Meta-Transfer Learning through Hard Tasks

Qianru Sun*, Yaoyao Liu*, Zhaozheng Chen, Tat-Seng Chua, and Bernt Schiele, *Fellow, IEEE*

Abstract—Meta-learning has been proposed as a framework to address the challenging few-shot learning setting. The key idea is to leverage a large number of similar few-shot tasks in order to learn how to adapt a base-learner to a new task for which only a few labeled samples are available. As deep neural networks (DNNs) tend to overfit using a few samples only, typical meta-learning models use shallow neural networks, thus limiting its effectiveness. In order to achieve top performance, some recent works tried to use the DNNs pre-trained on large-scale datasets but mostly in straight-forward manners, e.g., (1) taking their weights as a warm start of meta-training, and (2) freezing their convolutional layers as the feature extractor of base-learners. In this paper, we propose a novel approach called **meta-transfer learning (MTL)**, which learns to transfer the weights of a deep NN for few-shot learning tasks. Specifically, *meta* refers to training multiple tasks, and *transfer* is achieved by learning scaling and shifting functions of DNN weights (and biases) for each task. To further boost the learning efficiency of MTL, we introduce the **hard task (HT) meta-batch** scheme as an effective learning curriculum of few-shot classification tasks. We conduct experiments for five-class few-shot classification tasks on three challenging benchmarks, *mini*ImageNet, *tiered*ImageNet, and Fewshot-CIFAR100 (FC100), in both supervised and semi-supervised settings. Extensive comparisons to related works validate that our MTL approach trained with the proposed HT meta-batch scheme achieves top performance. An ablation study also shows that both components contribute to fast convergence and high accuracy.

Index Terms—few-shot learning, transfer learning, meta learning, image classification

1 Introduction

Although deep learning systems have achieved great performance when sufficient amounts of labeled data are available [1]–[3], there has been growing interest in reducing the required amount of data. Few-shot learning tasks have been defined for this purpose. The aim is to learn new concepts from a handful of training examples, e.g., from 1 or 5 training images [4]–[6]. Humans tend to be highly effective in this context, often grasping the essential connection between new concepts and their own knowledge, but it remains challenging for machine learning models. For instance, on the CIFAR-100 dataset, a classification model trained in the fully supervised mode achieves 76% accuracy for the 100-class setting [7], while the best-performing 1-shot model achieves only 45% in average for the much simpler 5-class setting [6]. On the other hand, in many real-world applications, we lack large-scale training data, as e.g., in the medical domain. It is thus desirable to improve machine learning models in order to handle few-shot settings.

Basically, the nature of few-shot learning with very scarce training data makes it difficult to train powerful machine learning models for new concepts. People explore a variety of methods in order to overcome this. A straight-forward idea is to increase the amount of available data by data augmentation techniques [8]. Several methods proposed to learn a data generator e.g. conditioned on Gaussian noises [9]–[11] or object attributes [12]. However, this data generator often under-performs when trained on few-shot data, which has been investigated by [13]. An alternative is to merge data from multiple tasks which, however, is often ineffective due to high variances of the data across tasks [11].

In contrast to data augmentation methods, meta-learning is a task-level optimization-based method [14]–[16]. It aims to transfer experience from similar few-shot learning tasks [5], [6], [17]–[24]. Related methods follow a unified training process that contains two loops. The inner-loop learns a base-learner for an individual task, and the outer-loop then uses the validation performance of the learned base-learner to optimize the meta-learner. A state-of-the-art representative method named Model-Agnostic Meta-Learning (MAML) learns to search for the optimal initialization state to fast adapt a base-learner to a new task [5]. Its task-agnostic property makes it possible to generalize to few-shot supervised/semi-supervised learning as well as unsupervised reinforcement learning [5], [17], [18], [21], [22], [25]–[27]. However, in our view, there are two main limitations of this type of approaches limiting their effectiveness: i) these methods usually require a large number of similar tasks for meta-training which is costly; and ii) each task is typically modeled by a low-complexity base-learner, such as a shallow neural network (SNN), to avoid model overfitting to few-shot training data, thus being unable to deploy the deeper and more powerful architectures. For example, for the *mini*ImageNet dataset [28], MAML uses a *shallow* CNN with only 4 CONV layers and its optimal performance was obtained by learning on $240k$ tasks ($60k$ iterations in total and each meta-batch contains 4 tasks).

In this paper, we propose a novel meta-learning method called **meta-transfer learning (MTL)** leveraging the advantages of both transfer learning and meta-learning. In a nutshell, MTL is a novel learning method that helps deeper neural networks based base-learners converge faster while reducing their probability to overfit when training on a few labeled data only. In particular, "transfer" means that DNN weights trained on large-scale data can be used in other tasks by two light-weight neuron operations: *Scaling* and *Shifting* (*SS*), i.e. $\alpha X + \beta$. "Meta" means that the parameters of

- *Qianru Sun is the corresponding author. qianrusun@smu.edu.sg. She and Zhaozheng Chen are with the School of Information Systems, Singapore Management University. Yaoyao Liu and Bernt Schiele are with the Department of Computer Vision and Machine Learning, Max Planck Institute for Informatics. Tat-Seng Chua is with the School of Computing, National University of Singapore. This work was done during Yaoyao's internship supervised by Qianru.*
- ** indicates equal contribution.*
- *The code is open-sourced at this link: https://github.com/yaoyao-liu/meta-transfer-learning.*

Manuscript received September 29, 2019; revised April 27, 2020; accepted August 17, 2020.

these *SS* operations can be viewed as hyper-parameters learned with few-shot learning tasks [19], [29], [30]. First, large-scale trained DNN weights offer a good initialization, enabling fast convergence of MTL with fewer tasks, e.g., only $8k$ tasks for *mini*ImageNet [28], 30 times fewer than MAML [5]. Second, light-weight operations on DNN neurons have less parameters to learn, e.g., less than $\frac{2}{49}$ if considering neurons of size 7×7 ($\frac{1}{49}$ for α and $< \frac{1}{49}$ for β), reducing the chance of overfitting to few-shot data. Third, these operations keep those trained DNN weights unchanged and thus avoid the problem of "catastrophic forgetting" which means forgetting general patterns when adapting to a specific task [31], [32]. Finally, these operations are conducted on the convolutional layers mostly working for image feature extraction, thus can generalize well to a variety of few-shot learning models, e.g., MAML [5], MatchingNet [28], ProtoNet [33], RelationNet [34] and SIB [24].

The second main contribution of this paper is an effective meta-training curriculum. Curriculum learning [35] and hard negative mining [36] both suggest that faster convergence and stronger performance can be achieved by better arrangements of training data, i.e., the few-shot training tasks in our case. Inspired by these ideas, we design our **hard task (HT) meta-batch** strategy to offer a challenging but effective learning curriculum. The conventional meta-batch contains a number of random tasks [5], but our HT meta-batch online re-samples harder ones according to past failure tasks with lowest validation accuracy. In addition, we add the meta-gradient regularization on each task that each task is optimized by using the weighted sum of meta-gradients of both current and previous tasks. The aim is to force the meta-learner not to forget old knowledge in afterward learning.

Our overall contribution is thus three-fold: i) we propose a novel **MTL** method that learns to transfer large-scale pre-trained DNN weights for solving few-shot learning tasks; ii) we propose a novel **HT meta-batch** learning strategy that forces meta-transfer to "grow faster and stronger through hardship"; and iii) we conduct extensive experiments on three few-shot learning benchmarks, namely *mini*ImageNet [28], *tiered*ImageNet [26] and Fewshot-CIFAR100 (FC100) [37], and achieve the state-of-the-art performance on both supervised and semi-supervised few-shot learning. **Compared to the conference version of the paper [6], this paper additionally presents** (1) the analysis of using different DNN architectures and baseline methods (by plug-in), e.g., ResNet-12, ResNet-18, ResNet-25 and WRN-28-10; (2) the new MTL variants that adapt our scaling and shifting functions to the state-of-the-art supervised as well as semi-supervised meta-learners, and achieve performance boosts consistently; (3) the discussion of new related works since the conference version; and (4) the results on the larger and more challenging benchmark – *tiered*ImageNet [26].

2 Related work

Research literature on few-shot learning exhibits great diversity, spanning from data augmentation [9]–[12] to supervised meta-learning [5], [6], [16]–[18], [20]–[24], [38], [39]. In this paper, we focus on the meta-learning based methods most relevant to ours and compared to in the experiments. Besides, we borrow the idea of transfer learning when leveraging the large-scale pre-training step in prior to meta-transfer. For task sampling, our HT meta-batch scheme is related to curriculum learning and hard negative sampling methods.

Meta-learning. We can divide meta-learning methods into three categories. 1) *Metric learning* methods [28], [33], [34], [40]–[44] learn a similarity space in which learning is particularly efficient for few-shot training examples. Examples of distance metrics include cosine similarity [28], [44], [45], Euclidean distance to the prototypical representation of a class [33], CNN-based relation module [34], ridge regression based [46], and graph model based [47], [48]. Some recent works also tried to generate task-specific feature representation for few-shot episodes based on metric learning, like [40], [49] 2) *Memory network* methods [29], [37], [44], [50], [51] learn to store "experience" when learning seen tasks and then generalize it to unseen tasks. The key idea is to design a model specifically for fast learning with a few training steps. A family of model architectures use external memory storage include Neural Turing Machines [52], Meta Networks [29], Neural Attentive Learner (SNAIL) [50], and Task Dependent Adaptive Metric (TADAM) [37]. For test, general meta memory and specific task information are combined to make predictions in neural networks. 3) *Gradient descent* based meta-learning methods [5], [6], [17], [18], [20]–[24], [39], [53] intend for adjusting the optimization algorithm so that the model can converge within a small number of optimization steps (with a few examples). The optimization algorithm can be explicitly modeled with two learning loops that outer-loop has a *meta-learner* that learns to adapt an inner-loop *base-learner* (to few-shot examples) through different tasks. For example, Ravi *et al*. [39] introduced a method that compresses the base-learners' parameter space in an LSTM meta-learner. Rusu *et al*. [25] designed a classifier generator as the meta-learner which output parameters for each specific base-learning task. Finn *et al*. [5] proposed a meta-learner called MAML that learns to effectively initialize a base-learner for a new task. Lee *et al*. [53] proposed MT-net, where the meta-learner determines a sub-space and a corresponding metric that task-specific learners can learn in, thus setting the degrees of freedom of task-specific learners to an appropriate amount. Lee *et al*. [23] presented a meta-learning approach with convex base-learners for few-shot tasks. Other related works in this category include Hierarchical Bayesian model [18], Bilevel Programming [19], and GAN based meta model [21]. Hu et al. [24] proposed to update base-learner with synthetic gradients generated by a variational posterior conditional on unlabeled data.

Among them, MAML is a fairly general optimization algorithm, compatible with any model that learns through gradient descent. Its meta-learner optimization is done by gradient descent using the validation loss of the base-learner. It is closely related to our MTL. An important difference is that MTL leverages transfer learning and benefits from referencing neuron knowledge in pre-trained deep nets. Although MAML can start from a pre-trained network, its element-wise fine-tuning makes it hard to learn deep nets without overfitting (validated in our experiments).

