Machine Learning (2020) 109:643–664
https://doi.org/10.1007/s10994-019-05838-7

Few-shot learning with adaptively initialized task optimizer: a practical meta-learning approach

Han-Jia Ye[1] · **Xiang-Rong Sheng**[1] · **De-Chuan Zhan**[1]

Received: 4 May 2019 / Revised: 12 July 2019 / Accepted: 6 September 2019 / Published online: 10 October 2019

Abstract
Considering the data collection and labeling cost in real-world applications, training a model with limited examples is an essential problem in machine learning, visual recognition, etc. Directly training a model on such few-shot learning (FSL) tasks falls into the over-fitting dilemma, which would turn to an effective task-level inductive bias as a key supervision. By treating the few-shot task as an entirety, extracting task-level pattern, and learning a *task-agnostic* model initialization, the model-agnostic meta-learning (MAML) framework enables the applications of various models on the FSL tasks. Given a training set with a few examples, MAML optimizes a model via fixed gradient descent steps from an initial point chosen beforehand. Although this general framework possesses empirically satisfactory results, its initialization neglects the task-specific characteristics and aggravates the computational burden as well. In this manuscript, we propose our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach for few-shot learning, which incorporates task context into the determination of the model initialization. This task-specific initialization facilitates the model optimization process so that it obtains high-quality model solutions efficiently. To this end, we decouple the model and apply a set transformation over the training set to determine the initial top-layer classifier. Re-parameterization of the first-order gradient descent approximation promotes the gradient back-propagation. Experiments on synthetic and benchmark data sets validate that our AVIATOR approach achieves the state-of-the-art performance, and visualization results demonstrate the *task-adaptive* features of our proposed AVIATOR method.

Keywords Few-shot learning · Meta-learning · Supervised-learning · Multi-task learning · Task-specific

Editors: Kee-Eung-Kim and Jun Zhu.

✉ De-Chuan Zhan
zhandc@lamda.nju.edu.cn

Han-Jia Ye
yehj@lamda.nju.edu.cn

[1] Nanjing University, Nanjing, China

1 Introduction

Although modern machine learning approaches achieve remarkable improvements in various real-world fields such as visual recognition (Krizhevsky et al. 2017), one of the key elements towards constructing such helpful models is a large training set (Russakovsky et al. 2015; Su et al. 2018). Taking the instance collection and labeling cost into consideration, learning "rich" knowledge from "small" data is necessary and important. For example, images of rare species are hard to collect, thus the model should be able to do visual recognition based on single or a few reference examples (Wang et al. 2018b); it is cumbersome to require a user recording multiple facial expressions into a system in advance (Tan et al. 2006); for a robot, imitating from one single demonstration of human must become a fantastic characteristic (Finn et al. 2017b). In addition to such instance collection difficulty for the "long-tailed" objects, to make it worse, there also exists labeling cost when dealing with bio-informatics (Huang et al. 2014) or thousands of new-coming items (Karlinsky et al. 2017). The task with only limited training examples from each class results in the few-shot learning (FSL) problem.

The training data in a machine learning task reveals the pattern of the data distribution, and the principles of statistical learning require enough training examples from the same distribution to make the model learnable and generalizable. Hence it seems that the FSL violates previous analyses on the machine learning field, and it is almost incredible to train a model through only a few training examples.

But how human can recognize novel objects with a few or even one single image? One possible reason lies in his/her rich experience, a.k.a. the the *inductive bias* (Baxter 2000), with seen objects (Lake et al. 2015). For example, a programmer adapts himself/herself to a new task rapidly based on his/her rich experience from related tasks.

Similar learning schema works in the few-shot classification environment. Specifically, a type of model inductive bias is extracted from the SEEN classes, and it is then applied to few-shot tasks composed by UNSEEN classes. To this end, meta-learning mimics the few-shot evaluation scenario during training, and figures out a common task configuration over the sampled few-shot tasks from the SEEN classes (Vilalta and Drissi 2002; Maurer et al. 2016; Thrun and Pratt 2012). Feature embedding (Koch et al. 2015; Vinyals et al. 2016; Snell et al. 2017; Triantafillou et al. 2017), embedding adaptation (Ye et al. 2018), and model optimization strategies (Ravi and Larochelle 2017) all can be *learned* in a meta-learning way.

Model-agnostic meta-learning (MAML) (Finn et al. 2017a; Nichol et al. 2018) is an important thread of meta-learning. Due to the observations that the quality of a model highly depends on the *initial point* of its optimization, to avoid over-fitting, MAML learns a *common* model initialization depicting the features across a wide range of few-shot tasks, then restricts any few-shot models to be optimized by a *fixed* number of gradient descent steps from this specified initialization. Although experiments verify the feasibility of MAML for FSL, there still exist several problems. First, a single optimization initialization point is too difficult to satisfy diverse tasks. Taking a binary task discerning "sunflower" and "dog" as an example. A good initialization tends to be composed by classifiers with discriminative coefficients towards plant and animal accordingly. While for another task containing "cat" and "rose", the previous plant-animal initialization cannot be applied to this "reverse" animal-plant case without careful updates. Furthermore, vanilla MAML requires the second order derivatives of all parameters during training, and recent literature shows it is hard to apply such a method without special hyper-parameter tuning tricks (Antoniou et al. 2018; Chen et al. 2019).

Due to the heterogeneity of classes from one task to another, in our manuscript, we propose our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach for few-shot learning,

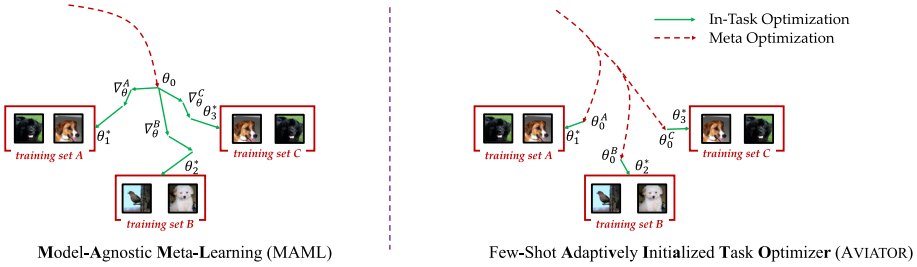


Fig. 1 The main flow of model-agnostic meta-learning (MAML) and our proposed AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach for few-shot learning. MAML figures out an initial point θ_0 for all tasks, and the final model of each task (say θ_A^*) is fine-tuned from θ_0 through a fixed number of gradient descent (say based on ∇_θ^A). While AVIATOR considers task characteristic, and sets specific initial points for each task

which transforms MAML into a practical FSL approach. AVIATOR incorporates the task context, i.e., the class information inside a training set, into the construction of the model initial point. Hence different tasks have *adaptive* model initialization, so as to obtain *task-specific* classifier effectively and efficiently. The notion of AVIATOR is illustrated in Fig. 1.

To this end, we decouple a model into the embedding part and the top-layer classifier. By treating the current few-shot training set as the input, a mapping is learned to encode the task characteristic and output an adaptive classifier initialization. To enable the gradient back-propagation, we turn to the first-order approximation of the gradient descent, and apply a simple re-parameterization strategy to implement and simplify the model training. During the synthetic experiments, we find the model initialization output by MAML is too conservative to cover all kinds of tasks, while the initial model used by AVIATOR handles model diversities effectively. We also validate the superiority of our task-adaptive optimizer approach w.r.t. the MAML variants on two benchmark data sets, i.e., *Mini*ImageNet and CUB. Concrete ablation studies and discussions illustrate the helpful task-specific feature of AVIATOR.

The main contributions of our manuscript can be summarized as follows:

- A practical task-adaptive optimizer for FSL which considers the task context.
- An effective meta-level optimization strategy with a low computational burden.
- Promising experimental results on benchmark data sets.

We start with the background and notations of the few-shot learning problem, and then we discuss the related approaches. Before diving into our AVIATOR approach in detail, we give a brief introduction to the MAML framework. The training and the re-parameterization strategy of AVIATOR are stated in Sect. 4. Finally are experiments and conclusion.

2 Related work

Although it is the fact that human is able to learn a novel concept through one single example (Li et al. 2006; Lake et al. 2011, 2015), it is still difficult to train a model under the data budget. For example, most visual recognition methods require thousands of training data to fit the huge deep learning models (Russakovsky et al. 2015; Krizhevsky et al. 2017). Considering both the data collection (Li and Zhou 2015) and labeling cost (Huang et al. 2014), it is necessary to enable a machine learning algorithm to have such a valuable ability—building classifiers with limited training examples, i.e., the one-shot and the few-shot Learning. One

naive idea is to fine-tune a pre-trained model or add strong regularizers. Neither these solutions cannot achieve satisfying results in practice due to the data scarcity and model complexity.

Tracing back to Baxter (2000) and Vilalta and Drissi (2002)), the importance of task-level inductive bias has been proposed and analyzed theoretically. Different from conventional models predicting over the instance level, meta-learning, a.k.a. learning-to-learn, extracts inductive bias across training *tasks*. A meta-model characterizes the task commonality and generalizes its prediction to those UNSEEN tasks from a related environment (Maurer 2009; Maurer et al. 2016; Denevi et al. 2018). Meta-learning approaches have been successfully applied in various fields, like long-tail class-imbalance classification (Wang et al. 2017b; Ren et al. 2018), domain adaptation (Motiian et al. 2017; Zhang et al. 2018), intimation learning (Yu et al. 2018), density estimation (Reed et al. 2017), unsupervised learning (Garg 2018; Hsu et al. 2018), data compression (Wang et al. 2018a), recommendation system (Vartak et al. 2017), and hyper-parameter tuning (Franceschi et al. 2017; Probst et al. 2019).

