Learning to Self-Train for Semi-Supervised Few-Shot Classification

Xinzhe Li \(^1\) Qianru Sun \(^2\) Yaoyao Liu \(^3\) Shibao Zheng \(^1\) Qin Zhou Tat-Seng Chua \(^5\) Bernt Schiele \(^6\)

\(^1\)Shanghai Jiao Tong University \(^2\)Singapore Management University \(^3\)Tianjin University \(^4\)Alibaba Group
\(^5\)National University of Singapore \(^6\)Max Planck Institute for Informatics, Saarland Informatics Campus

Abstract

Few-shot classification (FSC) is challenging due to the scarcity of labeled training data (e.g. only one labeled data point per class). Meta-learning has shown to achieve promising results by learning to initialize a classification model for FSC. In this paper we propose a novel semi-supervised meta-learning method called learning to self-train (LST) that leverages unlabeled data and specifically meta-learns how to cherry-pick and label such unsupervised data to further improve performance. To this end, we train the LST model through a large number of semi-supervised few-shot tasks. On each task, we train a few-shot model to predict pseudo labels for unlabeled data, and then iterate the self-training steps on labeled and pseudo-labeled data with each step followed by fine-tuning. We additionally learn a soft weighting network (SWN) to optimize the self-training weights of pseudo labels so that better ones can contribute more to gradient descent optimization. We evaluate our LST method on two ImageNet benchmarks for semi-supervised few-shot classification and achieve large improvements over the state-of-the-art method. Code is at github.com/xinzheli1217/learning-to-self-train.

1 Introduction

Today’s deep neural networks require large amounts of labeled data for supervised training and best performance [39, 8, 30]. Their potential applications to the small-data regimes are thus limited. There has been growing interest in reducing the required amount of data, e.g. to only 1-shot [12]. One of the most powerful methods is meta-learning that transfers the experience learned from similar tasks to the target task [3]. Among different meta strategies, gradient descent based methods are particularly promising for today’s neural networks [8, 32, 27]. Another intriguing idea is to additionally use unlabeled data. Semi-supervised learning using unlabeled data with a relatively small set of labeled ones has obtained good performance on standard datasets [21, 20]. A classic, intuitive and simple method is e.g. self-training. It first trains a supervised model with labeled data, and then enlarges the labeled set based on the most confident predictions (called pseudo labels) on unlabeled data [40, 35, 20]. It can outperform regularization based methods [17, 6, 9], especially when labeled data is scarce.

The focus of this paper is thus on the semi-supervised few-shot classification (SSFSC) task. Specifically, there are few labeled data and a much larger amount of unlabeled data for training classifiers. To tackle this problem, we propose a new SSFSC method called learning to self-train (LST) that successfully embeds a well-performing semi-supervised method, i.e. self-training, into the meta
gradient descent paradigm. However, this is non-trivial, as directly applying self-training recursively may result in gradual drifts and thus adding noisy pseudo-labels [41]. To address this issue, we propose both to meta-learn a soft weighting network (SWN) to automatically reduce the effect of noisy labels, as well as to fine-tune the model with only labeled data after every self-training step.

Specifically, our LST method consists of inner-loop self-training (for one task) and outer-loop meta-learning (over all tasks). LST meta-learns both to initialize a self-training model and how to cherry-pick from noisy labels for each task. An inner loop starts from the meta-learned initialization by which a task-specific model can be fast adapted with few labeled data. Then, this model is used to predict pseudo labels, and labels are weighted by the meta-learned soft weighting network (SWN). Self-training consists of re-training using weighted pseudo-labeled data and fine-tuning on few labeled data. In the outer loop, the performance of these meta-learners are evaluated via an independent validation set, and parameters are optimized using the corresponding validation loss.

In summary, our LST method learns to accumulate self-supervising experience from SSFSC tasks in order to quickly adapt to a new few-shot task. Our contribution is three-fold. (i) A novel self-training strategy that prevents the model from drifting due to label noise and enables robust recursive training. (ii) A novel meta-learned cherry-picking method that optimizes the weights of pseudo labels particularly for fast and efficient self-training. (iii) Extensive experiments on two versions of ImageNet benchmarks – miniImageNet [36] and tieredImageNet [24], in which our method achieves top performance.