Transfer learning. Transfer learning or knowledge transfer has the goal to transfer the information of trained models to solve unknown tasks, thereby reducing the effort to collect new training data. *What* and *how* to transfer are key issues to be addressed. Different methods are applied to different source-target domains and bridge different transfer knowledge [54]–[60]. For deep models, a powerful transfer method is adapting a pre-trained model for a new task, often called *fine-tuning* (*FT*). Models pre-trained on large-scale datasets have proven to generalize better than randomly initialized ones [61]. Another popular transfer method

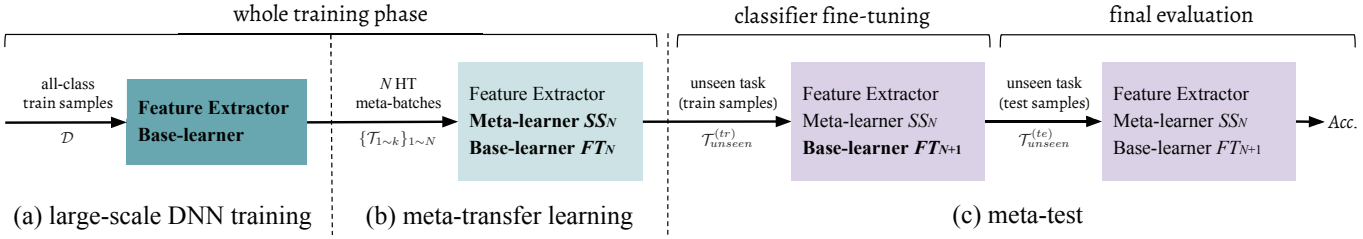


Fig. 1. The pipeline of our proposed few-shot learning method, including: (a) DNN pre-training on large-scale data, i.e. using the entire training dataset; and (b) meta-transfer learning (MTL) that learns the parameters of *Scaling* and *Shifting* (*SS*), on the basis of pre-trained feature extractor (Section 4.1). The learning process is scheduled by the proposed HT meta-batch (Section 4.2) and regularized by meta-gradient regularization (Section 4.3). In (c), it is meta-test on unseen task whose processing consists of a base-learner (classifier) *Fine-Tuning* (*FT*) stage and a final evaluation stage, described in the last paragraph in Section 3. Input data are along with arrows. Modules with names in bold get updated at corresponding phases.

is taking pre-trained networks as backbone and adding high-level functions, e.g. for object detection [62] and image segmentation [63], [64]. Besides, the knowledge to transfer can be from multi-modal category models, e.g. the word embedding models used for zero-shot learning [12], [65] and trained attribute models used for social relationship recognition [58].

In this paper, our meta-transfer learning leverages the idea of transferring pre-trained weights and our model meta-learns how to effectively transfer. The large-scale trained DNN weights are *what* to transfer, and the operations of *Scaling* and *Shifting* indicate *how* to transfer. Some existing few-shot learning methods [25], [50], [66]–[68] also deployed pre-trained DNNs. DNN weights in these methods are usually fixed for feature extraction or simply fine-tuned on each task. In contrast, our approach defines an explicit meta-learner to extract and apply usable knowledge of pre-learned DNNs to tackling the challenging few-shot learning tasks.

Curriculum learning & hard sample mining. Curriculum learning was proposed by Bengio *et al*. [35] and is popular for multi-task learning [69]–[71]. They showed that instead of observing samples at random it is better to organize samples in a meaningful way so that fast convergence, effective learning and better generalization can be achieved. Kumar *et al*. [72] introduced an iterative self-paced learning algorithm where each iteration simultaneously selects easy samples and learns a new parameter vector. Intuitively, the curriculum is determined by the pupil's abilities rather than being fixed by a teacher. Pentina *et al*. [73] use adaptive SVM classifiers to evaluate task difficulty for later organization. Most recently, Jiang *et al*. [74] designed a MentorNet that provides a "curriculum", i.e., sample weighting scheme, for StudentNet to focus on the labels which are probably correct. The trained MentorNet can be directly applied for the training of StudentNet on a new dataset. Differently, our MTL method does task difficulty evaluation online at the phase of test in each task, without needing any auxiliary model.

Hard sample mining was proposed by Shrivastava *et al*. [36] for object detection with DNNs. It treats image proposals overlapped with ground truth (i.e. causing more confusions) as hard negative samples. Training on more confusing data enables the detection model to achieve higher robustness and better performance [75]–[77]. Inspired by this, we sample harder tasks online and make our MTL learner "grow faster and stronger through more hardness". In our experiments, we show that this can be generalized to different architectures with different meta-training operations, i.e. *SS* and *FT*, referring to Figure 4.

3 Preliminary

In this section, we briefly introduce the unified episodic formulation in meta-learning, following related works [5], [25], [28], [37], [39]. Then, we introduce the task-level data denotations used at two phases, i.e., meta-train and meta-test.

Meta-learning has an episodic formulation that was proposed for tackling few-shot tasks first in [28]. It is different from traditional image classification, in three aspects: (1) the main phases are not train and test but meta-train and meta-test, each of which includes training and testing; (2) the samples in meta-train and meta-test are not datapoints but episodes, and each episode is a few-shot classification task; and (3) the objective is not classifying unseen datapoints but to fast adapt the meta-learned experience or knowledge to the learning of a new few-shot classification task.

The denotations of two phases, meta-train and meta-test, are as follows. A meta-train example is a classification task $\mathcal{T}$ sampled from a distribution $p(\mathcal{T})$. $\mathcal{T}$ is called episode, including a training split $\mathcal{T}^{(tr)}$ to optimize the base-learner, i.e., the classifiers in our model, and a test split $\mathcal{T}^{(te)}$ to optimize the meta-learner, i.e., the scaling and shifting parameters in our model. In particular, meta-train aims to learn from a number of episodes $\{\mathcal{T}\}$ sampled from $p(\mathcal{T})$. An unseen episode $\mathcal{T}_{unseen}$ in meta-test will start from that experience of the meta-learner and adapt the base-learner. The final evaluation is done by testing a set of unseen datapoints in $\mathcal{T}^{(te)}_{unseen}$.

Meta-train phase. This phase aims to learn a meta-learner from multiple episodes. In each episode, meta-training has a two-stage optimization. Stage-1 is called base-learning, where the cross-entropy loss is used to optimize the parameters of the base-learner. Stage-2 contains a feed-forward test on episode test datapoints. The test loss (also called meta loss) is used to optimize the parameters of the meta-learner. Specifically, given an episode $\mathcal{T} \in p(\mathcal{T})$, the base-learner $\theta_{\mathcal{T}}$ is learned from episode training data $\mathcal{T}^{(tr)}$ and its corresponding loss $\mathcal{L}_{\mathcal{T}}(\theta_{\mathcal{T}}, \mathcal{T}^{(tr)})$. After optimizing this loss, the base-learner has parameters $\tilde{\theta}_{\mathcal{T}}$. Then, the meta-learner is updated using meta loss $\mathcal{L}_{\mathcal{T}}(\tilde{\theta}_{\mathcal{T}}, \mathcal{T}^{(te)})$. After meta-training on all episodes, the meta-learner is optimized by meta losses $\{\mathcal{L}_{\mathcal{T}}(\tilde{\theta}_{\mathcal{T}}, \mathcal{T}^{(te)})\}_{\mathcal{T}\in p(\mathcal{T})}$. Therefore, the number of meta-learner updates equals to the number of episodes.

Meta-test phase. This phase aims to test the performance of the trained meta-learner for fast adaptation to unseen episodes. Given $\mathcal{T}_{unseen}$, the meta-learner $\tilde{\theta}_{\mathcal{T}}$ teaches the base-learner $\theta_{\mathcal{T}_{unseen}}$ to adapt to the objective of $\mathcal{T}_{unseen}$ by some means, e.g. through initialization [5]. Then, the test result on $\mathcal{T}^{(te)}_{unseen}$ is used to evaluate the meta-learning approach. If there are multiple unseen

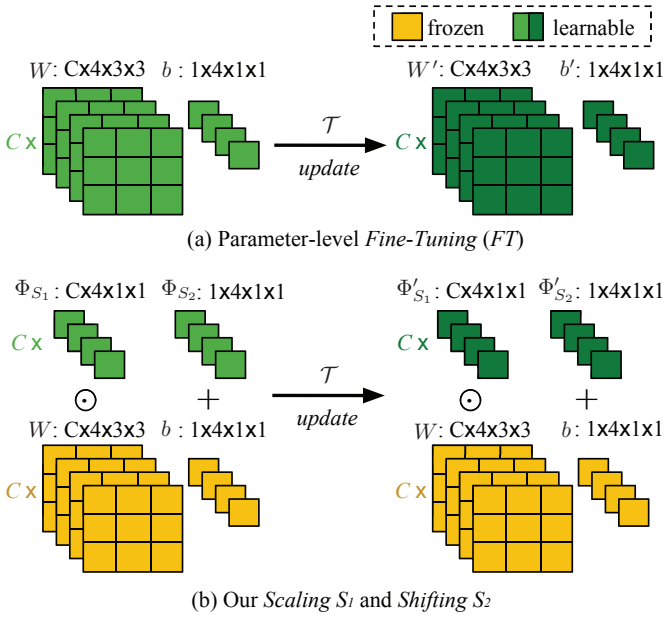


Fig. 2. Two kinds of meta operations on pre-trained weights. (a) Parameter-level *Fine-Tuning* (*FT*) is a conventional meta-train operation used in related works such as MAML [5], ProtoNets [33] and RelationNets [34]. Its update works for all neuron parameters, W and b. (b) Our neuron-level *Scaling* and *Shifting* (*SS*) operations in MTL. They reduce the number of learning parameters and avoid overfitting problems. In addition, they keep large-scale trained parameters (in yellow) frozen, preventing "catastrophic forgetting" [31], [32].

episodes $\{\mathcal{T}_{unseen}\}$, the average result on $\{\mathcal{T}^{(te)}_{unseen}\}$ will be the final evaluation.

4 Methodology

As shown in Figure 1, our method consists of three main training phases in order to achieve effective few-shot classifiers. First, we train a DNN on large-scale data, e.g., on *mini*ImageNet with 64 classes and 600 samples per class [28], and then fix convolutional layers as the Feature Extractor. Second, in the meta-transfer learning phase, our MTL learns the *Scaling* and *Shifting* (*SS*) parameters for the neurons of Feature Extractor, enabling the fast adaptation to few-shot episodes (Section 4.1). To boost the overall learning efficiency, we apply the HT meta-batch scheme (Section 4.2) and the meta-gradient regularization (Section 4.3) to the meta-train phase. The overall algorithm of our approach is given in Section 4.4. Finally in Section 4.5, we introduce how to plug this algorithm into existing methods.