Benefited from meta-learning's ability to generalize the prediction ability across tasks, in Few-Shot Learning (FSL), the inductive bias is first learned over few-shot tasks composed by SEEN classes, and then is evaluated for those few-shot tasks with UNSEEN classes. For example, few-shot classification can be implemented in a non-parametric way with soft nearest neighbor (Vinyals et al. 2016) or nearest center rule (Snell et al. 2017), so the representation function acts as the task-level inductive bias. The learned embedding pulls similar instances together and pushes dissimilar ones far away, such that a test instance can be classified even with a few labeled training examples (Koch et al. 2015). Considering the hypothesis complexity, the model training configurations also serve as a type of inductive bias. (Andrychowicz et al. 2016; Ravi and Larochelle 2017) meta-determines the optimization strategy for each task, including the learning rate and update directions as well. Other kinds of inductive biases are also explored. Hariharan and Girshick (2017) and Wang et al. (2018b)) learn a generation prior to augment examples given a single image; Dai et al. (2017) extracts logical derivations from related tasks; Wang et al. (2017a)) and Shyam et al. (2017) utilize the prior to attend images. An empirical study of few-shot learning approaches can be found in Chen et al. (2019).

Model-agnostic meta-learning (MAML) (Finn et al. 2017a) proposes another kind of inductive bias, i.e., the model initialization. After meta-learned a common model initialization among tasks, the classifier of a new few-shot task can be fine-tuned with several steps of gradient descent from that initial point. The universality of this MAML-type updates has be proved in Finn and Levine (2018). MAML have been applied in various scenarios, such as uncertainty estimation (Finn et al. 2018), robotics control (Yu et al. 2018; Clavera et al. 2018), neural translation (Gu et al. 2018), and language generation (Huang et al. 2018). In spite of the success, there still exist problems with such a framework. Nichol et al. (2018) handles the high computational burden of MAML with first-order approximation; Deleu and Bengio (2018) points out possible negative adaptation with too many updates; Antoniou et al. (2018) provides a bunch of tricks to tune the MAML framework; and Lee and Choi (2018) decomposes the MAML backbone to introduce task-specific metric. In our paper, we aim to incorporate the task context into the determination of the model initialization. By decoupling the model into embedding and top-layer classifier, we propose a re-parameterization strategy which enables the transition of the task information without touching the backbone. Our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach achieves good performance with low computational burden in practice.

Different from Ye et al. (2018) which transforms embedding with a transformer-based set function over the training set embedding output, AVIATOR adapts both top-layer classifier and embedding via gradient descent. In addition, Triantafillou et al. (2019) constructs an embed-

ding learning objective for class prototypes, and updates task embedding in the MAML style. AVIATOR introduces the task-adaptive property through the dynamic initialization, which not only incorporates task context but also accelerates the update process. More comparison can be found in the experiments part.

3 Notations and background

In this section, we describe the Few-Shot Learning (FSL) setting formally at first, and then present the main flow of the meta-learning methods dealing with FSL tasks. At last, we discuss the relationship between the current meta-learning and the previous analyzed learning-to-learn framework.

3.1 The few-shot learning problem

Following the literature, we define a N-way K-shot task as a classification problem with N classes in total and K examples in each class. In the few-shot scenario, the value K is very small, e.g., $K = 1$ or $K = 5$. The target of the Few-Shot Learning (FSL) task is to obtain an effective classifier based on these NK training examples, which is able to discern an unseen test instance among these N classes. The main difficulty of such FSL is the contradiction between the complex model and scarce data, which makes a model prone to over-fit.

We denote the training set (a.k.a. support set) of the problem as $\mathcal{D}_{\mathbf{train}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{NK}$. Each $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\mathbf{train}}$ are the instance and label of the ith example respectively, where $\mathbf{x}_i \in \mathbb{R}^D$, and the label $\mathbf{y}_i \in \{0, 1\}^N$ comes from a class set composed by N classes. The index of value 1 in the label vector $\mathbf{y}_i$ indicates the class of the instance $\mathbf{x}_i$.

Rather than learning a few-shot model over UNSEEN classes from scratch, the inductive bias extracted from related SEEN class tasks should be reused in advance. We use a superscript $\mathcal{S}$ to emphasize a set or an instance sampled from the SEEN class set if necessary. Different from UNSEEN few-shot tasks, classes in the SEEN set often have enough examples. Taking a rare-bird classification problem as an example. The target is to train a classifier for UNSEEN rare birds, where only a limited number of images of those birds can be collected in practice. To obtain inductive bias towards birds classification, we can take advantage of the public Caltech-UCSD Birds (CUB) (Wah et al. 2011) data sets which have 200 common birds and enough examples in each of the 200 classes. In other words, the public birds data set in this case serves as the SEEN class set, while the tasks with rare birds containing limited examples correspond to the UNSEEN class set. The target of the FSL is to utilize the SEEN set to assist the few-shot classification over the UNSEEN set.

3.2 Meta-learning for few-shot learning

Meta-learning is a popular approach to extract inductive bias from SEEN class set, which has been widely used in the few-shot scenarios (Vinyals et al. 2016; Finn et al. 2017a; Snell et al. 2017; Ye et al. 2018). A comparison between standard classifier training and the meta-learning paradigm can be found in Fig. 2.

The main idea of the meta-learning is to sample tasks from the SEEN sets to *mimic* the evaluation scenario, i.e., the N-way K-shot task. In particular, N-way K-shot tasks $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$ are sampled from the SEEN sets in a hierarchical way: first, N classes are chosen randomly from all SEEN classes, and then K examples in each of the N classes are selected in random.

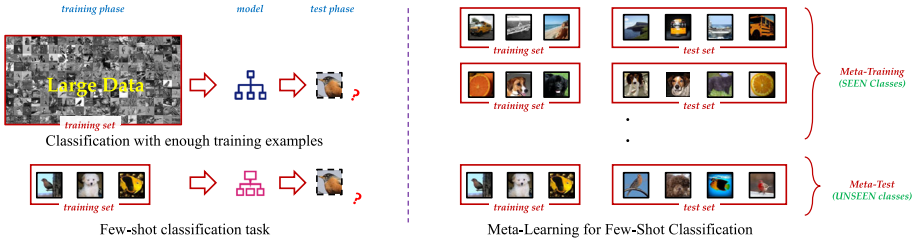


Fig. 2 Comparison between the few-shot classification and the standard supervised learning paradigm. Left: Different from standard machine learning paradigm training a model based on a large data set, few-shot classification considers the scenario there is only a limited number of instances in the training set. Right: The general flow of the meta-learning procedure for few-shot classification. By sampling few-shot tasks from the meat-training set (SEEN classes), the learned task inductive bias can be applied to the few-shot task from the meta-test set (UNSEEN classes)

Following this strategy, the SEEN class set is named as "meta-train", while the UNSEEN set is denoted as "meta-test".

A mapping f is constructed based on $\mathcal{D}_{\mathbf{train}}$, which outputs a classifier for the given N classes. In detail, for a test instance $\mathbf{x}_j$, its label can be predicted as

$$\hat{y}_j = f(\mathcal{D}_{\mathbf{train}})(\mathbf{x}_j). \tag{1}$$

The mapping f is learned with few-shot tasks from the meta-training set. To measure the performance of such a classifier mapping f over the N classes when facing a few-shot training set $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$, another test set (a.k.a. query set) $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ with these N classes is sampled. A good classifier mapping f will achieve low loss value after predicting the labels of all instances from the test set. Therefore, the objective of meta-learning can be summarized as

$$\min_{f} \sum_{(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}) \sim \mathcal{S}} \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(f(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})(\mathbf{x}^{\mathcal{S}}_j), \mathbf{y}^{\mathcal{S}}_j). \tag{2}$$

In Eq. 2, $(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}) \sim \mathcal{S}$ denote the enumeration of all sampled tasks from the SEEN class set. The loss function $\ell(\cdot, \cdot)$ measures the discrepancy between the prediction and true label for each instance in $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$. By optimizing the objective, a general mapping f is constructed, which maps a training task, even with a few instances in each class, to an effective classifier for the particular task (meta-training phase). Hence such mapping can also be applied to few-shot tasks from the UNSEEN class set (meta-test phase). The whole process is summarized in Alg. 1.

Remark 1 Although there are only limited training examples in each class, the classifier mapping f is *shared* among a lot of few-shot tasks. Furthermore, since such f is learned from plenty of tasks, enough number of tasks alleviate the burden of within task sample requirement from the meta-perspective to some extent.

3.3 Learning embedding for few-shot learning

A direct implementation of the classifier mapping f is the embedding function, $f = \phi : \mathbb{R}^D \to \mathbb{R}^d$, which extract features of the input examples and transformed them into a latent space with d dimensions. If the embedding ϕ makes similar objects close to each other while

Algorithm 1 The flow of the meta-learning for Few-Shot Classification.