2 Related works

Few-shot classification (FSC). Most FSC works are based on supervised learning. They can be roughly divided into four categories: (1) data augmentation based methods [15, 29, 37, 38] generate data or features in a conditional way for few-shot classes; (2) metric learning methods [36, 31] learn a similarity space of image features in which the classification should be efficient with few examples; (3) memory networks [18, 28, 22, 16] design special networks to record training “experience” from seen tasks, aiming to generalize that to the learning of unseen ones; and (4) gradient descent based methods [3, 4, 1, 23, 11, 7, 42, 32, 14] learn a meta-learner in the outer loop to initialize a base-learner for the inner loop that is then trained on a novel few-shot task. In our LST method, the outer-inner loop optimization is based on the gradient descent method. Different to previous works, we propose a novel meta-learner that assigns weights to pseudo-labeled data, particularly for semi-supervised few-shot learning.

Semi-supervised learning (SSL). SSL methods aim to leverage unlabeled data to obtain decision boundaries that better fit the underlying data structure [20]. The II-Model applies a simple consistency regularization [9], e.g. by using dropout, adding noise and data augmentation, in which data is automatically “labeled”. Mean Teacher is more stable version of the II-Model by making use of a moving average technique [34]. Visual Adversarial Training (VAT) regularizes the network against the adversarial perturbation, and it has been shown to be an effective regularization [17]. Another popular method is Entropy Minimization that uses a loss term to encourage low-entropy (more confident) predictions for unlabeled data, regardless of their real classes [6]. Pseudo-labeling is a self-supervised learning method that relies on the predictions of unlabeled data, i.e. pseudo labels [2]. It can outperform regularization based methods, especially when labeled data is scarce [20] as in our envisioned setting. We thus use this method in our inner loop training.

Semi-supervised few-shot classification (SSFSC). Semi-supervised learning on FSC tasks aims to improve the classification accuracy by adding a large number of unlabeled data in training. Ren et al. proposed three semi-supervised variants of ProtoNets [31], basically using Soft $k$-Means method to tune clustering centers with unlabeled data. A more recent work used the transductive propagation network (TPN) [13] to propagate labels from labeled data to unlabeled ones, and meta-learned the key hyperparameters of TPN. Differently, we build our method based on the simple and classical self-training [40] and meta gradient descent method [3, 32] without requiring a new design of a semi-supervised network. Rohrbach et al. [25] proposed to further leverage external knowledge, such as the semantic attributes of categories, to solve not only few-shot but also zero-shot problems. Similarly, we expect further gains of our approach when using similar external knowledge in our future work.
3 Problem definition and denotation

In conventional few-shot classification (FSC), each task has a small set of labeled training data called support set $\mathcal{S}$, and another set of unseen data for test, called query set $\mathcal{Q}$. Following [24], we denote another set of unlabeled data as $\mathcal{R}$ to be used for semi-supervised learning (SSL). $\mathcal{R}$ may or may not contain data of distracting classes (not included in $\mathcal{S}$).

Our method follows the uniform episodic formulation of meta-learning [36] that is different to traditional classification in three aspects. (1) Main phases are meta-train and meta-test (instead of train and test), each of which includes training (and self-training in our case) and test. (2) Samples in meta-train and meta-test are not datapoints but episodes (SSFSC tasks in our case). (3) Meta objective is not to classify unseen datapoints but to fast adapt the classifier on a new task. Let’s detail the denotations. Given a dataset $\mathcal{D}$ for meta-train, we first sample SSFSC tasks $\{\mathcal{T}\}$ from a distribution $p(\mathcal{T})$ such that each $\mathcal{T}$ has few samples from few classes, e.g. 5 classes and 1 sample per class. $\mathcal{T}$ has a support set $\mathcal{S}$ plus an unlabeled set $\mathcal{R}$ (with a larger number of samples) to train a task-specific SSFSC model, and a query set $\mathcal{Q}$ to compute a validation loss used to optimize meta-learners. For meta-test, given an unseen new dataset $\mathcal{D}_{un}$, we sample a new SSFSC task $\mathcal{T}_{un}$. “Unseen” means there is no overlap of image classes (including distracting classes) between meta-test and meta-train tasks. We first initialize a model and weight pseudo labels for this unseen task, then self-train the model on $\mathcal{S}_{un}$ and $\mathcal{R}_{un}$. We evaluate the self-training performance on a query set $\mathcal{Q}_{un}$. If we have multiple unseen tasks, we report average accuracy as the final evaluation.

