A Lazy Approach to Long-Horizon Gradient-Based Meta-Learning

Muhammad Abdullah Jamal[†] Liqiang Wang[†] Boqing Gong[♯]
[†]University of Central Florida [♯]Google

Abstract

Gradient-based meta-learning first trains task-specific models by an inner loop and then backpropagates meta-gradients through the loop to update the meta-model. To avoid high-order gradients, existing methods either take a small number of inner steps or approximate the meta-updates for the situations that the meta-model and task models lie in the same space. To enable long inner horizons for more general meta-learning problems, we instead propose an intuitive teacher-student strategy. The key idea is to employ a student network to adequately explore the search space of task-specific models, followed by a teacher's "leap" toward the regions probed by the student. The teacher not only arrives at a high-quality model but also defines a lightweight computational graph for the meta-gradients. Our approach is generic; it performs well when applied to four meta-learning algorithms over three tasks: few-shot learning, long-tailed object recognition, and adversarial blackbox attack.

1. Introduction

Humans can quickly learn the skills needed for new tasks by drawing from a fund of prior knowledge and experience. To grant machine learners this level of intelligence, meta-learning studies how to leverage past learning experiences to more efficiently learn for a new task [48]. A hallmark experiment design provides a meta-learner a variety of few-shot learning tasks (meta-training) and then desires it to solve previously unseen and yet related few-shot learning tasks (meta-test). This design enforces "learning to learn" because the few-shot training examples are insufficient for a learner to achieve high accuracy on any task in isolation.

Recent meta-learning methods focus on deep neural networks. Some learn recurrent neural networks as an update rule to a model [36, 2]. Some transfer attention schemes across tasks [30, 49]. Gradient-based meta-learning gains momenta recently following the seminal work [14]. It is model-agnostic meta-learning (MAML), learning a global model initialization from which a meta-learner can quickly derive task-specific models by using a few training examples.

In its core, MAML is a bilevel optimization problem [10]. The upper level searches for the best global initialization, and the lower level optimizes individual models, which all share the common initialization, for particular tasks sampled from a task distribution. This problem is hard to solve. [14] instead propose a "greedy" algorithm, which comprises two loops. The inner loop samples tasks and updates the task-specific models by k steps using the tasks' training examples. The k-step updates write a differentiable computation graph. The outer loop updates the common initialization by backpropagating meta-gradients through the computation graph. This method is "greedy" in that the number of inner steps is often small (e.g., $k = 1$). The outer loop takes actions before the inner loop sufficiently explores its search space.

This "greedy" algorithm is due to practical constraints that backpropagating meta-gradients through the inner loop incurs high-order derivatives, big memory footprints, and the risk of vanishing or exploding gradients. For the same reason, some related work also turns to greedy strategies, such as meta-attack [13] and learning to reweigh examples [38].

To this end, two questions arise naturally. *Would a less greedy gradient-based meta-learner (say, $k>10$ inner steps) achieve better performance? How to make it less greedy?*

Some first-order algorithms [14, 32, 15] have provided an affirmative answer to the first question above. [35] proposed a less "greedy" MAML by regularizing the inner loop. However, they are highly tailored in that the meta-model and task models lie in the same space, preventing them from tackling other meta-learning problems, for example, the long-tailed classification described later.

To answer the questions for more general meta-learning scenarios, we provide some preliminary results by introducing a lookahead optimizer [55] into the inner loop. It can be viewed as a teacher-student scheme. We use a student neural network to explore the search space for a given task adequately (by a large number k of updates), and a teacher network then takes a "leap" toward the regions visited by the student. As a result, the teacher network not only arrives at a high-performing model but also defines a very lightweight computational graph for the outer loop. In contrast to the traditionally "greedy" meta-learning framework used in MAML [14], meta-attack [13], learning to reweigh examples [38], etc., the teacher is "lazy". It sends a student to optimize for a task up to many steps and moves only once after that.

Our approach improves the gradient-based meta-learning framework rather than a single algorithm. Hence, we evaluate it on different methods and tasks, including MAML and Reptile [32] for few-shot learning, a two-component weighting algorithm [20] for long-tailed classification, and meta-attack [13]. Extensive results provide an affirmative answer to the first question above: long-horizon exploration in the inner loop improves a meta-learner's performance. We expect our approach, along with the compelling experimental results, can facilitate future work to address the second question above.

2. Related Work

Meta-learning has been a long-standing sub-field in machine learning [40, 45, 31]. Early approaches update a model's parameters by training a meta-learner [4, 5, 41]. This has been well studied in optimizing neural networks, and one such family of meta-learning learns an optimizer [36, 26, 2]. A specialized neural network takes gradients as input and outputs an update rule for the learner. In addition to the update rule, [36] also learn the weight initialization for few-shot learning. Finally, there are several approaches [29, 52] for training generic optimizers that can be applied broadly to different neural networks and datasets.

Under the context of few-shot learning, another family of meta-learning involves metric-learning based methods [49, 43, 30, 22, 33], which learn a metric space to benefit different few-shot learning tasks. The goal is to find the similarity between two samples regardless of their classes using some distance metric so that the similarity function can be used to classify the unseen classes at the test stage. Some recent studies along this line include Matching Networks [49], which employs the cosine similarity, Prototypical Networks [43], which uses the Euclidean distance to compute the similarity, Relation Network [44], which uses a relation module as the similarity function, ridge regression [6], and graph neural networks [39].

