Large-Scale Meta-Learning with Continual Trajectory Shifting

JaeWoong Shin [*1] **Hae Beom Lee** [*1] **Boqing Gong** [2 1] **Sung Ju Hwang** [1 3]

Abstract

Meta-learning of shared initialization parameters has shown to be highly effective in solving few-shot learning tasks. However, extending the framework to many-shot scenarios, which may further enhance its practicality, has been relatively overlooked due to the technical difficulties of meta-learning over long chains of inner-gradient steps. In this paper, we first show that allowing the meta-learners to take a larger number of inner gradient steps better captures the structure of heterogeneous and large-scale task distributions, thus results in obtaining better initialization points. Further, in order to increase the frequency of meta-updates even with the excessively long inner-optimization trajectories, we propose to estimate the *required shift* of the task-specific parameters with respect to the change of the initialization parameters. By doing so, we can arbitrarily increase the frequency of meta-updates and thus greatly improve the meta-level convergence as well as the quality of the learned initializations. We validate our method on a heterogeneous set of large-scale tasks and show that the algorithm largely outperforms the previous first-order meta-learning methods in terms of both generalization performance and convergence, as well as multi-task learning and fine-tuning baselines.

1. Introduction

Meta-learning (Schmidhuber, 1987; Thrun & Pratt, 1998) is a framework for learning a learning process itself by extracting common knowledge over a task distribution. As this meta-knowledge allows task learners to adapt to newly given tasks in a sample efficient manner, meta-learning has frequently been used for solving few-shot learning problems where each of the task learners is given only a few training examples (Lake et al., 2015; Vinyals et al., 2016; Santoro et al., 2016; Snell et al., 2017; Finn et al., 2017). While there exists a vast literature on meta-learning methods that tackle few-shot learning, one of the most popular approaches is the optimization-based method such as Model Agnostic Meta-Learning (MAML) (Finn et al., 2017), which aims to improve the generalization ability of few-shot learners by learning good initialization parameters, from which the model can rapidly adapt to novel tasks within only a few gradient steps.

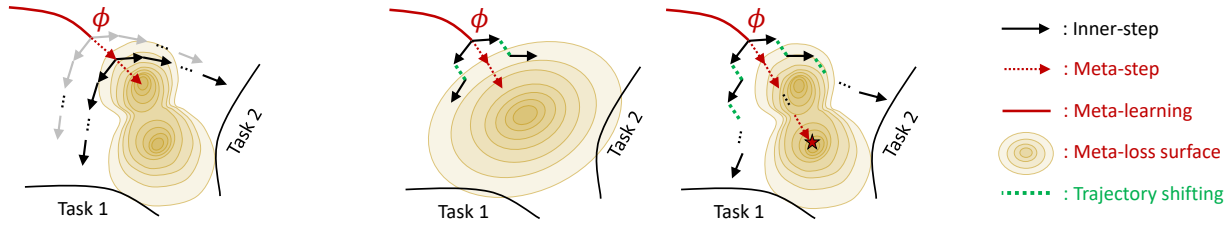
Then, a natural question is if the same meta-learning strategy is applicable to tasks with a larger number of examples, for instance STL10 (Coates et al., 2011) and Stanford Cars (Krause et al., 2013). It is well known that such standard learning tasks with a large number of training examples also benefit from good initialization parameters for better convergence and generalization, when compared with random initializations (Kornblith et al., 2019). A prevalent approach to enhance generalization for large-scale tasks is to pre-train the model with a large dataset such as ImageNet (Russakovsky et al., 2015), and further finetune the pretrained model parameters with the target dataset. This demonstrates that knowledge transfer is also highly beneficial for tasks with larger training sets.

However, the meta-learning of shared initialization parameters for many-shot learning problems has not received much attention. One reason may be that ImageNet pretraining has been practically effective for most of the standard object classification tasks or other computer vision problems. However, Kornblith et al. (2019) empirically show that ImageNet pretraining may not obtain meaningful performance gains on fine-grained classification tasks. This is because fine-grained classification tasks may require features that are more domain-specific or local for the discrimination of highly similar visual classes, for which the ImageNet features learned for general object classification may be ineffective. In other words, pretraining neural networks only on a single large-scale dataset will not sufficiently cover the heterogeneity of datasets and tasks that the model needs to handle at inference time. One of the effective ways to handle such heterogeneity is to train the model via meta-learning over a heterogeneous task distribution.

There have been several attempts to apply meta-learning to

[*]Equal contribution [1]Graduate School of AI, KAIST, South Korea [2]Google, LA [3]AITRICS, South Korea. Correspondence to: Sung Ju Hwang <sjhwang82@kaist.ac.kr>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).



(a) Previous meta-learning (b) Meta-learning with Continual Trajectory Shifting

Figure 1. **Concepts of large-scale meta-learning,** whose inner learning trajectories have to be long enough to fit each large-scale individual task. (a) Previous meta-learning which is vulnerable to bad meta-level local minima and waits for the excessively long inner learning trajectories between two meta-updates. (b) Our method that performs frequent meta-updates by interleaving them with the inner-learning trajectories, plus continual trajectory shiftings, and is less prone to bad local minima by gradually growing the trajectory length k.

large-scale settings, where the training set consists of a large number of instances (Nichol et al., 2018; Flennerhag et al., 2019; 2020). While these methods alleviate the computational cost for large-scale meta-learning, they are not truly scalable to tasks that are considered in conventional learning scenarios. The main difficulty of meta-learning with shared initialization for large-scale tasks, is that they require a large number of gradient steps to converge, since otherwise the meta-learner would suffer from the short horizon bias problem (Wu et al., 2018). Note that the computational cost of a single meta update increases linearly with respect to the number of inner gradient steps. Therefore, a single meta-gradient update for gradient-based meta-learning algorithms (e.g. MAML, Reptile (Nichol et al., 2018)) would require probably thousands of subsequent inner-gradient steps for the given tasks (See Figure 1(a)).

The key to this challenging problem of large-scale meta-learning, is how to perform frequent meta-updates for meta-convergence while allowing the learning trajectories of inner optimization problems to become sufficiently long. However, due to the strong dependency between the initialization parameters and learning trajectories for each task, naively updating the initialization parameters without correcting the learning trajectories may be suboptimal. We tackle this by proposing a novel idea: estimating the corresponding change of the task-specific parameters with respect to the change of the initialization point. If we can estimate such an update direction with a reasonable accuracy, then we will be able to arbitrarily increase the frequency of meta updates with the corresponding shifting for the task learning trajectories (See Figure 1(b)).

