META-LEARNING WITH NETWORK PRUNING FOR OVERFITTING REDUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

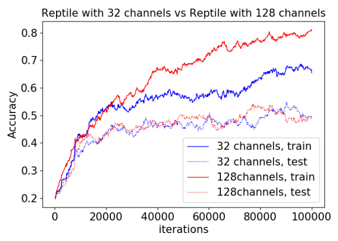
Meta-learning has achieved great success in few-shot learning. However, the existing meta-learning models have been evidenced to overfit on training tasks when using over-parameterized convolutional neural networks. This means that the generalization performance of the meta-learner cannot be improved by merely applying more complex networks. To remedy this deficiency, we propose a sparsity-constrained meta-learning approach to learn from training tasks a subnetwork from which first-order optimization methods can quickly converge towards the optimal network in testing tasks. Our theoretical analysis shows the benefit of sparsity for reducing the generalization gap of the estimated meta-learner. We have implemented our approach on top of Reptile assembled with varying network pruning routines including Dense-Sparse-Dense (DSD) and Iterative Hard Thresholding (IHT). Extensive experimental results on benchmark datasets with different over-parameterized deep networks demonstrate that our method not only effectively alleviates meta-overfitting but also in many cases improves the overall generalization performance when applied to few-shot classification tasks.

1 INTRODUCTION

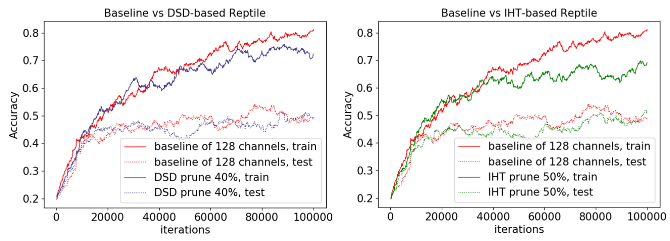
The ability of adapting to a new task with several trials is essential for artificial agents. The goal of few-shot learning (Santoro et al., 2016) is to build a model which is able to get the knack of a new task with limited training samples. Meta-learning (Schmidhuber, 1987; Bengio et al., 1990; Thrun & Pratt, 2012) provides a principled way to cast few-shot learning as the problem of *learning-to-learn*, which typically trains a hypothesis or learning algorithm to memorize the experience from previous tasks for a future task learning with very few samples. The practical importance of meta-learning has been witnessed in many vision and online/reinforcement learning applications including image classification (Ravi & Larochelle, 2016; Li et al., 2017), multi-arm bandit (Sung et al., 2017) and 2D navigation (Finn et al., 2017).

Among others, one particularly simple yet successful meta-learning paradigm is first-order optimization based meta-learning which aims to train hypotheses that can quickly adapt to unseen tasks by performing one or a few steps of (stochastic) gradient descent (Ravi & Larochelle, 2016; Finn et al., 2017). Reasons for the recent increasing attention to this class of gradient-optimization based methods include their outstanding efficiency and scalability exhibited in practice (Nichol et al., 2018).

Challenge and motivation. A challenge in the existing meta-learning approaches is their tendency to overfit (Mishra et al., 2018; Yoon et al., 2018). When training an over-parameterized meta-learner such as very deep and/or wide convolutional neural networks (CNN) which are powerful for representation learning, there are two sources of potential overfitting at play: the inter-task overfitting of meta-learner (or *meta-overfitting*) to the training tasks and the inner-task overfitting of task-specific learner to the task training data. There have been recent efforts put to deal with inner-task overfitting (Lee et al., 2019; Zintgraf et al., 2019). The study on the inter-task meta-overfitting, however, still remains under explored. Since in principle the optimization-based meta-learning is designed to learn fast from small amount of data in new tasks, we expect the meta-overfitting to play a more important role in influencing the overall generalization performance of the trained meta-learner. For an example, it can be observed from the curves in Figure 1(a) for MiniImageNet that although a 4-layer CNN with 128 channels achieves significantly higher training accuracy than the same network architecture with 32 channels, the two networks have quite comparable testing accuracy. It is thus



(a) Meta-overfitting



(b) Meta-overfitting reduction by DSD and IHT based Reptile

Figure 1: (a) The meta-overfitting phenomenon on MiniImageNet (5-way 1-shot) tasks when increasing the number of channels from 32 to 128 in a 4-layer CNN. (b) The proposed DSD and IHT based Reptile algorithms are effective in closing the gap between training accuracy and testing accuracy. Bested viewed in color.

crucial to develop robust methods that can well handle meta-overfitting especially when the number of training tasks is substantially smaller than the number of meta-parameters.

Sparsity model is a promising tool for high-dimensional machine learning with guaranteed statistical efficiency and robustness to overfitting (Maurer & Pontil, 2012; Yuan et al., 2018; Abramovich & Grinshtein, 2019). It has been theoretically and numerically justified by Arora et al. (2018) that s-parsity benefits considerably the generalization performance of deep neural networks. In the regime of compact deep learning, the so called *network pruning* technique has been widely studied and evidenced to work favorably in generating sparse subnetworks without compromising generalization performance (Frankle & Carbin, 2018; Jin et al., 2016; Han et al., 2016). Inspired by these remarkable success of sparsity models, it is reasonable to conjecture that sparsity would also be of help for enhancing the robustness of optimization based meta-learning to meta-overfitting.

Our contribution. In this paper, we present a novel sparse optimization based approach for meta-learning with over-parameterized neural networks. The problem is formulated as learning a sparse meta-initialization network from training tasks such that in a new task the learned subnetwork can quickly converge to the optimal solution via gradient descent. The core idea is to reduce meta-overfitting by controlling the counts of the non-zero parameters in the meta-learner. Theoretically, we have established a set of generalization gap bounds for the proposed sparse meta-learner showing the benefit of sparsity in improving its generalization performance. Practically, we have implemented our approach in a joint algorithmic framework of Reptile (Nichol et al., 2018) with network pruning, along with two specifications to network pruning methods, which are Dense-Sparse-Dense (DSD) (Han et al., 2016) and Iterative Hard Thresholding (IHT) (Jin et al., 2016), respectively. The actual performance of our approach has been extensively evaluated on few-shot classification tasks with over-parameterized deep/wide CNNs including ResNet-18 (He et al., 2016). Numerical results demonstrate that our method can effectively alleviate overfitting and achieve similar or even superior generalization performance to the conventional dense models. As illusrated in Figure 1(b), using 128-channel network on 5-way 1-shot tasks, our DSD and IHT based Reptile algorithms are effective in closing the gap between the training accuracy and testing accuracy.

2 Related Work

Optimization-based meta-learning. The family of optimization-based meta-learning approaches usually learns a good hypothesis which can fast adapt to unseen tasks (Ravi & Larochelle, 2016; Finn et al., 2017; Nichol et al., 2018; Khodak et al., 2019). Compared to the metric (Koch et al., 2015; Snell et al., 2017) and memory (Weston et al., 2014; Santoro et al., 2016) based meta learning algorithms, optimization based meta-learning algorithms are gaining increasing attention due to their simplicity, versatility and effectiveness. As a recent leading framework for optimization-based meta-learning, MAML (Finn et al., 2017) is designed to estimate a meta-initialization network which can be well fine-tuned in an unseen task via only one or few steps of minibatch gradient descent. Although simple in principle, MAML requires computing Hessian-vector product for back-propagation, which could be computationally expensive when the model is big. The first-order

MAML (FOMAML) is therefore proposed to improve the computational efficiency by simply ignoring the second-order derivatives in MAML. Reptile Nichol et al. (2018) is another approximated first-order algorithm which works favorably since it maximizes the inner product between gradients from the same task yet different minibatches, leading to improved model generalization. In (Lee et al., 2019), the meta-learner is treated as a feature embedding module of which the output is used as input to train a multi-class kernel support vector machine as base learner. To deal with overfitting, the CAVIA method (Zintgraf et al., 2019) decomposes the meta-parameters into the so called context parameters and shared parameters. The context parameters are updated for task adaption with limited capacity while the shared parameters are meta-trained for generalization across tasks.