Require: Seen class set $\mathcal{S}$
1: **for all** iteration = 1,… **do**
2: Sample N-way K-shot $(\mathcal{D}^{\mathcal{S}}_{\textbf{train}}, \mathcal{D}^{\mathcal{S}}_{\textbf{test}})$ from $\mathcal{S}$
3: **for all** $(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\textbf{test}}$ **do**
4: Predict $\hat{\mathbf{y}}^{\mathcal{S}}_j = f(\mathcal{D}^{\mathcal{S}}_{\textbf{train}})(\mathbf{x}^{\mathcal{S}}_j)$ based on Eq. 1
5: Compute loss $\ell(\hat{\mathbf{y}}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j)$ as Eq. 2
6: **end for**
7: Compute gradient $\nabla_f \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{X}^{\mathcal{S}}_{\textbf{test}}} \ell(\hat{\mathbf{y}}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j)$
8: Update the mapping f with selected optimizer.
9: **end for**
10: **return** Few-shot classifier mapping f.

dissimilar ones far away, then it is qualified for the few-shot classification task (Koch et al. 2015). For a test instance $\mathbf{x}_j$, the embedding function ϕ makes a prediction based on a soft nearest neighbor rule:

$$\hat{y}_j = f(\mathcal{D}_{\textbf{train}})(\mathbf{x}_j) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\textbf{train}}} \textbf{sim}(\mathbf{x}_j, \mathbf{x}_i)\mathbf{y}_i. \tag{3}$$

The $\textbf{sim}(\mathbf{x}_j, \mathbf{x}_i)$ measures the similarity between the test instance $\mathbf{x}_j$ and each training instances $\mathbf{x}_i$. Such measurement can be completed by a *normalized* version of cosine similarity (Vinyals et al. 2016) or negative euclidean distance (Snell et al. 2017). When there are more than one instance in each class, i.e., $K > 1$, instances in the same class can also be averaged to assist make final decision (Snell et al. 2017). By learning a good embedding, the important features for few-shot classification is stressed, which will also be used for few-shot tasks from the UNSEEN class set.

3.4 Discussion: the learning-to-learn framework

Instead of utilizing a test set to evaluate the quality of the mapping f in meta-learning as in Eq. 2, in the learning-to-learn (L2L) framework, it is the training set error that is used to provide the learning signal.[1]

As discussed in Maurer et al. (2016), the mapping $f = W \circ \phi$ is a composition of a linear classifier $W \in \mathbb{R}^{d \times N}$ and the embedding ϕ. The L2L objective is formulated as a bi-level optimization problem:

$$\begin{aligned} &\min_{W,\phi} \sum_{\mathcal{D}^{\mathcal{S}}_{\textbf{train}} \sim \mathcal{S}} \sum_{(\mathbf{x}^{\mathcal{S}}_i, \mathbf{y}^{\mathcal{S}}_i) \in \mathcal{D}^{\mathcal{S}}_{\textbf{train}}} \ell(W^\top \phi(\mathbf{x}_i), \mathbf{y}_i). \\ &\Rightarrow \min_{\phi} \sum_{\mathcal{D}^{\mathcal{S}}_{\textbf{train}} \sim \mathcal{S}} \min_{W} \sum_{(\mathbf{x}^{\mathcal{S}}_i, \mathbf{y}^{\mathcal{S}}_i) \in \mathcal{D}^{\mathcal{S}}_{\textbf{train}}} \ell(W^\top \phi(\mathbf{x}_i), \mathbf{y}_i). \end{aligned} \tag{4}$$

In Eq. 4, all tasks sampled from the meta-training set share the common embedding ϕ. On top of the embedding and the training set $\mathcal{D}^{\mathcal{S}}_{\textbf{train}}$, the classifier W for each task is generated by directly optimizing the linear classifier until convergence. Assuming all tasks are sampled from the *same task distribution*, the generalization ability of the learning-to-learn objective

[1] In this manuscript, we differentiate the meta-learning and learning-to-learn by their evaluation approach for the meta-learned mapping f. Hence meta-learning optimizes Eq. 2, while Eq. 4 is for learning-to-learn.

is guaranteed in Maurer et al. (2016). A similar objective is also used in the few-shot learning field (Nichol et al. 2018).

Remark 2 Comparing with the learning-to-learn framework (c.f. Eq. 4), the meta-learning framework (c.f. Eq. 2) has at least three main differences.

- Instead of focusing only on the few-shot training set $\mathcal{D}^S_{\mathbf{train}}$, in meta-learning, another test set $\mathcal{D}^S_{\mathbf{test}}$ is also sampled to evaluate the quality of the classifier mapping, which produces a more accurate measurement of the quality of f;
- All tasks sampling from the same task-distribution is an important assumption for L2L. Meta-learning weakens the assumption and directly applies the model learned on the tasks with seen classes during meta-training to those tasks with UNSEEN classes in the meta-test phase.
- In Eq. 4, only the embedding is updated across different tasks, while in the meta-learning framework, the parameter update of f would be more flexible.

4 Few-shot learning with adaptively initialized optimizer

In this section, we present the main idea of our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach for few-shot learning in detail. Before that, we provide a brief introduction to the model agnostic meta-learning approach.

4.1 Model-agnostic meta-learning (MAML)

Consider how we obtain a classifier $f : \mathbb{R}^D \to \{0, 1\}^N$ from the training set.[2] Given a training set $\mathcal{D}_{\mathbf{train}}$ of a task, the model can be obtained by optimizing the loss on the training set:

$$\min_{f_\theta} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\mathbf{train}}} \ell(f_\theta(\mathbf{x}_i), \mathbf{y}_i). \tag{5}$$

θ is the set of all learnable parameters in f. A simple way to optimize such objective is the gradient descent. Define a gradient operator $\Xi_\theta(\cdot)$:

$$\Xi_\theta(\mathcal{D}_{\mathbf{train}}, \theta_t) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\mathbf{train}}} \nabla_\theta \ell(f_{\theta_t}(\mathbf{x}_i), \mathbf{y}_i),$$

which computes the gradient w.r.t. θ given the training set and the current tth step's solution θ_t. We omit the $\mathcal{D}_{\mathbf{train}}$ in Ξ_θ for notation simplicity. Therefore, the full gradient descent update w.r.t. to the variable θ to solve the optimization problem in Eq. 5 is:

$$\theta_{t+1} = \theta_t - \eta \Xi_\theta(\theta_t) = \theta_t - \eta \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{\mathbf{train}}} \nabla_\theta \ell(f_{\theta_t}(\mathbf{x}_i), \mathbf{y}_i).$$

$\eta > 0$ is the step-size. Usually, starting from an initial θ_0, more than one gradient step will be used to get the final solution. If the value of the objective can be obtained during the optimization, complex step-size strategies can be utilized to accelerate the optimization.

The general idea of meta-learning aims to obtain a shared classifier mapping f among tasks. Consider the dilemma with the limited training instances in $\mathcal{D}_{\mathbf{train}}$, directly optimizing

[2] *With a bit abuse of the notation*, we also define the mapping f as the classifier which maps the training instance to a label.

the problem till convergence will make the model over-fit and hard to generalize. In addition, for a learner with non-convex objective, the quality of its solution depends on its initialization point a lot. Model-agnostic meta-learning (MAML) (Finn et al. 2017a) handles such problems with two strategies. First, an initialization of the model θ_0 is shared among tasks. Besides, to avoid over-fitting, only a fixed number of gradient descent steps (say T steps) are carried out. In other words, given a few-shot task $\mathcal{D}_{\mathbf{train}}$, its classifier is obtained by T successive gradient updates based on the common initialization θ_0. Therefore, the objective of MAML can be formulated as:

$$\min_{\theta_0} \sum_{(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}) \sim \mathcal{S}} \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(f_{\theta_0 - \eta \Xi_\theta(\theta_0)}(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})(\mathbf{x}^{\mathcal{S}}_j), \mathbf{y}^{\mathcal{S}}_j). \tag{6}$$

The MAML objective in Eq. 6 optimizes the initial parameter θ_0 for the model f. In each task, the task-specific classifier is obtained by one or more steps of gradient descent over θ_0, using the training set $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$. This objective also requires the updated model in each task has low loss value over instances from the corresponding test set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$. It is notable that in Eq. 6 and following objectives, we only do one inner-task gradient descent with step-size η. The objective with more steps can be easily extended. The output of MAML is the model initialization θ_0, so for a few-shot task from the UNSEEN class set, its task-specific model can also be optimized by gradient descent from this initial parameter θ_0.

Similar to the general meta-learning objective in Eq. 2, the loss computed on the test set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ supervises the search of the initialization point θ_0 among tasks. To optimize such initial parametric model f_{θ_0}, a meta-level stochastic gradient descent is leveraged. In detail, for a pair of $(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}})$, the update of θ_0 with a meta-level step-size $\gamma > 0$ is:

$$\begin{aligned} \theta_0 &= \theta_0 - \gamma \nabla_{\theta_0} \\ &= \theta_0 - \gamma \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \nabla \ell(f_{\theta_0 - \eta \Xi_\theta(\theta_0)}(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})(\mathbf{x}^{\mathcal{S}}_j), \mathbf{y}^{\mathcal{S}}_j). \end{aligned}$$

Owing to the fact that the gradient w.r.t. θ_0 is used for the inner update over the training set $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$, during the meta-update, the gradient w.r.t. the updated θ over the test set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ is related to the second order derivatives of θ_0, which will have high-computational burden. The MAML training follows the procedure in Algorithm 1.