4.1 Meta-transfer learning (MTL)

During pre-training, we merge all data $\mathcal{D}$ and derive the many-shot many-class model using the cross-entropy loss. The model is composed of the Feature Extractor Θ and a many-class classifier. The Θ keeps frozen in the following meta-training and meta-test phases, as shown in Figure 1. The many-class classifier is discarded, because few-shot episodes contain different classification objectives, e.g., 5-class instead of 64-class classification for *mini*ImageNet [28].

As shown in Figure 1(b), our MTL optimizes only the meta operations *Scaling* and *Shifting* (*SS*) through HT meta-batch training (Section 4.2). Figure 2 visualizes the difference of updating through *SS* and *FT* (*Fine-Tuning*). *SS* operations, denoted as Φ_{S_1} and Φ_{S_2}, do not change the frozen neuron weights of Θ during learning, while *FT* updates the complete Θ. Note that this *FT* is distinct from the Base-learner *FT* (on θ).

In the following, we expand the details of *SS* operations corresponding to Figure 1(b). Given an episode $\mathcal{T}$, the loss of $\mathcal{T}^{(tr)}$ is used to optimize the current base-learner (classifier) θ' by gradient descent:

$$\theta' \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\mathcal{T}^{(tr)}}\left([\Theta;\theta], \Phi_{S_{\{1,2\}}}\right), \tag{1}$$

where θ concerns a few classes, e.g., 5 classes, to classify each time in a novel few-shot setting. θ' corresponds to a temporal classifier working only in the current episode, initialized by the θ optimized by previous episodes (see Eq. (3)).

Φ_{S_1} is initialized by ones and Φ_{S_2} by zeros. Then, they are optimized by the meta loss of $\mathcal{T}^{(te)}$ as follows,

$$\Phi_{S_i} =: \Phi_{S_i} - \gamma \nabla_{\Phi_{S_i}} \mathcal{L}_{\mathcal{T}^{(te)}}\left([\Theta;\theta'], \Phi_{S_{\{1,2\}}}\right), i = 1, 2. \tag{2}$$

In this step, θ is updated with the same learning rate γ as in Eq. (2),

$$\theta =: \theta - \gamma \nabla_\theta \mathcal{L}_{\mathcal{T}^{(te)}}\left([\Theta;\theta'], \Phi_{S_{\{1,2\}}}\right). \tag{3}$$

Re-linking to Eq. (1), we note that the above θ' comes from the last epoch of base-learning on $\mathcal{T}^{(tr)}$.

Next, we describe how we apply $\Phi_{S_{\{1,2\}}}$ to the frozen neurons as shown in Figure 2(b). Given the trained Θ, for its l-th layer containing K neurons, we have K pairs of parameters, respectively as weight and bias, denoted as $\{(W_{i,k}, b_{i,k})\}$. Note that the neuron location l, k will be omitted for readability. Based on MTL, we learn K pairs of scalars $\{\Phi_{S_{\{1,2\}}}\}$. Assuming X is the input, we apply $\{\Phi_{S_{\{1,2\}}}\}$ to (W, b) as,

$$SS(X; W, b; \Phi_{S_{\{1,2\}}}) = (W \odot \Phi_{S_1})X + (b + \Phi_{S_2}), \tag{4}$$

where $\odot$ denotes the element-wise multiplication.

Taking Figure 2(b) as an example of a single 3×3 filter, after *SS* operations, this filter is scaled by Φ_{S_1} then the feature maps after convolutions are shifted by Φ_{S_2} in addition to the original bias b. Detailed steps of *SS* are given in Algorithm 1 in Section 4.4.

Figure 2(a) shows a typical parameter-level *FT* operation, which is in the meta optimization phase of our related work MAML [5]. It is obvious that *FT* updates the complete values of W and b, and has a large number of parameters, and our *SS* reduces this number to below $\frac{2}{9}$ in the example of the figure. In summary, *SS* can benefit the few-shot learning model in three aspects. 1) It starts from a strong initialization based on a large-scale trained DNN, yielding fast convergence for MTL. 2) It does not change DNN weights, thereby avoiding the problem of "catastrophic forgetting" [31], [32] when learning specific episodes in MTL. 3) It is light-weight, reducing the chance of overfitting of MTL in few-shot scenarios. In experiments, we compare *SS* with *FT* based on multiple baseline methods, and show the clear superiority of *SS* against the problem of "forgetting".

4.2 Hard task (HT) meta-batch

In this section, we introduce a method to schedule hard tasks in meta-training batches. The conventional meta-batch is composed of randomly sampled episodes, where the randomness implies random difficulties [5]. In our meta-training pipeline, we intentionally pick up failure cases in each episode and re-compose their data to be harder episodes for adverse re-training. The task flow is shown in Figure 3. We aim to force our meta-learner to "grow up through hardness".

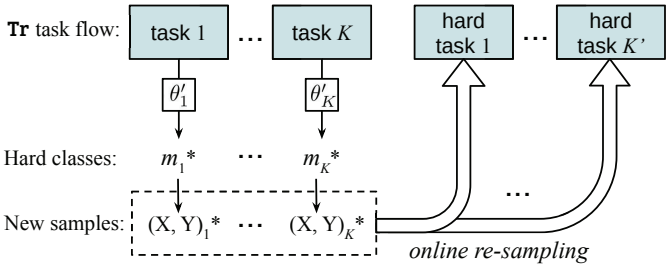


Fig. 3. The computation flow of online hard task sampling. During an HT meta-batch phase, the meta-training first goes through K random tasks then continues on re-sampled K' hard tasks.

Pipeline. Given a (M-class, N-shot) episode $\mathcal{T}$, a meta-batch $\{\mathcal{T}_{1\sim k}\}$ contains two splits, $\mathcal{T}^{(tr)}$ and $\mathcal{T}^{(te)}$, for base-learning and test, respectively. The base-learner is optimized by the loss of $\mathcal{T}^{(tr)}$ (in multiple epochs). *SS* parameters are then optimized by the loss of $\mathcal{T}^{(te)}$ once. During the loss computation on $\mathcal{T}^{(te)}$, we can also get the recognition accuracy for M classes. Then, we choose the lowest accuracy Acc_{m^*} to determine the most difficult class m^* (also called failure class) in the current episode.

After obtaining all failure classes $\{m^*\}$ from $\{\mathcal{T}_{1\sim k}\}$ in current meta-batch (k is the batch size), we re-sample episodes from the data indexed by $\{m^*\}$. Specifically, we assume $p(\mathcal{T}|\{m^*\})$ is the task distribution, we sample a "harder" episode $\mathcal{T}^{hard} \in p(\mathcal{T}|\{m^*\})$. Two important details are given below.

Choosing hard class m^*. We choose the failure class $m*$ from each episode by ranking the class-level accuracies instead of fixing a threshold. In a dynamic online setting as ours, it is more sensible to choose the hardest cases based on ranking rather than fixing a threshold ahead of time.

Two methods of hard tasking using $\{m^*\}$. Chosen $\{m^*\}$, we can re-sample episodes $\mathcal{T}^{hard}$ by (1) directly using the samples of m^*-th class in the current episode $\mathcal{T}$, or (2) indirectly using the index m^* to sample new samples of that class. In fact, setting (2) considers to include more data variance of m^*-th class and it works better than setting (1) in general.

4.3 Meta-gradient regularization

In order to further reduce the "catastrophic forgetting" problem, we deploy an easy and efficient meta-gradient regularization method for each training episode. Particularly, we apply this regularization to updating $\Phi_{S_{\{1,2\}}}$ and θ. Let q denote the index of current episode. Let $\nabla\mathcal{L}_{\mathcal{T}_r^{(te)}}$ be the gradient of the r-th episode. Eq. (2) and Eq. (3) can be rewritten as,

$$\begin{aligned}\Phi_{S_i} =:&\Phi_{S_i} - \gamma\nabla_{\Phi_{S_i}}\mathcal{L}_{\mathcal{T}_q^{(te)}}\left([\Theta;\theta'],\Phi_{S_{\{1,2\}}}\right)\\ &- \gamma\psi_1\sum_{r=q-p}^{q-1}\nabla_{\Phi_{S_i}}\mathcal{L}_{\mathcal{T}_r^{(te)}}\left([\Theta;\theta'],\Phi_{S_{\{1,2\}}}\right), i=1,2;\end{aligned} \tag{5}$$

$$\begin{aligned}\theta =:&\theta - \gamma\nabla_{\theta}\mathcal{L}_{\mathcal{T}_q^{(te)}}\left([\Theta;\theta'],\Phi_{S_{\{1,2\}}}\right)\\ &- \gamma\psi_2\sum_{r=q-p}^{q-1}\nabla_{\theta}\mathcal{L}_{\mathcal{T}_r^{(te)}}\left([\Theta;\theta'],\Phi_{S_{\{1,2\}}}\right),\end{aligned} \tag{6}$$

where ψ_1 and ψ_2 are two temperature scalars to balance the weights of the meta-gradients from current and previous episodes.

Algorithm 1: MTL with HT meta-batch strategy

Input: Task distribution $p(\mathcal{T})$ and corresponding dataset $\mathcal{D}$, learning rates α, β and γ
Output: Feature extractor Θ, base learner θ, *Scaling* and *Shifting* parameters $\Phi_{S_{\{1,2\}}}$

1 Randomly initialize Θ and θ;
2 **for** *samples in* $\mathcal{D}$ **do**
3 Evaluate $\mathcal{L}_{\mathcal{D}}([\Theta;\theta])$;
4 Optimize Θ and θ;
5 **end**
6 Initialize Φ_{S_1} by ones, initialize Φ_{S_2} by zeros; Reset θ for few-shot episodes; Randomly initialize θ; Initialize $\{m^*\}$ as an empty set.
7 **for** *iterations in meta-training* **do**
8 Randomly sample a batch of episodes $\{\mathcal{T}_{1\sim K}\} \in p(\mathcal{T})$;
9 **for** k *from* 1 *to* K **do**
10 **for** *samples in* $\mathcal{T}_k^{(tr)}$ **do**
11 Evaluate $\mathcal{L}_{\mathcal{T}_k^{(tr)}}$;
12 Optimize θ' by Eq. (1);
13 **end**
14 Optimize $\Phi_{S_{\{1,2\}}}$ and θ by Eq. (5) and Eq. (6);
15 **for** $m \in \{1\sim M\}$ **do**
16 Classify samples of m-th class in $\mathcal{T}_k^{(te)}$;
17 Compute Acc_m;
18 **end**
19 Get the returned m^*-th class, then add it to set $\{m^*\}$;
20 **end**
21 Sample hard tasks $\{\mathcal{T}^{hard}\} \subseteq p(\mathcal{T}|\{m^*\})$;
22 **for** k *from* 1 *to* K' **do**
23 Sample episode $\mathcal{T}_k^{hard} \in \{\mathcal{T}^{hard}\}$;
24 **for** *samples in* $\mathcal{T}_k^{hard,(tr)}$ **do**
25 Evaluate $\mathcal{L}_{\mathcal{T}_k^{hard,(tr)}}$;
26 Optimize θ' by Eq. (1);
27 **end**
28 Optimize $\Phi_{S_{\{1,2\}}}$ and θ by Eq. (5) and Eq. (6);
29 **end**
30 Empty $\{m^*\}$.
31 **end**

4.4 The overall algorithm

We elaborate the training process using our approach in Algorithm 1. There are two main training stages: large-scale DNN training (line 1-5) and meta-transfer learning (line 6-31). In particular, the proposed HT meta-batch sampling (with the subsequent training) are given on line 21-30. Note that the indices of failure classes are returned on line 19.