Network pruning. Early network weight pruning algorithms date back to Optimal Brain Damage (LeCun et al., 1990) and Optimal Brain Surgeon (Hassibi et al., 1993). A dense-to-sparse algorithm was developed by Han et al. (2015) to first remove near-zero weights and then fine tune the preserved weights. As a serial work of dense-to-sparse, the dense-sparse-dense (DSD) method (Han et al., 2016) was proposed to re-initialize the pruned parameters as zero and retrain the entire network after the dense-to-sparse pruning phase. The iterative hard thresholding (IHT) method (Jin et al., 2016) shares a similar spirit with DSD to conduct multiple rounds of iteration between pruning and retraining. Srinivas & Babu (2015) proposed a data-free method to prune the neurons in a trained network. In (Louizos et al., 2017), an L_0-norm regularized risk minimization framework was proposed to learn sparse networks during training. More recently, Frankle & Carbin (2018) introduced and studied the "lottery ticket hypothesis" which assumes that once a network is initialized, there should exist an optimal subnetwork, which can be learned by pruning, that performs as well as the original net or even superior.

Despite the remarkable success achieved by both meta-learning and network pruning, it still remains largely open to investigate the impact of network pruning on alleviating the meta-overfitting of optimization based meta-learning, which is of primal interest to our study in this paper.

3 Meta-Learning with Sparsity

In this section, we propose a sparse optimization based meta-learning approach for few-shot learning with over-parameterized neural networks. Our hope is to alleviate the overfitting of meta-learner by reducing its parameter counts without compromising accuracy.

3.1 Problem Setting

We consider the N-way K-shot problem as defined in (Vinyals et al., 2016). Tasks are sampled from a specific distribution $p(\mathcal{T})$ and will be divided into *meta training set* $\mathcal{S}^{tr}$, *meta validation set* $\mathcal{S}^{val}$, and *meta testing set* $\mathcal{S}^{test}$. Classes in different datasets are disjoint (i.e., the class in $\mathcal{S}^{tr}$ will not appear in $\mathcal{S}^{test}$). During training, each task is made up of support set $\mathcal{D}^{supp}$ and query set $\mathcal{D}^{query}$. Both $\mathcal{D}^{supp}$ and $\mathcal{D}^{query}$ are sampled from the same classes of $\mathcal{S}^{tr}$. $\mathcal{D}^{supp}$ is used to for training while $\mathcal{D}^{query}$ is used for evaluation. For a N-way K-shot classification task, we sampled N classes from dataset, and then K samples are sampled from each of these classes to form $\mathcal{D}^{supp}$, namely $\mathcal{D}^{supp} = \{(\boldsymbol{x}_c^k, y_c^k), k = 1, 2, ..., K; c = 1, 2, ..., N\}$. For example, for a 5-way 2-shot task, we sample 2 data-label pairs from each of 5 classes, thus, such a task has 10 samples. Usually, several other samples of the same classes will be sampled to compose $\mathcal{D}^{query}$. For example, $\mathcal{D}^{query}$ is used in Reptile (Nichol et al., 2018) in evaluation steps. We use the loss function $\ell(\hat{y}, y)$ to measure the discrepancy between the predicted label $\hat{y}$ and the true label y.

Notation. For an integer n, we denote $[n]$ as the abbreviation of the index set $\{1, ..., n\}$. We use $\odot$ to denote the element-wise product operator. We say a function $g : \mathbb{R}^p \mapsto \mathbb{R}$ is G-Lipschitz continuous if $|g(\theta) - g(\theta')| \leq G\|\theta - \theta'\|_2$, and g is H-smooth if it obeys $\|\nabla g(\theta) - \nabla g(\theta')\|_2 \leq H\|\theta - \theta'\|_2$. For any sparsity level $1 \leq s \leq p$, we say a function g is restricted μ_s-strongly convex if there exists $\mu_s > 0$ such that $g(\theta) - g(\theta') - \langle \nabla g(\theta'), \theta - \theta' \rangle \geq \frac{\mu_s}{2}\|\theta - \theta'\|^2, \forall \|\theta - \theta'\|_0 \leq s$.

3.2 Meta-Learning with Sparsity

Our ultimate goal is to learn a good initialization of parameters for a convolutional neural network $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, where θ is the model parameters set, from a set of training tasks such that the learned initialization network generalizes well to future unseen tasks. Inspired by the recent remarkable success of MAML (Finn et al., 2017) and the strong generalization capability of sparse deep learning models (Frankle & Carbin, 2018; Arora et al., 2018), we propose to learn from previous task experience a sparse subnetwork started from which the future task-specific networks can be efficiently learned using first-order optimization methods. To this end, we introduce the following layer-wise sparsity constrained stochastic first-order meta-learning formulation:

$$\min_{\theta} \mathcal{R}(\theta) := \mathbb{E}_{T\sim p(\mathcal{T})}\left[\mathcal{L}_{\mathcal{D}_T^{query}}\left(\theta - \eta\nabla_\theta \mathcal{L}_{\mathcal{D}_T^{supp}}(\theta)\right)\right], \text{ s.t. } \|\theta_l\|_0 \le k_l,\ l\in[L], \tag{1}$$

where $\mathcal{L}_{\mathcal{D}_T^{supp}}(\theta) = \frac{1}{NK}\sum_{(\boldsymbol{x}_c^k, y_c^k)\in\mathcal{D}_T^{supp}} \ell(f_\theta(\boldsymbol{x}_c^k), y_c^k)$ is the empirical risk for task T, η is a step-size constant, $||\theta_l||_0$ denotes the number of non-zero entries in the parameters of l-th layer θ_l, k_l controls the sparsity level of the l-th layer, and L is the total number of layers in the network.

In general, the mathematical formulation of task distribution $p(\mathcal{T})$ is unknown but we usually have access to a set of i.i.d. training tasks $S = \{T_i\}_{i=1}^M$ sampled from $p(\mathcal{T})$. Thus the following empirical version of the population form in equation 1 is alternatively considered for training:

$$\min_{\theta} \mathcal{R}_S(\theta) := \frac{1}{M}\sum_{i=1}^{M}\left[\mathcal{L}_{\mathcal{D}_{T_i}^{query}}\left(\theta - \eta\nabla_\theta \mathcal{L}_{\mathcal{D}_{T_i}^{supp}}(\theta)\right)\right], \text{ s.t. } \|\theta_l\|_0 \le k_l,\ l\in[L]. \tag{2}$$

To compare with MAML, our model shares an identical objective function but with the layer-wise sparsity constraints $\|\theta_l\|_0 \le k_l$ imposed for the purpose of enhancing learnability of the over-parameterized meta-initialization network. In view of the "lottery ticket hypothesis" (Frankle & Carbin, 2018), the model in equation 2 can be interpreted as a first-order meta-learner for estimating a subnetwork, or a "*winning ticket*", for future task learning. Given the strong statistical efficiency and generalization guarantees of sparsity models (Yuan et al., 2018; Arora et al., 2018), such a subnetwork is expected to achieve advantageous generalization performance over the dense initialization networks learned by vanilla MAML.

3.3 Generalization Performance Analysis

We now analyze the generalization performance of the sparsity constrained meta-learning model in equation 2. Let p be the total number of parameters in the over-parameterized network and $\Theta \subseteq \mathbb{R}^p$ be the domain of interest for θ. Let $k = \sum_{l=1}^{L} k_l$ be the total desired sparsity level of the subnetwork. Our first result is a uniform convergence bound for the k-sparse neural networks.