More recently, gradient-based meta-learning gains its momentum, and a variety of methods have been proposed in this vein. The most notable one among them might be MAML [14], where the goal is to learn the network weight initialization so that it can adapt to unseen tasks rapidly. There have been extensions to improve MAML. Meta-SGD [27] learns the learning rates along with the weight initialization. Regularization techniques [54, 21] are introduced to MAML to mitigate over-fitting. [34] preconditions on the gradients in the inner loop by learning a curvature. Despite MAML's popularity, it is still computationally expensive and consumes large memory due to the computation of high-order derivatives. The authors show that the first-order approximation, which neglects the gradients of the inner loop during meta-optimization, performs about the same as the original MAML. Another first-order meta-learning method is Reptile [32], which decouples the inner and outer optimization steps. iMAML [35] provides an approximate solution for meta-gradients by using an algorithm based on conjugate gradients, and its low-level optimization is similar to Meta-MinibatchProx [56]. The idea is to add an ℓ_2 regularizer in the inner loop, allowing the updated parameters close to the initial parameters. Similar to iMAML, [28, 3] provide approximate solutions for optimizing hyperparameters and simulator parameters respectively.

3. "Greedy" Gradient-Based Meta-Learning

We first review gradient-based meta-learning from the perspective of "search space carving".

Notations. Let $P_{\mathcal{T}}$ denote a task distribution. For each task drawn from the distribution $\mathcal{T} \sim P_{\mathcal{T}}$, we have a training set $\mathcal{D}_{tr}$ and a validation set $\mathcal{D}_{val}$, both in the form of $\{(x_1, y_1), (x_2, y_2), \cdots\}$ where x_m and y_m are respectively an input and a label. We learn a predictive model for a task by minimizing the empirical loss $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi)$ over the training set while using the validation set to choose hyper-parameters (e.g., early stopping), where ϕ collects all trainable parameters of the model. Similarly, we denote by $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi)$ the loss calculated over the validation set.

Meta-learning as "space carving". Instead of focusing on an isolated task, meta-learning takes a global view and introduces a meta-model, parameterized by θ, that can improve the learning efficiency for all individual tasks drawn from the task distribution $P_{\mathcal{T}}$. The underlying idea is to derive a task-specific model ϕ from not only the training set $\mathcal{D}_{tr}$ but also the meta-model θ, i.e., $\phi \in \mathcal{M}(\theta, \mathcal{D}_{tr})$. We refer to $\mathcal{M}(\theta, \mathcal{D}_{tr})$ the "carved" search space for the task-specific model ϕ, where the "carving" function is realized as an attention module in [49, 30], as a conditional neural process in [17, 18], as a gradient-based update rule in [14, 34, 27, 32], and as a regularized optimization problem in [35, 56].

An optimal meta-model θ^* is supposed to yield the best task-specific models in expectation,

$$\begin{aligned} \theta^* \leftarrow \arg\min_{\theta} \; & \mathbb{E}_{\mathcal{T}\sim P_{\mathcal{T}}, \mathcal{D}_{val}\sim\mathcal{T}} \, \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi^*(\theta)) \\ \text{subject to } \phi^*(\theta) \leftarrow & \arg \min_{\phi\in\mathcal{M}(\theta,\mathcal{D}_{tr})} \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi). \end{aligned} \tag{1}$$

One can estimate the optimal meta-model θ^* from some tasks and then use it to "carve" the search space, $\mathcal{M}(\theta^*, \mathcal{D}_{tr})$, for novel tasks' models.

Gradient-based meta-learning. One of the notable meta-learning methods is MAML [14], which uses a gradient-based update rule to "carve" the search space for a task-specific model,

$$\begin{aligned} \mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr}) := \{\phi_0 \leftarrow \theta\} \cup \{\phi_j \,|\, \phi_j \leftarrow \phi_{j-1} \\ -\alpha\nabla_\phi \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi_{j-1}),\ j = 1, 2, \cdots, k\} \end{aligned} \tag{2}$$

where the meta-model θ becomes an initialization to the task-specific model ϕ_0, the other candidate models $\phi_1, \cdots, \phi_k$ are obtained by gradient descent, and $\alpha > 0$ is a learning rate. Substituting it into equation (1), $\phi_k \in \mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr})$ is naturally a solution to the lower-level optimization problem, and MAML solves the upper-level optimization problem by meta-gradient descent,

$$\theta \leftarrow \theta - \beta \mathbb{E}_{\mathcal{T} \sim P_{\mathcal{T}}, \mathcal{D}_{val} \sim \mathcal{T}} \nabla_\theta \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi_k(\theta)), \qquad (3)$$

where β is a learning rate, and $\phi_k(\theta)$ indicates the dependency on the meta-model θ. The meta-gradient must back-propagate through the chain of updates in eq. (2), which has to be short (e.g., $k = 1$) to avoid big memory footprints, high-order derivatives, and the risk of vanishing or exploding gradients.

We say MAML is "greedy" in that it descends meta-gradients for the meta-model θ before it runs adequate updates to the task-specific model ϕ. As an increasing number of works adopt the gradient-based "search space carving" for task-specific models [27, 35, 34, 16, 53], they also bear greedy algorithms. Relaxing the greedy strategy may benefit not one, but a variety of, high-order meta-learning methods.

4. A "Lazy" Approach to Gradient-Based Meta-Learning

In this section, we describe a "lazy" meta-learning approach, which is readily applicable to different gradient-based meta-learning algorithms. We first describe the general approach as an improvement to MAML and then customize it for few-shot learning, long-tailed classification, and meta-attack.