4.5 Plug MTL into baseline methods

Conventional supervised few-shot learning methods include metric learning based (e.g., ProtoNets [33], MatchingNets [28], and RelationNets [34]) and optimization based (e.g., MAML [5]). For semi-supervised few-shot learning, there are Masked Soft k-Means [26], TPN [48], and LST [27]. The neural network architecture in these methods is often composed of two modules, i.e. convolutional-layer feature extractor Θ and fully-connected-

layer classifier θ. Our MTL operations *SS* are conducted on convolutional neurons, so they are generic and easy to plug in Θ.

First, we pre-train Θ on a many-shot classification task using the whole set of $\mathcal{D}$. Then, we plug-in *Scaling* and *Shifting* weights Φ_{SS} on each neuron of Θ and update them with meta loss. Given an episode $\mathcal{T}$, we feed training images $x^{(tr)}$ and test images $x^{(te)}$ to the feature extractor $\Theta \odot \Phi_{SS}$, and obtain the embedding $e^{(tr)}$ and $e^{(te)}$, respectively. We apply different classifier architectures [5], [26]–[28], [33], [34], [48] to train classifiers with $e^{(tr)}$ and $y^{(tr)}$, and then test with $e^{(te)}$ resulting in the predictions $\hat{y}^{(te)}$. We then compute the test loss using $\hat{y}^{(te)}$ and $y^{(te)}$. Using this loss, we proceed meta-gradient back-propagation to update Φ_{SS} as well as the original meta-learner proposed in the baseline methods, e.g. the initialization network of base-learner θ in MAML [5]. In experiments, we report all our plug-in results compared to those of using *FT* operations (see Table 4 and Table 5).

5 Experiments

We evaluate the proposed approach in terms of few-shot recognition accuracy and model convergence speed. Below we describe the datasets we evaluate on and detailed settings, followed by the comparisons to state-of-the-art methods, validations on several baseline methods with *SS* plugin, and an ablation study regarding the key components of our approach, i.e., *SS* operations, HT meta-batch, and meta-gradient regularization. In the end, we demonstrate the statistical numbers and Gaussian fitting curves for the meta-learned *SS* parameters.

5.1 Datasets

We conduct few-shot learning experiments on three benchmarks, *mini*ImageNet [28], *tiered*ImageNet [26] and Fewshot-CIFAR100 (FC100) [37]. *mini*ImageNet is the most widely used in related works [5], [18], [19], [39], [78], and the later ones are more recently published with a larger scale and a more challenging setting, i.e., lower image resolution and stricter training-test splits.

***mini*ImageNet [28].** It was proposed especially for the few-shot learning evaluation [28]. Its complexity is high due to the use of ImageNet images, but it requires less resource and infrastructure than running on the full ImageNet dataset [79]. In total, there are 100 classes with 600 samples of 84×84 color images per class. These 100 classes are divided into 64, 16, and 20 classes respectively for sampling episodes for meta-train, meta-validation and meta-test, following related works [5], [18], [19], [39], [78].

***tiered*ImageNet [26].** Compared to *mini*ImageNet, it is a larger subset of ImageNet with 608 classes (779, 165 images) grouped into 34 super-class nodes. These nodes are partitioned into 20, 6, and 8 disjoint sets respectively for meta- training, validation, and test. The corresponding sub-classes are used to build the classification tasks in each of which the 5 sub-classes are randomly sampled. As argued in [26], this super-class based training-test split results in a more challenging and realistic regime with meta-test and validation episodes that are less similar to meta-train episodes.

Fewshot-CIFAR100 (FC100) [37]. This dataset is based on the popular object classification dataset CIFAR100 [80]. Its training-test splits are also based on super-classes [37]. In total, it contains 100 object classes (600 images per class) belonging to 20 super-classes. meta-train data are from 60 classes belonging to 12 super-classes. Meta-validation and meta-test sets contain 20 classes belonging to 4 super-classes, respectively. Comparing to the ImageNet subsets above, FC100 offers a more challenging scenario with lower image resolution, i.e. each sample is a 32×32 color image. In addition, the super-class gap on FC100 is more significant than that on ImageNet datasets.

Semi-supervised splits. On *mini*ImageNet and *tiered*ImageNet, we follow the semi-supervised task splitting method used in previous works [26], [27], [48]. In addition to the supervised data (same as above), we use 30 (50) unlabeled images per class for every 1-shot (5-shot) episode. In a more difficult setting, we use unlabeled data from 3 distracting classes (same number of samples with non-distracting classes) that are excluded in the support set [26], [48]

5.2 Implementation details

Episode sampling. We use the same episode sampling method as related works [5], on all datasets. Specifically, (1) we consider the 5-class classification, (2) during meta-train, we sample 5-class, 1-shot (or 5-shot) episodes to contain 1 (or 5) samples for train episode and 15 (uniform) samples for episode test, and (3) during meta-validation and meta-test, we sample 5-class, 1-shot (or 5-shot) episodes to contain 1 (or 5) samples for train episode and 1 (uniform) sample for episode test. Note that in some related works, e.g., [37], 32 samples are used for episode test on 5-shot episodes. Using such a larger number of test samples results in the lower standard variance of recognition accuracies.

In total, we sample at most $20k$ episodes ($10k$ meta-batches) for meta-train (same for the cases *w/* and *w/o* HT meta-batch), and sample 600 random episodes for both meta-validation and meta-test [5]. Note that we choose the trained models which have the highest meta-validation accuracies, for meta-test.

Network architectures. We present the details of network architectures for Feature Extractor parameters Θ, MTL meta-learner with *Scaling* and *Shifting* (*SS*) parameters Φ_{S_1}, Φ_{S_2}, and MTL base-learner (classifier) parameters θ. For Θ, in our conference version [6], we used ResNet-12 and 4CONV which are also commonly used in previous works [5], [23], [28], [37], [39], [50], [78]. In this journal version, we implement two deeper architectures – ResNet-18, ResNet-25 and WRN-28-10 which have been adopted in newly published related works [24], [40], [67], [81], and we achieve the top performance using ResNet-25 and WRN-28-10. In specific, **4CONV** consists of 4 layers with 3×3 convolutions and 32 filters, followed by batch normalization (BN) [82], a ReLU nonlinearity, and 2×2 max-pooling. MTL only works with the following deep nets. **ResNet-12** contains 4 residual blocks and each block has 3 CONV layers with 3×3 kernels. At the end of each residual block, a 2×2 max-pooling layer is applied. The number of filters starts from 64 and is doubled every next block. Following 4 blocks, there is a mean-pooling layer to compress the output feature maps to a feature embedding. **ResNet-18** contains 4 residual blocks and each block has 4 CONV layers with 3×3 kernels. The number of filters starts from 64 and is doubled every next block. Before the residual blocks, there is one additional CONV layer with 64 filter and 3×3 kernels at the beginning of the network. The residual blocks are followed by an average pooling layer. The ResNet-18 backbone we use exactly follow [2] except that the last FC layer is removed. **ResNet-25** is exactly the same as the released code of [41], [67]. Three residual

blocks are used after an initial convolutional layer. Each block has 4 CONV layers with 3×3 kernels. The number of filters starts from 160 and is doubled every next block. After a global average pooling layer, it leads to a 640-dim embedding. **WRN-28-10** has its depth and width set to 28 and 10, respectively. After a global average pooling in the last layer of the backbone, it gets a 640-dimensional embedding. For this backbone, we resize the input image to $80 \times 80 \times 3$ for a fair comparison with related methods [6], [24]. Other details are same with those of ResNet-25 [25], [41]. Note that we employ this architecture using the code of SIB [24] and implement only our *SS* operations to it.

For the architecture of Φ_{S_1} and Φ_{S_2}, actually, they are generated according to the architecture of Θ, as introduced in Section 4.1. For example, when using ResNet-25 in MTL, Φ_{S_1} and Φ_{S_2} also have 25 layers, respectively.

For the architecture of θ (the parameters of the base-learner), we empirically find that in our cases a single FC layer (as θ) is faster to train and more effective for classification than multiple layers, taking the most popular dataset *mini*ImageNet as an example. Results are given in Table 1, in which we can see the performance drop when changing this θ to multiple layers.

Base-learning	**Dim. of θ**	***mini*ImageNet**	
		1-shot	**5-shot**
θ (2 FC layers)	512, 5	59.1 ± 1.9	70.7 ± 0.9
θ (3 FC layers)	1024, 512, 5	56.2 ± 1.8	68.7 ± 0.9
Θ, θ	5	59.6 ± 1.8	71.6 ± 0.9
θ (**Ours**)	5	**60.6** ± 1.9	**74.3** ± 0.8

TABLE 1
The 5-way, 1-shot and 5-shot classification accuracy (%) on *mini*ImageNet, for choosing the best architecture of base-learner (i.e., the classifier θ). "meta-batch" and "ResNet-12 (pre)" are used.

Pre-training stage. For the phase of DNN training on large-scale data, the model is trained by Adam optimizer [83]. The learning rate is initialized as 0.001, and decays to its half every $5k$ iterations until it is lower than 0.0001. We set the keep probability of the dropout as 0.9 and batch-size as 64. The pre-training stops after $10k$ iterations. Note that for hyperparameter selection, we randomly choose 550 samples each class as the training set, and the rest as validation. After the grid search for hyperparameters, we fix them and mix up all samples (64 classes, 600 samples each class) to do the final pre-training. Image samples in these steps are augmented by horizontal flipping.

Meta-train stage. This is a task-level training in which the base-learning in one task considers a training step for optimizing base-learner, followed by a validation step for optimizing meta-learner. The base-learner θ is optimized by batch gradient descent with the learning rate of 0.01. It is updated with 20 and 60 epochs respectively for 1-shot and 5-shot episodes on the *mini*ImageNet and *tiered*ImageNet datasets, and 20 epochs for all episodes on the FC100 dataset. Specially when using ResNet-25, we use 100 epochs for all episodes on all datasets. The meta-learner, i.e. the parameters of the *SS* operations, is optimized by Adam optimizer [83]. Its learning rate is initialized as 0.001, and decays to the half every $1k$ iterations until 0.0001. The size of meta-batch is set to 2 (episodes) due to the memory limit. For meta-gradient regularization, each time we deploy 8 previous episodes to compute meta gradients, and set temperature scalars ψ_1 and ψ_2 both as 1.0.