Remark 3 One simple way to reduce the computational burden of MAML is to use the *first-order gradient* to do the inner task update. In particular, define $\mathbf{sg}(\cdot)$ as an operator which stops the gradient for the input variable, then we can use $\mathbf{sg}(\Xi_\theta(\theta_0))$ to replace the inner gradient. Therefore, although the gradients in the inner update are related to the intermediate variable θ_t and the loss on the training set, the updated parameter θ of the model can be formulated as the θ_0 minus some *constants* gradient values. So the final objective of the the first-order MAML is

$$\min_{\theta_0} \sum_{(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}) \sim \mathcal{S}} \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(f_{\theta_0 - \eta \mathbf{sg}(\Xi_\theta(\theta_0))}(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})(\mathbf{x}^{\mathcal{S}}_j), \mathbf{y}^{\mathcal{S}}_j). \tag{7}$$

This first-order MAML achieves a little lower performance degrade over the benchmark evaluation Finn et al. (2017a), while it accelerates the meta-training process a lot. Another variant of the first-order MAML updates, Reptile (Nichol et al. 2018), returns to the learning-to-learn objective and optimizes the loss on the training set:

$$\min_{\theta_0} \sum_{\mathcal{D}^S_{\mathbf{train}}} \sum_{(\mathbf{x}^S_j, \mathbf{y}^S_j) \in \mathcal{D}^S_{\mathbf{train}}} \ell(f_{\theta_0 - \eta \mathbf{sg}(\Xi_\theta(\theta_0))}(\mathcal{D}^S_{\mathbf{train}})(\mathbf{x}^S_j), \mathbf{y}^S_j). \tag{8}$$

Reptile empirically gets better few-shot classification results than the first-order MAML, but it still remains a lot of hyper-parameters to tune during the meta-training.

Remark 4 We can think of MAML as a surrogate implementation of the bi-level learning-to-learn objective in Eq. 4. First, it is the model initialization that shared among tasks; then, instead of optimizing the model over the training set completely, it simplifies such optimization process with *a fixed number of* gradient steps. So in MAML, a good enough task-specific classifier is assumed to be obtained thanks to a good initialization. There are two main advantages of such decisions. On the one hand, it gets rid of the inner-task minimization problem and simplifies the holistic optimization. Furthermore, the conservative fixed number of gradient descent fits the few-shot learning problem with the limit of the training data.

Remark 5 There are several hyper-parameters in the training process of MAML. In addition to the number of steps (T) and learning rate (η) in the inner-task optimization, the meta-level learning rate and schedule should also be determined for model training. With so many hyper-parameters, however, it is hard to tune MAML in practice (Chen et al. 2019). Antoniou et al. (2018) provide a lot of tricks and objective transformations to make MAML work better.

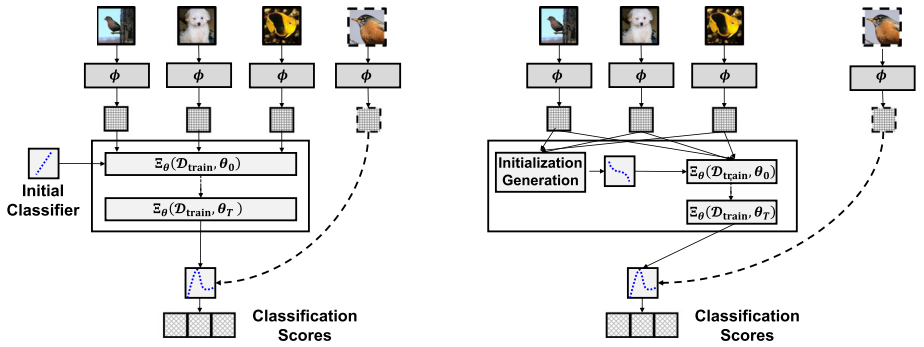
Remark 6 MAML optimizes an initial model over the same form of tasks, which can also be used for the sample-efficient reinforcement learning (RL). Different from few-shot classification scenarios where tasks during the model evaluation phase are sampled from the UNSEEN classes, the tasks in RL fit the assumption of the task distribution. Finn et al. (2017a) has verified the usage of MAML in several RL problems. In this paper, we focus on applying MAML for few-shot classification.

4.2 Using the adaptively initialized optimizer

We decouple the model $f_\theta : \mathbb{R}^D \to \mathbb{R}^N$ into two parts. To make a prediction on an instance, the feature extractor $\phi : \mathbb{R}^D \to \mathbb{R}^d$ maps the input into an embedding space, and a top-layer classifier $W \in \mathbb{R}^{d \times N}$ carries out the final prediction.[3] Each column of W corresponds to each on of the N classes. In other words, $f(\mathbf{x}) = W^\top \phi(\mathbf{x})$. The feature embedding extracts general representations among classes (Achille and Soatto 2018), while the top-layer classifier encodes the relationship between an embedding feature and a particular class.

Consider a binary example classifying dog and cat. The embedding ϕ reveals the features among all possible classes, while the columns of W depict the relatedness between features and classes in the task. For example, a classifier for "dog" will have larger weights for those discriminative features between dog and others. Focusing on the top-layer classifier, it is noteworthy that there exists a *one-to-one correspondence* between the (columns of the) classifier with those classes. In this case, MAML would construct an initial W_0 with coefficients biased towards dog and cat respectively, so that an effective final classifier could be obtained easily with limited steps of gradient descent on a few examples. However, when facing another task classifying cat and dog, a class-level permutation of the previous task, the previous learned initial classifier W_0 will have a negative effect due to the incorrespondence between classes and their stressed features. In other words, a single model initialization cannot handle heterogeneous tasks effectively.

[3] We omit the bias of the classifier for discussion simplicity.



(a) MAML **(b) AVIATOR**

Fig. 3 The workflow of the model-agnostic meta-learning (MAML) and our proposed AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach. Instead of updating the task classifier based on a non-informative initialization, AVIATOR takes task characteristic into account for an *adaptively* initialized model. $\Xi(\cdot)$ is the gradient operator

Inspired by this observation, we aim to make the learned model initialization *adaptable* to all kinds of tasks by incorporating the task context. By encoding the task property through a limited number of instances, the model initialization could be better estimated after such "first impression". After that, more details could be fulfilled by inner-task gradient updates. In the "dog and cat" example, the few-shot training set will facilitate to recognize the relationship between the class and a particular classifier, then an adapted initial classifier could be easily generated via fully reusing the previously learned initial model. Therefore, we propose our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach for few-shot learning (Fig. 3).

Since the embedding function ϕ extracts general discriminative features among classes, we attribute the main problem of MAML to the non-informative top-layer classifier initialization. In our AVIATOR approach, we use the training set embedding of the current task as the context, and use another mapping g to generate the *initialization* of W. It is notable that we do not treat the output of g as the new classifier, and still use an adaptively updated W for final classification. In the inner-task optimization, it is the gradient w.r.t. W (and ϕ) used to update the model, and g is only used to initialize the W.

The main difficulty lies in the fact that g only affects the initial value of W and will be untouched in the following gradient updates and classification process. In other words, after obtaining the updated classifier over the few-shot training set $\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}$, the loss over the corresponding test set $\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}$ has nothing to do with g. Therefore, this kind of adaptive strategy blocks the gradient flow to g, which makes it hard to update g during the meta-training stage. To solve this problem, we propose to re-parameterize the gradient update flow with the first-order approximation.

4.2.1 Enable the gradient flow

The inner-task updates of the model f_θ depends on the initialization and the gradient of the model parameters. Denote $\theta = \{W, \phi\}$, we can expand the update process as

$$W = W_0 - \eta \Xi_W(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, W_0), \quad \phi = \phi_0 - \eta \Xi_\phi(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \phi_0).$$

Same as the previous discussion, we list one gradient step here, and more steps can be easily extended. The next-step gradient depends on the training set loss with the previous-step

model parameter. When using the first-order updates, we stop computing the gradient for $\Xi_W(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, W_0)$ (resp. $\Xi_\phi(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}, \phi_0)$) w.r.t. W (resp. ϕ), and the final updated W (resp. ϕ) is a function of W_0 (resp. ϕ_0). Denote $\phi(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}) = \{\phi(\mathbf{x}); \forall \mathbf{x} \in \mathcal{D}^{\mathcal{S}}_{\mathbf{train}}\}$ the embedding of the training set, then we use a function g on $\phi_0(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})$ to generate the initial top-layer classify W_0. To enable the gradient descent for the initialization generation function g, we use the following three-step re-parameterization strategy: First, the task-specific initial classifier W_0 is generated based on $W_0 = g(\phi_0(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}}))$. Then the *value* of such initial point is assigned to an intermediate initialization $\hat{W}_0 = \mathbf{sg}(W_0)$, which is used to compute the following gradient but is not related to g. After computing the gradient w.r.t. this $\hat{W}_0$, we re-introduce g to generate the final classifier $W = g(\phi_0(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})) - \eta \mathbf{sg}(\Xi_W(\hat{W}_0))$.