Theorem 1. *Assume that the domain of interest Θ is bounded by R and the loss function $\ell(f_\theta(\boldsymbol{x}), y)$ is G-Lipschitz continuous and H-smooth with respect to θ. Then for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the random draw of S, the generalization gap is uniformly upper bounded as*

$$\sup_{\|\theta_l\|_0\le k_l, l\in[L]} |\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \le \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k/\delta) + k^2\log(M)\log(ep/k)}{M}}\right).$$

We remark that the dominant term $\mathcal{O}\left(\sqrt{k^2\log(M)\log(ep/k)/M}\right)$ in the above uniform convergence bound is paid for not knowing the sparsity pattern of an arbitrary k-sparse subnetwork. Comparing to the $\mathcal{O}\left(\sqrt{p\log(M)\log(p)/M}\right)$ uniform bound in Lemma 1 (see Appendix A.1) for dense networks, the uniform bound established in Theorem 1 is stronger when $p \gg k^2$, which shows the benefit of sparsity constraint imposed on the meta-initialization network. Although quite general, the uniform generalization bound established in the above theorem is still relatively loose due to the k^2 factor appeared in the bound. We next show how to improve upon such a uniform bound via further exploring the optimality of the estimated subnetwork.

Let $\mathcal{J} = \{J \subseteq [p] : |J_l| = k_l\}$ be the set of index set with layer-wise cardinality $k_l, l\in[L]$, and let $\mathcal{J}^* := \{J : J \in \mathcal{J} | \min_{\text{supp}(\theta)=J}\mathcal{R}(\theta) \le \min_{\text{supp}(\theta')\in\mathcal{J}}\mathcal{R}(\theta')\}$. Our analysis also relies on the quantity $\Delta := \min_{\text{supp}(\theta')\in\mathcal{J}\setminus\mathcal{J}^*}\mathcal{R}(\theta') - \min_{\text{supp}(\theta)\in\mathcal{J}^*}\mathcal{R}(\theta)$ which measures the gap between the smallest k-sparse meta-loss value and the second smallest k-sparse meta-loss value. The following theorem shows the improved generalization gap bound for the sparse meta-learner $\theta_S := \arg\min_{\text{supp}(\theta)\in\mathcal{J}}\mathcal{R}_S(\theta)$.

Theorem 2. *Assume that the domain of interest is bounded by R and the loss function $\ell(f_\theta(\boldsymbol{x}), y)$ is G-Lipschitz continuous and H-smooth with respect to θ. Assume that M is sufficiently large such that $\Delta = \Omega\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k/\delta)+k^2\log(M)\log(ep/k)}{M}}\right)$. Then for any $\delta \in (0,1)$ the generalization gap of θ_S is upper bounded by*

$$|\mathcal{R}(\theta_S) - \mathcal{R}_S(\theta_S)| \leq \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k|\mathcal{J}^*|/\delta)}{M}}\right).$$

Remark 1. *As is expected that the larger the gap value Δ is, the easier the condition $\Delta = \Omega\left(\sqrt{k^2\log(M)\log(ep/k)/M}\right)$ can be fulfilled with respect to the meta-sample size M.*

Theorem 2 mainly conveys the message that the generalization gap of θ_S is upper bounded by $\mathcal{O}\left(\sqrt{k\log(M)\log(k|\mathcal{J}^*|)/M}\right)$ with high probability. Since $|\mathcal{J}^*|$ is expected to be much smaller than $|\mathcal{J}| = \mathcal{O}\left((p/k)^k\right)$, this bound would be substantially tighter than the corresponding uniform convergence bound established in Theorem 1 when $|\mathcal{J}^*| \ll |\mathcal{J}|$ (e.g., $|\mathcal{J}^*| = \mathcal{O}(1)$). Specially, the following corollary shows that if further assuming restricted strong convexity and certain signal strength condition on the meta-loss function $\mathcal{R}$, then $\mathcal{J}^*$ will be a singleton and thus the bound in Theorem 2 holds with $|\mathcal{J}^*| = 1$. We denote $\theta_{\min}$ the smallest (in magnitude) nonzero element of θ.

Corollary 3. *Assume that $\mathcal{R}(\theta)$ is μ_{2k}-strongly convex. Suppose that there exists a k-sparse vector $\bar{\theta}$ such that $\|\bar{\theta}_l\|_0 \leq k_l, \forall l \in [L]$ and $\bar{\theta}_{\min} > 2\sqrt{2k}\|\nabla\mathcal{R}(\bar{\theta})\|_\infty/\mu_{2k}$. Assume that the conditions in Theorem 2 hold. Then for any $\delta \in (0,1)$ the generalization gap of θ_S is upper bounded by $|\mathcal{R}(\theta_S) - \mathcal{R}_S(\theta_S)| \leq \mathcal{O}\left(\sqrt{k\log(M)\log(k/\delta)/M}\right)$.*

Remark 2. *Particularly, if there exists a layer-wise sparse stationary point $\bar{\theta}$ such that $\nabla\mathcal{R}(\bar{\theta}) = 0$, then the signal strength condition $\bar{\theta}_{\min} > 2\sqrt{2k}\|\nabla\mathcal{R}(\bar{\theta})\|_\infty/\mu_{2k}$ can be naturally satisfied.*

4 ALGORITHM

We have implemented the proposed sparse meta-learning model 2 based on Reptile (Nichol et al., 2018) which is a scalable method for solving the meta-learning model of the form 2 but without layer-wise sparsity constraint. In order to handle the sparsity constraint, we follow the principles behind the widely applied *dense-sparse-dense* (DSD) (Han et al., 2016) and *iterative hard thresholding* (IHT) (Jin et al., 2016) network pruning algorithms to alternate the Reptile iteration between pruning insignificant weights in each layer and retraining the pruned network.

4.1 MAIN ALGORITHM: REPTILE WITH ITERATIVE NETWORK PRUNING

The algorithm of our network-pruning-based Reptile method is outlined in Algorithm 1. The learning procedure contains a pre-training phase followed by an iterative procedure of network pruning and retraining. We want to emphasize that since our ultimate goal is not to do network compression, but to reduce meta-overfitting via controlling the sparsity level of the meta-initialization network, the final output of our algorithm is typically dense after the retraining phase which has been evidenced in practice to be effective for improving the prediction accuracy. In the following subsections, we describe the key components of our algorithm in details.