HT meta-batch. Hard tasks are sampled every time after running 10 meta-batches, i.e., the failure classes used for sampling hard tasks are from 20 episodes as each meta-batch contains 2 episodes. The number of hard tasks is selected for different settings by validation: 10 and 4 hard tasks respectively for the 1-shot and 5-shot experiments, on the *mini*ImageNet and *tiered*ImageNet datasets; and respectively 20 and 10 hard tasks for the 1-shot, 5-shot experiments, on the FC100 dataset.

Ablative settings. In order to show the effectiveness of our *SS* operations, we carefully design several ablative settings: two baselines without meta-learning but more classic learning, named as *update**, four baselines of *Fine-Tuning* (*FT*) on different numbers of parameters in the outer-loop based on MAML [5], named as *FT**, and two *SS* variants on smaller numbers of parameters, named as *SS**. Table 6 shows the results in these settings, for which we simply use the classical architecture (ResNet-12) containing 4 residual blocks named $\Theta 1 \sim \Theta 4$ and an FC layer θ (classifier). The bullet names used in the Table are explained as follows.

update $[\Theta; \theta]$ **(or** θ**).** There is no meta-train phase. During test phase, each episode has its whole model $[\Theta; \theta]$ (or the classifier θ) updated on $\mathcal{T}^{(tr)}$, and then tested on $\mathcal{T}^{(te)}$.

FT θ **(**$[\Theta 4; \theta]$ **or** $[\Theta 3; \Theta 4; \theta]$ **or** $[\Theta; \theta]$**).** These are straightforward ways to reduce the quantity of meta-learned parameters. For example, "$[\Theta 3; \Theta 4; \theta]$" does not update the the first two residual blocks which encode the low-level image features. Specially, "θ" means only the classifier parameters are updated during meta-train.

SS $[\Theta 4; \theta]$ **(or** $[\Theta 3; \Theta 4; \theta]$ **or** $[\Theta; \theta]$**).** During the meta-train, *SS* parameters are defined and used on $\Theta 4$. Low-level residual blocks, e.g. $\Theta 1$, deploy the pre-trained weights without meta-level update.

SS $[\Theta; \theta]$**,** ***regularized*****.** Our method of meta-transfer learning on the whole backbone and with meta-gradient regularization.

5.3 Comparison to the state-of-the-art

Table 2 and Table 3 present the overall comparisons to related works, on the *mini*ImageNet, *tiered*ImageNet, and FC100 datasets. Note that these numbers are the meta-test results of the meta-trained models which have the highest meta-validation accuracies. On the *mini*ImageNet, models on 1-shot and 5-shot are meta-trained for $6k$ and $10k$ iterations, respectively. On the *tiered*ImageNet, iterations for 1-shot and 5-shot are at $8k$ and $10k$, respectively. On the FC100, iterations are all at $3k$.

***mini*ImageNet.** In Table 2, we can see that "SIB + *SS* $[\Theta; \theta]$" and "*SS* $[\Theta; \theta]$, HT meta-batch" achieve top performances for 1-shot and 5-shot tasks, respectively. Regarding the network architecture, we can see that models using deeper ones, e.g. ResNet-25 and WRN-28-10, outperform 4CONV-based models by quite large margins, e.g. 4CONV models have the best 1-shot result with 55.51% [48] which is 9.4% lower than 64.9% (our method on ResNet-25). This clearly validates our contribution of utilizing deeper neural networks to tackle the few-shot classification problems.

***tiered*ImageNet.** In Table 3, we give the results on the larger dataset — *tiered*ImageNet. Since this dataset is newly proposed [26], its results of using previous methods [5], [33], [34] were reported by [26], [48]. From the table, we again confirm that "SIB + *SS* $[\Theta; \theta]$" outperforms others, e.g. it achieves around a margin of 2.6% over the original SIB [24] on 1-shot tasks. An interesting observation is that on this larger and more challenging dataset, our deeper version of MTL (ResNet-25) outperforms the

Few-shot Learning Method		Backbone	*mini*ImageNet (test)	
			1-shot	5-shot
Data augmentation	Adv. ResNet, [9]	WRN-40 (pre)	55.2	69.6
	Delta-encoder, [10]	VGG-16 (pre)	58.7	73.6
Metric learning	MatchingNets, [28]	4 CONV	43.44 ± 0.77	55.31 ± 0.73
	ProtoNets, [33]	4 CONV	49.42 ± 0.78	68.20 ± 0.66
	RelationNets, [34]	4 CONV	50.44 ± 0.82	65.32 ± 0.70
	Graph neural network, [47]	4 CONV	50.33 ± 0.36	66.41 ± 0.63
	Ridge regression, [46]	4 CONV	51.9 ± 0.2	68.7± 0.2
	TransductiveProp, [48]	4 CONV	55.51	69.86
Memory network	Meta Networks, [29]	5 CONV	49.21 ± 0.96	–
	SNAIL, [50]	ResNet-12 (pre)$^{\diamond}$	55.71 ± 0.99	68.88 ± 0.92
	TADAM, [37]	ResNet-12 (pre)†	58.5 ± 0.3	76.7 ± 0.3
	Cross-Modulation Nets, [44]	4 CONV	50.94 ± 0.61	66.65 ± 0.67
	Isotropic Gaussian, [51]	ResNet-34 (pre)	56.3 ± 0.4	73.9 ± 0.3
Gradient descent	MAML, [5]	4 CONV	48.70 ± 1.75	63.11 ± 0.92
	MAML++, [22]	4 CONV	52.15 ± 0.26	68.32 ± 0.44
	Meta-LSTM, [39]	4 CONV	43.56 ± 0.84	60.60 ± 0.71
	Hierarchical Bayes, [18]	4 CONV	49.40 ± 1.83	–
	MT-net, [53]	4 CONV	51.70 ± 1.84	–
	Bilevel Programming, [19]	ResNet-12$^{\diamond}$	50.54 ± 0.85	64.53 ± 0.68
	MetaGAN, [21]	ResNet-12	52.71 ± 0.64	68.63 ± 0.67
	adaResNet, [78]	ResNet-12‡	56.88 ± 0.62	71.94 ± 0.57
	MetaOptNet, [23]	ResNet-12	62.64 ± 0.35	78.63 ± 0.68
	LEO, [25]	WRN-28-10 (pre)	61.67 ± 0.08	77.59 ± 0.12
	LGM-Net, [49]	MetaNet+4CONV	69.13 ± 0.35	71.18 ± 0.68
	CTM, [40]	ResNet-18 (pre)	64.12 ± 0.82	80.51 ± 0.13
	SIB, [24]	WRN-28-10 (pre)	***70.0*** ± 0.6	79.2 ± 0.4
Ours	*FT* $[\Theta;\theta]$, HT meta-batch	ResNet-12 (pre)	58.7 ± 1.8	73.2 ± 0.8
	SS $[\Theta;\theta]$, HT meta-batch	ResNet-12 (pre)	61.4 ± 1.8	75.9 ± 0.8
	SS $[\Theta;\theta]$, HT meta-batch	ResNet-18 (pre)	62.0 ± 1.9	76.0 ± 0.8
	SS $[\Theta;\theta]$, HT meta-batch	ResNet-25 (pre)	64.9 ± 1.8	**81.2** ± 0.8
	SIB + *SS* $[\Theta;\theta]$	WRN-28-10 (pre)	**71.0** ± 0.7	***81.0*** ± 0.4

$^{\diamond}$Additional 2 convolutional layers ‡Additional 1 convolutional layer †Additional 72 fully connected layers

TABLE 2
The 5-way, 1-shot and 5-shot classification accuracy (%) on *mini*ImageNet datasets. "pre" means including our pre-training step with all training datapoints. The **best** and ***second best*** results are highlighted. Note that (1) the standard variance is affected by the number of episode test samples, and our sample splits are the same with MAML [5]; and (2) our methods with *SS* $[\Theta;\theta]$ all use meta-gradient regularization.

Few-shot Learning Method	Backbone	*tiered*ImageNet (test)		FC100 (test)	
		1-shot	5-shot	1-shot	5-shot
ProtoNets, [33] (by [26])	4 CONV	53.31 ± 0.89	72.69 ± 0.74	–	–
ProtoNets, [33] (by us)	ResNet-25	65.30 ± 1.70	83.00 ± 0.70	41.1 ± 1.8	58.6 ± 0.8
RelationNets, [34] (by [48])	4 CONV	54.48 ± 0.93	71.32 ± 0.78	–	–
TransductiveProp, [48]	4 CONV	57.41 ± 0.94	71.55 ± 0.74	–	–
MAML, [5] (by us)	4CONV	49.0 ± 1.8	66.5 ± 0.9	38.1 ± 1.7	50.4 ± 1.0
MAML++, [22] (by us)	4CONV	51.5 ± 0.5	70.6 ± 0.5	38.7 ± 0.4	52.9 ± 0.4
TADAM, [37]	ResNet-12 (pre)†	62.13 ± 0.31	81.92 ± 0.30	40.1 ± 0.4	56.1 ± 0.4
MetaOptNet [23]	ResNet-12	65.99 ± 0.72	81.56 ± 0.53	41.1 ± 0.6	55.5 ± 0.6
CTM, [40]	ResNet-18 (pre)	68.41 ± 0.39	***84.28*** ± 1.73	–	–
LEO, [25]	WRN-28-10 (pre)	66.33 ± 0.05	81.44 ± 0.09	–	–
SIB, [24] (by us)	WRN-28-10 (pre)	***72.9*** ± 0.6	82.8 ± 0.4	45.2 ± 0.6	55.9 ± 0.4
FT $[\Theta;\theta]$, HT meta-batch	ResNet-12 (pre)	64.7 ± 1.7	78.5 ± 0.8	42.0 ± 1.8	55.2 ± 0.8
SS $[\Theta;\theta]$, HT meta-batch	ResNet-12 (pre)	65.3 ± 1.8	81.2 ± 0.8	45.3 ± 1.8	57.5 ± 0.8
SS $[\Theta;\theta]$, HT meta-batch	ResNet-18 (pre)	68.1 ± 1.8	82.3 ± 0.8	45.5 ± 1.8	***57.9*** ± 0.8
SS $[\Theta;\theta]$, HT meta-batch	ResNet-25 (pre)	72.3 ± 1.8	**85.6** ± 0.8	**46.1** ± 1.8	**61.4** ± 0.8
SIB + *SS* $[\Theta;\theta]$	WRN-28-10 (pre)	**75.5** ± 0.7	84.3 ± 0.4	***45.9*** ± 0.7	56.7 ± 0.4

†Additional 72 fully connected layers.

TABLE 3
The 5-way, 1-shot and 5-shot classification accuracy (%) on *tiered*ImageNet and FC100 datasets. "pre" means including our pre-training step with all training datapoints. "by [*]" means the results were reported in [*]. "by us" means our implementation using open-sourced code. The **best** and ***second best*** results are highlighted. Note that (1) the standard variance is affected by the number of episode test samples, and our sample splits are the same with MAML [5]; and (2) our methods with *SS* $[\Theta;\theta]$ all use meta-gradient regularization.