In summary, we have the following objective for AVIATOR:

$$\begin{aligned} \min_{\phi_0, g} \quad & \sum_{\mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\mathbf{test}}} \ell(f_{\theta_0 - \eta \mathbf{sg}(\Xi_f(\hat{\theta}_0))}(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})(\mathbf{x}^{\mathcal{S}}_j), \mathbf{y}^{\mathcal{S}}_j) \\ \text{s.t.} \quad & \theta_0 = \{W_0, \phi_0\};\ \hat{\theta}_0 = \{\mathbf{sg}(W_0), \mathbf{sg}(\phi_0)\},\ W_0 = g(\phi_0(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})). \end{aligned} \tag{9}$$

4.2.2 Implementation of the classifier initialization mapping

The function g maps the embedding of the training set to the corresponding classifier, which generates an adaptive initialization based on the "first impression" of data. There are two main rules to design g. First, there exists a *strong relationship* between the instances in a particular class and its corresponding classifier. In addition, the influence of other classes should act as a set, i.e., their influences will be invariant for their permutations. Hence we implement g based on the idea of deep sets (Zaheer et al. 2017), where the effect of a set can be depicted as the transformed sum of the set instance embedding. Given the training set embedding $\phi(\mathcal{D}^{\mathcal{S}}_{\mathbf{train}})$, we use the mean of a particular class instances and the mean of the complementary classes instances as the context for a classifier. For the nth class, denote

$$\mathbf{p}_n = \frac{1}{K} \sum_{\mathbf{y}_i = n} \phi(\mathbf{x}_i),\ \mathbf{p}^{\complement}_n = \frac{1}{(N-1)K} \sum_{\mathbf{y}_i \neq n} \phi(\mathbf{x}_i). \tag{10}$$

Here the $\mathbf{y}_i = n$ and $\mathbf{y}_i \neq n$ mean the instances in or not in the nth class respectively. After concatenating $\mathbf{p}_n$ and $\mathbf{p}^{\complement}_n$, we use a two-layer fully connected network to generate the d-dimensional classifier for the nth class. The bias can be generated in a similar way. We use this simple implementation to show the effectiveness of the AVIATOR idea. More complicated implementations can be found in the ablation study part in the experiments. The whole flow of our AVIATOR approach can be found in Algorithm 2.

Remark 7 With the decoupled top-layer classifier W and the embedding ϕ, a straightforward way to construct such adaptive model is to make W become the output of g, i.e., $W = g(\phi(\mathcal{D}_{\mathbf{train}}))$. So we have $\hat{y_j} = g(\phi(\mathcal{D}_{\mathbf{train}}))^\top \phi(\mathbf{x}_j)$. It incoporates the few-shot training set into the determiniation of the whole initialization, and increase the complexity by the mapping composition. The main difference between this implementation and ours is that it includes g as a part of the predictor and also updates g during the inner-task optimization. Since g could be complicated, the inner updates over g will not only increase the computational burden but also make the model prone to over-fit. While in AVIATOR, g only help determine the initialization of the linear top-layer classifier W, and will not participate in the inner-task update. Empirically, we find this naive task-adaptive solution is hard to train and easy to over-fit.

Algorithm 2 The flow of the AVIATOR approach for Few-Shot Classification.

Require: Seen class set $\mathcal{S}$, inner-task step-size η, number of steps T, meta-update step-size γ.
for all iteration = 1,… **do**
  Sample N-way K-shot $(\mathcal{D}^{\mathcal{S}}_{\textbf{train}}, \mathcal{D}^{\mathcal{S}}_{\textbf{test}})$ from $\mathcal{S}$
  Construct task-specific classifier initialization $W_0 = g(\phi(\mathcal{D}^{\mathcal{S}}_{\textbf{train}}))$
  Assign the value of W_0 to an intermediate variable $\hat{W}_0 = \textbf{sg}(W_0)$
  for all update iteration = 0,…t…T **do**
    Inner-task update: $W_{t+1} = W_t - \eta \Xi_W(W_t),\ \phi_{t+1} = \phi_t - \eta \Xi_\phi(\phi_t)$
  end for
  Construct final classifier $W_T = g(\phi(\mathcal{D}^{\mathcal{S}}_{\textbf{train}})) - \eta \textbf{sg}(\Xi_W(W_0)) - \eta \textbf{sg}(\Xi_W(W_1)) - \cdots$
  Set $\theta = \{W_T, \phi_T\}$
  for all $(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{D}^{\mathcal{S}}_{\textbf{test}}$ **do**
    Predict $\hat{\mathbf{y}}^{\mathcal{S}}_j = f_\theta(\mathbf{x}^{\mathcal{S}}_j))$
    Compute loss $\ell(\hat{\mathbf{y}}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j)$ as Eq. 2
  end for
  Compute gradient $\nabla_{\phi,g} \sum_{(\mathbf{x}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j) \in \mathcal{X}^{\mathcal{S}}_{\textbf{test}}} \ell(\hat{\mathbf{y}}^{\mathcal{S}}_j, \mathbf{y}^{\mathcal{S}}_j)$
  Update the parameter $\{\phi, g\}$ with selected optimizer and step-size γ.
end for
return Few-Shot classifier mapping f_θ.

Remark 8 It is notable that due to the fixed initial classifier used in the vanilla MAML, the form of training and evaluation few-shot tasks should be consistent. In other words, when sampling 5-way tasks during the meta-training phase, the initial classifier is learned as a $d \times 5$ matrix, which is *hard to generalize to tasks with other numbers of classes*. In contrast, AVIATOR generates the classifier initialization for each class based on the context of the current class and the holistic effect of other classes. Thus it is easy for AVIATOR to deal with tasks with more classes, which makes our proposed method practical.

5 Experiments

We verify the effectiveness of our proposed AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach through both synthetic and real benchmark data sets. We will introduce the details of the model setups, including data sets, implementations, and evaluation protocols at first. Then we describe the results. At last we analyze the ablated model.

5.1 Setups

Here we introduce the general setups for few-shot classification in our experiments.

5.1.1 Data sets

Two benchmarks are used in our experiments. The *mini*ImageNet dataset (Vinyals et al. 2016) is a subset of the ILSVRC-12 dataset (Russakovsky et al. 2015). There are totally 100 classes and 600 examples in each class. For evaluation, we follow the split of Ravi and Larochelle (2017) and use 64 of 100 classes for meta-training, 16 for validation, and 20 for meta-test (model evaluation). In other words, a model is trained on few-shot tasks sampled from the 64 SEEN classes set during meta-training, and the best model is selected based on the few-shot

classification performance over the 16 class set. The final model is evaluated based on few-shot tasks sampled from the 20 UNSEEN classes. Following (Vinyals et al. 2016; Finn et al. 2017a; Snell et al. 2017), all images are first re-scaled to the fixed size $3 \times 84 \times 84$ during the pre-processing.

The Caltech-UCSD Birds (CUB) 200-2011 data set (Wah et al. 2011) is designed for fine-grained classification, which contains a total of 11,788 images of birds over 200 species. Due to the similarity between classes, it constructs difficult classification tasks when given limited training examples. Following the configuration of Triantafillou et al. (2017) and Chen et al. (2019), we use the provided bounding box of CUB to crop the center object of each image. All images in CUB are also resized to $3 \times 84 \times 84$. 100 classes are used for meta-training, 50 classes are used for model selection, and the last 50 classes are served as the UNSEEN set for model evaluation. Since there is no public released split from Triantafillou et al. (2017), we randomly shuffle all 200 classes in CUB, and choose SEEN and UNSEEN sets by ourselves.

5.1.2 Evaluation protocols

We use the same evaluation protocol over all benchmark data sets (Vinyals et al. 2016; Finn et al. 2017a; Snell et al. 2017; Triantafillou et al. 2017), i.e., the performance of 1-shot 5-way and 5-shot 5-way classification. We keep the same configuration of tasks between meta-training and meta-test. In other words, for the 1-shot 5-way problem, we keep sampling the 1-shot 5-way training set (a.k.a. support set in the literature) from SEEN class set during meta-training. Besides, to evaluate the optimized classifier for a particular N-way problem, another 15 examples from each of the N classes are sampled as the test set (a.k.a. query set in the literature) to provide the loss and supervision. Most previous methods are evaluated on 600 tasks sampled from the UNSEEN class set (Vinyals et al. 2016; Finn et al. 2017a; Snell et al. 2017; Triantafillou et al. 2017; Antoniou et al. 2018; Chen et al. 2019), which introduces high variance. Different from that, in our experiments, we test the model over 10,000 sampled few-shot tasks (Rusu et al. 2018; Ye et al. 2018). The mean accuracy and 95% confidence interval are recorded for comparison.

5.1.3 Implementation details

Following the setting of most existing methods (Vinyals et al. 2016; Finn et al. 2017a; Snell et al. 2017; Triantafillou et al. 2017), we use 4 identical neural network blocks to implement the embedding backbone ϕ. In each of the block, four components, i.e., a 3×3 convolution with 64 filters, a batch normalization (Ioffe and Szegedy 2015), a ReLU activation, and a 2×2 max-pooling, are stacked upon each other. Different from previous methods, we add another global max-pooling layer following the last block, which outputs 64-dimensional embeddings at last. The global pooling not only deals with the spatial transformation effectively but also relieves the optimization burden a lot.

