11-2021

Learning to teach and learn for semi-supervised few-shot image classification

Xinzhe LI

Jianqiang HUANG

Yaoyao LIU

Qin ZHOU

Shibao ZHENG

See next page for additional authors

Citation

LI, Xinzhe; HUANG, Jianqiang; LIU, Yaoyao; ZHOU, Qin; ZHENG, Shibao; SCHIELE, Bernt; and SUN, Qianru. Learning to teach and learn for semi-supervised few-shot image classification. (2021). *Computer Vision and Image Understanding*. 212, 1-10.
Available at: https://ink.library.smu.edu.sg/sis_research/6628

Published in Computer Vision and Image Understanding. November 2021, pp. 1-10.
https://doi.org/10.1016/j.cviu.2021.103270

Learning to teach and learn for semi-supervised few-shot image classification

Xinzhe Li [a], Jianqiang Huang [b], Yaoyao Liu [c], Qin Zhou [b], Shibao Zheng [a,*], Bernt Schiele [c], Qianru Sun [d,*]

[a] *Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, 201100, China*
[b] *Alibaba DAMO Academy, Hangzhou, 310012, China*
[c] *Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, 66123, Germany*
[d] *School of Computing and Information Systems, Singapore Management University, 178902, Singapore*

A R T I C L E I N F O

Communicated by Nikos Paragios

MSC:
41A05
41A10
65D05
65D17

Keywords:
Few-shot learning
Meta-learning
Semi-supervised learning

A B S T R A C T

This paper presents a novel semi-supervised few-shot image classification method named Learning to Teach and Learn (LTTL) to effectively leverage unlabeled samples in small-data regimes. Our method is based on self-training, which assigns pseudo labels to unlabeled data. However, the conventional pseudo-labeling operation heavily relies on the initial model trained by using a handful of labeled data and may produce many noisy labeled samples. We propose to solve the problem with three steps: firstly, *cherry-picking* searches valuable samples from pseudo-labeled data by using a soft weighting network; and then, cross-teaching allows the classifiers to teach mutually for rejecting more noisy labels. A feature synthesizing strategy is introduced for cross-teaching to avoid clean samples being rejected by mistake; finally, the classifiers are fine-tuned with a few labeled data to avoid gradient drifts. We use the meta-learning paradigm to optimize the parameters in the whole framework. The proposed LTTL combines the power of meta-learning and self-training, achieving superior performance compared with the baseline methods on two public benchmarks.

1. Introduction

Today's deep neural networks (DNNs) often require large amounts of labeled training data to achieve their best performance (Yann et al., 2015; He et al., 2016; Shelhamer et al., 2017). However, collecting sufficient samples with correct labels is expensive for some tasks due to the scarcity of expert knowledge. In this case, DNNs tend to overfit and perform poorly with only a few samples.

Researchers have explored a variety of methods in order to overcome the challenging problem. One commonly used solution is meta-learning, which follows a unified two-loop training process: (1) Inner-loop: updating the parameters of the networks (also denoted as base-learners) with gradient descent on the support set of a single (few-shot) task; (2) Outer-loop: optimizing the hyper-parameters (also denoted as meta-learners) needed by the inner-loop on the query sets from multiple tasks. A representative approach is Model-Agnostic Meta-Learning (MAML), which learns to initialize parameters for the network in order to fast adapt them to new similar tasks (Finn et al., 2017).

Another intriguing idea is to increase the amount of training data by exploiting unlabeled samples, which can be obtained much more quickly and cheaply. A classic and intuitive method is self-training 1u (Yarowsky, 1995; Triguero et al., 2015; Berthelot et al., 2019) which first trains a network with labeled data (training the initial model) and then enlarges the labeled set based on the most confident predictions on the unlabeled set (recursive pseudo-labeling). It can outperform regularization-based methods (Miyato et al., 2016; Laine and Aila, 2017), especially when labeled data is scarce.