4.1. General Approach

Given a meta-model θ, we "carve" the search space for task-specific models $\phi \in \mathcal{M}(\theta, \mathcal{D}_{tr})$ by a teacher-student scheme. The key idea is to let a student explore the search space adequately using the training set of a task-specific model without worrying the length of the update chain because a teacher will examine the explored regions by the student, followed by a one-step "leap". Hence, one can update the meta-model by backpropagating meta-gradients through the teacher's "leap", not the student's update chain (ignoring that the chain starts from the meta-model). Figure 1 illustrates the main idea.

An exploratory student acts exactly the same as the gradient-based updates in MAML except that it explores the feasible space by a large number of steps ($k > 10$), resulting in $k + 1$ checkpoints of a task-specific model $\phi \in \mathcal{M}_{\text{MAML}}(\theta, \mathcal{D}_{tr}) = \{\phi_j, j = 0, \cdots, k\}$. It is clear from Section 3 that we cannot backpropagate the meta-gradients through the long chain of checkpoints, $\phi_0, \cdots, \phi_k$, made by the exploratory student.

A lazy teacher sits at the initialization $\phi_0 = \theta$ until the student stops. It then takes a "leap" towards the region explored by the student. The teacher essentially defines another "carved search space" for the task-specific model ϕ,

$$\mathcal{M}_{\text{LAZY}}(\theta, \mathcal{D}_{tr}) := \gamma\theta + (1 - \gamma)\mathcal{R}_{k-b+1,\cdots,k} \qquad (4)$$

where $\gamma \in [0, 1]$. The region $\mathcal{R}_{k-b+1,\cdots,k}$ is a convex hull of the last b checkpoints the student visited:

$$\mathcal{R}_{k-b+1,\cdots,k} := \alpha_{k-b+1}\phi_{k-b+1} + \alpha_{k-b+2}\phi_{k-b+2} + \cdots + \alpha_k\phi_k, \qquad (5)$$

where the coefficients $\{\alpha\}$ are non-negative and their sum equals 1, i.e., $\alpha_{k-b+1} + \cdots + \alpha_k = 1$. The last b checkpoints presumably cover a high-quality task-specific model ϕ by a better chance than the first few checkpoints. We shall experiment with $b = 3$ and $b = 1$.

Any task-specific model ϕ in this "lazy" space $\mathcal{M}_{\text{LAZY}}(\theta, \mathcal{D}_{tr})$ is determined by the hyper-parameters γ and $\alpha_{k-b+1}, \cdots, \alpha_k$, over which we conduct a grid search to minimize the validation loss $\mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\phi)$. This is similar in spirit to meta-SGD [27], which uses the validation data to search for the learning rates.

Denote by $\hat{\gamma}\theta + (1 - \hat{\gamma})\hat{\phi}$ the task-specific model as a result of the grid search. Notably, it is only one hop away from the meta-model θ, making it easy to compute meta-gradients. Concretely, the meta-gradient descent for the meta model θ becomes $\theta \leftarrow \theta - \beta\mathbb{E}_{\mathcal{T} \sim P_{\mathcal{T}}, \mathcal{D}_{val} \sim \mathcal{T}} \nabla_\theta \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{val}}(\hat{\gamma}\theta + (1 - \hat{\gamma})\hat{\phi})$, which is apparently more manageable than the gradients in eq. (3) when $k > 1$.

Algorithm 1 "Lazy" Meta-Learning

```
Require: A distribution over tasks $P_{\mathcal{T}}$
Require: Learning rates $\eta, \beta$
Ensure: The meta model $\theta$
 1: Randomly initialize the meta-model $\theta$
 2: while not done do
 3:     Sample a batch of tasks $\{\mathcal{T}_i \sim P_{\mathcal{T}}\}$
 4:     for all $\{\mathcal{T}_i\}$ do
 5:         Sample data $\mathcal{D}_{tr}$ and $\mathcal{D}_{val}$ for $\mathcal{T}_i$
 6:         $\phi_{i,0} \leftarrow \theta$
 7:         for $j = 1, 2, \cdots, k$ do                                  //student
 8:             $\phi_{i,j} \leftarrow \phi_{i,j-1} - \eta\nabla_\phi \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{tr}}(\phi_{i,j-1})$
 9:         end for
10:         Grid-search $\mathcal{M}_{\text{LAZY}}(\theta, \mathcal{D}_{tr})$ such that $\mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}$ is
    minimized at $\hat{\gamma}_i\theta + (1 - \hat{\gamma}_i)\hat{\phi}_i$                     //teacher
11:         $\phi_i(\theta) \leftarrow \hat{\gamma}_i\theta + (1 - \hat{\gamma}_i)\hat{\phi}_i$            //teacher
12:     end for
13:     $\theta \leftarrow \theta - \beta\nabla_\theta \sum_i \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}(\phi_i(\theta))$
14: end while
```

Algorithm 1 presents our "lazy" approach in detail. In the outer while-loop, we sample a batch of tasks $\{\mathcal{T}_i\}$ (Line