Before the meta-training stage, we try to find a good initialization for the embedding ϕ. In particular, we add a linear layer on the backbone output and optimize a 64-way classification problem on the meta-training set with the cross-entropy loss function. Stochastic gradient descent with ADAM (Kingma and Ba 2014) is used to complete such optimization. The 16 classes for model selection also assist the choice of the pre-trained model. After each epoch, we use the current embedding and measure the nearest neighbor based few-shot classification performance on the sampled few-shot tasks from these 16 classes. The most suitable embedding function is recorded. After that, such learned backbone is used to initialize

Table 1 The range of hyper-parameters in three families of curves

Curve families	α	β	γ
Linear ($y = \alpha x + \beta$)	$[-3, 3]$	$[-3, 3]$	–
Square ($y = \alpha(x - \beta)^2 + \gamma$)	$[-0.15, -0.02] \cup [0.02, 0.15]$	$[-3.0, 3.0]$	$[-3.0, 3.0]$
Sine ($y = \alpha sin(\beta x + \gamma)$)	$[0.1, 5.0]$	$[0.5, 2.0]$	$[0.5, 2\pi]$

the embedding part ϕ of the whole model. For the meta-learning phase, ADAM is used with an initial learning rate 1e−4 for the backbone part and 10 times faster rates for the top layers. The learning rate of all layers will be halved after optimizing 2000 tasks. During the experiments, we find the pre-train stage facilitates the learning of few-shot optimizer a lot, which is consistent with Rusu et al. (2018) and Ye et al. (2018). Besides, we set the inner-task learning rate $\eta = 0.05$, with $T = 15$ gradient descent updates. The sizes of two hidden layers in the initialization generation function are 128 and 64 respectively.

5.2 Synthetic regression tasks

We first use a simple regression task to verify the ability of AVIATOR to deal with complex tasks. In each few-shot regression task, a limited number of points are provided for curve fitting, and the model is required to predict the value of unseen points (sampled from the same curve) precisely. To increase the difficulty of this task, three curve families are used, namely the linear ($y = \alpha x + \beta$), square ($y = \alpha(x - \beta)^2 + \gamma$), and sine ($y = \alpha sin(\beta x + \gamma)$). The range of parameters in the function families are listed in Table 1.

Regression tasks in both meta-training and meta-test stages are sampled in a hierarchical way. First, one of the three curve families are determined, then the parameters of a curve are sampled from the pre-specified range. The training and test points in regression tasks are perturbed by normal distribution $\mathcal{N}(0, 0.3)$ from the oracle curve. Five points are used as the training set. After fitting the curve with square loss, the model is evaluated on 10 test points. Both MAML and AVIATOR are meta-trained and meta-tested with 10,000 tasks. A 4-layer MLP, with ReLU activation and hidden layer dimension 100 is used as the backbone for both methods. Mean Square Error (MSE) is computed to evaluate the ability of the few-shot regression model. Since the heterogeneity of the tasks, AVIATOR is expected to get better results.

After meta-training, MAML and AVIATOR achieve 3.043 and 1.926 MSE respectively. One regression task for each curve family is randomly chosen for illustration (in Fig. 4). It can be found that MAML uses the same model initialization in all cases, which is non-informative for the complex task. The initial curve of AVIATOR obviously adapts for different scenarios.

5.3 Synthetic classification tasks

To visualize the classification boundary, we test MAML and AVIATOR on a 2-D classification task. We generate 100 classes in total based on the normal distribution, with mean sampled from $[0, 1]^2$ and variance equals $0.05I$. Half of the 100 classes are used for meta-training, half of the remaining classes are used for model selection, and the last 25 classes are for meta-test. Equipped with a two-layer MLP as the encoder, MAML and AVIATOR are trained by 50,000 5-shot 3-way tasks, and tested on 10,000 tasks. The average few-shot classification

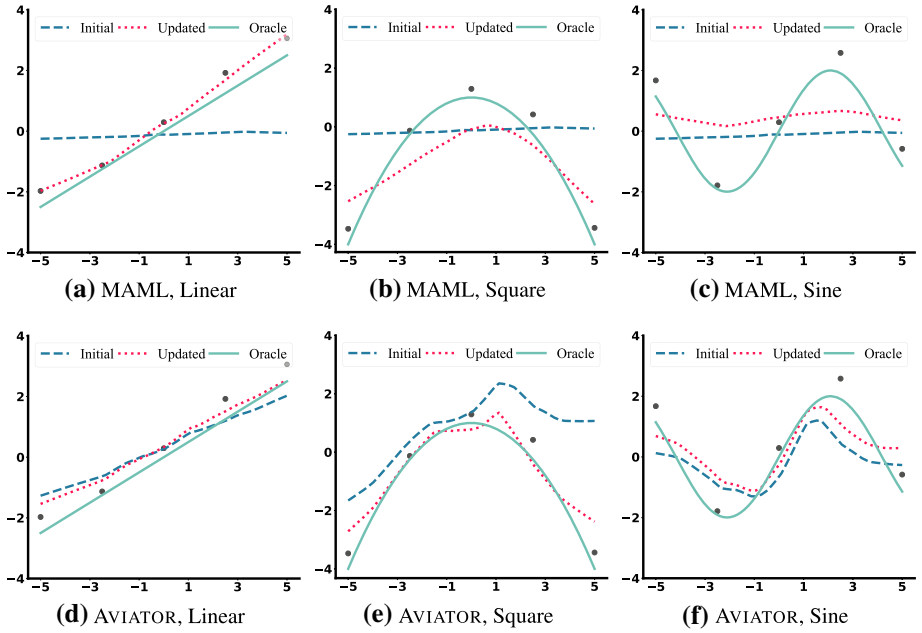


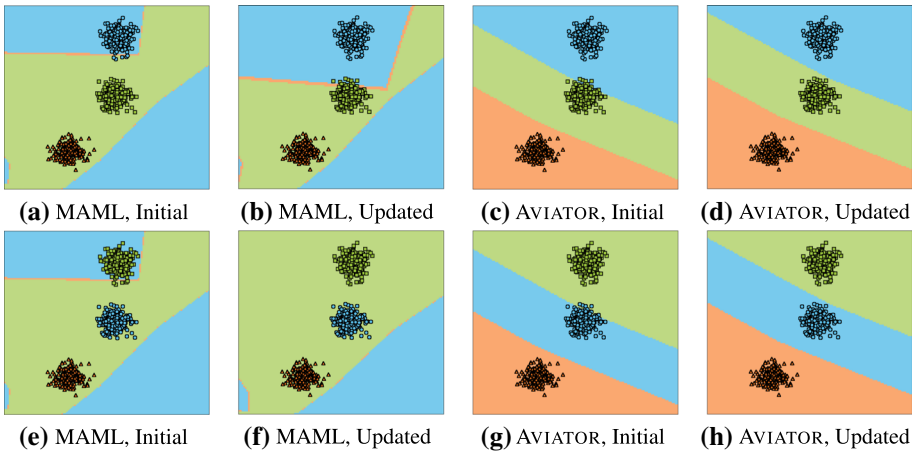
Fig. 4 Initial and updated regression models of MAML and AVIATOR on 3 different types of curves. Learning with 5 noisy examples (the black dots), the model is asked to fit the curve

Table 2 The 3-way classification performance of MAML and AVIATOR on the synthetic classification data set. The average accuracy using inner-task step-size 0.01 and different numbers of updates are reported

T	1	5	10	15
MAML	$35.34 \pm 0.36\%$	$43.19 \pm 0.30\%$	$45.45 \pm 0.30\%$	$45.42 \pm 0.29\%$
AVIATOR	$83.6 \pm 0.30\%$	$86.57 \pm 0.26\%$	$87.13 \pm 0.25\%$	$87.59 \pm 0.25\%$

accuracy of MAML and AVIATOR are listed in Table 2. By observing the results, we can find AVIATOR achieves much better classification results than MAML. Besides, to achieve the same level quality of the few-shot classification model (w.r.t. the test accuracy), AVIATOR requires much fewer steps. So the adaptively initialized model can effectively reduce the computational burden of the gradient-based meta-learning approach.

One 3-way task is randomly selected for visualization, as shown in Fig. 5. The first row in the figure corresponds to the task, while the second row presents the same task with a permutation of classes. 200 instances from each class are drawn to show the range of the class, and different classes instances/classification boundaries are denoted by different colors. From the results, MAML uses the uniform non-informative initial classifier, which is difficult to apply on various tasks, so that more gradient descent steps are required to update the initial model to fit a specific task. While for AVIATOR, it deals with two permuted tasks well, and the adaptively generated initial classifier is close to the final updated ones.