In this paper, we show that first-order Taylor expansion together with the first-order approximation of Jacobian over the learning trajectories (Finn et al., 2017; Nichol et al., 2018; Flennerhag et al., 2019) yields a surprisingly simple but effective shifting rule, shifting the entire learning trajectories to the direction and the amount for each meta update. By doing so, we can perform more frequent meta-updates compared to existing optimization-based meta-learning algorithms, while preserving the connection between the initialization point and the task learning trajectories up to the approximation error. Our method enjoys significantly faster convergence over existing first-order meta-learning algorithms, and the learned initialization by our method leads to better generalization performance as well.

We validate our method by meta-training over a heterogeneous set of standard, many-shot learning tasks such as Aircraft (Maji et al., 2013), CUB (Wah et al., 2011), and Fashion MNIST (Xiao et al., 2017b) that require at least a thousand of gradient steps for an accurate estimation of meta gradients. We then meta-test the learned initial model parameters by finetuning with a similar set of diverse datasets such as Stanford Cars (Krause et al., 2013) and STL10 (Coates et al., 2011).

We summarize our contributions as follows:

- We show that large-scale meta-learning requires substantially a larger number of inner gradient steps than what are reqruied for few-shot learning.
- We show that gradually extending the length of inner-learning trajectories lowers the risk of converging to poor meta-level local optima.
- To this end, we propose a novel and an efficient algorithm for large-scale meta-learning that frequently performs meta-optimization even with excessively long inner-learning trajectories.
- We verify our algorithm on a heterogeneous set of tasks, on which it achieves significant improvements over existing meta-learning algorithms in terms of meta-convergence and generalization performance.

2. Related Work

Meta-learning Meta-learning (Schmidhuber, 1987; Thrun & Pratt, 1998) aims to learn how to learn on novel tasks, without overfitting to seen tasks. Meta-learning is usually done by assuming a task distribution (Vinyals et al., 2016; Ravi & Larochelle, 2017) from which tasks

are sampled and a meta-learner which solves them by extracting common meta-knowledge among the given tasks. Many recent works have demonstrated the effectiveness of such a strategy in few-shot learning settings where the learner should adapt to novel tasks with few training samples for each task (Lee & Choi, 2018; Mishra et al., 2018; Rusu et al., 2019; Liu et al., 2019; Lee et al., 2019). A popular approach to meta-learning is to learn a common metric space over a task distribution (Vinyals et al., 2016; Snell et al., 2017; Yang et al., 2017; Oreshkin et al., 2018), that can be used for the prediction for a novel task. For classification, the space could be learned to map each instance (query instance) closer to either another instance from the same class, or the prototype of the class. However, in many-shot scenario we target, it is not trivial to fully exploit the task information without taking a sufficient number of gradient steps. Therefore, we focus more on optimization-based meta-learning methods (Finn et al., 2017) that are model-agnostic, whose goal is to learn a shared initialization parameters from which each of the target tasks can adapt after taking some amount of gradient steps. The shared initialization parameters are meta-learned by backpropagating through the learning trajectories.

Efficient Meta-learning Early optimization-based meta-learning algorithms usually require computing the second-order derivatives in order to obtain the meta-gradients (Finn et al., 2017). Yet, due to the heavy computational cost in computing them, many prior works propose to use a first order approximation to obtain the meta-gradient (Finn et al., 2017; Nichol et al., 2018; Flennerhag et al., 2019), based on the empirical observation that given a sufficiently small stepsize for the inner-gradient steps, curvature information around a local region, i.e. Hessian, can be safely ignored. Other ways to efficiently compute meta-gradients include Rajeswaran et al. (2019b) and Song et al. (2020). However, despite their computational efficiency, none of the existing gradient-based meta-learning methods are truly scalable to large-scale meta-learning that involves a large number of inner-gradient steps, since this will slow down the meta-convergence as the meta-update frequency will decrease as the trajectory lengths for the inner-gradient steps increase. In this paper, we propose a novel algorithm for effectively *increase the frequency of meta-updates* while preserving the connection between the shared initialization and task learning trajectories.

Transfer and multi-task learning It is possible to use transfer learning as an alternative of meta-learning for large-scale sceanarios to avoid the excessive computational cost associated with it. Specifically, finetuning from a pretrained network on a large dataset such as ImageNet (Russakovsky et al., 2015) is a simple yet an effective method that is known to perform well in practice. Dhillon et al. (2020) recently showed that a simple variant of finetuning strategy in transductive setting outperforms most of the current sophisticated meta-learning algorithms. Yet, finetuning strategies are based on a strong assumption that a feature extractor learned from a single big dataset is beneficial in boosting the generalization performance of the target datasets, which may not hold when the target tasks have largely different distributions from the source task (Kornblith et al., 2019). While more sophisticated transfer learning or domain adaptation methods can tackle this problem (Jang et al., 2019), it remains an important question whether we can learn intialization parameters that can generalize well even to tasks with large distributional shifts, such as fine-grained classification. Also, although there exist abundant datasets that may contribute to meta-knowledge, it is not trivial to decide which datasets to use for pretraining. Multi-task learning (MTL) is an effective way to achieve generalization across tasks. However a naive MTL approach with joint training of multiple tasks is vulnerable to negative transfer under a heterogeneous task distribution, degrading the quality of the learned feature extractor that will be used for finetuning the target tasks. Optimization based meta-learning can be a natural solution to tackle the negative transfer problem, since it finds an intialization point that can lead to optimal solutions for heterogeneous tasks, rather than trying to find a solution that are jointly optimal for all tasks.

3. Approach

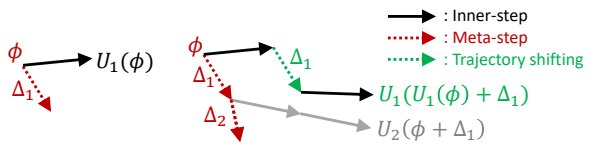
We start by describing our problem setup. Our goal is to learn shared initialization parameters ϕ that can lead to good solutions for diverse tasks after task-specific adaptation. Suppose that we have T tasks (or datasets) $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}$ that will be used for meta-training, and each of the tasks has a large number of training examples. We further assume that there exist substantial distributional discrepancies among the tasks. In this large-scale heterogeneous meta-learning setup, it is crucial for the task-specific model parameters $\theta^{(t)}$ to fully adapt to a given task $\mathcal{D}^{(t)}$ by taking a sufficient number (K) of gradient steps (e.g. $K = 1,000$ steps) from the shared initialization ϕ. We also let the inner-optimization processes repeat M times in order to make the initialization parameters ϕ fully converge. As a result, we expect the meta-learned ϕ to work well on new tasks or datasets.