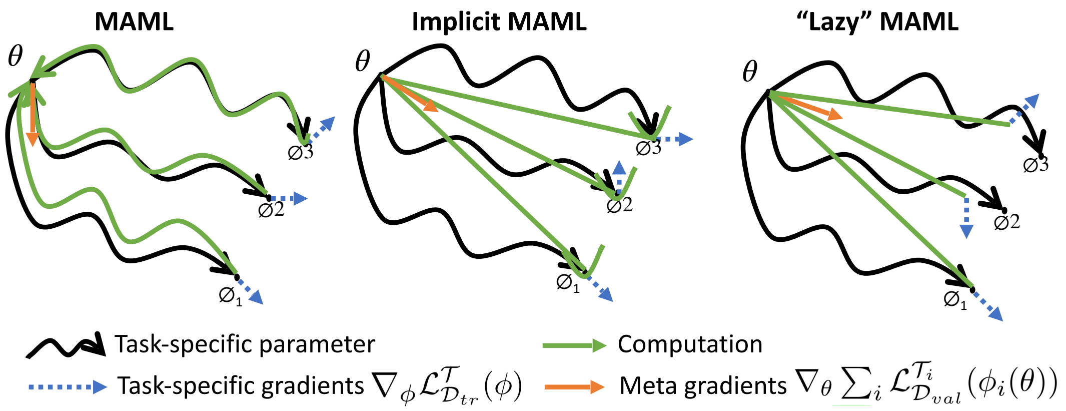

MAML
Implicit MAML
"Lazy" MAML
θ
$\varnothing_1$
$\varnothing_2$
$\varnothing_3$
Task-specific parameter
Computation
Task-specific gradients $\nabla_\phi \mathcal{L}^{\mathcal{T}}_{\mathcal{D}_{tr}}(\phi)$
Meta gradients $\nabla_\theta \sum_i \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}(\phi_i(\theta))$

Figure 1. To compute the meta-gradients $\nabla_\theta \sum_i \mathcal{L}^{\mathcal{T}_i}_{\mathcal{D}_{val}}(\phi_i(\theta))$, MAML [14] differentiates through the inner updates, the implicit MAML [35] approximates local curvatures, while we differentiate through the "lazy" teacher's one-step "leap". The exploratory student may make many steps of inner updates before the teacher's "leap".

3, or L3) and use them to make a gradient update to the meta-model θ (L13). All task-specific models $\{\phi_{i,0}\}$ are initialized to the current meta-model θ (L6). For each task $\mathcal{T}_i$, the student first runs gradient descent with respect to the task-specific model ϕ_i up to k steps (L8), and the teacher then takes a "leap" from the initial meta-model θ according to the checkpoints visited by the student (L10–11).

Remarks. Our "lazy" teacher is motivated by the lookahead optimizer [55]. They have some key differences as follows due to the meta-learning setup. We initialize multiple task-specific models by the meta-model. Moreover, we dynamically choose the "leap" rate γ by a validation set. Finally, the validation data allows us to take advantage of not one checkpoint, but a region around the checkpoints visited by the student.

Like Reptile, our approach allows the inner loop to make many steps of updates to task-specific models. Moreover, we share the same update rule as Reptile by the end of the many-step exploration. However, we apply that rule to the task-specific models, while Reptile essentially uses it to update the meta-model. Unlike Reptile, we use meta-gradients to update the meta-model. This difference is subtle and vital, making it straightforward to apply our approach to the two-component weighting algorithm for long-tailed classification (and other meta-like algorithms) but unclear how to do it for Reptile.

We share the same goal, to make MAML less "greedy", as the recently proposed implicit gradients (iMAML) [35]. iMAML changes the lower-level problem in eq. (1) to an ℓ_2-regularized problem, which lends an analytical expression for the meta-gradient. But it is expensive to compute and has to be approximated by a conjugate gradient algorithm. The ℓ_2 regularization also falls short in capturing structural relations between a meta-model and task-specific models.

4.2. Few-Shot Learning, Long-Tailed Classification, and Meta-Attack

Since the "lazy" teacher does not change the innermost loop of gradient-based meta-learning — it instead "leaps" over the chain of updates to the task-specific model ϕ, we can apply it to different algorithms. We evaluate it on few-shot learning, long-tailed classification, and meta-attack, in which meta-learning based methods have led to state-of-the-art results.

Few-shot learning in this paper concerns an N-way-K-shot classification problem. To customize Algorithm 1 for this problem, we randomly select N classes for each task $\mathcal{T}_i$ and then draw from each class $K+1$ examples with labels, K of which are assigned to the training set $\mathcal{D}_{tr}$ and one is to the validation set $\mathcal{D}_{val}$. Besides, we choose the hyper-parameter γ_i by using the task-specific model's classification accuracy on the validation set, instead of the loss in L10, Algorithm 1.

There is an interesting "trap" in few-shot learning, identified as over-fitting by memorization [53]. The tasks $\{\mathcal{T}_i\}$ drawn from a distribution $P_{\mathcal{T}}$ are supposed to be i.i.d., but they could be correlated in the following scenario. Suppose there exists a global order of all classes. If we maintain this order among the N classes in each task, the meta-model could over-fit the tasks seen during meta-training by memorizing the functions that solve these tasks, and it would fail to generalize to new tasks. Hence, it is important to randomly shuffle the N classes every time we sample them for a task (e.g., "dogs" and "cats" are respectively labeled as 0 and 1 in a two-way classification task, and yet they are shuffled to 1 and 0 in another two-way task).

We will empirically show that our approach is less prone to over-fitting than MAML even without class shuffling. A possible reason is that we use longer chains of updates $(\phi_0, \cdots, \phi_k, k > 10)$ to learn the functions that solve the individual tasks, making them harder to memorize.

Long-tailed classification emerges as an inevitable challenge as object recognition makes progress toward large-scale, fine-grained classes [47, 51], which often exhibit a long-tailed distribution. To uplift infrequent classes, [20] propose to weigh each training example by two components, a fixed component w_y to balance different classes [11] and a trainable component ϵ_i. We improve their learning method by a "lazy" teacher, as described in Algorithm 2. It alternatively optimizes the per-example weight ϵ_i (using a balanced validation set) and a recognition network θ (using the long-tailed training set), in the same spirit as meta learning (cf. Algorithm 1 vs. L5-12 in Algorithm 2). We insert a "lazy" teacher model to L6, let it take a "leap" in L12, and then backpropagate the gradient with respect to the per-example weight ϵ_i through the "leap".