4 Learning to self-train (LST)

The computing flow of applying LST to a single task is given in Figure 1. It contains pseudo-labeling unlabeled samples by a few-shot model pre-trained on the support set; cherry-picking pseudo-labeled samples by hard selection and soft weighting; re-training on picked “cherries”, followed by a fine-tuning step; and the final test on a query set. On a meta-train task, final test acts as a validation to output a loss for optimizing meta-learned parameters of LST, as shown in Figure 2.
4.1 Pseudo-labeling & cherry-picking unlabeled data

**Pseudo-labeling.** This step deploys a supervised few-shot method to train a task-specific classifier $\theta$ on the support set $S$. Pseudo labels of the unlabeled set $R$ are then predicted by $\theta$. Basically, we can use different methods to learn $\theta$. We choose a top-performing one – meta-transfer learning (MTL) [32] (for fair comparison we also evaluate this method as a component of other semi-supervised methods [24, 13]) that is based on simple and elegant gradient descent optimization [3]. In the outer loop meta-learning, MTL learns scaling and shifting parameters $\Phi_{ss}$ to fast adapt a large-scale pre-trained network $\Theta$ (e.g. for 64 classes and 600 images per class on miniImageNet [36]) to a new learning task. In the inner loop base-learning, MTL takes the last fully-connected layer as classifier $\theta$ and trains it with $S$.

In the following, we detail the pseudo-labeling process on a task $T$. Given the support set $S$, its loss is used to optimize the task-specific base-learner (classifier) $\theta$ by gradient descent:

$$ \theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L(S; [\Phi_{ss}, \theta_{t-1}]), \quad (1) $$

where $t$ is the iteration index and $t \in \{1, ..., T\}$. The initialization $\theta_0$ is given by $\theta'$ which is meta-learned (see Section 4.2). Once trained, we feed $\theta_T$ with unlabeled samples $R$ to get pseudo labels $Y^R$ as follows,

$$ Y^R = f_{[\Phi_{ss}, \theta_T]}(R), \quad (2) $$

where $f$ indicates the classifier function with parameters $\theta_T$ and feature extractor with parameters $\Phi_{ss}$ (the frozen $\Theta$ is omitted for simplicity).

**Cherry-picking.** As directly applying self-training on pseudo labels $Y^R$ may result in gradual drifts due to label noises, we propose two countermeasures in our LST method. The first is to meta-learn the SWN that automatically reweighs the data points to up-weight the more promising ones and down-weights the less promising ones, i.e. learns to cherry-pick. Prior to this step we also perform hard selection to only use the most confident predictions [35]. The second countermeasure is to fine-tune the model with only labeled data (in $S$) after every self-training step (see Section 4.2).

Specifically, we refer to the confident scores of $Y^R$ to pick-up the top $Z$ samples per class. Therefore, we have $Z C$ samples from $C$ classes in this pseudo-labeled dataset, namely $R_p$. Before feeding $R_p$ to re-training, we compute their soft weights by a meta-learned soft weighting network (SWN), in order to reduce the effect of noisy labels. These weights should reflect the relations or distances between pseudo-labeled samples and the representations of $C$ classes. We refer to a supervised method called RelationNets [33] which makes use of relations between support and query samples for traditional few-shot classification.

First, we compute the prototype feature of each class by averaging the features of all its samples. In the 1-shot case, we use the unique sample feature as prototype. Then, given a pseudo-labeled sample $(x_i, y_i) \in R_p$, we concatenate its feature with $C$ prototype features, then feed them to SWN. The weight on the $c$-th class is as follows,

$$ w_{i,c} = f_{\Phi_{swn}} \left( \left[ f_{\Phi_{ss}}(x_i); \sum_k \frac{f_{\Phi_{ss}}(x_{c,k})}{K} \right] \right), \quad (3) $$

where $c$ is the class index and $c \in [1, ..., C]$, $k$ is the sample index in one class and $k \in [1, ..., K]$, $x_{c,k} \in S$, and $\Phi_{swn}$ denotes the parameters of SWN whose optimization procedure is given in Section 4.2. Note that $\{w_{i,c}\}$ have been normalized over $C$ classes through a softmax layer in SWN.