4.1.1 MODEL PRETRAINING

For model pre-training, we run a few number of Reptile iteration rounds to generate a relatively good initialization. In each loop of the Reptile iteration, we first sample a mini-batch of meta-tasks $\{T_i\}_{i=1}^s$ from the task distribution $p(\mathcal{T})$. Then for each task T_i, we compute the adapted parameters via (stochastic) gradient descent as $\tilde{\theta}_{T_i} = \theta^{(0)} - \eta\nabla_\theta\mathcal{L}_{\mathcal{D}_{T_i}^{supp}}(\theta^{(0)})$, where $\tilde{\theta}_{T_i}$ denotes the task-specific parameters learned from each task T_i, $\theta^{(0)}$ is the current initialization of model parameters, η is the inner-task learning rate, and $\mathcal{D}_{T_i}^{supp}$ denotes the support set of task T_i. When all the task-specific parameters are updated, the initialization parameters will be updated according to $\theta^{(0)} =$

```
Algorithm 1: Reptile with Iterative Network Pruning
Input  : inner loop learning rate $\eta$, outer loop learning rate $\beta$, layer-wise sparsity level $\{k_l\}_{l=1}^L$,
         mini-batch batch size $s$ for meta training.
Output: $\theta^{(t)}$.
Initialization Randomly initialize $\theta^{(0)}$.
/* Pre-training with Reptile                                                    */
while the termination condition is not met do
    Sample a mini-batch tasks $\{T_i\}_{i=1}^s$ of size $s$;
    For each task $T_i$, compute the task-specific adapted parameters using gradient descent:
        $\tilde{\theta}_{T_i} = \theta^{(0)} - \eta \nabla_\theta \mathcal{L}_{\mathcal{D}_{T_i}^{supp}}(\theta^{(0)});$
    Update the parameters: $\theta^{(0)} = \theta^{(0)} + \beta \left( \frac{1}{s} \sum_{i=1}^s \tilde{\theta}_{T_i} - \theta^{(0)} \right).$
end
for $t = 1, 2, ...$ do
    /* Pruning phase                                                            */
    Generate a network zero-one mask $\mathcal{M}^{(t)}$ whose non-zero entries at each layer $l$ are those top $k_l$
    entries in $\theta_l^{(t)}$;
    Compute $\theta_{\mathcal{M}}^{(t)} = \theta^{(t)} \odot \mathcal{M}^{(t)}$;
    /* Subnetwork fine-tune with Reptile                                        */
    while the termination condition is not met do
        Sample a mini-batch tasks $\{T_i\}_{i=1}^s$ of size $s$;
        For each task $T_i$, compute the adapted parameters using gradient descent:
            $\tilde{\theta}_{T_i} = \theta_{\mathcal{M}}^{(t)} - \eta \nabla_\theta \mathcal{L}_{\mathcal{D}_{T_i}^{supp}}(\theta_{\mathcal{M}}^{(t)});$
        Update the parameters: $\theta^{(t)} = \theta_{\mathcal{M}}^{(t)} + \beta \left( \frac{1}{s} \sum_{i=1}^s \tilde{\theta}_{T_i} - \theta_{\mathcal{M}}^{(t)} \right);$
        Compute $\theta_{\mathcal{M}}^{(t)} = \theta^{(t)} \odot \mathcal{M}^{(t)}$;
    end
    /* Retraining phase                                                         */
    Set $\theta^{(t)} \leftarrow \theta_{\mathcal{M}}^{(t)}$;
    while the termination condition is not met do
        Sample a mini-batch tasks $\{T_i\}_{i=1}^s$ of size $s$;
        For each task $T_i$, compute the adapted parameters using gradient descent:
            $\tilde{\theta}_{T_i} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}_{\mathcal{D}_{T_i}^{supp}}(\theta^{(t)});$
        Update the parameters: $\theta^{(t)} = \theta^{(t)} + \beta \left( \frac{1}{s} \sum_{i=1}^s \tilde{\theta}_{T_i} - \theta^{(t)} \right).$
    end
end
```

$\theta^{(0)} + \beta \left(\frac{1}{s} \sum_{i=1}^s \tilde{\theta}_{T_i} - \theta^{(0)} \right)$ with learning rate β. Here we follow Reptile to use $\frac{1}{s} \sum_{i=1}^s \tilde{\theta}_{T_i} - \theta^{(0)}$ as an approximation to the negative meta-gradient, which has been evidenced to be effective for scaling up the MAML-type first-order meta-learning models (Nichol et al., 2018).

4.1.2 Iterative Network Pruning and Retraining

After model pre-training, we proceed to the main loop of our Algorithm 1 that carries out iterative network pruning and retraining.

Pruning phase. In this phase, we first greedily truncate out of the model a portion of near-zero parameters which are unlikely to contribute significantly to the model performance. To do so, we generate a network binary mask $\mathcal{M}^{(t)}$ whose non-zero entries at each layer l are those top k_l (in magnitude) entries in $\theta_l^{(t)}$, and compute $\theta_{\mathcal{M}}^{(t)} = \theta^{(t)} \odot \mathcal{M}^{(t)}$ as the sparsity restriction of $\theta^{(t)}$. Then we fine-tune the subnetwork over the mask $\mathcal{M}^{(t)}$ by applying Reptile restrictively to this subnetwork with initialization $\theta_{\mathcal{M}}^{(t)}$. Our numerical experience suggests that sufficient steps of subnetwork fine-tuning tends to substantially improve the stability and convergence behavior of the method.

The fine-tuned subnetwork $\theta_{\mathcal{M}}^{(t)}$ at the end of the pruning phase is expected to reduce the chance of overfitting to noisy data. However, it is also believed that such subnetwork will reduce the capacity of the network, which could in turn lead to potentially biased learning with higher training loss. To remedy this issue, inspired by the retraining trick introduced in (Han et al., 2016) for network pruning, we propose to restore the pruned weights that would be beneficial for enhancing the model representation power to improve the overall generalization performance in average.

Retraining phase. In this phase, the layer-wise sparsity constraints are relaxed and the pruned parameters are re-activated for fine-tuning. The retraining procedure is almost identical to the pre-training phase, but with the main difference that the former is initialized with the subnetwork generated by the pruning phase while the latter uses random initialization. Such a retraining operation restores the representation capacity of the pruned parameters, which tends to lead to improved generalization performance in practice. For theoretical justification, roughly speaking, since the sparse meta-initialization network obtained in the pruning phase generalizes well in light of Theorem 1 and Theorem 2, it is expected to serve as a good initialization for future retraining via gradient descent. Then according to the uniform stability theory of gradient descent methods in (Hardt et al., 2016) the output dense network will also generalize well if the retraining phase converges fast.

4.2 Two Special Implementations

Reptile with DSD pruning. The DSD method is an effective network pruning approach for preventing the learned model from capturing noise during the training (Han et al., 2016). By implementing the main loop with $t = 1$, the proposed Algorithm 1 reduces to a DSD-based Reptile method for first-order meta-learning.

Reptile with IHT pruning. The IHT method (Jin et al., 2016) is another representative network pruning approach which shares a similar dense-sparse-dense spirit with DSD. Different from the one-shot weight pruning and network training by DSD, IHT is designed to perform multiple rounds of iteration between pruning and retraining, and hence is expected to have better chance to find an optimal sparse subnetwork than DSD does. By implementing the main loop of Algorithm 1 with $t > 1$, we actually obtain a variant of Reptile with IHT-type network pruning.

5 Experiments

This section is devoted to illustrating the empirical performance of our proposed algorithms when evaluated on two benchmark datasets Omniglot (Lake et al., 2011) and MiniImageNet (Vinyals et al., 2016). The baseline algorithms considered for comparison include Reptile and CAVIA (Zintgraf et al., 2019). The model used throughout the experiment contains 4 sequential modules. Each module has a convolutional layer with 3×3 kernel, followed by a batch normalization and a ReLU activation. Additionally for the experiments on MiniImageNet, a 2×2 max-pooling pooling is used on the batch normalization layer output while for Omniglot a stride of 2 is used in convolution. The above network structure design is consistent with those considered for Reptile in (Nichol et al., 2018). We test with varying channel number $\{32, 64, 128, 256\}$ in each convolution layer to show the robustness of our algorithms to meta-overfitting.

There are several additional hyper-parameters in our approach including the number of iterations for pre-training, pruning and retraining phases. These hyper-parameters are selected from a smal-l number of candidate configurations based on our numerical experience with optimal validation performance.

5.1 Few-Shot Classification on Omniglot

Dataset. The Omniglot dataset has 1623 characters from 50 alphabets. Each character contains 20 instances drawn by different individuals. The size of each image is 28×28. We randomly select

1200 characters for meta training and the rest are used for meta testing. Following (Santoro et al., 2016), we also adopt a data augmentation strategy based on image rotation to enhance performance.

Experimental Settings. In DSD-based Reptile training, we set the iteration numbers of pre-training, pruning/fine-tuning and retraining as $3 \times 10^4, 5 \times 10^4$ and 2×10^4, respectively. For IHT-based Reptile model training, we first pre-train the model. Then we iterate between the sparse model fine-tuning (with 1.5×10^4 iterations) and dense-model retraining (with 5×10^3 iterations) for $t = 4$ rounds. The setting of other model training related parameters is identical to those in Nichol et al. (2018). For this group of experiments, the inner-task learning rate is 0.001 and the number of inner-task gradient descent steps is 5. The inter-task learning rate is initialized as 1.0 and the value decays during the entire learning process.

Results. We present the results of 5-way classification accuracy comparison in Table 1. More results are available in Table 4 in Appendix B. The results of Reptile with varying channel number are treated as baseline results. The results in Table 1 show that when using fewer channels (e.g. 32 and 64), both DSD and IHT based Reptile algorithms achieve comparable performance to baselines. When the channel size increases to 128 and 256, slightly improved performance can be observed. This is consistent with our analysis that overfitting is more likely to happen when channel number is relatively large and weight pruning helps to reduce overfitting, which then leads to accuracy improvement with retraining operation.