3.1. Limitations of previous methods

Naturally, backpropagating through a learning process requires to compute the second-order derivatives such as Hessians (Finn et al., 2017). Due to its heavy computational cost, we focus on the first-order meta-learning algorithms such as FOMAML (Finn et al., 2017), Reptile (Nichol et al., 2018), or Leap (Flennerhag et al., 2019) which are more suitable for large-scale meta-learning. We sketch the method in Algorithm 1. For instance, the meta-gradient of FOMAML is $\mathsf{MetaGrad}(\phi; \theta_K^{(t)}) = \nabla_\theta \mathcal{L}_K^{(t)}|_{\theta=\theta_K^{(t)}}$ where $\mathcal{L}_k^{(t)}$

Algorithm 1 Previous meta-learning algorithms

Input: A set of tasks $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}$
Input: Inner-learning rate α, meta-learning rate β
Output: Meta-learned initialization ϕ
Randomly initialize ϕ.
for $m = 1$ to M **do** ▷ Repeating inner-opt. processes
  for $t = 1$ to T **do**
    $\theta_0^{(t)} \leftarrow \phi$ ▷ Resetting each task learner
    for $k = 1$ to K **do** ▷ Inner-optimization
      $\theta_k^{(t)} \leftarrow \theta_{k-1}^{(t)} - \alpha \nabla_\theta \mathcal{L}_k^{(t)}|_{\theta=\theta_{k-1}^{(t)}}$
    end for
  end for
  $\phi \leftarrow \phi - \beta \frac{1}{T} \sum_{t=1}^{T} \mathsf{MetaGrad}(\phi; \theta_K^{(t)})$ ▷ Meta-update
end for



(a) $k = 1$ (b) $k = 2$

Figure 2. Illustration of the proposed continual trajectory shifting.

denotes loss of task t at step k, and the Reptile gradient is $\mathsf{MetaGrad}(\phi; \theta_K^{(t)}) = \phi - \theta_K^{(t)}$. They consist of only the first-order terms and thus are cheaper to compute than meta-gradients with the second-order derivatives.

However, the previous meta-learning methods have to reinitiate each inner-optimization process right after every single meta-update, because the inner-learning process should remain consistent with the updated initialization point and start a new learning trajectory from there (see Figure 1). This makes the large-scale meta-learning inefficient. We see from Algorithm 1 that the interval between the current meta-update and the previous one (in red) linearly increases with respect to K, the total length of every inner-optimization trajectory. For example, if we have $T = 10$ tasks and $K = 1,000$, then we need to take $10 \times 1,000 = 10,000$ gradient steps before making a single meta-gradient step, which quickly becomes computationally expensive no matter how efficient the approximation is for computing each meta gradient. For meta-learning, we often need to perform a large number of meta-updates to ensure the convergence of the meta-model ϕ, but having long trajectories of inner-gradient steps for large-scale tasks would prevent us from making sufficient meta-updates within a computing budget.

3.2. Continual trajectory shifting

Our key idea is to interleave the meta-updates with the inner-optimization processes to reduce the long wait between two adjacent meta-updates. This is made possible by continual trajectory shifting described as follows. We first introduce some notations. Let $U_k(\phi)$ denote a function that takes the initialization ϕ as input and outputs θ_k, the k-th step parameters for solving a task. Here we drop

Algorithm 2 Meta-learning with continual shifting

Input: A set of tasks $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(T)}$
Input: Inner-learning rate α, meta-learning rate β
Output: Meta-learned initialization ϕ
Randomly initialize ϕ
for $m = 1$ to M **do** ▷ Reapating inner-opt. processes
  $\theta_0^{(t)} \leftarrow \phi$ for $t = 1, \ldots, T$ ▷ Resetting task learners
  for $k = 1$ to K **do** ▷ Inner-opt. for all tasks
    for $t = 1$ to T **do** ▷ Parallel for loop
      $\theta_k^{(t)} \leftarrow \theta_{k-1}^{(t)} - \alpha \nabla_\theta \mathcal{L}_k^{(t)}|_{\theta=\theta_{k-1}^{(t)}}$
    end for
    $\Delta_k \leftarrow -\beta \frac{1}{T} \sum_{t=1}^{T} \mathsf{MetaGrad}(\phi; \theta_k^{(t)})$
    $\phi \leftarrow \phi + \Delta_k$ ▷ Meta-update
    $\theta_k^{(t)} \leftarrow \theta_k^{(t)} + \Delta_k$ for $t = 1, \ldots, T$ ▷ Shifting
  end for
end for

the task dependency for notational brevity. For instance, if we use vanilla stochastic gradient descent, then we have $U_k(\phi) := \phi - \alpha \sum_{i=0}^{k-1} \nabla_\theta \mathcal{L}_i|_{\theta=\theta_i}$ where $\theta_0 := \phi$ and α is the inner learning rate. [1] Denote by $\Delta_1, \Delta_2, \ldots$ the series of meta-updates induced by all tasks, such that the shared initialization evolves as $\phi, \phi + \Delta_1, \phi + \Delta_1 + \Delta_2, \cdots$.

Now we show that *we can perform k meta-updates within k inner-gradient steps*, unlike the previous meta-learning methods. Note that Reptile gradient $\phi - \theta_k$ depends only on the task-specific parameters $\theta_k := U_k(\phi)$. Based on this property[2], we propose to estimate $U_1(\phi), U_2(\phi + \Delta_1), \ldots, U_k(\phi + \Delta_1 + \cdots + \Delta_{k-1})$ from a single inner-optimizatin process, and perform the meta-updates with them for every step up to k. Specifically, in Figure 2(a), we compute the first meta-update Δ_1 based on the single-step task-specific parameters $U_1(\phi)$. Then in the next step $k = 2$ in Figure 2(b), in order to compute the next meta-update Δ_2 w.r.t. the new initialization point $\phi + \Delta_1$, we propose to approximate $U_2(\phi + \Delta_1)$ with $U_1(U_1(\phi) + \Delta_1)$, which we can obtain *without actually taking gradient steps at* $\phi + \Delta_1$. We generalize the approximation as follows:

$$\begin{aligned} &U_k(\phi + \Delta_1 + \cdots + \Delta_{k-1}) \\ &\approx U_1(\cdots U_1(U_1(\phi) + \Delta_1) \cdots + \Delta_{k-1}) \end{aligned} \tag{1}$$

Eq. (1) means that for every inner-step from 1 to k, we continuously shift the task-specific learning trajectory by *the same direction and amount of each meta-update*, thereby allowing the task-learning trajectory to remain consistent with the series of updates for the initialization parameters. See Figure 2(b) and Algorithm 2 for the detailed procedure. We name this method as *Continual Trajectory Shifting*.

