2, pp. 294–308, 2017.

[65] Zhou, W., Zhang, H. -T., and Wang, J., "An efficient sparse Bayesian learning algorithm based on Gaussian-scale mixtures", *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[66] Roth, V., "The generalized LASSO", *IEEE Trans Neural Netw*, Vol. 15, No. 1, pp. 16–28, 2004.

[67] Titsias, M., "Variational learning of inducing variables in sparse Gaussian processes", in *Artificial intelligence and statistics*, pp. 567–574, 2009.

[68] Hensman, J., Matthews, A., and Ghahramani, Z., "Scalable variational Gaussian process classification", in *Artificial Intelligence and Statistics*, pp. 351–360, 2015.

[69] Hensman, J., Fusi, N., and Lawrence, N. D., "Gaussian processes for Big data", in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.

[70] Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P., "Stochastic variational deep kernel learning", *Advances in Neural Information Processing Systems*, Vol. 29, 2016.

[71] Uhrenholt, A. K., Charvet, V., and Jensen, B. S., "Probabilistic selection of inducing points in sparse Gaussian processes", in *Uncertainty in Artificial Intelligence*, pp. 1035–1044, 2021.