Few-shot Learning Method	Operation	*mini*ImageNet		*mini*ImageNet (*tiered*Pre)		*tiered*ImageNet	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNets [33]	*SS*	56.7 ± 1.9	72.0 ± 0.9	62.0 ± 1.9	77.9 ± 1.0	62.2 ± 2.1	78.1 ± 0.9
	FT	55.2 ± 1.9	70.8 ± 0.9	57.2 ± 1.9	75.9 ± 0.9	54.9 ± 2.0	73.0 ± 1.0
MatchingNets [28]	*SS*	58.1 ± 1.8	66.9 ± 0.9	63.6 ± 1.7	73.2 ± 0.9	64.5 ± 1.9	73.9 ± 0.9
	FT	57.4 ± 1.7	67.5 ± 0.8	61.1 ± 1.8	72.6 ± 0.8	62.4 ± 1.8	73.5 ± 0.8
RelationNets [34]	*SS*	57.2 ± 1.8	71.1 ± 0.9	61.5 ± 1.8	74.9 ± 0.9	***65.6*** ± 1.9	77.5 ± 0.9
	FT	56.0 ± 1.8	69.0 ± 0.8	58.9 ± 1.8	72.0 ± 0.8	62.2 ± 1.8	76.0 ± 0.9
MTL (FC)	*SS*	**60.6** ± 1.9	***74.3*** ± 0.8	***65.7*** ± 1.8	***78.4*** ± 0.8	***65.6*** ± 1.7	***78.7*** ± 0.9
MAML [5] (FC)	*FT*	58.3 ± 1.9	71.6 ± 0.9	61.6 ± 1.9	73.5 ± 0.8	62.0 ± 1.8	70.6 ± 0.9
MTL (Cosine)	*SS*	58.2 ± 1.8	**74.6** ± 0.8	**66.1** ± 1.8	**79.7** ± 0.9	**67.1** ± 1.8	**80.0** ± 0.8
MAML [5] (Cosine)	*FT*	***59.8*** ± 1.8	72.8 ± 0.9	59.9 ± 1.9	76.5 ± 0.7	65.1 ± 1.9	78.2 ± 0.8

TABLE 4
The 5-way, 1-shot and 5-shot classification accuracy (%) on *mini*ImageNet and *tiered*ImageNet datasets. "meta-batch" and "ResNet-12 (pre)" are used. "(*tiered*Pre)" means the pre-training stage is finished on the *tiered*ImageNet. We implement the public code of related methods [5], [28], [33], [34], [45] in our framework by which we are able to conduct different meta operations, i.e. *FT* $[\Theta;\theta]$ and *SS* $[\Theta;\theta]$. The **best** and ***second best*** results are highlighted in each block. Note that (1) cosine classifiers have been used in MatchingNets [33] and Baseline++ [45] for few-shot classification; and (2) MAML in this table is not exactly the same with original MAML [5], as it works on deep neural networks and does not update convolutional layers during base-training.

	*mini*ImageNet		*tiered*ImageNet		*mini*ImageNet w/$\mathcal{D}$		*tiered*ImageNet w/$\mathcal{D}$	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Masked Soft k-Means [26]	50.7 ± 0.3	64.4 ± 0.2	52.4 ± 0.4	69.9 ± 0.2	49.0 ± 0.3	63.0 ± 0.1	51.4 ± 0.4	69.1 ± 0.3
Masked Soft k-Means *w/* MTL	58.6 ± 1.8	72.2 ± 0.8	65.1 ± 0.8	80.1 ± 0.8	57.2 ± 1.8	71.0 ± 0.8	63.5 ± 1.8	80.2 ± 0.8
TPN [48]	52.8 ± 0.3	66.4 ± 0.2	55.7 ± 0.3	71.0 ± 0.2	50.4 ± 0.8	64.9 ± 0.7	53.5 ± 0.9	69.9 ± 0.8
TPN *w/* MTL	59.6 ± 1.8	72.3 ± 0.8	68.3 ± 1.9	80.4 ± 0.8	59.1 ± 1.8	71.0 ± 0.8	67.7 ± 1.9	80.3 ± 0.8
LST [27] *w/o* MTL	64.7 ± 1.9	74.8 ± 0.8	73.3 ± 1.6	82.9 ± 0.8	59.8 ± 1.9	74.8 ± 0.8	70.2 ± 1.6	81.9 ± 0.8
LST *w/* MTL	70.1 ± 1.9	78.7 ± 0.8	77.7 ± 1.6	85.2 ± 0.8	64.1 ± 1.9	77.4 ± 0.8	73.5 ± 1.6	83.4 ± 0.8

TABLE 5
Semi-supervised 5-way, 1-shot and 5-shot classification accuracy (%) on *mini*ImageNet and *tiered*ImageNet. "meta-batch" and "ResNet-12 (pre)" are used. "w/$\mathcal{D}$" means additionally including the unlabeled data from 3 distracting classes (5 unlabeled samples per class) that are **excluded** in the "5-way" classes of the task [26], [27], [48].

shallower one (ResNet-12) by 7% on 1-shot, which is twice the margin on *mini*ImageNet (3.5%). This shows our idea of transferring knowledge from pre-trained DNNs is more promising for handling harder few-shot settings.

FC100. In Table 3, we also show the results on the FC100. We report the numbers of TADAM [37] and MetaOptNet [23] given in original papers, and obtain the results of classical methods, i.e. MAML [5], MAML++ [22], RelationNets [34] and MatchineNets [28], by implementing their open-sourced code on the deeper pre-trained networks. From the table, we can see that our approach consistently outperforms MAML and its improved version MAML++ by large margins, e.g. over 7% for 1-shot tasks. Besides, it surpasses TADAM and MetaOptNet by 6% and 5%, respectively. Implementing *SS* operations on SIB brings around 1% gains over the original both for 1-shot and 5-shot.

5.4 Plug-in evaluation

The *Scaling* and *Shifting* (*SS*) operations in our proposed MTL approach work on pre-trained convolutional neurons thus are easy to be applied to other CNN-based few-shot learning models. Detailed plug-in steps are given in Section 4.5.

Table 4 shows the results of implementing *SS* operations on supervised models, i.e. ProtoNets [33], MatchingNets [28], RelationNets [34], MAML [5] (with a single FC layer as the base-learner [6]), and MAML [5] (with a cosine distance classifier as the base-learner [28], [45]). In their original methods, *FT* is the meta-level operation and 4CONV is the uniform architecture. For an easy and fair comparison, we also implement the results of using *FT* on deeper networks (e.g. ResNet-12). Note that more results of using ResNet-18 are provided in the Appendix. Regarding the table, we have three columns. "*mini*ImageNet (*tiered* Pre)" denotes that the model is pre-trained on *tiered*ImageNet and its weights are then meta-transferred to the learning of few-shot models on *mini*ImageNet episodes. We can see that in all settings, (1) the best performance is achieved by our proposed MTL, e.g., MTL (FC) outperforms MAML (FC) by 3.6% and 8.1% on *tiered*ImageNet 1-shot and 5-shot, respectively; and (2) classical methods using *SS* get consistent improvements over the original version of using *FT*, e.g., RelationNets [34] gains 3.4% and 1.5% on *tiered*ImageNet 1-shot and 5-shot, respectively.

In addition, we verify the generalization ability of our MTL to semi-supervised few-shot learning (SSFSL) methods [26], [27], [48]. Our results on the *mini*ImageNet and *tiered*ImageNet datasets are presented in Table 5. Note that "w/$\mathcal{D}$" indicates the more challenging setting of including 3 distracting classes in the unlabeled set (see Section 5.1). From Table 5, we can see that three models "*w/* MTL" obtain consistent improvements (over their originals) by quit large margins, e.g. the highest as 14.2% on *tiered*ImageNet 1-shot w/$\mathcal{D}$.

Settings	*mini*ImageNet		*mini*ImageNet (tieredPre)		FC100		FC100 (tieredPre)	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
update $[\Theta;\theta]$	45.3 $\pm$ 1.9	64.6 $\pm$ 0.9	54.4 $\pm$ 1.8	73.7 $\pm$ 0.8	38.4 $\pm$ 1.8	52.6 $\pm$ 0.9	37.6 $\pm$ 1.9	52.7 $\pm$ 0.9
update θ	50.0 $\pm$ 1.8	66.7 $\pm$ 0.9	51.3 $\pm$ 1.8	70.3 $\pm$ 0.8	39.3 $\pm$ 1.9	51.8 $\pm$ 0.9	38.3 $\pm$ 1.8	52.9 $\pm$ 1.0
FT θ	55.9 $\pm$ 1.9	71.4 $\pm$ 0.9	61.6 $\pm$ 1.8	73.5 $\pm$ 0.9	41.6 $\pm$ 1.9	54.9 $\pm$ 1.0	40.4 $\pm$ 1.9	54.7 $\pm$ 0.9
FT $[\Theta 4;\theta]$	57.2 $\pm$ 1.8	71.6 $\pm$ 0.8	62.3 $\pm$ 1.8	73.9 $\pm$ 0.9	40.9 $\pm$ 1.8	54.3 $\pm$ 1.0	41.2 $\pm$ 1.8	53.6 $\pm$ 1.0
FT $[\Theta 3, \Theta 4;\theta]$	58.1 $\pm$ 1.8	70.9 $\pm$ 0.8	63.0 $\pm$ 1.8	74.8 $\pm$ 0.9	41.5 $\pm$ 1.8	53.7 $\pm$ 0.9	40.7 $\pm$ 1.9	53.8 $\pm$ 0.9
FT $[\Theta;\theta]$	58.3 $\pm$ 1.8	71.6 $\pm$ 0.8	63.2 $\pm$ 1.9	75.7 $\pm$ 0.8	41.6 $\pm$ 1.9	54.4 $\pm$ 1.0	41.1 $\pm$ 1.9	54.5 $\pm$ 0.9
SS $[\Theta 4;\theta]$	59.2 $\pm$ 1.8	73.1 $\pm$ 0.9	64.0 $\pm$ 1.8	76.9 $\pm$ 0.8	42.4 $\pm$ 1.9	55.1 $\pm$ 1.0	42.7 $\pm$ 1.9	55.9 $\pm$ 1.0
SS $[\Theta 3, \Theta 4;\theta]$	59.4 $\pm$ 1.8	73.4 $\pm$ 0.8	64.5 $\pm$ 1.8	77.2 $\pm$ 0.8	42.5 $\pm$ 1.9	54.5 $\pm$ 1.0	43.4 $\pm$ 1.8	56.4 $\pm$ 1.0
SS $[\Theta;\theta]$	***60.6*** $\pm$ 1.8	***74.3*** $\pm$ 0.8	***65.7*** $\pm$ 1.8	***78.4*** $\pm$ 0.9	**43.6** $\pm$ 1.9	**55.4** $\pm$ 1.0	**43.5** $\pm$ 1.9	***57.1*** $\pm$ 1.0
SS $[\Theta;\theta]$, *regularized*	**61.0** $\pm$ 1.8	**74.5** $\pm$ 0.8	**66.2** $\pm$ 1.8	**79.1** $\pm$ 0.8	***43.5*** $\pm$ 1.7	***55.3*** $\pm$ 0.8	**43.5** $\pm$ 1.9	**57.2** $\pm$ 0.8

TABLE 6
The 5-way, 1-shot and 5-shot classification accuracy (%) using ablative models, on two datasets. "meta-batch" and "ResNet-12 (pre)" are used. "(tieredPre)" means the pre-training stage is finished on the *tiered*ImageNet. The **best** and ***second best*** results are highlighted.