One important aspect of our method is that k, the inner-trajectory length used to compute each meta-update, grad-

[1] We do not impose any restrictions on the type of optimizers for $U_k(\phi)$. See **the supplementary file** for more discussion.

[2] Note that we can use any meta-gradients with similar property.

ually increases from 1 to maximum K. In Figure 2(a), we compute Δ_1 with the single-step task learner $U_1(\phi)$, and in Figure 2(b) we compute Δ_2 with $U_1(U_1(\phi)+\Delta_1)$, which is an approximation of the two-step task learner $U_2(\phi+\Delta_1)$. Later we will discuss the effect of gradually increasing k as a meta-level regularizer.

3.3. Approximation error

We next analyze the approximation error in Eq. (1), which is central to our continual trajectory shifting method. We first show $U_k(\phi+\Delta) \approx U_k(\phi)+\Delta$ can be derived by applying two different approximations. The first approximation is Taylor expansion:

$$\begin{aligned} &U_k(\phi+\Delta) \\ &= U_k(\phi) + \frac{\partial U_k(\phi)}{\partial \phi}\Delta + \frac{1}{2}\Delta^\top \frac{\partial^2 U_k(\phi)}{\partial \phi^2}\Delta + \cdots \\ &= U_k(\phi) + \frac{\partial U_k(\phi)}{\partial \phi}\Delta + O(\beta^2) \end{aligned} \tag{2}$$

$O(\beta^2)$ is because $\Delta = -\beta \cdot \mathsf{MetaGrad}(\phi;\theta_k) = O(\beta)$. Therefore, the first order Taylor approximation with Eq. (2) is reasonable if $\beta > 0$ is sufficiently small.

Secondly, we apply the Jacobian approximation frequently used by the first-order meta-learning algorithms (Finn et al., 2017; Nichol et al., 2018; Flennerhag et al., 2019):

$$\begin{aligned} \frac{\partial U_k(\phi)}{\partial \phi} &= \frac{\partial U_k(\phi)}{\partial U_{k-1}(\phi)} \cdots \frac{\partial U_1(\phi)}{\partial \phi} = \prod_{i=0}^{k-1}(I-\alpha H_i) \\ &= I + O(\alpha h k) \end{aligned} \tag{3}$$

where we let $\theta_0 := \phi$, $\alpha > 0$ is the inner-learning rate, H_i is the Hessian at step i, and h denotes an upper bound of norm of Hessians (e.g. spectral norm). As long as $\alpha h k > 0$ is significantly smaller than 1, we can safely approximate Eq. (3) with the identity matrix I. Applying Eq. (2) and Eq. (3), we have

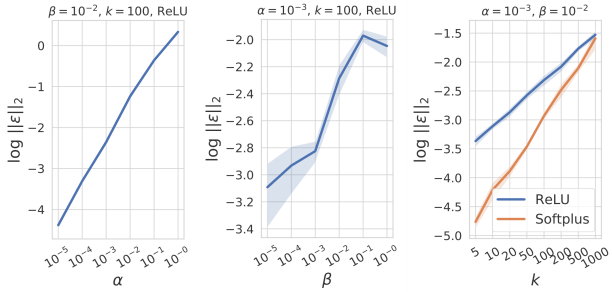
$$U_k(\phi+\Delta) = U_k(\phi) + \Delta + O(\beta\alpha h k + \beta^2). \tag{4}$$

Based on Eq. (4), we can derive the complexity of the approximation error caused by Eq. (1):

$$\begin{aligned} &U_k\left(\phi + \Delta_1 + \cdots + \Delta_{k-1}\right) \\ &= U_1(\cdots U_1(U_1(\phi)+\Delta_1)\cdots + \Delta_{k-1}) \\ &\quad + O(\beta\alpha h k^2 + \beta^2 k). \end{aligned} \tag{5}$$

See **the supplementary file** for the derivation.

Empirical analysis. Then, is the approximation error in Eq. (5) empirically manageable? To answer the question, we define the error $\varepsilon := U_k(\phi+\Delta_1+\cdots+\Delta_{k-1}) - U_1(\cdots U_1(U_1(\phi)+\Delta_1)\cdots+\Delta_{k-1})$ and collect the norm of



(a) α (b) β (c) k, Activation

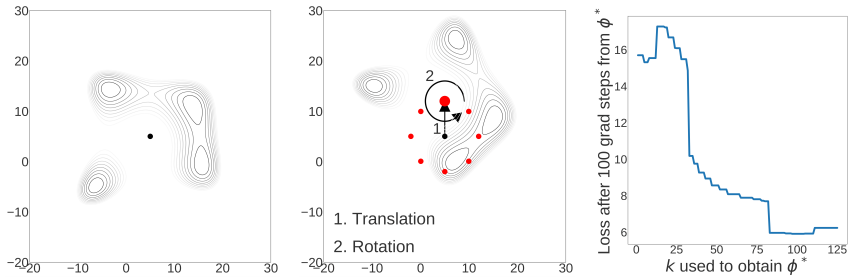
Figure 3. **Approximation error** versus inner-learning rate α, meta-learning rate β, inner-learning trajectory length k, and the type of network activations. We report the mean and 95% confidence intervals over 10 draws of inner-learning trajectories. See **the supplementary file** for the detailed experimental setup.

ε empirically. We see from Figure 3 that the error sharply increases in proportion to α, β, and k. Especially, Figure 3(c) shows the difficulty of managing the error for the large-scale tasks that require a large number of gradient steps. Further, the use of ReLU activations and max-pooling introduces additional errors (Balduzzi et al., 2017a;b). It is because the Taylor expansion assumes infinitely differentiable functions, but ReLU and max-pooling are not differentiable at certain points. See Figure 3(c) which shows that the networks with ReLU activations yield more inaccurate approximations over ones with Softplus activations. In conclusion, for most of the modern convolutional networks and large-scale tasks, we cannot guarantee that the proposed approximation will be highly accurate. However, we empirically found that the method still works very well even with the large approximation error. We provide a plausible interpretation about the results in the next subsection.