However, one critical problem hinders the usage of this semi-supervised approach for the few-shot scenarios. Specifically, the pseudo-labeling operation relies heavily on the initial model trained by using the labeled set. If insufficient labeled samples are provided to train the initial model, the pseudo-labeling operation will produce many noisy labels and severely harm the subsequent training procedures.

To take advantage of self-training for semi-supervised few-shot classification (SSFSC) and alleviate the interference of noisy labels produced by pseudo-labeling, we propose a novel **learning to teach and learn (LTTL)** method. LTTL follows the same two-loop training process of meta-learning:

In the inner-loop, the whole process starts with adapting the task-specific network using the meta-learned initialization parameters just as (Finn et al., 2017; Sun et al., 2019b; Rusu et al., 2019). And then three steps are used: (1) searching the informative samples among the unlabeled set with the proposed **soft weighting network (SWN)** (we name this procedure as *cherry-picking*); (2) performing **cross-teaching** for the double-branch classifiers to reject more noisy labeled samples; (3) fine-tuning the classifiers with only labeled samples from support

* Corresponding authors.
E-mail addresses: sbzh@sjtu.edu.cn (S. Zheng), qianrusun@smu.edu.sg (Q. Sun).

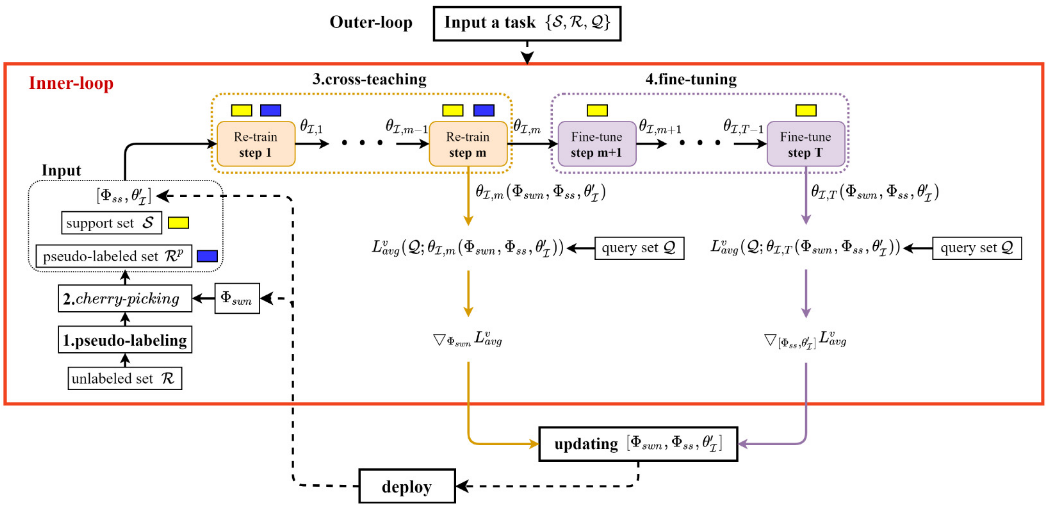

Outer-loop
Input a task $\{\mathcal{S},\mathcal{R},\mathcal{Q}\}$
Inner-loop
3.cross-teaching
4.fine-tuning
Re-train step 1
Re-train step m
Fine-tune step m+1
Fine-tune step T
$\theta_{\mathcal{I},1}$
$\theta_{\mathcal{I},m-1}$
$\theta_{\mathcal{I},m}$
$\theta_{\mathcal{I},m+1}$
$\theta_{\mathcal{I},T-1}$
Input
$[\Phi_{ss},\theta'_{\mathcal{I}}]$
support set $\mathcal{S}$
pseudo-labeled set $\mathcal{R}^p$
2.cherry-picking
Φ_{swn}
1.pseudo-labeling
unlabeled set $\mathcal{R}$
$\theta_{\mathcal{I},m}(\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}})$
$\theta_{\mathcal{I},T}(\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}})$
$L^v_{avg}(\mathcal{Q};\theta_{\mathcal{I},m}(\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}}))$
$L^v_{avg}(\mathcal{Q};\theta_{\mathcal{I},T}(\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}}))$
query set $\mathcal{Q}$
query set $\mathcal{Q}$
$\nabla_{\Phi_{swn}} L^v_{avg}$
$\nabla_{[\Phi_{ss},\theta'_{\mathcal{I}}]} L^v_{avg}$
updating $[\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}}]$
deploy