Setting	*mini*ImageNet		*tiered*ImageNet	
	1-shot	5-shot	1-shot	5-shot
FT $[\Theta;\theta]$	58.3 $\pm$ 1.9	71.6 $\pm$ 0.9	61.6 $\pm$ 1.9	73.5 $\pm$ 0.8
FT $[\Theta;\theta]$, HT	58.7 $\pm$ 1.8	73.2 $\pm$ 0.8	64.7 $\pm$ 1.7	78.5 $\pm$ 0.8
SS $[\Theta;\theta]$	60.6 $\pm$ 1.9	74.3 $\pm$ 0.8	65.6 $\pm$ 1.7	78.7 $\pm$ 0.9
SS $[\Theta;\theta]$, HT	61.4 $\pm$ 1.8	75.9 $\pm$ 0.8	65.3 $\pm$ 1.8	81.2 $\pm$ 0.8

TABLE 7
Ablation results for the 5-way, 1-shot and 5-shot classification accuracy (%) on *mini*ImageNet and *tiered*ImageNet datasets. "ResNet-12 (pre)" is used.

5.5 Ablation study

Table 6 shows the results of ablation studies on the *mini*ImageNet and FC100. Figure 4 demonstrates the performance gap between *w/* and *w/o* HT meta-batch in terms of recognition accuracy and converging speed on the FC100. Table 7 summarizes the accuracies of *w/* and *w/o* HT meta-batch on ImageNet-based datasets.

MTL *vs*. No meta-learning. Table 6 shows the results of *No meta-learning* methods on the top block. Compared to these, our approach achieves significantly better performance, e.g., the largest margins on *mini*ImageNet are 11.0% for 1-shot and 7.8% for 5-shot. This validates the effectiveness of meta-learning method for tackling few-shot learning problems. Between two *No meta-learning* methods, we can see that updating both feature extractor Θ and classifier θ is inferior to updating θ only (Θ is pre-trained), e.g., around 5% reduction on *mini*ImageNet 1-shot. One reason is that in few-shot settings, there are too many parameters to optimize with few-shot data. This is our motivation to learn only θ during base-learning (see Table 1).

SS $[\Theta;\theta]$ **works better than light-weight *FT* variants.** Table 6 shows that our approach with *SS* $[\Theta;\theta]$ achieves the best performances for all few-shot settings. *SS* actually meta-learns a smaller set of transferring parameters on $[\Theta;\theta]$ than *FT*. People may argue that *FT* is weaker because it learns a larger set of initialization parameters, whose quantity is equal to the size of $[\Theta;\theta]$, causing the model to overfit to few-shot data. In the middle block of Table 6, we show the ablation study of freezing low-level pre-trained layers and meta-learn only the high-level layers (e.g. $\Theta 4$ of ResNet-12) by *FT* operations. It is obvious that they all yield inferior performances than using our *SS*. An additional observation is that our methods *SS** perform consistently better than *FT**.

Meta-gradient regularization is effective. On the last two rows of Table 6, we validate the effectiveness of meta-gradient regularization (see Section 4.3). From the results we can see deploying such cross-task memory regularization (*SS* $[\Theta;\theta]$, *regularized*) achieves better performance than using *SS* $[\Theta;\theta]$ whose meta gradients each time come from an individual episode. This is because our regularization forces meta-learner to be less forgetting about previous episodes, and additionally stabilize the meta-gradients of each few-shot episode.

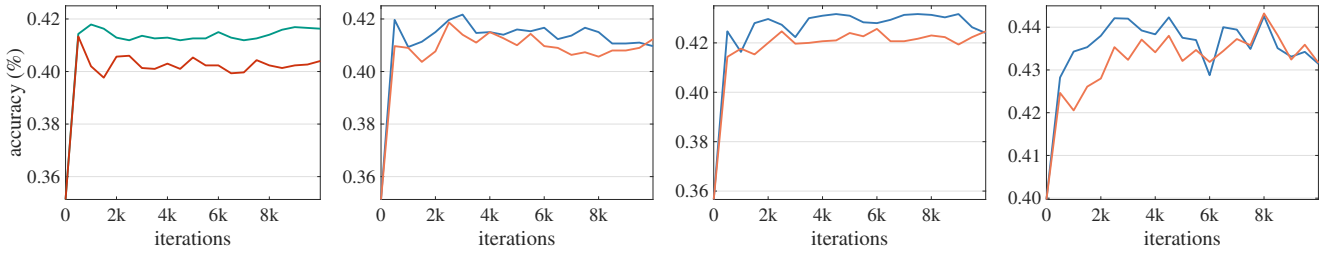
Accuracy gain by HT meta-batch. HT meta-batch is basically a curriculum learning scheme, and can be generalized to the models with different network architectures. In Table 7, we show the ablation results for HT meta-batch on ImageNet-based datasets. Comparing *SS* $[\Theta;\theta]$, HT (HT meta-batch) with *SS* $[\Theta;\theta]$, HT meta-batch improves the results by an average accuracy of 1.5%. In Figure 4, we show the validation curves including the cases of using *FT* as well as *SS* during meta-training on the FC100 dataset. Red curves are the results of using conventional meta-batch [5]. It is clear that HT meta-batch boosts both *FT* and *SS* based meta-learners.

Speed of convergence of MTL with HT meta-batch. From Figure 4 (a)-(d), we can see that impressively, the models using our proposed HT meta-batch require only $1 \sim 4k$ episodes to converge to a good performance. Note that (1) each iteration contains 2 training episodes, and (2) MAML [5] without deep pre-trained networks used over $200k$ episodes to achieve the best performance on miniImageNet. We attest this to three reasons. First, our methods start from the pre-trained deep neural networks. Second, our *SS* needs to learn only $< \frac{2}{9}$ parameters of the number of *FT* parameters. Third, our HT meta-batch is a hard negative mining step and brings accelerations by learning challenging tasks [36].

5.6 Statistical data of *SS*

We evaluate to what extent neuron weights and biases (e.g., on ResNet-12) have drifted after *SS* operations. We present the statistics on the learned *SS* weights in Table 8. Each number in the table shows how much the weight (or bias) drifts from original weight (or bias) pre-trained using large-scale data. Each dot curve in (a) and (b) presents the distribution of those numbers, matching well with the Gaussian distribution (in red).

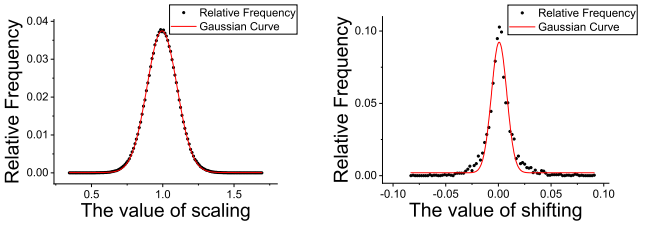
We find that *Scaling* parameters are more scattered than *Shifting* on three datasets. The average shift of mean values x_c of *Scaling* is 4.51×10^{-3} higher than that of *Shifting* 7.13×10^{-4}



(a) *FT*, ResNet-12 (b) *SS*, ResNet-12 (c) *SS*, ResNet-18 (d) *SS*, ResNet-25

Fig. 4. The 5-way, 1-shot meta-validation accuracy plots on the FC100, using *FT* (pre-trained ResNet-12 and MAML [5]) and our MTL on different pre-trained networks. Red curve uses the original meta-batch [5] and others use our proposed HT meta-batch.

(a) *mini*ImageNet (b) *mini*ImageNet



		***mini*ImageNet**	***tiered*ImageNet**	**FC100**
Scaling	y_0	9.58×10^{-5}	1.95×10^{-4}	8.01×10^{-5}
	x_c	9.94×10^{-1}	9.98×10^{-1}	9.93×10^{-1}
	w	2.11×10^{-1}	1.49×10^{-1}	2.27×10^{-1}
	A	9.87×10^{-3}	9.79×10^{-3}	9.90×10^{-3}
Shifting	y_0	1.99×10^{-3}	1.32×10^{-3}	1.30×10^{-3}
	x_c	7.52×10^{-4}	7.61×10^{-4}	6.28×10^{-4}
	w	1.46×10^{-2}	1.53×10^{-2}	1.70×10^{-2}
	A	1.65×10^{-3}	1.73×10^{-3}	1.71×10^{-3}

TABLE 8
Statistical values of *SS* parameters, i.e. to see how much network parameters drifted after the meta-training using *SS*. The experiments are conducted with the settings of ResNet-12, meta-batch, 5-way and 1-shot. *Scaling* and *Shifting* parameters are counted with bin size 0.01 and 0.002, respectively. Relative frequency of each *SS* value is computed. All dots match a fit to the Gaussian distribution ($y = y_0 + \frac{A}{w\sqrt{\pi/2}} e^{-2(\frac{x-x_c}{w})^2}$). x_c and w are the values of mean and standard deviation, respectively. y_0 and A are two parameters of the distribution to enable the exact fit.

(note that initialization for *Scaling* parameter is 1 and for *Shifting* is 0). The standard deviation w shows also higher for *Scaling*. We think these are due to the fact that convolution neuron weights (rather than neuron biases) encode the most of image representation knowledge. We also see the differences among three datasets: the parameter drifting is more obvious on smaller datasets such as the FC100. In other words, the gap between pre-trained model and meta-learner is more significant on such dataset. We think this is because the feature representations learned on small-size data are not as generalizable as those learned on larger-scale data.

6 Conclusions

In this paper, we show that our novel MTL model trained with HT meta-batch learning curriculum achieves the top performance for tackling few-shot learning problems. The key operations of MTL on pre-trained DNN neurons proved to be highly efficient for adapting the learning experience to the unseen task. The superiority was particularly achieved in the extreme 1-shot cases on three challenging benchmarks – *mini*ImageNet, *tiered*ImageNet and FC100. The generalization ability of our method is validated by implementing MTL on the classical supervised few-shot models as well as the state-of-the-art semi-supervised few-shot models. The consistent improvements by MTL prove that large-scale pre-trained deep networks can offer a good "knowledge base" to conduct efficient few-shot learning on. In terms of learning scheme, HT meta-batch showed consistently good performance for the ablative models. On the more challenging FC100 benchmark, it showed to be particularly helpful for boosting convergence speed. This design is independent of any specific model or architecture and can be generalized well whenever the hardness of task is easy to evaluate in online iterations.

Acknowledgments

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant. This research is as part of NExT++, a research is supported by the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative.