Table 1: Results on Omniglot with different number of channels and pruning rates. The "±" indicates 95% confidence intervals.

Methods	**5-way 1-shot**	**5-way 5-shot**
Reptile baseline (32)	**96.63±0.17**%	99.31±0.07%
Reptile baseline (64)	97.68±0.10%	99.48±0.06%
Reptile baseline (128)	97.99±0.11%	99.60±0.13%
Reptile baseline (256)	98.05±0.13%	99.65±0.06%
DSD+Reptile (32, 10%)	96.42±0.17%	99.38±0.07%
DSD+Reptile (64, 10%)	97.64±0.02%	99.50±0.05%
DSD+Reptile (128, 10%)	98.04±0.10%	99.61±0.10%
DSD+Reptile (256, 10%)	**98.12±0.12**%	**99.68±0.05**%
IHT+Reptile (32, 20%)	96.54±0.17%	99.57±0.06%
IHT+Reptile (64, 30%)	97.77±0.15%	99.52±0.06%
IHT+Reptile (128, 20%)	**98.22±0.12**%	99.64±0.05%
IHT+Reptile (256, 10%)	98.16±0.12%	**99.66±0.05**%

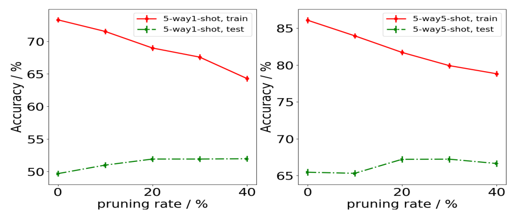
5.2 Few-shot Classification on MiniImageNet

Dataset. The MiniImageNet dataset consists of 100 classes from the ImageNet dataset (Krizhevsky et al., 2012) and each class contains 600 images of size 84 × 84×3. There are 64 classes are used for training, 12 classes for validation and 24 classes for testing.

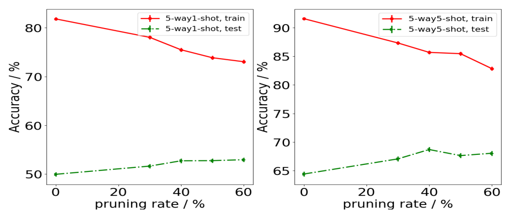
Experimental Settings. For DSD-based Reptile, with 32 channels, we set the iteration numbers for the pre-traning, pruning and retraining phases respectively as 3×10^4, 5×10^4 and 2×10^4, while with $64, 128, 256$ channels, the corresponding number is 3×10^4, 6×10^4 and 10^4 respectively. For IHT-based Reptile, the setting of iteration numbers is the same as those in Section 5.1. Besides, we also evaluate our method on ResNet-18 (He et al., 2016) with DSD-based Reptile. For ResNet-18, we set different pruning rates for residual blocks with different channels. The first convolutional layer is not pruned. For residual block with 64 channels, 128 channels, 256 channels, 512 channels and fully-connected layer, the corresponding pruning rates are 40%, 60%, 60%, 70%, 70% and the corresponding iteration numbers are 3×10^4, 6×10^4, 10^4 iterations, respectively. For this group of experiments, the inner-task learning rate is 0.001 and the number of inner-task iteration loops is 8. The initial inter-task learning rate is 1.0 which will decay during the learning process.

Results. The classification accuracies of DSD-based Reptile approach, IHT-based Reptile approach and baselines on MiniImageNet dataset are presented in Table 2. More results are provided in Table 5 in Appendix B. From these results, we can observe that our methods consistently outperform the considered baselines. The following observations can be highlighted:

- In the 32-channel setting in which the model is less prone to overfit, when applying DSD-based Reptile with 40% pruning rate, the accuracy gain is 0.5% on 5-way 1-shot tasks and 1.5% on 5-way 5-shot tasks. In the 64-channel setting, our IHT-based Reptile approach respectively improves about 1.5% and 1.95% over the baselines on 5-way 1-shot tasks and 5-way 5-shot tasks.



(a) DSD-based Reptile in the 64-channel setting



(b) IHT-based Reptile in the 128-channel setting

Figure 2: Demonstration of overfitting reduction with network pruning: (a) DSD-based Reptile in the setting of 64-channel and 5-way tasks. (b) IHT-based Reptile in the setting of 128-channel and 5-way tasks. In most cases, the gap between training accuracy and testing accuracy reduces as the pruning rate increases, which confirms that both our DSD and IHT-based Reptile approaches can considerably ease meta-overfitting.

In the setting of 128-channel, the accuracy of DSD-based Reptile on 5-way 1-shot tasks is nearly 3% higher than the baseline while on 5-way 5-shot tasks the gain is about 4.47%.

- As expected that ResNet-18 exhibits severe overfitting and DSD-based Reptile can improve $\sim 8\%$ upon baseline in this case.
- In the 128-channel setting, the accuracies of our approaches are all higher than those of CAVIA, and the highest gain of accuracy is over 3%.

It is also worth noting from Table 2 that the accuracy of our algorithms tends to increase as the channel size increases while the baselines behave oppositely. Although in 256-channel case the performance of IHT-based approach drops compared with the 128-channel setting, it still achieves $\sim$ 1.2% accuracy gain over the baseline on 5-way 1-shot tasks and $\sim$ 3.32% on 5-way 5-shot tasks. These results clearly confirm the robustness of our algorithms to the meta-overfitting suffered from the over-parameterization of CNNs.

Table 2: Results on MiniImageNet with different number of channels and pruning rates.

Methods	**5-way 1-shot**	**5-way 5-shot**
Reptile baseline (32)	50.30±0.40%	64.27±0.44%
Reptile baseline (64)	51.08±0.44%	65.46±0.43%
Reptile baseline (128)	49.96±0.45%	64.40±0.43%
Reptile baseline (256)	48.60±0.44%	63.24±0.43%
CAVIA baseline (32)	47.24±0.65%	59.05±0.54%
CAVIA baseline (128)	49.84±0.68%	64.63±0.54%
CAVIA baseline (512)	51.82±0.65%	65.85±0.55%
DSD+Reptile (32, 40%)	50.83±0.45%	65.24±0.44%
DSD+Reptile (64, 30%)	51.91±0.45%	67.23±0.43%
DSD+Reptile (128, 50%)	52.08±0.45%	**68.87±0.42**%
DSD+Reptile (256, 60%)	**53.00±0.45**%	68.04±0.42%
IHT+Reptile(32, 20%)	50.26±0.47%	63.63±0.45%
IHT+Reptile(64, 40%)	52.59±0.45%	67.41±0.43%
IHT+Reptile(128, 40%)	**52.73±0.45**%	**68.69±0.42**%
IHT+Reptile(256, 60%)	49.85±0.44%	66.56±0.42%
ResNet-18(baseline)	39.22±0.45%	-
DSD+ResNet-18	47.23±0.47%	-

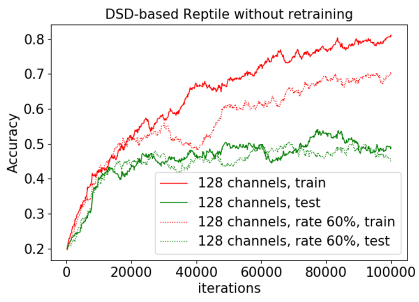
Figure 2 plots the evolving curves of training accuracy and testing accuracy under varying pruning rates from 0 to 40% for DSD and IHT based Reptile. From these curves we can clearly observe that in most cases, the gap between training accuracy and testing accuracy reduces when the pruning rate increases, which confirms our theoretical predictions in Theorem 1 and Theorem 2 that sparsity matters in alleviating meta-overfitting.