Fig. 1. The two-loop training procedure of LTTL. *cherry-picking* is used to process unlabeled samples for the inner-loop process. The updating procedure of inner-loop contains m steps of re-training and $T-m$ steps of fine-tuning. For re-training process, cross-teaching is used to deal with pseudo-labeled data obtained from *cherry-picking*. The meta-parameters $[\Phi_{swn},\Phi_{ss},\theta'_{\mathcal{I}}]$ are updated in the outer-loop by using the validation losses L^v_{avg}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

set to avoid the drifting problem. Although *cherry-picking* exploits valuable information from the unlabeled set, the operation relies on the initial model's performance, just as the standard self-training process. It is impossible for cherry-picking to deal with the remaining noisy labeled samples in the subsequent updating procedure. Therefore, in this paper, we propose to use double-branch classifiers as the base-learners and let them update their parameters in a cross-teaching manner. The noisy labeled samples will be filtered out gradually in the updating steps. Moreover, in order to solve the problem that the cross-teaching process may wrongly reject some clean samples, new training samples are synthesized with the mixup method (Zhang et al., 2018a).

In the outer loop, meta-learners (initialization parameters of the classifiers and SWN) are optimized by using the validation losses computed with the query set of the task. The whole process is illustrated in Fig. 1.

In a nutshell, our proposed LTTL learns to accumulate *semi-supervised learning experience* from SSFSC tasks to adapt the network to new similar tasks quickly. The contributions of this paper can be summarized as

(1) A novel self-training strategy is proposed for semi-supervised few-shot classification (SSFSC), which effectively exploits unlabeled data with pseudo-labeling. It can also exploit recursive training to improve performance.
(2) A novel meta-learned framework for processing unlabeled data, which prevents the models from drifting due to label noise. The framework contains several steps: cherry-picking, cross-branch teaching, and clean data fine-tuning.
(3) Our method achieves satisfying performance on two versions of ImageNet benchmarks — miniImageNet (Vinyals et al., 2016) and tieredImageNet (Ren et al., 2018a).

Compared to the conference version of our paper (Li et al., 2019), this version additionally presents the following contributions:

(1) We propose to use double-branch classifiers with cross-teaching updating manner to replace the original simple classifier in Li et al. (2019). Cross-teaching first gradually learn to filter out noisy labeled samples, and then it synthesizes new training samples to fetch up the rejected ones, which can effectively reduce the effect of noisy labeled data and improve the performance.
(2) We use a new evaluation process with a larger meta-test dataset containing 6000 random unseen tasks, obtaining more stable test results.
(3) More thorough experiments and analyses are provided to demonstrate the favorable performance of our method.

2. Related work

Few-shot classification (FSC). Most FSC works are based on supervised learning. They can be roughly divided into three categories: (1) data augmentation based methods (Wang et al., 2018; Xian et al., 2019; Zhang et al., 2018b) directly deal with the data deficiency problem in FSC, which conditionally generate training data or features for classifiers. (2) metric learning methods (Sung et al., 2018; Chen et al., 2019; Hou et al., 2019) learn a distance metric for image features which is effective with few training samples. (3) gradient descent methods in meta-learning (Finn et al., 2017, 2018; Antoniou et al., 2019; Zhang et al., 2018b; Sun et al., 2019b,a; Liu et al., 2020), which are commonly used, exploit a two-loop training process that learns meta-learners in the outer loop to adjust the updating algorithm of base-learners in the inner loop so that they can be effectively trained on a new few-shot task. MAML (Finn et al., 2017) and its variants (Finn et al., 2018; Antoniou et al., 2019) meta-learn the initial parameters for the CNN model. Zhang et al. (2018b) proposed MetaGAN, which meta-learns to generate fake samples for the inner loop. In our LTTL, we also adopt the two-loop training process of the meta-learning framework. Different from previous works, we propose a new inner-loop updating process and meta-learners that deal with noisy labels from pseudo-labeled samples, particularly for semi-supervised few-shot learning.