4.2 Self-training on cherry-picked data

As shown in Figure 2 (inner loop), our self-training contains two main stages. The first stage contains a few steps of re-training on the pseudo-labeled data $R_p$ in conjunction with support set $S$, and the second are fine-tuning steps with only $S$.

We first initialize the classifier parameters as $\theta_0 \leftarrow \theta'$, where $\theta'$ is meta-optimized by previous tasks in the outer loop. We then update $\theta_0$ by gradient descent on $R_p$ and $S$. Assuming there are $T$ iterations, re-training takes the first $1 \sim m$ iterations and fine-tuning takes the rest $m + 1 \sim T$. For $t \in \{1, ..., m\}$, we have

$$ \theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L(S \cup R_p; [\Phi_{swn}, \Phi_{ss}, \theta_{t-1}]), \quad (4) $$
We evaluate the proposed LST method in terms of few-shot image classification accuracy in semi-supervised settings. Below we describe the two benchmarks we evaluate on, details of settings, comparisons to state-of-the-art methods, and an ablation study.
5.1 Datasets and implementation details

Datasets. We conduct our experiments on two subsets of ImageNet [26]. miniImageNet was firstly proposed by Vinyals et al. [36] and has been widely used in supervised FSC works [3,23,32,27,7,5], as well as semi-supervised works [13,24]. In total, there are 100 classes with 600 samples of 84 × 84 color images per class. In the uniform setting, these classes are divided into 64, 16, and 20 respectively for meta-train, meta-validation, and meta-test. tieredImageNet was proposed by Ren et al. [24]. It includes a larger number of categories, 608 classes, than miniImageNet. These classes are from 34 super-classes which are divided into 20 for meta-train (351 classes), 6 for meta-validation (97 classes), and 8 for meta-test (160 classes). The average image number per class is 1281, which is much bigger than that on miniImageNet. All images are resized to 84 × 84. On both datasets, we follow the semi-supervised task splitting method used in previous works [24,13]. We consider the 5-way classification, and sample 5-way, 1-shot (5-shot) task to contain 1 (5) samples as the support set S and 15 samples (a uniform number) samples as the query set Q. Then, on the 1-shot (5-shot) task, we have 30 (50) unlabeled images per class in the unlabeled set R. After hard selection, we filter out 10 (20) samples and only use the rest 20 (30) confident ones to do soft weighting and then re-training. In the recursive training, we use a larger unlabeled data pool containing 100 samples from which each iteration we can sample a number of samples, i.e. 30 (50) samples for 1-shot (5-shot).

Network architectures of Θ and Φ_{ss} are based on ResNet-12 (see details of MTL [32]) which consist of 4 residual blocks and each block has 3 CONV layers with 3 × 3 kernels. At the end of each block, a 2 × 2 max-pooling layer is applied. The number of filters starts from 64 and is doubled every next block. Following residual blocks, a mean-pooling layer is applied to compress the feature maps to a 512-dimension embedding. The architecture of SWN consists of 2 CONV layers with 3 × 3 kernels in 64 filters, followed by 2 FC layers with the dimensionality of 8 and 1, respectively.

Hyperparameters. We follow the settings used in MTL [32]. Base-learning rate α (in Eq. 1, Eq. 4 and Eq. 6) is set to 0.01. Meta-learning rates β_{1} and β_{2} (in Eq. 7 and Eq. 8) are set to 0.001 initially and decay to the half value every 1k meta iterations until a minimum value 0.0001 is reached. We use a meta-batch size of 2 and run 15k meta iterations. In recursive training, we use 6 (3) recursive stages for 1-shot (5-shot) tasks. Each recursive stage contains 10 re-training and 30 fine-tuning steps.