5.3 Ablation Study

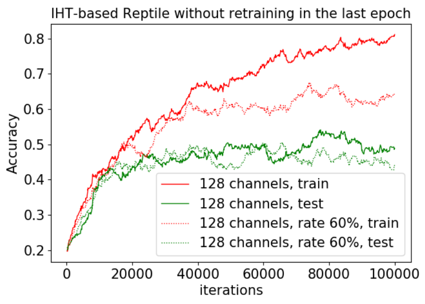
In order to gain better understanding of the impact of dense retraining on the task-specific testing performance, we further carry out an ablation study. For the purpose of fair comparison, only the retraining phase is removed and other settings are the same as proposed in Section 5.1 and 5.2. This

Table 3: Results of ablation study in the 5-way setting. The "±" shows 95% confidence intervals.

Methods	**Retraining**	**5-way 1-shot**	**5-way 5-shot**
Reptile baseline(64)	-	51.08±0.44%	65.46±0.43%
Reptile baseline(128)	-	49.96±0.45%	64.40±0.43%
DSD+Reptile(64, 40%)	×	43.92±0.43%	60.09±0.45%
DSD+Reptile(128, 60%)	×	47.06±0.44%	55.07±0.44%
IHT+Reptile(64, 40%)	×	40.03±0.41%	60.59±0.45%
IHT+Reptile(128, 60%)	×	42.01±0.42%	52.71±0.45%
DSD+Reptile(64, 40%)	√	**51.96**±**0.45**%	**66.64**±**0.43**%
DSD+Reptile(128, 60%)	√	**52.27**±**0.45**%	**68.44**±**0.42**%
IHT+Reptile(64, 40%)	√	**52.59**±**0.45**%	**67.41**±**0.43**%
IHT+Reptile(128, 60%)	√	**52.95**±**0.45**%	**68.04**±**0.42**%



(a) Ablation study on DSD-based Reptile



(b) Ablation study on IHT-based Reptile

Figure 3: Ablation study: Overfitting reduction by DSD and IHT based Reptile *without* retraining.

study is conducted on MiniImageNet dataset for both DSD-based and IHT-based Reptile approaches.

The results of ablation study are listed in Table 3. It can be clearly seen from this group of results that the retraining phase plays an important role in the accuracy performance of our method. Under the same pruning rate, without the retraining phase, the accuracy of both DSD-based and IHT-based Reptile approach drops dramatically. For an instance, in the 64-channel case with 40% pruning rate, the variant of IHT-based Reptile without retraining phase suffers from a $\sim$ 11% drop in accuracy compared with the baseline. On the other side, as shown in Figure 3 that sparsity structure of the network does help to reduce the gap between training accuracy and testing accuracy even with the retraining phase removed. This confirms the benefit of sparsity for generalization gap reduction as revealed by Theorem 1 and 2. Therefore, the network pruning phase makes the model robust to overfitting but in the meanwhile tend to suffer from the deteriorated training loss. The retraining phase helps to restore the capacity of the model to further improve the overall generalization performance.

6 Conclusion

In this paper, we proposed a sparsity-constrained meta-learning approach for overfitting reduction when using over-parameterized neural networks. We have theoretically proved that the generalization gap bounds of the sparse meta-learner have polynomial dependence on the sparsity level rather than the number of parameters. Our approach has been implemented in a scalable meta-learning framework of Reptile with the sparsity level of parameters maintained by network pruning routines including dense-sparse-dense and iterative hard thresholding. Extensive experimental results on benchmark few-shot classification datasets confirm our theoretical predictions and demonstrate the power of network pruning and retraining for improving the generalization performance of gradient-optimization-based meta-learning approaches.

REFERENCES

Felix Abramovich and Vadim Grinshtein. High-dimensional classification by sparse logistic regression. *IEEE Transactions on Information Theory*, 65(5):3068–3079, 2019.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pp. 254–263, 2018.

Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN*, 1990.

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*, pp. 1126–1135. JMLR. org, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1225–1234, 2016.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xiaojie Jin, Xiaotong Yuan, Jiashi Feng, and Shuicheng Yan. Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*, 2016.

Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. *arXiv preprint arXiv:1902.10644*, 2019.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning. 2017.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Andreas Maurer and Massimiliano Pontil. Structured sparsity and generalization. *Journal of Machine Learning Research*, 13(Mar):671–690, 2012.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1DmUzWAW`.

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

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

Philippe Rigollet. 18. s997: High dimensional statistics. *Lecture Notes, Cambridge, MA, USA: MIT Open-CourseWare*, 2015.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850, 2016.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1:2, 1987.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, 2009.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

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

J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pp. 7332–7342, 2018.

Xiao-Tong Yuan, Ping Li, and Tong Zhang. Gradient hard thresholding pursuit. *Journal of Machine Learning Research*, 18:1–43, 2018.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pp. 7693–7702, 2019.

A Technical Proofs

A.1 Proof of Theorem 1

We need the following lemma which guarantees the uniform convergence of $\mathcal{R}_S(\theta)$ towards $\mathcal{R}(\theta)$ for all θ when the loss function is Lipschitz continuous and smooth, and the optimization is limited on a bounded domain. The lemma is essentially an implication of the uniform convergence bound in (Shalev-Shwartz et al., 2009, Theorem 5) to our problem regime without imposing sparsity constraint.

Lemma 1. *Assume that the domain of interest $\Theta \subseteq \mathbb{R}^p$ is bounded by R and the loss function $\ell(f_\theta(\boldsymbol{x}), y)$ is G-Lipschitz continuous and H-smooth with respect to θ. Then for any $\delta \in (0,1)$, the following bound holds with probability at least $1-\delta$ over the random draw of sample set S for all $\theta \in \Theta$,*

$$|\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \leq \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{p\log(M)\log(p/\delta)}{M}}\right).$$

Proof. Let us denote $f(\theta) := \mathcal{L}_{\mathcal{D}^{supp}_{T_i}}\left(\theta - \eta\nabla_\theta \mathcal{L}_{\mathcal{D}^{supp}_{T_i}}(\theta)\right)$. Since $\ell(f_\theta(\boldsymbol{x}), y)$ is G-Lipschitz continuous with respect to θ, we can show that

$$\begin{aligned}|f(\theta) - f(\theta')| \leq& G\|\theta - \eta\nabla_\theta\mathcal{L}_{\mathcal{D}^{supp}_{T_i}}(\theta) - \theta' + \eta\nabla_\theta\mathcal{L}_{\mathcal{D}^{supp}_{T_i}}(\theta')\| \\ \leq& G\left(\|\theta - \theta'\| + \eta\|\nabla_\theta\mathcal{L}_{\mathcal{D}^{supp}_{T_i}}(\theta) - \nabla_\theta\mathcal{L}_{\mathcal{D}^{supp}_{T_i}}(\theta')\|\right) \\ \leq& G(1+\eta H)\|\theta - \theta'\|,\end{aligned}$$

which indicates that f is $G(1+\eta H)$-Lipschitz continuous. The desired bound then follows readily from the uniform convergence bound in (Shalev-Shwartz et al., 2009, Theorem 5). □

Based on this lemma, we can readily prove the main result in the theorem.