3.4. Meta-level curriculum learning with increasing k

Recall from Section 3.2 that our method computes each meta-update with the gradually increasing k, the number of inner-gradient steps. The original motivation of gradually increasing k came from interleaving every inner-optimization step with a meta-update, but we find that it introduces another benefit: a regularization effect. This is because our algorithm could be considered as an instance of curriculum learning at the meta-level. Curriculum learning (Bengio et al., 2009) is a learning strategy where we present training examples from easy to more difficult ones, thereby sequentially controlling the complexity of the loss landscape. It has been empirically shown that the strategy improves the speed of convergence and the quality of local optima.

In our case, the number of inner-gradient steps k used to compute each meta-gradient determines the complexity of the meta-training loss landscape. Starting from $k=1$, the meta-learner first seeks to find a slightly better initialization point ϕ than the old one based on the very limited information about the task learning trajectories due to the

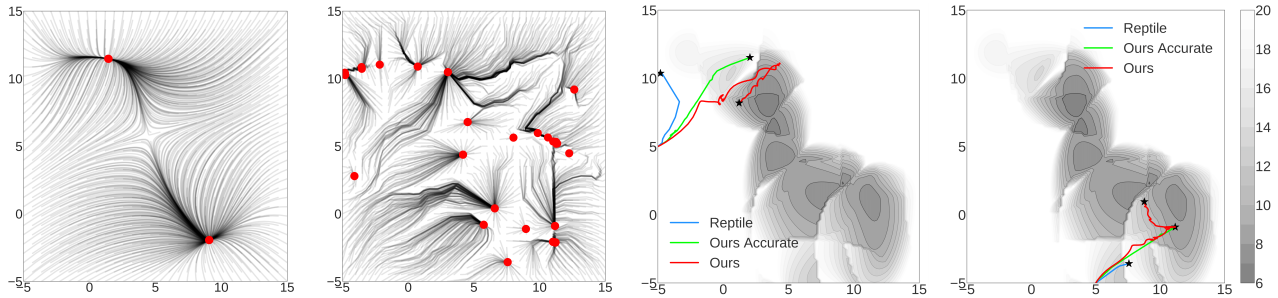


(a) Template function (b) Task 1 loss function $\mathcal{L}^{(1)}$ (c) Short horizon bias

	Reptile	Ours Accurate	Ours
Length of inner-opt. trajectories	$K, .., K$	$1, 2, .., K$	$K, .., K$
# Repetitions of inner-optimizations	MK	MK	M
# Total cumulative meta-updates	MK		
# Total cumulative inner-gradient steps	MK^2	$\frac{MK(K+1)}{2}$	MK

(d) Computational cost

Figure 4. **(a)** Template function used to generate the task loss functions. **(b)** Task 1 loss function $\mathcal{L}^{(1)}$ obtained by applying translation (straight arrow) and random rotation (round arrow). **(c)** Task-average loss after 100 gradient steps from ϕ^* vs. k used to obtain the optimal initialization parameters ϕ^* over the tasks. **(d)** Computational cost in terms of total cumulative number of inner-gradient steps.



(a) $k = 5$ (b) $k = 100$ (c) Starting point: $(-5, 5)$ (d) Starting point: $(5, -5)$

Figure 5. **(a,b) Meta-learning trajectories of Reptile** with the length of inner-optimization fixed as k. We collect the trajectories by initiating them from the various points in the grid of ϕ space. Meta-level local optima are shown by the red dots. **(c,d) Background contour: Task-average loss** after taking 100 gradient steps from each point. The darker the better quality of the initialization point. **Lines: Meta-learning trajectories** of ϕ obtained from the baselines and our algorithm.

short horizon bias (Wu et al., 2018). The bias simplifies the meta-level loss landscape and thus lowers the risk of falling into bad local minima, which is especially beneficial for the early stage of meta-training (See Figure 1(b), left). After alleviating the risk, the meta-learner gradually increases k to have more complex loss surfaces and find more informative local minima with longer horizons (See Figure 1(b), right). It partly explains how our method finds better initialization parameters than those by the previous meta-learning with a fixed length of inner learning trajectories. See Figure 5(a), 5(b), and Section 4.1 for the discussions with real examples.

Curriculum learning and approximation error. The curriculum learning effect also partly explains how our model performs well even with the fairly large approximation error. Note that the risk of bad local optima is significant at the beginning of the meta-training. Since our approximation is relatively accurate when k is small, our model can enjoy the curriculum learning effect even if the approximation error goes up as k increases.

4. Experiments

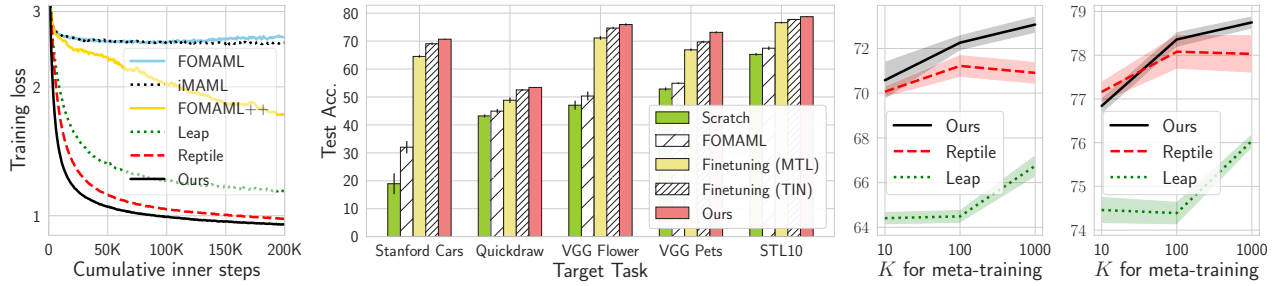
We first examine how and why our method outperforms the baselines with synthetic experiments. We then verify the effectiveness of our method on a set of large-scale heterogeneous tasks, comparing to finetuning baselines and first-order meta-learning algorithms.

4.1. Synthetic experiments

We first experiment with a synthetic task distribution to provide insights on how our algorithm works.