(a) MAML, Initial **(b)** MAML, Updated **(c)** AVIATOR, Initial **(d)** AVIATOR, Updated

(e) MAML, Initial **(f)** MAML, Updated **(g)** AVIATOR, Initial **(h)** AVIATOR, Updated

Fig. 5 Initial and updated classification models of MAML and AVIATOR on 2 different 3-way tasks. Colors of instances denote the class, and the shadow region are the classification boundary of the specified model. Each row corresponds to a task, and these two tasks only differ in the order of the classes

Table 3 The average classification accuracy and 95% confidence interval when different methods are evaluated on few-shot tasks sampled from *Mini*ImageNet. All methods use the 4-layer ConvNet Backbone

*Mini*ImageNet	1-Shot 5-Way	5-Shot 5-Way
MatchNet (Vinyals et al. 2016)	$48.14 \pm 0.78\%$	$63.48 \pm 0.66\%$
ProtoNet (Snell et al. 2017)	$47.74 \pm 0.84\%$	$66.68 \pm 0.68\%$
RelationNet (Sung et al. 2017)	$49.31 \pm 0.85\%$	$66.60 \pm 0.69\%$
MAML (Finn et al. 2017a)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (reproduce) (Chen et al. 2019)	$46.47 \pm 0.82\%$	$62.71 \pm 0.71\%$
MAML+(Antoniou et al. 2018)	$52.40 \pm 1.13\%$	$67.15 \pm 0.26\%$
ProtoMAML (Triantafillou et al. 2019)	$52.05 \pm 0.21\%$	$67.36 \pm 0.18\%$
AVIATOR (ours)	$\mathbf{54.97} \pm \mathbf{0.22}\%$	$\mathbf{68.37} \pm \mathbf{0.17}\%$

Values in bold denote the ones with the highest classification accuracy

5.4 Benchmark results

Table 3 shows the results of various few-shot learning methods on the *Mini*ImageNet data sets, where both 1-shot 5-way and 5-shot 5-way tasks are investigated. The main results of comparison methods are cited from a recent empirical study on few-shot learning (Chen et al. 2019). Different from the reported value of Finn et al. (2017a), there exists a noticeable gap of MAML's results between the reported and reproduced one. By adding some specific training strategies, Antoniou et al. (2018) improves the few-shot classification ability of MAML. We also re-implement the ProtoMAML (Triantafillou et al. 2019) approach, which constructs an embedding learning objective based on the relationship between the prototypes and the classifiers. Owing to the adaptive embedding in ProtoMAML, it can improve over vanilla MAML a lot, as reported in their paper. Our AVIATOR approach gets the best performance among all previous results, which validates the importance of the task-adaptive initialization in the few-shot inner-task update. It is also notable that our results are evaluated on 10,000

Table 4 The average classification accuracy and 95% confidence interval when different methods are evaluated on few-shot tasks sampled from CUB. All methods use the 4-layer ConvNet Backbone

CUB	1-Shot 5-Way	5-Shot 5-Way
MatchNet (Vinyals et al. 2016)	$61.16 \pm 0.89\%$	$72.86 \pm 0.70\%$
ProtoNet (Snell et al. 2017)	$51.31 \pm 0.91\%$	$70.77 \pm 0.69\%$
RelationNet (Sung et al. 2017)	$62.45 \pm 0.98\%$	$\mathbf{76.11 \pm 0.69}\%$
MAML (reproduce) (Chen et al. 2019)	$55.92 \pm 0.95\%$	$72.09 \pm 0.76\%$
ProtoMAML (Triantafillou et al. 2019)	$64.28 \pm 0.23\%$	$75.06 \pm 0.19\%$
AVIATOR (ours)	$\mathbf{67.59 \pm 0.23}$ %	75.32 ± 0.17 %

Values in bold denote the ones with the highest classification accuracy

Table 5 Comparison between different MAML-based improvements on *Mini*ImageNet benchmark with the 4-layer ConvNet backbone

*Mini*ImageNet	1-Shot 5-Way	5-Shot 5-Way
MAML (reproduce) (Chen et al. 2019)	$46.47 \pm 0.78\%$	$62.71 \pm 0.66\%$
Meta-SGD (Li et al. 2017)	$47.74 \pm 0.84\%$	$66.68 \pm 0.68\%$
MT-Net (Lee and Choi 2018)	$51.70 \pm 1.84\%$	–
MAML+ (Antoniou et al. 2018)	$52.40 \pm 1.13\%$	$67.15 \pm 0.26\%$
ProtoMAML (Triantafillou et al. 2019)	$52.05 \pm 0.21\%$	$67.36 \pm 0.18\%$
AVIATOR (ours)	$\mathbf{54.97 \pm 0.22}\%$	$\mathbf{68.37 \pm 0.17}\%$

Values in bold denote the ones with the highest classification accuracy

tasks, which are more convinced with a lower confidence interval. Similar trends can also be found in CUB data set in Table 4.

5.5 Ablation studies and discussions

In this subsection, we analyze various properties of our proposed AVIATOR approach on the *Mini*ImageNet data set.

5.5.1 The influence of different components in the inner-task update

There exist various key components to update the classifier within a task. In addition to focusing on the model initialization in MAML and AVIATOR, Meta-SGD (Li et al. 2017) adds another learnable parameter to assist the gradient descent; MT-Net (Lee and Choi 2018) enhances the construction of the backbone, which decomposes the embedding function into two task-common and task-specific feature generation flows; Antoniou et al. (2018) propose an orthogonal way to train MAML, some special tricks like the accumulated annealed step-wise losses are used. Since ProtoMAML (Triantafillou et al. 2019) dynamically updates the embedding for each task in a MAML way, the further performance improvement of ProtoMAML verifies the importance of considering task context and make the method adaptive. Table 5 lists the results of all these methods. AVIATOR easily gets the best performance with only an addition of a simple top-layer initialization generation function.

Table 6 Comparison between different methods on *Mini*ImageNet benchmark with the 4-layer ConvNet backbone. By using the attention, AVIATOR$^+$ achieves better results

*Mini*ImageNet	1-Shot 5-Way	5-Shot 5-Way
MAML (reproduce) (Chen et al. 2019)	$48.14 \pm 0.78\%$	$63.48 \pm 0.66\%$
MAML+ (Antoniou et al. 2018)	$52.40 \pm 1.13\%$	$67.15 \pm 0.26\%$
FEAT (Ye et al. 2018)	$55.15 \pm 1.13\%$	$\mathbf{71.61 \pm 0.26}\%$
AVIATOR (ours)	$54.97 \pm 0.22\%$	$68.37 \pm 0.17\%$
AVIATOR$^+$ (ours)	$\mathbf{55.60 \pm 0.21}\%$	$71.26 \pm 0.17\%$

Values in bold denote the ones with the highest classification accuracy

5.5.2 Different implementation of the *g*

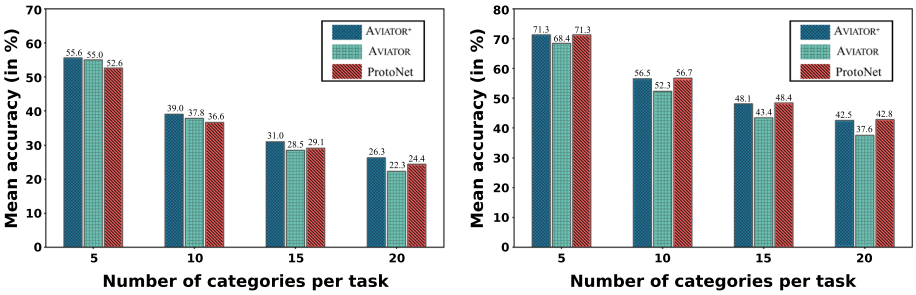
In previous experiments, we use the simple two-layer fully connected network to implement the the initialization generation function g. Attention mechanism (Vaswani et al. 2017) can also be introduced to generate the top-layer linear classifier. Denote the embedding prototype of the training set as $P = [\mathbf{p}_1, \ldots, \mathbf{p}_N] \in \mathbb{R}^{d \times N}$. For the nth class, the corresponding classifier is generated by $\mathbf{p}_n + \mathbf{Softmax}(\mathbf{p}_n^\top M P) P$. Here M is a learnable matrix, and $\mathbf{Softmax}(\cdot)$ is used to normalize the similarity value between one class and other prototypes. The usage of the attention can depict the relationship between classes better, which results in the AVIATOR$^+$ approach. The results of all methods are shown in Table 6, where AVIATOR$^+$ can further improve the few-shot classification performance. We also list the results of FEAT, which also takes advantage of the self-attention to adapt the embedding. It is notable that different from the embedding-based approach FEAT, AVIATOR$^+$ incorporates task context with an optimization-based strategy, and could be further improved with other helpful objectives (Ye et al. 2018).

5.5.3 Generalization of AVIATOR to more ways

One promising feature of the embedding-based few-shot learning approach is their ability to generalize to different forms of tasks. As in matching network (Vinyals et al. 2016) and prototypical network (Snell et al. 2017), by learning a few-shot facilitated embedding, tasks with a various number of classes can all be classified via the nearest neighbor rule. For MAML-based approaches, the main obstacle is the strong relationship between the classifier and the number of classes. By generating initial classifier based on the training set embedding adaptively, our AVIATOR approach handles different ways of tasks effectively. With models trained by 1-shot 5-way tasks, we test the *same* model on 1-shot {5, 10, 15, 20}-way tasks. The results of our re-implemented ProtoNet (Snell et al. 2017), AVIATOR, and AVIATOR$^+$ can be found in Fig. 6. Our AVIATOR and its variants can get the best performance in most scenarios.

6 Conclusion

Model-agnostic meta-learning (MAML) is an important flow of the few-shot learning research, which capsules the model information into the gradient and supervises the model update by the loss on the training set. Current results observe the difficulties of



(a) 1-Shot **(b)** 5-Shot

Fig. 6 Testing AVIATOR variants and our re-produced ProtoNet on *Mini*ImageNet with different numbers of ways. After meta-train with 5-way, these models are evaluated on {5, 10, 15, 20} ways 1-shot/5-shot tasks

MAML tackling complicated tasks, and its weakness to adaptively build the correspondence between a class and an initialized model. We propose our AdaptiVely InitiAlized Task OptimizeR (AVIATOR) approach to incorporate the task context and enable the gradient-based meta-learning applicable on various kinds of few-shot learning tasks. Using a re-parameterization strategy, the task-specific initialization facilitates the inner-task training a lot. Visualization experiments on synthetic data show the adaptively generated model initialization of AVIATOR is consistent with the first impression of the task. Empirical results on benchmark data sets verify the superiority of AVIATOR as well. The improvement of AVIATOR over MAML can be applied to almost all fields where MAML works well. We will investigate more real applications in the future.