Algorithm 2 "Lazy" Two-Component Weighting for Long-Tailed Recognition

```
Require: A training set $\mathcal{D}_{tr}$ whose class frequency is long-tailed, a balanced validation set $\mathcal{D}_{val}$
Require: Class-wise weights $\{w_y\}$ estimated by using [11]
Require: Learning rates $\eta$, $\tau$, pre-training steps $t_1$, fine-tuning steps $t_2$
1: Train a recognition network, parameterized by $\theta$, for $t_1$ steps by a standard cross-entropy loss
2: for $t = t_1 + 1, \cdots, t_1 + t_2$ do
3:     Sample a mini-batch $B$ from the training set $\mathcal{D}_{tr}$
4:     Set $\epsilon_i \leftarrow 0, \forall i \in B$, and denote by $\epsilon := \{\epsilon_i, i \in B\}$
5:     Compute $\mathcal{L}_B(\theta, \epsilon) := \frac{1}{|B|}\sum_{i\in B}(w_{y_i} + \epsilon_i)\mathcal{L}_i(\theta)$   //$\mathcal{L}_i$ is a cross-entropy over the $i$-th input
6:     Update $\tilde{\theta}(\epsilon) \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_B(\theta, \epsilon)$   // The "lazy" teacher, which depends on $\epsilon$
7:     Initialize a student model by setting $\phi_0 \leftarrow \tilde{\theta}(\epsilon)$
8:     for $j = 1, 2, ..., k$ do
9:         Update the student model by gradient descent $\phi_j \leftarrow \phi_{j-1} - \eta\nabla_\phi\mathcal{L}_B(\phi_{j-1}, \epsilon)$
10:    end for
11:    Grid search for $\gamma$ s.t. the teacher's "leap", $\gamma\tilde{\theta}(\epsilon) + (1-\gamma)\phi_k$, yields high accuracy on $\mathcal{D}_{val}$
12:    Update $\epsilon \leftarrow \epsilon - \tau\nabla_\epsilon\mathcal{L}_{\mathcal{D}_{val}}(\gamma\tilde{\theta}(\epsilon) + (1-\gamma)\phi_k)$
13:    Compute $\mathcal{L}_B(\theta, \epsilon)$ (cf. Line 5) and update $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_B(\theta, \epsilon)$
14: end for
```

Meta-attack [13] is a query-efficient blackbox attack algorithm on deep neural networks. Recent work has shown that one can manipulate an image recognition network's predictions by adding very small perturbations to benign inputs. However, if the network's architecture and weights are unknown (blackbox), it takes a large number of queries into the network to find a valid adversarial example. To improve the query efficiency, [13] propose to learn a meta-model from many whitebox neural networks and then generalize it to blackbox attacks. They train this meta-model by using the same meta-learning framework as Algorithm 1. Therefore, it is straightforward to improve their inner loop by our "lazy" teacher; we postpone the detailed algorithm to supplementary materials.

5. Experiments

We evaluate the "lazy", long-horizon meta-learning approach by plugging it into different algorithms with applications to few-shot learning, long-tailed recognition, and meta-attack.

5.1. Few-Shot Learning

We experiment with four datasets for few-shot learning: Omniglot [24], MiniImageNet [50], TieredImageNet [37], and CIFAR-FS [6]. The experiment protocols and implementation details largely follow MAML [14] and Reptile [32]. Please refer to supplementary materials for more details.

Table 1. Our approach applied to MAML and Reptile for five-way few-shot classification on MiniImageNet (Accuracy $\pm$ 95% confidence interval over 2000 runs)

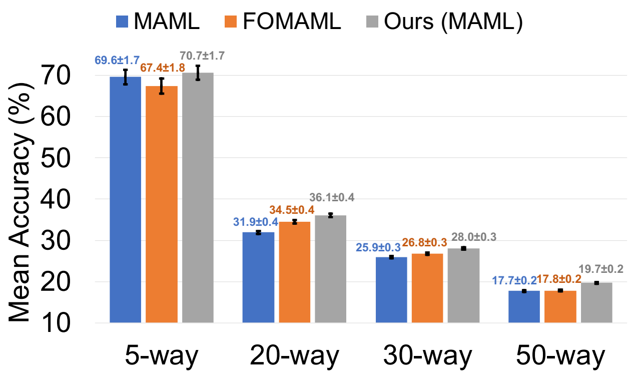
Method	MiniImageNet	
	1-shot	5-shot
MAML [14]	48.70 ± 1.84	63.11 ± 0.92
"Lazy" MAML ($b = 1$)	48.26 ± 1.78	64.13 ± 1.90
"Lazy" MAML ($b = 3$)	48.17 ± 1.84	63.73 ± 1.10
Reptile [32]	49.97 ± 0.32	65.99 ± 0.58
"Lazy" Reptile ($b = 1$)	51.50 ± 1.00	67.22 ± 0.97
"Lazy" Reptile ($b = 3$)	52.67 ± 1.01	68.77 ± 0.98

Our approach permits long-horizon inner updates and involves a convex hull of the last few checkpoints. In Table 1, we first experiment with the last b=3 and b=1 checkpoints. We test them with two representative meta-learning algorithms: MAML (cf. Algorithm 1) and Reptile (replacing Line 13 (L13) in Algorithm 1 with $\theta \leftarrow \theta - \beta\sum_i(\theta - \phi_i(\theta))$). The intervals are 0.05 in the grid search (L10), and the search range for the learning rate γ is between 0.75 and 0.95.