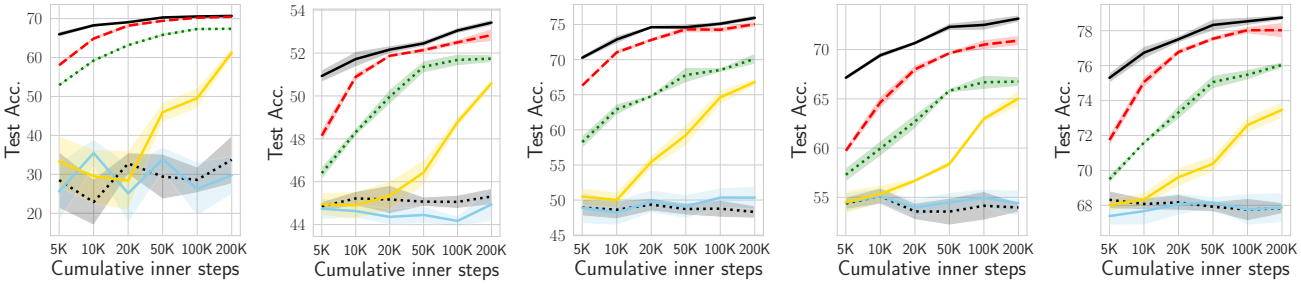
Task distribution. We define a 2D function $f(x, y) = \left\{(x^2 - 10x + y + 9)^2 + (x + y^2 - 10y + 13)^2\right\}/3$ which has four global minima (Figure 4(a)). We shift this template function toward each of the red dots in Figure 4(b), which form a circle centered at $(5, 5)$, and randomly rotate around each dot to generate eight task losses $\mathcal{L}^{(1)} \dots \mathcal{L}^{(8)}$. Although the tasks share the same loss surface shape, they are heterogeneous since the rotations are random. We use all the eight tasks for meta-training to analyze the meta-convergence of different methods.

Baselines. We compare **Ours** with **Reptile** (Nichol et al., 2018) and **Ours Accurate**. Ours Accurate computes each of the meta-updates $\Delta_1, \dots, \Delta_k$ without the approximation errors for the task-specific parameters in Eq. (5). Specifically, Ours Accurate directly computes $U_1(\phi), U_2(\phi + \Delta_1), \dots, U_k(\phi + \Delta_1 + \dots + \Delta_{k-1})$ by repeatedly reinitiating the inner-learning processes after each meta-update, which is computationally far more inefficient than Ours. See Figure 4(d) for the computational cost for each method. Note that we let the methods perform the same number of total meta-updates. **Experimental setup:** We use $\alpha = 0.05$, $\beta = 0.1$, $K = 100$, and $M = 3$. We set the inner-optimizer to SGD with momentun ($\mu = 0.9$). See **the supplementary**



(a) Meta-convergence (b) Meta-testing performance of the baselines (c) VGG Pets (d) STL10

Figure 6. **(a) Meta-training convergence** measured as task-average training loss vs. cumulative inner-gradient steps. **(b) Meta-testing performance** of the baselines. **(c) Meta-testing performance** vs. K used for meta-training.



(a) Stanford Cars (b) Quickdraw (c) VGG Flowers (d) VGG Pets (e) STL10

Figure 7. **Meta-testing performance** to show how efficient each method is in terms of cumulative inner-gradient steps spent for meta-training. We report mean and 95% confidence intervals over 5 runs.

file for more information.

Results and analysis. We make the following observations from the synthetic experiment. Firstly, we should use a large K for meta-training if we want to use a large K for meta-testing. Figure 4(c) demonstrates the existence of the short horizon bias. It shows that the optimal initialization ϕ^* obtained with small K (e.g. $K = 25$) cannot provide good performance even if we take a sufficient number of gradient steps from there (e.g. $K = 100$).

Also, if we use a large K during meta-training, we can allow the meta-learner to avoid bad local optima at the early stage. It is done by gradually increasing the trajectory length k from 1 to maximum K, which is used to compute each meta-gradient. In order to demonstrate this effect, we visualize the meta loss landscape over ϕ with various k in Figure 5(a) and 5(b), by simply collecting the meta-learning trajectories starting from various points in the spatial grid of ϕ. We see from Figure 5(b) that there exist many local minima for large k. It is because the longer horizons make the task-specific parameters to sensitively react to a small change in the initialization ϕ, making the direction of meta-gradient frequently change over the space of ϕ. As a result, comparing the local optima in Figure 5(b) with the map of initialization quality in Figure 5(c), we see that many of the the local optima are of low quality and also attract the meta-learner even from the beginning. See Figure 5(c) and 5(d) that Reptile gets stuck in a bad local minimum, whereas Ours and Ours Accurate can circumvent it. It shows that Ours and Ours Accurate actually make use of the much simpler loss landscape provided by smaller k, effectively lowering the risk of bad local minima. Note that the short horizon bias introduced by smaller k is only temporary as we gradually increase k up to maximum K over the course of inner-optimization processes.

Lastly, Figure 5(c) and 5(d) show that although Ours and Ours Accurate reveal dissimilar meta-learning trajectories in general, the early part of the trajectories are quite similar to each other. It means that the early part of Ours is accurate enough to enjoy the curriculum learning effect. The approximation error would increase as k grows, but the figures show that it does not necessarily lead to worse solutions. It explains why the performances of Ours remain robust to the approximation error.

4.2. Image classification

Next, we verify our method on a realistic large-scale and heterogeneous task distribution with multiple datasets.