Semi-supervised few-shot classification (SSFSC). Semi-supervised learning on FSC tasks aims to improve classification accuracy by adding many unlabeled data in training. Ren et al. (2018a) proposed three semi-supervised variants of ProtoNets (Snell et al., 2017), basically using the Soft k-Means method to improve the clustering centers (prototypes) with the additional unlabeled data. Liu et al. (2019) proposed a transductive propagation network (TPN) to propagate labels from labeled data to unlabeled ones with a relation graph and meta-learned the key hyperparameters of TPN. Yu et al. (2020) designed a transfer learning framework to utilize the auxiliary information from labeled base-class data and unlabeled novel-class data. The model is further enhanced by using MixMatch (Berthelot et al., 2019). Simon et al. (2020) learned dynamic classifiers from a few samples based on a subspace method. Differently, we build our method based on the classical and straightforward self-training (Yarowsky, 1995) and meta-learning method (Finn et al., 2017; Sun et al., 2019b) without requiring a new design of a semi-supervised network. Rohrbach et al. (2013) proposed to further leverage external knowledge, such as the semantic attributes of categories, to solve not only the few-shot but also zero-shot problems. In this paper, we exploit a simple but effective semi-supervised technique — self-training (Dong-Hyun, 2013), to enhance the inner-loop of meta-learning process for dealing with the SSFSC problem.

Learning from noisy labels. In our work, pseudo labels can be regarded as noisy labels. In the field of noisy label learning, many popular methods are about sample selection/weighting that automatically adjust the weight of noisy samples during training (Ren et al., 2018b; Han et al., 2018; Arazo et al., 2019; Huang et al., 2019; Li et al., 2020). Their weighting is often relying on the loss or entropy of each noisy sample. For instance, Ren et al. (2018b) meta-learned the soft weight for each training sample. Huang et al. (2019) filtered out samples that have high average losses after several training epochs. Arazo et al. (2019) learned a Beta mixture model from the "loss data" and use this model to assign weights on training samples. Our proposed method utilizes cross-branch sample selection (Han et al., 2018; Yu et al., 2019; Li et al., 2020), which is practical and convenient to exploit for our self-training process.

3. Problem definition and denotation

There are three different training settings for few-shot image classification: **supervised, semi-supervised and transductive** settings. In the **standard supervised** setting, only a few labeled samples (called support set) are provided for each novel class. In the **transductive** setting, the test samples of novel classes (also called query samples) are assumed available and regarded as unlabeled samples, which give additional information about the distribution of novel classes on top of the support set. In the **semi-supervised** setting, an additional unlabeled set is given for providing for training the model, while the query set is unknown during training and only used at inference time. Our proposed method adopts the semi-supervised setting as Ren et al. (2018a).

Specifically, the episodic training of meta-learning framework (Vinyals et al., 2016) is used, which consists of the **meta-train** and **meta-test** phases. The input of the two phases are **tasks**, not data-points such as images. Let us revisit the meta-learning framework and detail the denotations.

Tasks of meta-learning (semi-supervised setting). Tasks used in meta-learning can be "sampled" or constructed from a large-scale dataset. For each task, a fixed number of classes are first randomly sampled (C classes are totally sampled), and then the corresponding data points are sampled for each class. Following the settings in Ren et al. (2018a), these samples are further divided into three splits: a small labeled training set called support set $\mathcal{S}$, another labeled set used to evaluate the performance called query set $\mathcal{Q}$, and an unlabeled set $\mathcal{R}$ used for semi-supervised learning.