Comparing methods. In terms of SSFSC, we have two methods, namely Soft Masked k-Means [24] and TPN [13] to compare with. Their original models used a shallow, i.e. 4CONV [3] trained from scratch, for feature extraction. For fair comparison, we implement the MTL as a component of their models in order to use deeper nets and pre-trained models which have been proved better. In addition, we run these experiments using the maximum budget of unlabeled data, i.e. 100 samples per class. We also compare to the state-of-the-art supervised FSC models which are closely related to ours. They are based on either data augmentation [15,29] or gradient descent [3,23,7,5,42,19,27,32,10].

Ablative settings. In order to show the effectiveness of our LST method, we design following settings belonging to two groups: with and without meta-training. Following are the detailed ablative settings. no selection denotes the baseline of once self-training without any selection of pseudo labels. hard denotes hard selection of pseudo labels. hard with meta-training means meta-learning only [Φ_{ss}, θ_{T}]. soft denotes soft weighting on selected pseudo labels by meta-learned SWN. recursive applies multiple iterations of self-training based on fine-tuned models, see Section 4.2. Note that this recursive is only for the meta-test task, as the meta-learned SWN can be repeatedly used. We also have a comparable setting to recursive called mixing in which we mix all unlabeled subsets used in recursive and run only one re-training round (see the last second paragraph of Section 4.2).

5.2 Results and analyses

We conduct extensive experiments on semi-supervised few-shot classification. In Table 1 we present our results compared to the state-of-the-art FSC methods, respectively on miniImageNet and tieredImageNet. In Table 2 we provide experimental results for ablative settings and comparisons with the state-of-the-art SSFSC methods. In Figure 3 we show the effect of using different numbers of re-training steps (i.e. varying m in Figure 2).

Overview for two datasets with FSC methods. In the upper part of Table 1 we present SSFSC results on miniImageNet. We can see that LST achieves the best performance for the 1-shot (70.1%) setting, compared to all other FSC methods. Besides, it tackles the 5-shot episodes with an accuracy
**Few-shot Learning Method**  
**Backbone**  
**miniImageNet (test)**  

<table>
<thead>
<tr>
<th>Data augmentation</th>
<th>miniImageNet (test)</th>
<th>miniImageNet (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
</tbody>
</table>

**Gradient descent**  

<table>
<thead>
<tr>
<th>Few-shot Learning Method</th>
<th>Backbone</th>
<th>miniImageNet (test)</th>
<th>tieredImageNet (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAML, [3] (by [13])</td>
<td>ResNet-12</td>
<td>51.67 ± 1.81</td>
<td>70.30 ± 0.08</td>
</tr>
<tr>
<td>LEO, [27]</td>
<td>ResNet-12 (pre)</td>
<td>66.33 ± 0.05</td>
<td>81.44 ± 0.09</td>
</tr>
<tr>
<td>MTL, [32] (by us)</td>
<td>ResNet-12 (pre)</td>
<td>65.6 ± 1.8</td>
<td>78.6 ± 0.9</td>
</tr>
<tr>
<td>MetaOpt-SVM, [10]†</td>
<td>ResNet-12</td>
<td>65.99 ± 0.72</td>
<td>81.56 ± 0.53</td>
</tr>
</tbody>
</table>

LST (Ours)  

<table>
<thead>
<tr>
<th>Few-shot Learning Method</th>
<th>Backbone</th>
<th>miniImageNet (test)</th>
<th>tieredImageNet (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
</tbody>
</table>

Table 1: The 5-way, 1-shot and 5-shot classification accuracy (%) on miniImageNet and tieredImageNet datasets. “pre” means pre-trained for a single classification task using all training datapoints. Note that this is a reference table to show how much we gain by considering unlabeled data.