Table 1 shows that there is no significant difference between $b = 3$ and $b = 1$, so we shall employ $b = 1$ for the remaining experiments. Moreover, the "lazy" variation improves the vanilla Reptile, but not MAML, probably because the five-way one/five-shot learning is too simple for MAML to take advantage of the long-horizon inner updates. We next study many-way few-shot learning tasks, which are arguably more complex.

5.1.1 MAML vs. "Lazy" MAML for many-way few-shot learning

We switch to the TieredImageNet dataset since there are only 20 classes in MiniImageNet's meta-test set. The left panel of Figure 2 shows the results of MAML, FOMAML and "Lazy" MAML for N-way-five-shot learning, where N varies in



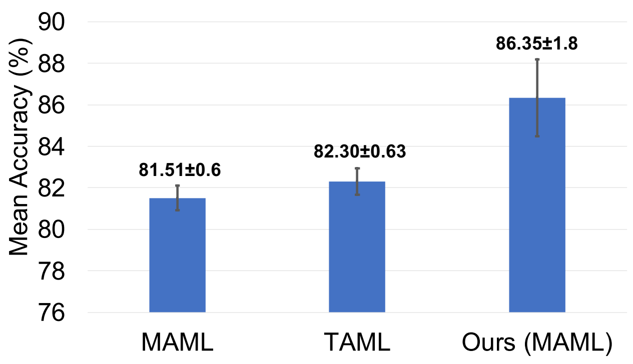


Figure 2. Left: Mean Accuracy (%) for N-way-five-shot classification on TieredImageNet. Right: Mean Accuracy (%) for 20-way-one-shot non-i.i.d. [53] classification tasks on Omniglot.

$\{5, 20, 30, 50\}$, and the student runs for $k = 10, 15, 20, 20$ inner steps, respectively. The "lazy" variation is on par with MAML for the five-way classification, and it outperforms MAML, and FOMAML for 20-way, 30-way, and 50-way five-shot classifications. This trend indicates that the many-way few-shot learning problems desire more inner updates to the task-specific models, amplifying the benefit of the "lazy" teacher.

5.1.2 Many-shot Classification

The Figure 3 shows the results of MAML and "Lazy" MAML for five-way-K-shot learning on MiniImageNet. We vary K in $\{1, 5, 20, 50\}$ and let the student run for $k = 10, 15, 15, 20$ steps, respectively. Under the 1-shot and 5-shot settings, our approach is comparable to MAML, but it significantly outperforms MAML for 20-shot and 50-shot classifications. This trend indicates that more training data desires more steps of exploration for a task-specific model and hence magnifies the benefit of our teacher-student scheme introduced to MAML.

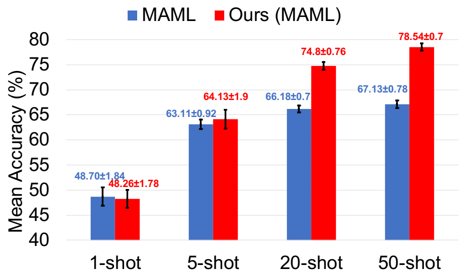


Figure 3. Mean Accuracy (%) for five-way K-shot classification on MiniImageNet.

5.1.3 "Lazy" MAML is less prone to over-fitting by memorization than MAML

The right panel of Figure 2 shows some 20-way-one-shot classification results on Omniglot when we learn from non-i.id. tasks, i.e., by maintaining a global order of all training classes. This global order creates a shortcut for meta-learning methods; they may memorize the order from the meta-training tasks and fail to generalize to meta-test tasks [53]. We can see that the "lazy" teacher boosts MAML by a large margin and outperforms TAML [21], indicating that it is less prone to over-fitting by memorization. A plausible reason is that the $k = 15$ steps taken by the exploratory student make it harder to memorize than the one-step update in MAML or TAML.

5.1.4 Five-way-few-shot learning

We compare our approach with state-of-the-art meta-learning methods for five-way few-shot learning problems on four datasets. The results are shown in Tables 2 for MiniImageNet and TieredImageNet . For our own approach, we study both the MAML-style update to the meta-model (ours (MAML), L13 in Algorithm 1) and the Reptile-style [32] update (ours (Reptile), replacing Line 13 (L13) in Algorithm 1 with $\theta \leftarrow \theta - \beta \sum_i (\theta - \phi_i(\theta))$) for MiniImageNet and TieredImageNet. Batch normalization with test data yields about 2% improvement over the normalization with the training data only, and we report the results of both scenarios.

It can be seen that our results are better than or comparable with those of the competing methods. In general, the improvements by our teacher-student scheme are more significant on 5-shot settings than on 1-shot settings, verifying the trend in Section 5.1.1 that more training data can better leverage the exploratory student in our method. Besides, ours (Reptile) outperforms ours (MAML) probably for two reasons. One is that ours (Reptile) uses more than k shots of training examples per class for a k-shot learning problem during meta-training, following the experiment setup of Reptile [32]. The other is that the second-order gradients in ours (MAML) make the training procedure less stable than Reptile. We hypothesize that a many-shot setting would be less sensitive to both factors. Indeed, we verified this hypothesis by another five-way-50-shot learning experiment with ours (Reptile), which yields $76.17 \pm 0.32\%$ on MiniImageNet and is lower than 78.54 ± 0.70 by ours (MAML).

Table 2. Five-way few-shot classification accuracies (%) on MiniImageNet and TieredImageNet. The ± shows 95% confidence intervals computed over 2000 tasks.