Proof of Theorem 1. For any fixed supporting set $J \in \mathcal{J}$, by applying Lemma 1 we obtain that the following uniform convergence bound holds for all w with supp$(\theta) \subseteq J$ with probability at least $1-\delta$ over S:

$$|\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \leq \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k/\delta)}{M}}\right).$$

Since by constraint the parameter vector θ is always k-sparse, we thus have supp$(\theta) \in \mathcal{J}$. Then by union probability we get that with probability at least $1-\delta$, the following bound holds for all w with $\|\theta\|_0 \leq k$:

$$|\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \leq \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k/\delta) + k\log(M)\log(|\mathcal{J}|)}{M}}\right).$$

It remains to bound the cardinality $|\mathcal{J}|$. From (Rigollet, 2015, Lemma 2.7) we know $|\mathcal{J}| = \binom{p}{k} \leq \left(\frac{ep}{k}\right)^k$, which then implies the desired generalization gap bound. This completes the proof. □

A.2 Proof of Theorem 2

Proof of Theorem 2. Let us now consider the following set of events associated with S:

$$\mathcal{E}_1 : \left\{S \in \mathcal{T}^M : \sup_{\text{supp}(\theta)\in\mathcal{J}} |\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| < \frac{\Delta}{2}\right\}, \quad \mathcal{E}_2 : \left\{S \in \mathcal{T}^M : \text{supp}(\theta_S) \in \mathcal{J}^*\right\},$$

$$\mathcal{E}_3 : \left\{S \in \mathcal{T}^M : \sup_{\text{supp}(\theta)\in\mathcal{J}^*} |\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \leq \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k|\mathcal{J}^*|/\delta)}{M}}\right)\right\},$$

and

$$\mathcal{E} : \left\{ S \in \mathcal{T}^M : |\mathcal{R}(\theta_S) - \mathcal{R}_S(\theta_S)| \le \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k|\mathcal{J}^*|/\delta)}{M}} \right) \right\}.$$

We first show that $\mathcal{E}_1 \subseteq \mathcal{E}_2$. Indeed, if the event $\mathcal{E}_1$ occurs, then from the optimality of θ_S we can see that for any θ with supp$(\theta) \in \mathcal{J}$

$$\mathcal{R}(\theta_S) < \mathcal{R}_S(\theta_S) + \frac{\Delta}{2} \le \mathcal{R}_S(\theta) + \frac{\Delta}{2} \le \mathcal{R}(\theta) + \Delta.$$

Particularly, for any θ with supp$(\theta) \in \mathcal{J}^*$, the above bound leads to $\mathcal{R}(\theta_S) - \mathcal{R}(\theta) < \Delta$ which according to the definition of Δ immediately implies supp$(\theta_S) \in \mathcal{J}^*$. Therefore, we must have $\mathcal{E}_1 \subseteq \mathcal{E}_2$. By invoking Theorem 1 we know that the following holds with probability at least $1-\delta/2$:

$$\sup_{\text{supp}(\theta)\in\mathcal{J}} |\mathcal{R}(\theta) - \mathcal{R}_S(\theta)| \le \mathcal{O}\left(GR(1+\eta H)\sqrt{\frac{k\log(M)\log(k/\delta) + k^2\log(M)\log(ep/k)}{M}} \right) < \frac{\Delta}{2},$$

which implies that under the given conditions on M,

$$\mathbb{P}(\mathcal{E}_2) \ge 1 - \delta/2.$$

Next we verify that $\mathcal{E} \cap \mathcal{E}_2 \supseteq \mathcal{E}_3 \cap \mathcal{E}_2$. Indeed, for any $S \in \mathcal{E}_3 \cap \mathcal{E}_2$, it follows immediately that $S \in \mathcal{E}$ under the condition on the sample size M, and thus $S \in \mathcal{E} \cap \mathcal{E}_2$. By invoking Lemma 1 with union probability over $\mathcal{J}^*$ we obtain that

$$\mathbb{P}(\mathcal{E}_3) \ge 1 - \delta/2.$$

Then, we can derive

$$\mathbb{P}(\mathcal{E}) \ge \mathbb{P}(\mathcal{E} \cap \mathcal{E}_2) \ge \mathbb{P}(\mathcal{E}_3 \cap \mathcal{E}_2) \ge 1 - \mathbb{P}(\overline{\mathcal{E}}_3) - \mathbb{P}(\overline{\mathcal{E}}_2) \ge 1 - \delta.$$

This proves the desired generalization gap bound. □

A.3 Proof of Corollary 3

The following lemma, which shows that the support of θ^* is unique under the provided conditions, is key to the proof the theorem.

Lemma 2. *Assume that $\mathcal{R}(\theta)$ is μ_{2k}-strongly convex. Suppose that there exists a k-sparse vector $\bar{\theta}$ such that $\|\bar{\theta}_l\|_0 \le k_l, \forall l \in [L]$ and*

$$\bar{\theta}_{\min} > \frac{2\sqrt{2k}\|\nabla\mathcal{R}(\bar{\theta})\|_\infty}{\mu_{2k}}.$$

Then $\bar{J} = supp(\bar{\theta})$ is the only element in $\mathcal{J}^*$.

Proof. Let us consider any fixed $J \in \mathcal{J}^*$ and denote $\theta^* = \arg\min_{\text{supp}(\theta)=J}\mathcal{R}(\theta)$. Then from the definition of $\mathcal{J}^*$ we have $\mathcal{R}(\theta^*) \le \mathcal{R}(\bar{\theta})$. Since $\mathcal{R}(\theta)$ is μ_{2k}-strongly convex,

$$\begin{aligned}
\mathcal{R}(\theta^*) \ge& \mathcal{R}(\bar{\theta}) + \langle \nabla R(\bar{\theta}), \theta^* - \bar{\theta} \rangle + \frac{\mu_{2k}}{2}\|\theta^* - \bar{\theta}\|^2 \\
\ge& \mathcal{R}(\bar{\theta}) - \|\nabla_{J\cup\bar{J}}\mathcal{R}(\bar{\theta})\|\|\theta^* - \bar{\theta}\| + \frac{\mu_{2k}}{2}\|\theta^* - \bar{\theta}\|^2 \\
\ge& \mathcal{R}(\theta^*) - \|\nabla_{J\cup\bar{J}}\mathcal{R}(\bar{\theta})\|\|\theta^* - \bar{\theta}\| + \frac{\mu_{2k}}{2}\|\theta^* - \bar{\theta}\|^2,
\end{aligned}$$

where the second inequality follows from Cauchy-Schwarz inequality. The above inequality then implies

$$\|\theta^* - \bar{\theta}\| \le \frac{2\|\nabla_{J\cup\bar{J}}\mathcal{R}(\bar{\theta})\|}{\mu_{2k}} \le \frac{2\sqrt{2k}\|\nabla\mathcal{R}(\bar{\theta})\|_\infty}{\mu_{2k}}.$$

Assume that $J \ne \bar{J}$. Then the above bound leads to

$$\bar{\theta}_{\min} \le \|\theta^* - \bar{\theta}\| \le \frac{2\sqrt{2k}\|\nabla\mathcal{R}(\bar{\theta})\|_\infty}{\mu_{2k}},$$

which contradicts the assumption on $\bar{\theta}$. Therefore, it must be true that $J = \bar{J}$ for any $J \in \mathcal{J}^*$. □

Proof of Corollary 3. The bound follows readily by combining Lemma 2 and Theorem 2. □

B Additional Experimental Results

This appendix contains full experimental results for Omniglot, MiniImageNet datasets, the consolidated version of the tables found in the main text is included as well.

B.1 Results on Omniglot dataset

Table 4: Few Shot Classification results on Omniglot dataset for 4-layer convolutional network with different channels on 5-way 1-shot and 5-way 5-shot tasks. The "±" shows 95% confidence intervals over tasks. The evaluation baselines are run by us.