References

[1] L. Yann, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521(7553), p. 436, 2015.
[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[3] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.
[4] F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.
[5] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.
[6] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *CVPR*, 2019, pp. 403–412.
[7] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *ICLR*, 2016.
[8] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid data dreaming for object tracking," *arXiv*, vol. 1703.09554, 2017.
[9] A. Mehrotra and A. Dukkipati, "Generative adversarial residual pairwise networks for one shot learning," *arXiv*, vol. 1703.08033, 2017.
[10] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, R. S. Feris, A. Kumar, R. Giryes, and A. M. Bronstein, "Delta-encoder: an effective sample synthesis method for few-shot object recognition," in *NeurIPS*, 2018, pp. 2850–2860.
[11] Y. Wang, R. B. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *CVPR*, 2018, pp. 7278–7286.
[12] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "f-VAEGAN-D2: A feature generating framework for any-shot learning," in *CVPR*, 2019, pp. 10 275–10 284.
[13] S. Bartunov and D. P. Vetrov, "Few-shot generative modelling with generative matching networks," in *AISTATS*, 2018, pp. 670–678.

[14] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, 1992, pp. 6–8.

[15] D. K. Naik and R. Mammone, "Meta-neural networks that learn by learning," in *IJCNN*, 1992, pp. 437–442.

[16] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to learn*. Springer, 1998, pp. 3–17.

[17] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *NeurIPS*, 2018, pp. 9537–9548.

[18] E. Grant, C. Finn, S. Levine, T. Darrell, and T. L. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," in *ICLR*, 2018.

[19] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *ICML*, 2018, pp. 1563–1572.

[20] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layer-wise metric and subspace," in *ICML*, 2018, pp. 2933–2942.

[21] R. Zhang, T. Che, Z. Grahahramani, Y. Bengio, and Y. Song, "Metagan: An adversarial approach to few-shot learning," in *NeurIPS*, 2018, pp. 2371–2380.

[22] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," in *ICLR*, 2019.

[23] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *CVPR*, 2019, pp. 10 657–10 665.

[24] S. X. Hu, P. G. Moreno, X. S. Y. Xiao, N. D. Lawrence, G. Obozinski, A. Damianou, and F. Champs-sur Marne, "Empirical bayes meta-learning with synthetic gradients," in *ICLR*, 2020.

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

[26] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," in *ICLR*, 2018.

[27] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *NeurIPS*, 2019, pp. 10 276–10 286.

[28] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NIPS*, 2016, pp. 3630–3638.

[29] T. Munkhdalai and H. Yu, "Meta networks," in *ICML*, 2017, pp. 2554–2563.

[30] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few shot learning," *arXiv*, vol. 1707.09835, 2017.

[31] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *NIPS*, 2017, pp. 6467–6476.

[32] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, 1989, pp. 3–17.

[33] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017, pp. 4077–4087.

[34] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*, 2018, pp. 1199–1208.

[35] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[36] A. Shrivastava, A. Gupta, and R. B. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016, pp. 761–769.

[37] B. N. Oreshkin, P. Rodríguez, and A. Lacoste, "TADAM: task dependent adaptive metric for improved few-shot learning," in *NeurIPS*, 2018, pp. 719–729.

[38] H. E. Geoffrey and P. C. David, "Using fast weights to deblur old memories," in *CogSci*, 1987, pp. 177–186.

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

[40] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang, "Finding task-relevant features for few-shot learning by category traversal," in *CVPR*, 2019, pp. 1–10.

[41] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *CVPR*, 2020.

[42] R. Hou, H. Chang, M. Bingpeng, S. Shan, and X. Chen, "Cross attention network for few-shot classification," in *NeurIPS*, 2019, pp. 4005–4016.

[43] N. Dvornik, C. Schmid, and J. Mairal, "Diversity with cooperation: Ensemble methods for few-shot classification," in *ICCV*, 2019, pp. 3723–3731.

[44] H. Prol, V. Dumoulin, and L. Herranz, "Cross-modulation networks for few-shot learning," *arXiv*, vol. 1812.00273, 2018.

[45] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *ICLR*, 2019.

[46] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *ICLR*, 2019.

[47] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *ICLR*, 2018.

[48] Y. Liu, J. Lee, M. Park, S. Kim, and Y. Yang, "Learning to propagate labels: Transductive propagation network for few-shot learning," in *ICLR*, 2019.

[49] H. Li, W. Dong, X. Mei, C. Ma, F. Huang, and B. Hu, "Lgm-net: Learning to generate matching networks for few-shot learning," in *ICML*, 2019, pp. 3825–3834.

[50] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "Snail: A simple neural attentive meta-learner," in *ICLR*, 2018.

[51] M. Bauer, M. Rojas-Carulla, J. B. Świątkowski, B. Schölkopf, and R. E. Turner, "Discriminative k-shot learning using probabilistic models," *arXiv*, vol. 1706.00326, 2017.

[52] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016, pp. 1842–1850.

[53] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layer-wise metric and subspace," in *ICML*, 2018, pp. 2933–2942.

[54] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 22, no. 2, pp. 199–210, 2011.

[55] J. Yang, R. Yan, and A. G. Hauptmann, "Adapting SVM classifiers to data with shifted distributions," in *ICDM Workshops*, 2007.

[56] Y. Wei, Y. Zhang, J. Huang, and Q. Yang, "Transfer learning via learning to transfer," in *ICML*, 2018, pp. 5085–5094.

[57] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *CVPR*, 2018, pp. 3712–3722.

[58] Q. Sun, B. Schiele, and M. Fritz, "A domain based approach to social relation recognition," in *CVPR*, 2017, pp. 435–444.

[59] Y. Liu, Y. Su, A. Liu, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *CVPR*, 2020.

[60] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2018, pp. 3942–3951.

[61] D. Erhan, Y. Bengio, A. C. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[62] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *CVPR*, 2017, pp. 3296–3297.

[63] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *ICCV*, 2017, pp. 2980–2988.

[64] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[65] M. Rohrbach, S. Ebert, and B. Schiele, "Transfer learning in a transductive setting," in *NIPS*, 2013, pp. 46–54.

[66] R. Keshari, M. Vatsa, R. Singh, and A. Noore, "Learning structure and strength of CNN filters for small sample size training," in *CVPR*, 2018, pp. 9349–9358.

[67] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *CVPR*, 2018, pp. 7229–7238.

[68] T. R. Scott, K. Ridgeway, and M. C. Mozer, "Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning," in *NeurIPS*, 2018, pp. 76–85.

[69] N. Sarafianos, T. Giannakopoulos, C. Nikou, and I. A. Kakadiaris, "Curriculum learning for multi-task classification of visual attributes," in *ICCV Workshops*, 2017.

[70] D. Weinshall, G. Cohen, and D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," in *ICML*, 2018, pp. 5235–5243.

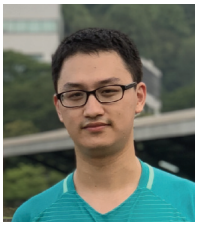
[71] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *ICML*, 2017, pp. 1311–1320.

[72] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *NIPS*, 2010, pp. 1189–1197.

[73] A. Pentina, V. Sharmanska, and C. H. Lampert, "Curriculum learning of multiple tasks," in *CVPR*, 2015, pp. 5492–5500.

[74] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *ICML*, 2018, pp. 2309–2318.
[75] O. Canévet and F. Fleuret, "Large scale hard sample mining with monte carlo tree search," in *CVPR*, 2016, pp. 5128–5137.
[76] B. Harwood, V. Kumar, G. Carneiro, I. Reid, and T. Drummond, "Smart mining for deep metric learning," in *ICCV*, 2017, pp. 2840–2848.
[77] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 886–893.
[78] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler, "Rapid adaptation with conditionally shifted neurons," in *ICML*, 2018, pp. 3661–3670.
[79] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[80] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.
[81] H. Ye, H. Hu, D. Zhan, and F. Sha, "Learning embedding adaptation for few-shot learning," *arXiv*, vol. 1812.03664, 2018.
[82] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.
[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, vol. 1412.6980, 2014.

Qianru Sun is a Tenure-Track Assistant Professor in the School of Information Systems, Singapore Management University, since 2019. From 2018 to 2019, she was a Joint Research Fellow at the National University of Singapore and the MPI for Informatics. From 2016 to 2018, she held the Lise Meitner Award Fellowship and worked at the MPI for Informatics. In 2016, she obtained her Ph.D. degree from Peking University. In 2014, she was a visiting student at the University of Tokyo. Her research interests are computer vision and machine learning that aim to develop efficient algorithms and systems for visual understanding.



Yaoyao Liu is a Ph.D. student in Computer Science at Max Planck Institute for Informatics, working with Prof. Dr. Bernt Schiele and Prof. Qianru Sun. From 2018 to 2019, he was a research intern at the School of Computing, National University of Singapore, working with Prof. Tat-Seng Chua and Prof. Qianru Sun. Before this, he obtained his bachelor's degree at Qiushi Honors College, Tianjin University. His research includes few-shot learning, meta learning, incremental learning and image generation.

Zhaozheng Chen is a Ph.D. student in the School of Information Systems, Singapore Management University, supervised by Prof. Qianru Sun. He obtained his bachelor degree at the School of Computer Science and Technology, Shandong University in 2019. His research interests are computer vision and machine learning. Specific efforts have been made in few-shot image classification, object detection and social relationship recognition.

Tat-Seng Chua is the KITHCT Chair Professor at the School of Computing, National University of Singapore. He is also the distinguish Visiting Professor of Tsinghua University. He was the Founding Dean of the School from 1998-2000. He is now the Director of a joint research Center between NUS and Tsinghua (NExT) to research into big unstructured multi-source multimodal data analytics. He holds a PhD degree from the University of Leeds, UK, since Feb 1983. Before that, he got his bachelor degree in the Civil Engineering and Computer Science, University of Leeds, UK, in Jun 1979. His main research interests are in multimedia information retrieval and social media analytics. In particular, his research focuses on the extraction, retrieval and question-answering of text, video and live media arising from the Web and social networks.

Bernt Schiele has been Max Planck Director at MPI for Informatics and Professor at Saarland University since 2010. He studied computer science at the University of Karlsruhe, Germany. He worked on his master thesis in the field of robotics in Grenoble, France, where he also obtained the "diplome d'etudes approfondies d'informatique". In 1994 he worked in the field of multi-modal human-computer interfaces at Carnegie Mellon University, Pittsburgh, PA, USA in the group of Alex Waibel. In 1997 he obtained his PhD from INP Grenoble, France under the supervision of Prof. James L. Crowley in the field of computer vision. The title of his thesis was "Object Recognition using Multidimensional Receptive Field Histograms". Between 1997 and 2000 he was postdoctoral associate and Visiting Assistant Professor with the group of Prof. Alex Pentland at the Media Laboratory of the Massachusetts Institute of Technology, Cambridge, MA, USA. From 1999 until 2004 he was Assistant Professor at the Swiss Federal Institute of Technology in Zurich (ETH Zurich). Between 2004 and 2010 he was Full Professor at the computer science department of TU Darmstadt.