<table>
<thead>
<tr>
<th>mini</th>
<th>tiered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td>fully supervised (upper bound)</td>
<td>80.4</td>
</tr>
<tr>
<td>no meta</td>
<td></td>
</tr>
<tr>
<td>hard</td>
<td></td>
</tr>
<tr>
<td>recursive, hard</td>
<td></td>
</tr>
<tr>
<td>hard (Φ₁, θ₀)</td>
<td></td>
</tr>
<tr>
<td>soft</td>
<td></td>
</tr>
<tr>
<td>meta</td>
<td></td>
</tr>
<tr>
<td>hard, soft</td>
<td></td>
</tr>
<tr>
<td>recursive, hard, soft</td>
<td></td>
</tr>
<tr>
<td>mixing, hard, soft</td>
<td></td>
</tr>
</tbody>
</table>

| Masked Soft k-Means with MTL |        |        |        |        |
| TPN with MTL                  |        |        |        |        |
| Masked Soft k-Means [24]      |        |        |        |        |
| TPN [13]                      |        |        |        |        |

Table 2: Classification accuracy (%) in ablative settings (middle blocks) and related SSFSC works (bottom block), on miniImageNet (“mini”) and tieredImageNet (“tiered”). “fully supervised” means the labels of unlabeled data are used. “w/D” means using unlabeled data from 3 distracting classes that are excluded in the support set [13, 24]. The results of using a small unlabeled set, 5 per class [24], are given in the supplementary materials.

This result is slightly better than 78.6% reported by [10], which uses various regularization techniques like data augmentation and label smoothing. Compared to the baseline method MTL [32], LST improves the accuracies by 8.9% and 3.2% respectively for 1-shot and 5-shot, which proves the efficiency of LST using unlabeled data. In the lower part of Table 1, we present the results on tieredImageNet. Our LST performs best in both 1-shot (77.7%) and 5-shot (85.2%) and surpasses the state-of-the-art method [10] by 11.7% and 3.6% respectively for 1-shot and 5-shot. Compared to MTL [32], LST improves the results by 12.1% and 6.6% respectively for 1-shot and 5-shot.
All three figures show that re-training indeed achieves better results, but too many re-training steps we still obtain the best results compared to others.

Recursive, hard, soft with them, our method (which also shows that SWN and the hard selection strategy are complementary. (b) recursive, hard (no meta) with MTL [32]; and (c) recursive, hard (no meta) simply initialized by pre-trained ResNet-12 model [32]. Results on tieredImageNet are given in the supplementary.

Hard selection. In Table 2 we can see that the hard selection strategy often brings improvements. For example, compared to no selection, hard can boost the accuracies of 1-shot and 5-shot by 3.3% and 1.1% respectively on miniImageNet, 2.4% and 0.4% respectively on tieredImageNet. This is due to the fact that selecting more reliable samples can relieve the disturbance brought by noisy labels. Moreover, simply repeating this strategy (recursive, hard) brings about 1% average gain.

SWN. The meta-learned SWN is able to reduce the effect of noisy predictions in a soft way, leading to better performance. When using SWN individually, soft achieves comparable results with two previous SSFSC methods [24, 13]. When using SWN in cooperation with hard selection (hard, soft) achieves 0.9% improvement on miniImageNet for both 1-shot and 5-shot compared to hard(\(\Phi_{ss}, \theta'\)), which also shows that SWN and the hard selection strategy are complementary.

Recursive self-training. Comparing the results of recursive, hard with hard, we can see that by doing recursive self-training when updating \(\theta\), the performances are improved in both “meta” and “no meta” scenarios. E.g., it boosts the results by 5.1% when applying recursive training to hard, soft for minImageNet 1-shot. However, when using mixing, hard, soft that learns all unlabeled data without recursive, the improvement reduces by 3.9%. These observations show that recursive self-training can successfully leverage unlabeled samples. However, this method sometimes brings undesirable results in the cases with distractors. E.g., compared to hard, the recursive, hard brings 0.4% and 0.5% reduction for 1-shot on miniimagenet and tieredImagenet respectively, which might be due to the fact that disturbances caused by distractors in early recursive stages propagate to later stages.

Comparing with the state-of-the-art SSFSC methods. We can see that Masked Soft k-Means [24] and TPN [13] improve their performances by a large margin (more than 10% for 1-shot and 7% for 5-shot) when they are equipped with MTL and use more unlabeled samples (100 per class). Compared with them, our method (recursive, hard, soft) achieves more than 7.4% and 4.5% improvements respectively for 1-shot and 5-shot cases with the same amount of unlabeled samples on miniimagenet. Similarly, our method also surpasses TPN by 5.6% and 1.9% for 1-shot and 5-shot on tieredImagenet. Even though our method is slightly more effected when adding distractors to the unlabeled dataset, we still obtain the best results compared to others.