Method	BN w/	Mini-ImageNet		TieredImageNet	
	Test	1-shot	5-shot	1-shot	5-shot
MAML [14]	✗	46.21 ± 1.76	61.12 ± 1.01	49.60 ± 1.83	66.58 ± 1.78
MAML [14]	✓	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	69.60 ± 1.73
Meta-Curvature [34]	✓	48.83 ± 1.80	62.63 ± 0.93	50.30 ± 1.99	66.14 ± 0.95
iMAML [35]	✓	49.30 ± 1.88	63.47 ± 0.90	51.51 ± 1.80	69.92 ± 1.70
Ours (MAML)	✓	48.26 ± 1.78	64.13 ± 1.90	51.03 ± 1.70	70.67 ± 1.72
FOMAML [14]	✗	45.53 ± 1.58	61.02 ± 1.12	48.01 ± 1.74	64.07 ± 1.72
Reptile [32]	✗	47.07 ± 0.26	62.74 ± 0.37	49.12 ± 0.43	65.99 ± 0.42
Meta-MinibatchProx [56]	✗	47.81 ± 1.00	63.18 ± 1.00	49.97 ± 0.93	66.60 ± 0.91
Ours (Reptile)	✗	48.14 ± 0.94	64.64 ± 0.92	51.15 ± 0.95	68.84 ± 0.90
FOMAML [14]	✓	48.07 ± 1.75	63.15 ± 0.91	50.12 ± 1.82	67.43 ± 1.80
Reptile [32]	✓	49.97 ± 0.32	65.99 ± 0.58	51.34 ± 0.4	68.73 ± 0.40
Meta-MinibatchProx [56]	✓	50.08 ± 1.00	66.28 ± 0.98	53.71 ± 1.04	69.78 ± 0.95
Ours (Reptile)	✓	$\mathbf{51.50} \pm \mathbf{1.00}$	$\mathbf{67.22} \pm \mathbf{0.97}$	$\mathbf{54.41} \pm \mathbf{1.00}$	$\mathbf{72.21} \pm \mathbf{0.94}$

Table 3. Five-way few-shot classification accuracies (%) on Omniglot and CIFAR-FS. The ± shows 95% confidence intervals computed over 1000 tasks.

Method	BN w/	Omniglot		CIFAR-FS	
	Test	1-shot	5-shot	1-shot	5-shot
MAML [14]	✓	98.70 ± 0.40	$\mathbf{99.90} \pm \mathbf{0.10}$	56.50 ± 1.90	70.50 ± 0.90
iMAML [35]	✓	$\mathbf{99.16} \pm \mathbf{0.35}$	99.67 ± 0.12	-	-
Reptile [32]	✗	95.39 ± 0.09	98.90 ± 0.10	53.12 ± 1.34	69.40 ± 1.30
Ours (Reptile)	✗	95.44 ± 0.57	98.92 ± 0.29	54.64 ± 1.30	70.56 ± 1.20
FOMAML [14]	✓	98.30 ± 0.50	99.20 ± 0.20	55.6 ± 1.88	69.52 ± 0.91
Reptile [32]	✓	97.68 ± 0.04	99.48 ± 0.06	57.50 ± 0.45	71.88 ± 0.42
Ours (Reptile)	✓	98.20 ± 0.38	99.70 ± 0.16	$\mathbf{59.36} \pm \mathbf{1.44}$	$\mathbf{74.90} \pm \mathbf{1.28}$

The results on Omniglot and CIFAR-FS are reported in Table 3. We only report ours (Reptile) due to its low computation cost. It can be seen that our results are better than or comparable with those of the competing methods.

In Appendix A, we present more results of the few-shot learning. Section A.5 investigates the proposed "lazy" approach with Reptile-style update for N-way-five-shot learning on TieredImageNet. Section A.4 further compares MAML and "lazy" MAML by their computation memory costs.

5.2. Long-Tailed Classification

Following the experiment setup in [11] and [20], we use the CIFAR-LT-100 dataset [11] to compare our Algorithm 2 with several long-tailed recognition methods. [11] created multiple long-tailed datasets by removing training examples from CIFAR-100 [23] according to different power law distributions. In each version, we compute an imbalance factor as the ratio between the sizes of the head class and the tail class. We run $k = 5$ steps in the innermost loop of Algorithm 2.

Table 4 shows the test errors (%) under different imbalance factors. We can see that our teacher-student scheme boosts the original two-component weighting approach [20] under all the imbalance factors. The results are especially interesting in that Algorithm 2 is not exactly a meta-learning method, though it shares the same framework as the gradient-based meta-learning due to the two nested optimization loops. Besides, compared with the other competing methods, our results establish a new state of the art for the long-tailed object recognition.

5.3. Meta-Attack

We evaluate the "lazy" meta-attack on MNIST [25] and CIFAR-10 [23]. We follow [13] for the experiment setup and all training details, including the network architectures used to generate gradients for input images, the attack models, meta-attack models, and evaluation metrics for both the

Table 4. Test top-1 errors (%) of ResNet-32 on CIFAR-LT-100 under different imbalance settings.

Method ↓ / Imbalance factor →	200	100	50	20
Standard cross-entropy training	65.16	61.68	56.15	48.86
Class-balanced cross-entropy training [11]	64.30	61.44	55.45	48.47
Class-balanced fine-tuning [12]	61.78	58.17	53.60	47.89
Learning to reweight [38]	67.00	61.10	56.83	49.25
Meta-weight [42]	63.38	58.39	54.34	46.96
Two-component weighting [20]	60.69	56.65	51.47	44.38
Lazy two-component weighting (ours)	**58.67**	**53.46**	**48.24**	**43.68**

Table 5. Untargeted adversarial attack results on MNIST and CIFAR10. We achieve comparable success rates and average ℓ_2 distortions with other methods by using a smaller number of queries.