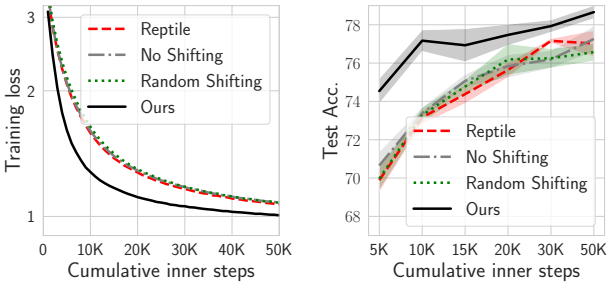
Datasets. We consider large-scale datasets with the number of instances roughly ranging from $5,000$ up to $100,000$. For images larger than 84×84, we resize their width and height into one of $\{28, 32, 64, 84\}$ for faster training. See **the supplementary file** for more information. For **meta-training**, we use 7 datasets: **Tiny ImageNet** (tin), **CIFAR100** (Krizhevsky et al., 2009), **Stanford Dogs** (Khosla et al., 2011), **Aircraft** (Maji et al., 2013), **CUB** (Wah et al., 2011), **Fashion-MNIST** (Xiao et al., 2017a), and **SVHN** (Netzer et al., 2011)). Tiny ImageNet (TIN) and CIFAR100 are benchmark classification datasets of general cat-

egories. We class-wisely divide TIN into two splits. Other datasets include fine-grained classifications that would require a sufficient amount of task-specific adaptations (e.g. Aircraft), and grey-scale images (Fashion-MNIST). We **meta-test** the trained model on 5 datasets: **Stanford Cars**, **QuickDraw** (Ha & Eck, 2017), **VGG Flowers** (Nilsback & Zisserman, 2008), **VGG Pets** (Parkhi et al., 2012), and **STL10**, which are also highly heterogeneous.

Experimental setup. We use ResNet20 frequently used for images of size 32×32 (e.g. CIFAR datasets). We use random cropping and horizontal flipping as data augmentations, following the convention. For **meta-training**, we use the same $\alpha = 0.01$, $K = 1,000$, and $M = 200$ for all the baselines and our model, except for β that we found in the range of $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$. We use SGD with momentum ($\mu = 0.9$) and weight decay ($\lambda = 0.0005$) as the inner optimizer. For **meta-testing**, we train $K = 1,000$ steps for each dataset. We use SGD with Nesterov momentum optimizer ($\mu = 0.9$) with an appropriate learning rate scheduling. The starting learning rate is $\alpha = 0.1$ and we use $\lambda = 0.0005$. See **the supplementary file** for more detail. The code is also publicly available[3].

Baselines. We first compare with **Finetuning** baselines. We consider finetuning from the initialization obtained with multi-headed **multi-task learning (MTL)**, where we pretain a single shared feature extractor across the source tasks while the final dense layers are exclusive from one task to the others. We also consider finetuning from the initialization obtained by learning **only with Tiny ImageNet (TIN)** dataset in order to alleviate the negative transfer issue that may come with MTL. We next consider the following **meta-learning** methods. **FOMAML:** Meta-gradient of this method (Finn et al., 2017) is simply the last-step inner gradient. **FOMAML++:** (Antoniou et al., 2019) is a variant of FOMAML which periodically accumulates the intermediate meta-gradients (Multi-Step Loss Optimization in MAML++). **iMAML:** (Rajeswaran et al., 2019a) compute meta-gradient by estimating local curvature at the last step, based on Implicit Function Theorem. **Reptile:** Meta-gradient of this method (Nichol et al., 2018) is defined as the average of differences between initialization and task-specific parameters. **Leap:** This method (Flennerhag et al., 2019) defines meta-objective as a sum of the task trajectory lengths, and its meta-gradient is computed with the similar first-order approximation.

Results and analysis. First of all, we see from Figure 6(a) that Ours achieve much faster meta-convergence than other meta-learning methods, thanks to more frequent meta-updates with the proposed continual trajectory shifting. Our method thus achieves competitive meta-testing performances significantly faster than other meta-learning methods (Figure 7(a-e)[4]). Note that Reptile significantly outperforms Leap in our experiments. We carefully tuned the meta-learning rate of each method, and found that Reptile allows much greater meta-learning rate ($\beta = 1.0$) than Leap ($\beta = 0.1$).



(a) Meta-convergence (b) Meta-testing performance

Figure 8. **Ablation study**. (b) VGG Pets

We also compare with the other baselines in Figure 6(b). FOMAML, FOMAML++, and iMAML perform much worse than other baselines. For FOMAML, its meta-gradient is simply the last-step inner-gradient, which can be arbitrarily uninformative for the meta-learner (Flennerhag et al., 2019). iMAML estimates the meta gradient at the last step by implicitly incorporating the learning trajectory based on Implicit Function Theorem, but the results tell that the method is not as effective as explicit methods such as Reptile. FOMAML++ outperforms FOMAML, demonstrating the importance of considering the whole inner-trajectory when computing the meta-gradients (Flennerhag et al., 2019).

For the finetuning baselines, finetuning with MTL significantly underperforms the finetuning with only TIN. It demonstrates the effect of negative transfer problem that frequently happens when we jointly learn with multiple heterogeneous datasets. On the other hand, our method outperforms both of the finetuning baselines, indicating that meta-learning of the shared initialization can be an effective alternative for the negative transfer problem, instead of finding a jointly optimal feature extractor for all the tasks. Lastly, Figure 6(c) and 6(d) shows that the performance improves as we increase the inner trajectory length up to $K = 1,000$, demonstrating the effect of short horizon bias (See also Figure 4(c)).

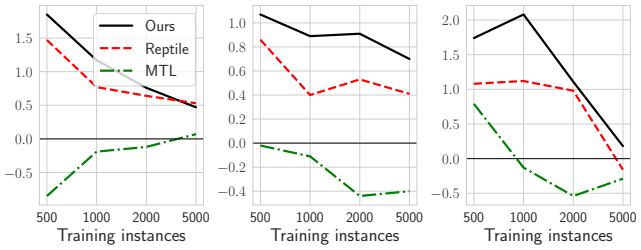
Ablation study. We perform the ablation study whether the proposed shifting for the task-learning trajectories (Line 13 in Algorithm 2) is the source of performance improvements. We see from Figure 8 that our model without shifting (**No Shifting**) or the shifting with the same magnitude but with random direction (**Random Shifting**) performs almost the same as Reptile, demonstrating the effectiveness of the proposed shifting rule.

[3] https://github.com/JWoong148/ContinualTrajectoryShifting

[4] Note that x-axis is cumulative inner steps at meta-training, not training steps at meta-testing.

Table 1. **Classification accuracies obtained with various pre-training methods (%)** when the target dataset contains 1,000 images. We report the mean accuracies and the 95% confidence intervals over 5 runs.