Meta-train phase. In this phase, the model is trained on multiple training tasks, such that the trained model can master new concepts quickly with only a few training samples. Each meta-train episode contains a two-loop optimization process: *Inner-loop is also called base-learning*, where the models θ (named as base-learners) are updated through the training loss on the support set $\mathcal{S}$ and unlabeled set $\mathcal{R}$ by performing gradient descent:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L\left(\mathcal{S} \cup \mathcal{R}; \theta_{t-1}\right), \tag{1}$$

where α is the learning rate used in the inner-loop. *Outer-loop is also called meta-learning*, where learnable hyper-parameters Φ (e.g., initial parameters of the model, which are named as meta-learners) needed by the base-learning are optimized with the validation loss on the query set $\mathcal{Q}$:

$$\Phi \leftarrow \Phi - \beta \nabla_{\Phi} L^{v}\left(\mathcal{Q}; \theta_T(\Phi)\right), \tag{2}$$

where β is the learning rate used in the outer-loop. θ_T can be regarded as a function of Φ here.

Meta-test phase. The performance of **meta-train** is evaluated in this phase. Given an unseen ("unseen" means there is no overlap of classes between meta-test and meta-train tasks) task $\mathcal{T}_{un}$, we train a new few-shot model with the given support set $\mathcal{S}_{un}$ as well as unlabeled set $\mathcal{R}_{un}$ of the task as Eq. (1), and the learned hyper-parameters Φ are directly used during the training process. We finally evaluate the model's performance on the query set $\mathcal{Q}_{un}$ of the task. If there are multiple unseen tasks, their average accuracy is reported as the final evaluation.

4. Learning to Teach and Learn (LTTL)

We introduce our proposed LTTL according to the two-loop training process of meta-learning as in (Finn et al., 2017; Sun et al., 2019b; Rusu et al., 2019).

4.1. Inner-loop: learning classifiers for each SSFSC task

The inner-loop of our method is illustrated in Fig. 1 (inside the red box) and Fig. 2. It consists of following several steps: **(1) assigning pseudo labels for the unlabeled dataset** $\mathcal{R}$ with initially trained classifiers; **(2) applying *cherry-picking* to pseudo-labeled samples** by using hard-selection and soft-weighting two operations; **(3) performing cross-teaching updating process** for the double-branch classifiers; and **(4) fine-tuning the classifiers with "clean" data from the support set.** We denote the process that training classifiers with pseudo-labeled data as **re-training**.

4.1.1. Pseudo-labeling

Pseudo-labeling is the first step for the inner-loop. This step first trains task-specific classifiers $\theta_{\mathcal{I}}$ on the support set $\mathcal{S}$. And then pseudo labels of the unlabeled samples in $\mathcal{R}$ are predicted by $\theta_{\mathcal{I}}$. We choose a simple but top-performing method — meta-transfer learning (MTL) (Sun et al., 2019b) as the baseline. Specifically, MTL fixes the feature extracting part (convolutional layers) of the network in the inner-loop and only exploits the final fully-connected (FC) layer as the classifier. This baseline is more suitable for complex base-learning processes due to its good property that takes up minor computing resources. We detail the pseudo-labeling operation for one single classifier in the following.

Given the support set $\mathcal{S}$, a classification loss is used to optimize the classifier θ by performing a few gradient descent steps:

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla_{\theta_{t-1}} L\left(\mathcal{S}; \theta_{t-1}\right), \tag{3}$$

where $t \in \{1, \ldots, T_{pre}\}$ is the updating step, $L\left(\mathcal{S}; \theta_{t-1}\right)$ is the classification loss (e.g., cross-entropy loss l_{ce}) computed with $\mathcal{S}$:

$$L\left(\mathcal{S}; \theta_{t-1}\right) = \frac{1}{|\mathcal{S}|} \sum_{(x_i, y_i) \in \mathcal{S}} l_{ce}\left(f_{\theta_{t-1}}(x_i), y_i\right), \tag{4}$$

where $f_{\theta_{t-1}}(\cdot)$ indicates the classifier (the final fully-connected layer) with parameters θ_{t-1}. The initialization θ_0 is given by θ' meta-learned in the outer-loop. After the updating process, we feed unlabeled samples $\mathcal{R} = \left\{x_i^u\right\}$ to the classifier to get pseudo labels $\left\{\hat{y}_i^u\right\}$ as follows,

$$\hat{y}_i^u = \arg\max\left(f_{\theta_{T_{pre}}}(x_i^u)\right), \quad x_i^u \in \mathcal{R}. \tag{5}$$

The most confident classes selected by the $\arg\max(\cdot)$ function are used as the labels. Our proposed method has multiple classifiers (denoted as $\theta_{\mathcal{I}}$, where $\mathcal{I}$ is the index set). The classifiers are updated identically by Eq. (3), and the pseudo labels are obtained by using the average predictions of the classifiers. However, due to the extreme scarcity of labeled data in the support set, initially trained $f_{\theta_{T_{pre}}}(\cdot)$ may produce many noisy labels.

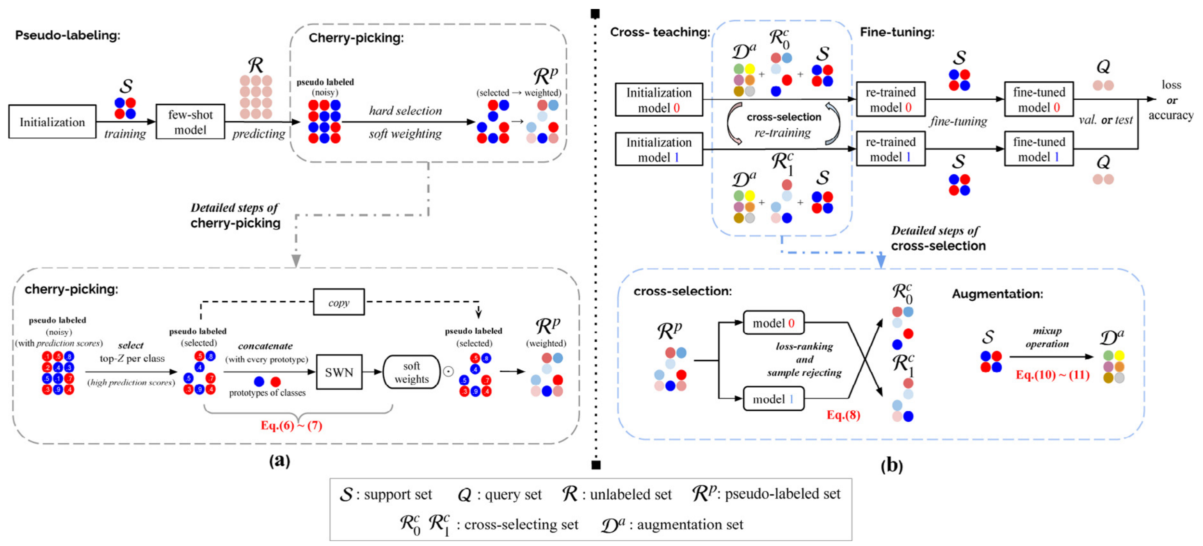


Fig. 2. Details of applying **LTTL** to a single (2-class, 2-shot) task in the inner-loop. (a) Pseudo-labeling and cherry-picking operations for processing the unlabeled set $\mathcal{R}$. Hard-selection and soft weighting are subsequently used in *cherry-picking* to select informative samples. (b) Cross-teaching operation for rejecting more noisy labels. Double-branch classifiers with indexes 0 and 1 are used in this process, which reject noisy labeled samples according to the loss values for each other. After re-training with the pseudo-labeled set $\mathcal{R}^p$, the classifiers are further fine-tuned with the clean set $\mathcal{S}$ to avoid the drifting problem.