Dataset / Target model	Method	Success Rate	Avg. ℓ_2	Avg. Queries
MNIST / Net4	Zoo [8]	1.00	1.61	21,760
	Decision boundary [1]	1.00	1.85	13,630
	Opt-attack [9]	1.00	1.85	12,925
	AutoZoom [46],	1.00	1.86	2,412
	Bandits [19]	0.73	1.99	3,771
	Meta-attack [13]	1.00	1.77	749
	Lazy meta-attack (ours)	1.00	1.65	**566**
CIFAR10 / Resnet18	Zoo [8]	1.00	0.30	8,192
	Decision boundary [1]	1.00	0.30	17,010
	Opt-attack [9]	1.00	0.33	20,407
	AutoZoom [46]	1.00	0.28	3,112
	Bandits [19]	0.91	0.33	4,491
	FW-black [7]	1.00	0.43	5,021
	Meta-attack [13]	0.94	0.34	1,583
	Lazy meta-attack (ours)	0.98	0.45	**1,061**

datasets, to name a few. The learning rates in the inner and outer loops are both 0.01. We let the student run $k = 8$ and $k = 10$ steps in the innermost loop for MNIST and CIFAR-10, respectively. Table 5 shows the results of untargeted attack, namely, the attack is considered successful once it alters the recognition network's prediction to any incorrect class. **Appendix B** includes the results of targeted attack. In addition to the original meta-attack [13], Table 5 also presents several existing blackbox attack methods for comparison. We can see that meta-attack and our "lazy" meta-attack yield about the same success rates as the other blackbox attacks. The second-to-the-right column is about the average ℓ_2 distortion an attacker makes to an input, the lower the better. The rightmost column is about the number of queries an attacker makes into the recognition network, the lower the better. The "lazy" meta-attack is able to achieve comparable success rates and ℓ_2 distortion rates with the other methods yet by using a smaller number of queries. Both meta-attack and its "lazy" version significantly outperform the other methods in terms of the query efficiency, indicating the generalization capability of the meta-attack model from known whitebox neural networks to unknown blackbox networks.

6. Conclusion

We propose a teacher-student scheme for the gradient-based meta-learning algorithms to allow them run more steps of inner updates to task-specific models while being immune to the risk of vanishing or exploding gradients. The student explores the tasks-specific model's feasible space up to many steps, and the "lazy" teacher takes a one-step "leap" towards the region explored by the student. As a result, the teacher defines a lightweight computation graph and yet it takes advantage of the adequately explored checkpoints by the student. This approach is generic; we apply it to different problems, include few-shot learning, long-tail recognition, and meta-attack and various meta-learning methods. Experiments verify the benefit of long-horizon inner updates in gradient-based meta-learning.

References

[1] Wieland Brendel *, Jonas Rauber *, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018. 8

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016. 1, 2

[3] Harkirat Singh Behl, Atilim Günes Baydin, Ran Gal, Philip H. S. Torr, and Vibhav Vineet. Autosimulate: (quickly) learning synthetic data generation. abs/2008.08424, 2020. 2

[4] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Optimality in Biological and Artificial Networks*. 1995. 2

[5] Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, 1991. 2

[6] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *CoRR*, 2018. 2, 5

[7] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. A frank-wolfe framework for efficient and effective adversarial attacks, 2018. 8

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, 2017. 8

[9] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jin-Feng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019. 8

[10] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007. 1

[11] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 5, 7, 8

[12] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[13] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. *arXiv preprint arXiv:1906.02398*, 2019. 1, 2, 5, 7, 8

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017. 1, 2, 4, 5, 7

[15] Sebastian Flennerhag, Pablo G Moreno, Neil D Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. *arXiv preprint arXiv:1812.01054*, 2018. 1

[16] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019. 3

[17] Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018. 2

[18] Jonathan Gordon, Wessel Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. In *ICLR*, 2020. 2

[19] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019. 8

[20] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective, 2020. 2, 5, 7, 8

[21] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 6

[22] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015. 2

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 7

[24] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011. 5

[25] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34, 1998. 7

[26] Ke Li and Jitendra Malik. Learning to optimize, 2016. 2

[27] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017. 2, 3

[28] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. abs/1911.02590, 2019. 2

[29] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-dickstein. Learned optimizers that outperform on wall-clock and validation loss, 2019. 2

[30] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017. 1, 2

[31] D. K. Naik and R. J. Mammone. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 1992. 2

[32] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 1, 2, 5, 6, 7

[33] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning, 2018. 2

[34] Eunbyung Park and Junier B Oliva. Meta-curvature. In *Advances in Neural Information Processing Systems*, pages 3309–3319, 2019. 2, 3, 7

[35] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019. 1, 2, 3, 4, 7

[36] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 1, 2

[37] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 5

[38] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018. 1, 8

[39] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018. 2

[40] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, 1987. 2

[41] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. 1992. 2

[42] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting, 2019. 8

[43] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017. 2

[44] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning, 2017. 2

[45] Sebastian Thrun and Lorien Pratt. *Learning to Learn: Introduction and Overview*. Springer US, 1998. 2

[46] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks, 2018. 8

[47] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 5

[48] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002. 1

[49] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 1, 2

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016. 5

[51] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2–a large-scale benchmark for instance-level recognition and retrieval. *arXiv preprint arXiv:2004.01804*, 2020. 5

[52] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize, 2017. 2

[53] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019. 3, 4, 6

[54] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020. 2

[55] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9593–9604, 2019. 1, 4

[56] Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, pages 1532–1542, 2019. 2, 7