ImageNet pre-training	Target dataset CIFAR100	CIFAR10	SVHN	Dogs	Pets	Flowers	Food	CUB	DTD	Avg.
+ None	41.95±0.29	81.60±0.28	60.09±0.98	55.56±0.29	83.48±0.15	87.01±0.38	36.95±0.37	34.32±0.46	59.39±0.53	60.04
+ MTL	42.79±0.54	82.33±0.20	59.05±1.09	55.00±0.28	83.29±0.25	87.04±0.34	36.84±0.37	34.19±0.88	58.86±0.49	59.93
+ Reptile	47.98±0.14	**84.58±0.12**	62.39±0.72	56.97±0.12	84.25±0.22	87.22±0.31	37.35±0.22	35.44±0.48	58.98±0.59	61.68
+ Ours	**48.34±0.21**	84.42±0.15	**62.82±0.56**	**57.53±0.48**	**84.65±0.11**	**87.54±0.21**	**37.84±0.20**	**36.40±0.20**	**59.53±0.25**	**62.12**



(a) Pets (b) Food (c) CUB

Figure 9. Accuracy improvements (%) over ImageNet Finetuning.

4.3. Improving on ImageNet Pre-trained Model

We demonstrate that our method is capable of improving on the ImageNet finetuning under limited data regime.

Datasets. For **meta-training**, we construct a heterogeneous data distribution by class-wisely dividing the original ImageNet dataset into 8 subsets based on the WordNet class hierarchy. We then meta-train the model over the obtained subsets. See Figure 4 and Table 4 in the **Supplementary file** for more information. We then **meta-test** with the 9 benchmark image classification datasets described in Table 5 in the **Supplementary file**.

Experimental setup. We use ResNet18 (He et al., 2016) suitable for the images of size 224×224. We use random cropping and horizontal flipping as data augmentations. **Meta-training:** For MTL, Reptile and our model, we start from the ImageNet pretrained model for the meta-training to converge faster and reach a better solution than meta-training from scratch. We use SGD with momentum ($\mu = 0.9$) as the inner-optimizer without applying the weight decay ($\lambda = 0$). We set the batch size to 256. For **MTL**, we set the learning rate to 0.01 and train for $50,000$ steps. For **Reptile** and **our model**, we use $\alpha = 0.01$, $K = 1,000$, and $M = 50$, but use different meta-learning rate ($\beta = 1$ for Reptile and $\beta = 0.001$ for our model), since the optimal meta-learning rate differs across two methods. We meta-train for $50,000$ steps. **Meta-testing:** We subsample each target dataset such that each contains $1,000$ training datapoints. We normalize each input image with the mean and standard deviation of RGB channels across the whole training datapoints. We set the batch size to 256. We train $K = 1,000$ steps with Nesterov momentum optimizer ($\mu = 0.9$). The starting learning rate is $\alpha = 0.01$, and we step-wisely decay α at 400, 700, and 900 steps by multiplying 0.2. We use the weight decay $\lambda = 0.0001$.

Results and analysis Table 1 shows the results. We empirically observe that our method is more effective when the task-specific learning suffers from overfitting. In order to clearly see this effect, we subsample each target dataset to contain only $1,000$ images and compare the performances. Table 1 shows that our method consistently outperforms the baselines across most of the datasets when each target dataset has a limited number of instances. Figure 9 confirms that on most of the target datasets, the performance improvements from the base ImageNet finetuning increase as the size of each dataset gets smaller. We conjecture that the performance improvements comes from the *smoother* initial model parameter learned with our meta-learning algorithm, which may correspond to a stronger prior over the model parameters that can effectively regularize the task-specific learning with the small datasets.

5. Conclusion

In this paper, we tackled the challenging problem of large-scale meta-learning. We first showed that a large number of inner-gradient steps allows to capture the structure of large-scale meta-learning well. We then improve the meta-learning efficiency with the *continual trajectory shifting*, which continuously shifts the inner-learning trajectories w.r.t. the frequent update of the initialization point. By doing so, unlike the previous meta-learning algorithms, the task-learners no longer need to reinitiate the learning trajectory for every meta-update, thereby allowing to arbitrarily increase the meta-update frequency. We investigated why and how our model works well with synthetic experiment and also validated the effectiveness of our method on the large-scale experiments with image datasets. We believe that our work make a meaningful step toward applying meta-learning to large-scale real-world tasks.

Acknowledgement

This work was supported by Google AI Focused Research Award, the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST))

References

https://tiny-imagenet.herokuapp.com/.

Antoniou, A., Edwards, H., and Storkey, A. How to train your MAML. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJGven05Y7.

Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In *ICML*, 2017a.

Balduzzi, D., McWilliams, B., and Butler-Yeoman, T. Neural taylor approximations: Convergence and exploration in rectifier networks. In *ICML*, 2017b.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *ICML*, 2009.

Coates, A., Ng, A., and Lee, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. In *ICLR*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, 2017.

Flennerhag, S., Moreno, P. G., Lawrence, N., and Damianou, A. Transferring Knowledge across Learning Processes. In *ICLR*, 2019.

Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R. Meta-learning with warped gradient descent. In *ICLR*, 2020.

Ha, D. and Eck, D. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. URL http://arxiv.org/abs/1704.03477.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Jang, Y., Lee, H., Hwang, S. J., and Shin, J. Learning What and Where to Transfer. In *ICML*, 2019.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, CVPR*, 2011.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *CVPR*, 2019.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

Krizhevsky, A., Hinton, G., et al. Learning Multiple Layers of features from Tiny Images. 2009.

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

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

Lee, Y. and Choi, S. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018.

Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S., and Yang, Y. LEARNING TO PROPAGATE LABELS: TRANSDUCTIVE PROPAGATION NETWORK FOR FEW-SHOT LEARNING. In *ICLR*, 2019.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-Grained Visual Classification of Aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *ICLR*, 2018.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading Digits in Natural Images with Unsupervised Feature Learning. 2011.

Nichol, A., Achiam, J., and Schulman, J. On First-Order Meta-Learning Algorithms. *arXiv e-prints*, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

Oreshkin, B., Rodríguez López, P., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/072b030ba126b2f4b2374f342be9ed44-Paper.pdf.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-Learning with Implicit Gradients. In *NeurIPS*, 2019b.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

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

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NIPS*, 2017.

Song, X., Gao, W., Yang, Y., Choromanski, K., Pacchiano, A., and Tang, Y. Es-maml: Simple hessian-free meta learning. In *ICLR*, 2020.

Thrun, S. and Pratt, L. (eds.). *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-8047-9.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching Networks for One Shot Learning. In *NIPS*, 2016.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wu, Y., Ren, M., Liao, R., and Grosse., R. Understanding short-horizon bias in stochastic meta-optimization. In *ICLR*, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017a.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017b.

Yang, F. S. Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning.(2017). 2017.