Acknowledgements This research was supported by the The National Key R&D Program of China (2018YFB1004300), NSFC (61773198, 61751306, 61632004), and the program A for Outstanding Ph.D. candidate of Nanjing University.

References

Achille, A., & Soatto, S. (2018). Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, *19*, 50:1–50:34.

Andrychowicz, M., Denil, M., Colmenarejo, S. G., Hoffman, M. W., Pfau, D., Schaul, T., et al. (2016). Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, *29*, 3981–3989.

Antoniou, A., Edwards, H., & Storkey, A. J. (2018). How to train your MAML. CoRR arXiv:1810.09502.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Chen, W. Y., Liu, Y.C., Kira, Z., Wang, Y.C.F., & Huang, J. B. (2019). A closer look at few-shot classification. CoRR arXiv:1904.04232.

Clavera, I., Nagabandi, A., Fearing, R.S., Abbeel, P., Levine, S., & Finn, C. (2018). Learning to adapt: Meta-learning for model-based control. CoRR arXiv:1803.11347.

Dai, W. Z., Muggleton, S., Wen, J., Tamaddoni-Nezhad, A., & Zhou, Z. H. (2017). Logical vision: One-shot meta-interpretive learning from real images. In *Proceedings of the 27th international conference on inductive logic programming, Orléans, France* (pp. 46–62).

Deleu, T., & Bengio, Y. (2018). The effects of negative adaptation in model-agnostic meta-learning. CoRR arXiv:1812.02159.

Denevi, G., Ciliberto, C., Stamos, D., & Pontil, M. (2018). Learning to learn around A common mean. *Advances in Neural Information Processing Systems*, *31*, 10190–10200.

Finn, C., Abbeel, P., & Levine, S. (2017a). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th international conference on machine learning, Sydney, Australia* (pp. 1126–1135).

Finn, C., & Levine, S. (2018). Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *Proceeding of the 6th international conference on learning representations, Vancouver, Canada*.

Finn, C., Xu, K., & Levine, S. (2018). Probabilistic model-agnostic meta-learning. *Advances in Neural Information Processing Systems*, *31*, 9537–9548.

Finn, C., Yu, T., Zhang, T., Abbeel, P., & Levine, S. (2017b). One-shot visual imitation learning via meta-learning. In *Proceedings of the 1st annual conference on robot learning, Mountain View, CA* (pp. 357–368).

Franceschi, L., Donini, M., Frasconi, P., & Pontil, M. (2017). A bridge between hyperparameter optimization and larning-to-learn. CoRR arXiv:1712.06283.

Garg, V. (2018). Supervising unsupervised learning. *Advances in Neural Information Processing Systems*, *31*, 4996–5006.

Gu, J., Wang, Y., Chen, Y., Li, V. O. K., & Cho, K. (2018). Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 conference on empirical methods in natural language processing, Brussels, Belgium* (pp. 3622–3631).

Hariharan, B., & Girshick, R. B. (2017). Low-shot visual recognition by shrinking and hallucinating features. In *IEEE international conference on computer vision* (pp. 3037–3046). Italy: Venice.

Hsu, K., Levine, S., & Finn, C. (2018). Unsupervised learning via meta-learning. CoRR arXiv:1810.02334

Huang, P. S., Wang, C., Singh, R., Yih, W., & He, X. (2018). Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies, New Orleans, LA* (pp. 732–738).

Huang, S. J., Jin, R., & Zhou, Z. H. (2014). Active learning by querying informative and representative examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(10), 1936–1949.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on machine learning, Lille, France* (pp. 448–456).

Karlinsky, L., Shtok, J., Tzur, Y., & Tzadok, A. (2017). Fine-grained recognition of thousands of object categories with single-example training. In *IEEE conference on computer vision and pattern recognition, Honolulu, HI* (pp. 965–974).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. CoRR arXiv:1412.6980.

Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop* (Vol. 2) https://sites.google.com/site/deeplearning2015/home.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Lake, B. M., Salakhutdinov, R., Gross, J., & Tenenbaum, J. B. (2011). One shot learning of simple visual concepts. In *Proceedings of the 33th annual meeting of the cognitive science society, Boston, MA*.

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

Lee, Y., & Choi, S. (2018). Gradient-based meta-learning with learned layerwise metric and subspace. In *Proceedings of the 35th international conference on machine learning, Stockholm, Sweden* (pp. 2933–2942).

Li, F. F., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(4), 594–611.

Li, Y. F., & Zhou, Z. H. (2015). Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(1), 175–188.

Li, Z., Zhou, F., Chen, F., & Li, H. (2017). Meta-SGD: Learning to learn quickly for few shot learning. CoRR arXiv:1707.09835.

Maurer, A. (2009). Transfer bounds for linear feature learning. *Machine Learning*, *75*(3), 327–350.

Maurer, A., Pontil, M., & Romera-Paredes, B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, *17*, 81:1–81:32.

Motiian, S., Jones, Q., Iranmanesh, S. M., & Doretto, G. (2017). Few-shot adversarial domain adaptation. *Advances in Neural Information Processing Systems*, *30*, 6673–6683.

Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms. CoRR arXiv:1803.02999.

Probst, P., Boulesteix, A. L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, *20*, 53:1–53:32.

Ravi, S., & Larochelle, H. (2017). Optimization as a model for few-shot learning. In *In international conference on learning representations*.
Reed, S. E., Chen, Y., Paine, T., van den Oord, A., Eslami S. M. A., Rezende, D. J., Vinyals, O., & de Freitas, N. (2017). Few-shot autoregressive density estimation: Towards learning to learn distributions. CoRR arXiv:1710.10304.
Ren, M., Zeng, W., Yang, B., & Urtasun, R. (2018). Learning to reweight examples for robust deep learning. In *Proceedings of the 35th international conference on machine learning, Stockholm, Sweden* (pp. 4331–4340).
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.
Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., & Hadsell, R. (2018). Meta-learning with latent embedding optimization. CoRR arXiv:1807.05960.
Shyam, P., Gupta, S., & Dukkipati, A. (2017). Attentive recurrent comparators. In *Proceedings of the 34th international conference on machine learning, Sydney, Australia* (pp. 3173–3181).
Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, *30*, 4080–4090.
Su, D., Zhang, H., Chen, H., Yi, J., Chen, P. Y., & Gao, Y. (2018). Is robustness the cost of accuracy? A comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the 15th European conference on computer vision, Munich, Germany* (pp. 644–661).
Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., & Hospedales, T. M. (2017). Learning to compare: Relation network for few-shot learning. CoRR arXiv:1711.06025.
Tan, X., Chen, S., Zhou, Z. H., & Zhang, F. (2006). Face recognition from a single image per person: A survey. *Pattern Recognition*, *39*(9), 1725–1745.
Thrun, S., & Pratt, L. (2012). *Learning to learn*. New York: Springer.
Triantafillou, E., Zemel, R. S., & Urtasun, R. (2017). Few-shot learning through an information retrieval lens. *Advances in Neural Information Processing Systems*, *30*, 2252–2262.
Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P. A., & Larochelle, H. (2019). Meta-dataset: A dataset of datasets for learning to learn from few examples. CoRR arXiv:1903.03096.
Vartak, M., Thiagarajan, A., Miranda, C., Bratman, J., & Larochelle, H. (2017). A meta-learning perspective on cold-start recommendations for items. *Advances in Neural Information Processing Systems*, *30*, 6907–6917.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*, 6000–6010.
Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95.
Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, *29*, 3630–3638.
Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). The Caltech-UCSD birds-200-2011 dataset. Technical report.
Wang, P., Liu, L., Shen, C., Huang, Z., van den Hengel, A., & Shen, H.T. (2017a). Multi-attention network for one shot learning. In *IEEE conference on computer vision and pattern recognition, Honolulu, HI* (pp. 6212–6220).
Wang, T., Zhu, J. Y., Torralba, A., & Efros, A. A. (2018a). Dataset distillation. CoRR arXiv:1811.10959.
Wang, Y., Ramanan, D., & Hebert, M. (2017b). Learning to model the tail. *Advances in Neural Information Processing Systems*, *30*, 7032–7042.
Wang, Y., Girshick, R. B., Hebert, M., & Hariharan, B. (2018b). Low-shot learning from imaginary data. CoRR arXiv:1801.05401.
Ye, H. J., Hu, H., Zhan, D. C., & Sha, F. (2018). Learning embedding adaptation for few-shot learning. CoRR arXiv:1812.03664.
Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., & Levine, S. (2018). One-shot imitation from observing humans via domain-adaptive meta-learning. CoRR arXiv:1802.01557.
Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. *Advances in Neural Information Processing Systems*, *30*, 3394–3404.
Zhang, Y., Wei, Y., & Yang, Q. (2018). Learning to multitask. *Advances in Neural Information Processing Systems*, *31*, 5776–5787.