In order to better understand our method and validate the robustness, we present more in-depth results regarding the key components, namely re-training steps, distracting classes, pseudo labeling accuracies, and using different architectures as backbone, in the following texts.

Number of re-training steps. In Figure 3 we present the results for different re-training steps. Figure 3(a), (b) and (c) show different settings respectively: LST; recursive, hard that uses the off-the-shelf MTL method; and recursive, hard that replaces MTL with pre-trained ResNet-12 model. All three figures show that re-training indeed achieves better results, but too many re-training steps may lead to drifting problems and cause side effects on performance. The first two settings reach best performance at 10 re-training steps while the third one needs 20 re-training steps. That means MTL-based methods (LST and the recursive, hard) achieve faster convergence compared to the one directly using pre-trained ResNet-12 model.

Quantitative analyses on the number of distracting classes. In Figure 4 we show the effects of distracting classes on our LST and related methods (improved versions with MTL) [24, 34]. More
Figure 4: Classification accuracy on miniImageNet 1-shot (a) and tieredImageNet 1-shot (b), using different numbers of distracting classes.

distracting classes cause more performance deduction for all methods. Our LST achieves the top performance, especially more than 2% higher than TPN [13] in the hardest case with 7 distracting classes. Among our different settings, we can see that LST with less re-training steps, i.e., a smaller \( m \) value, works better for reducing the effect from a larger number of distracting classes.

The performance of pseudo-labeling. Taking the miniImageNet 1-shot as an example, we record the accuracy of pseudo labeling for meta-training and meta-test (based on our best model recursive, hard, soft), in Table 3 and Table 4 respectively. In meta-training, we can see the accuracy grows from 47.0% (iter=0) to 71.5% (iter=15k), and it reaches saturation after 2k iterations. There are 6 recursive stages during meta-test. From stage-2 to stage-6, the average accuracy of 600 meta-test tasks using our best method increases from 59.8% to 68.8%.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>0</th>
<th>0.2k</th>
<th>0.5k</th>
<th>1k</th>
<th>2k</th>
<th>5k</th>
<th>10k</th>
<th>15k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>47.0</td>
<td>64.1</td>
<td>65.9</td>
<td>70.0</td>
<td>71.2</td>
<td>70.9</td>
<td>71.3</td>
<td>71.5</td>
</tr>
</tbody>
</table>

Table 3: Pseudo-labeling accuracies (%) during the meta-training process, on miniImageNet 1-shot.

<table>
<thead>
<tr>
<th>Stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>59.8</td>
<td>63.6</td>
<td>65.1</td>
<td>66.9</td>
<td>67.9</td>
<td>68.8</td>
</tr>
</tbody>
</table>

Table 4: Pseudo-labeling accuracies (%) at six recursive stages of meta-test, on miniImageNet 1-shot. Stage-1 is initialization.

Generalization ability. Our LST approach is in principle able to generalize to other optimization-based FSC methods. To validate this, we replace MTL with a classical method called MAML [3]. We implement the experiments of MAML-based LST (using recursive, hard, soft) and compare with the same 4CONV-arch model TPN [13]. On miniImageNet 1-shot, our method gets the accuracy of 54.8% (52.0% for \( w/D \)), outperforming TPN by 2.0% (1.6% for \( w/D \)). On the more challenging dataset tieredImageNet (1-shot) we achieve even higher superiority, i.e., 2.9% (2.0% for \( w/D \)).

6 Conclusions

We propose a novel LST approach for semi-supervised few-shot classification. A novel recursive-learning-based self-training strategy is proposed for robust convergence of the inner loop, while a cherry-picking network is meta-learned to select and label the unsupervised data optimized in the outer loop. Our method is general in the sense that any optimization-based few-shot method with different base-learner architectures can be employed. On two popular few-shot benchmarks, we found consistent improvements over both state-of-the-art FSC and SSFSC methods.

Acknowledgments

This research is part of NExT research which is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its IRC@SG Funding Initiative. It is also partially supported by German Research Foundation (DFG CRC 1223), and National Natural Science Foundation of China (61772359, 61671289, 61771301, 61521062).
References


[34] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In NIPS, 2017.