Methods	**5-way 1-shot**	**5-way 5-shot**
Reptile baseline(32)	96.63±0.17%	99.31±0.07%
Reptile baseline(64)	97.68±0.10%	99.48±0.06%
Reptile baseline(128)	97.99±0.11%	99.60±0.13%
Reptile baseline(256)	98.05±0.13%	99.65±0.06%
DSD+Reptile(32, 10%)	96.42±0.17%	**99.38±0.07**%
DSD+Reptile(32, 20%)	95.98±0.18%	99.33±0.07%
DSD+Reptile(32, 30%)	96.22±0.17%	99.23±0.08%
DSD+Reptile(32, 40%)	96.53±0.17%	99.37±0.07%
IHT+Reptile(32, 10%)	**96.65±0.16**%	99.49±0.06%
IHT+Reptile(32, 20%)	96.54±0.17%	**99.57±0.06**%
IHT+Reptile(32, 30%)	96.45±0.17%	99.52±0.06%
IHT+Reptile(32, 40%)	96.21±0.18%	99.48±0.07%
DSD+Reptile(64, 10%)	**97.64±0.02**%	**99.50±0.05**%
DSD+Reptile(64, 20%)	97.60±0.07%	99.49±0.04%
DSD+Reptile(64, 30%)	97.47±0.05%	99.49±0.05%
DSD+Reptile(64, 40%)	97.43±0.01%	99.45±0.03%
IHT+Reptile(64, 10%)	97.63±0.14%	99.49±0.06%
IHT+Reptile(64, 20%)	97.60±0.13%	**99.57±0.06**%
IHT+Reptile(64, 30%)	**97.77±0.15**%	99.52±0.06%
IHT+Reptile(64, 40%)	97.51±0.1%	99.48±0.07%
DSD+Reptile(128, 10%)	**98.04±0.10**%	99.61±0.10%
DSD+Reptile(128, 20%)	97.99±0.10%	99.62±0.12%
DSD+Reptile(128, 30%)	97.96±0.12%	**99.63±0.12**%
DSD+Reptile(128, 40%)	97.99±0.10%	99.61±0.10%
IHT+Reptile(128, 10%)	98.12±0.12%	99.63±0.06%
IHT+Reptile(128, 20%)	**98.22±0.12**%	99.64±0.05%
IHT+Reptile(128, 30%)	98.01±0.13%	**99.65±0.05**%
IHT+Reptile(128, 40%)	98.06±0.12%	99.63±0.06%
DSD+Reptile(256, 10%)	**98.12±0.12**%	**99.68±0.05**%
DSD+Reptile(256, 20%)	98.02±0.13%	99.66±0.05%
DSD+Reptile(256, 30%)	97.96±0.13%	99.67±0.05%
DSD+Reptile(256, 40%)	97.99±0.10%	99.63±0.06%
IHT+Reptile(256, 10%)	**98.16±0.12**%	99.66±0.05%
IHT+Reptile(256, 20%)	98.08±0.13%	**99.69±0.05**%
IHT+Reptile(256, 30%)	98.05±0.13%	99.64±0.05%
IHT+Reptile(256, 40%)	97.90±0.13%	99.65±0.05%

We performed our methods on CNNs with 32, 64, 128, 256 channels respectively as mentioned in Section 5. The baselines and all the results of Omniglot dataset are reported in Figure 4. For each case, both DSD-based Reptile approach and IHT-based Reptile approach are evaluated on various pruning rates. The settings are the same as proposed in Section 5.1.

For 32-channel case and 64-channel cases, which is less prone to be overfitting, both DSD-based Reptile approach and IHT-based Reptile approach can perform well. On 5-way 1-shot task settings, it can be observed that the accuracy doesn't drop too much and can achieve similar performance as baselines. The accuracy of the worst result is the case of 32-channel CNNs with 20% pruning rate which drops about 0.65%, and the best one is the case of 64-channel CNNs with 30% pruning rate that improves about 0.1%. On 5-way 5-shot task settings, most cases outperform the baseline. As shown in the table, even in 32-channel case, when the pruning rate is 20%, the performance of IHT-based Reptile imporves about 0.26%. For 128-channel and 256-channel cases, our method consistently achieves better performance than the baselines. The most important is that with channels added, the baselines keep increasing, but our method can still improve the performance, which can manifest the impressive advantage of our method. Table **??** presents the hyper-parameters used in all the Omniglot experiments.

B.2 Results on MiniImageNet dataset

Here we report the detailed results of experiments on MiniImageNet dataset. We conduct our experiments with ResNet-18 and CNNs with 32, 64, 128, 256 channels based on the settings detailed in 5.2.

Table 5: Few Shot Classification results on MiniImageNet dataset for 4-layer convolutional network with different channels on 5 way setting. The "±" shows 95% confidence intervals over tasks. The evaluation baselines are run by us.

Methods	**5-way 1-shot**	**5-way 5-shot**
Reptile baseline(32)	50.30±0.40%	64.27±0.44%
Reptile baseline(64)	51.08±0.44%	65.46±0.43%
Reptile baseline(128)	49.96±0.45%	64.40±0.43%
Reptile baseline(256)	48.60±0.44%	63.24±0.43%
CAVIA baseline (32)	47.24±0.65%	59.05±0.54%
CAVIA baseline (128)	49.84±0.68%	64.63±0.54%
CAVIA baseline (512)	51.82±0.65%	65.85±0.55%
DSD+Reptile(32, 10%)	50.65±0.45%	65.29±0.44%
DSD+Reptile(32, 20%)	49.94±0.43%	64.65±0.43%
DSD+Reptile(32, 30%)	50.18±0.43%	**65.78±0.41**%
DSD+Reptile(32, 40%)	**50.83±0.45**%	65.24±0.44%
IHT+Reptile(32, 10%)	**50.45±0.45**%	63.91±0.46%
IHT+Reptile(32, 20%)	50.26±0.47%	63.63±0.45%
IHT+Reptile(32, 30%)	50.21±0.44%	**65.05±0.45**%
IHT+Reptile(32, 40%)	49.74±0.46%	64.15±0.45%
DSD+Reptile(64, 10%)	50.98±0.45%	65.31±0.43%
DSD+Reptile(64, 20%)	51.91±0.45%	67.21±0.43%
DSD+Reptile(64, 30%)	51.91±0.45%	**67.23±0.43**%
DSD+Reptile(64, 40%)	**51.96±0.45**%	66.64±0.43%
IHT+Reptile(64, 10%)	52.23±0.45%	66.08±0.43%
IHT+Reptile(64, 20%)	52.13±0.46%	66.78±0.43%
IHT+Reptile(64, 30%)	51.98±0.45%	66.14±0.43%
IHT+Reptile(64, 40%)	**52.59±0.45**%	**67.41±0.43**%
DSD+Reptile(128, 30%)	51.98±0.45%	68.16±0.43%
DSD+Reptile(128, 40%)	52.15±0.45%	68.19±0.43%
DSD+Reptile(128, 50%)	52.08±0.45%	**68.87±0.42**%
DSD+Reptile(128, 60%)	**52.27±0.45**%	68.44±0.42%
IHT+Reptile(128, 30%)	51.64±0.45%	67.05±0.43%
IHT+Reptile(128, 40%)	52.73±0.45%	**68.69±0.42**%
IHT+Reptile(128, 50%)	52.76±0.45%	67.63±0.43%
IHT+Reptile(128, 60%)	**52.95±0.45**%	68.04±0.42%
DSD+Reptile(256, 60%)	**53.00±0.45**%	**68.04±0.42**%
IHT+Reptile(256, 60%)	**49.85±0.44**%	**66.56±0.42**%
ResNet-18(baseline)	39.22±0.45%	-
DSD+ResNet-18	**47.23±0.47**%	-

From the table, we can obviously discover that our method achieves remarkable performance. From the perspective of channels, for each case, the performance of our method improves obviously and most results outperform the baselines. Meanwhile, our method reveals more impressive performance with more channels since that while the accuracy of baseline decreases with channels added in the network, the improvement becomes more prominent. For example, with DSD-based Reptile approach on 5-way 1-shot task settings, for 32-channel case, the accuracy of the best performance increases by about 0.5%, but for 64-channel, 128-channel case, the best improvement respectively reaches 2.31% and 4.4%. For IHT-based Reptie settings, we can notice such phenomenon as well. From the perspective of the pruning rate, we notice that for networks with different number of channels, when under the same condition of pruning rate, networks with more channels will achieve better performance. Although the overfitting problem on ResNet-18 is not addressed, it can be observed that our method eases such phenomenon obviously. Our method also outperforms CAVIA (Zintgraf et al., 2019), which can increase the network size without overfitting. These results demonstrate that by introducing dense-sparse-dense pruning strategy to meta-learning, the pruning phase can help to avoid learning the useless and thus ease the overfitting, and the retraining phase which preserves the capacity of the model assist to improve the performance of the generalization.