4.1.2. *Cherry-picking*

After pseudo-labeling, we propose *cherry-picking* to take advantage of the obtained pseudo-labeled data. *cherry-picking* has two main operations: **hard-selection** that picks the most confident samples just like the normal self-training paradigm, and **soft-weighting** that up-weighs more informative samples as well as down-weighs less useful ones, as shown in the third part of Fig. 2.

Specifically, we pick up Z samples which have the largest prediction scores (by using the initial classifiers) for each class, thus $Z \times C$ samples coming from C classes are selected in the pseudo-labeled set. We denoted this newly obtained set as $\mathcal{R}^p$. Instead of using $\mathcal{R}^p$ directly in the re-training process, we compute soft weights for the data with a soft-weighting network (SWN) to enhance the effect of those more valuable samples. As for the structure design of SWN, the obtained weights should reflect the relations between samples and the semantic representations of all the classes, so we refer to a metric-learning FSC method (Sung et al., 2018) which effectively makes use of relations between support and query samples for classification.

First, we compute the prototype of each class by averaging the features of all its samples in the support set (in the 1-shot case, the unique sample feature is used as the prototype):

$$P_c = \frac{\sum_k f_{\Phi_{ss}}(x_{c,k})}{K}, \tag{6}$$

where $c \in [1, \ldots, C]$ is the class index, $k \in [1, \ldots, K]$ is the sample index in one class, $x_{c,k} \in \mathcal{S}$. We use $f_{\Phi_{ss}}(\cdot)$ to denote the feature extractor part of the model and Φ_{ss} denotes its parameters. The inputs of $f_{\Phi_{ss}}(\cdot)$ are images and the outputs are feature representations.

And then, given a pseudo-labeled sample $(x_i, y_i) \in \mathcal{R}^p$, we concatenate its feature with each of C prototypes $\{P_c\}$ and input them to SWN. The soft weight for the cth class is as follows,

$$w_{i,c} = f_{\Phi_{swn}}\left(\left[f_{\Phi_{ss}}(x_i); P_c\right]\right), \tag{7}$$

where $[\cdot;\cdot]$ is the concatenate operation and Φ_{swn} denotes the parameters of SWN. The inputs for the SWN (denoted as $f_{\Phi_{swn}}(\cdot)$) are concatenated features $\left[f_{\Phi_{ss}}(x_i); P_c\right]$ and outputs are the soft weights $\{w_{i,c}\}$. Both $f_{\Phi_{ss}}(\cdot)$ and $f_{\Phi_{swn}}(\cdot)$ are CNNs in our method. After obtaining $\{w_{i,c}\}$, they are further normalized over all the classes through a softmax layer.

In *cherry-picking*, hard-section can reject some noisy-labeled data. However, the operation can only be used at the start of the inner-loop, and many noisy-labeled samples still remain in the pseudo-labeled set, which harms the subsequent retraining process.

4.1.3. *Cross-teaching of double-branch classifiers*

To further reduce the interference of noisy labels after cherry-picking, we re-design the re-training scheme in the inner-loop, which can reject more noisy labeled samples dynamically. Specifically, we refer to the sample selection method in data cleaning (Han et al., 2018; Yu et al., 2019; Wang et al., 2019; Huang et al., 2019), and improve the original self-training process using two classifiers $\theta_{\mathcal{I}}$ $(\mathcal{I} = \{0, 1\})$, which allows each one to "teach" the other how to learn with the noisy labels. We name the new proposed re-training process as *cross-teaching*, it first performs cross-selection as well as feature synthesizing for *cherry-picked* data, and then updates the parameters of classifiers mutually by letting them "teach" each other.

Cross-selection & feature synthesizing. During training, the DNN models tend to first learn knowledge from "easy" samples with small losses (Arpit et al., 2017). According to this finding, we can leverage each sample's loss value to identify noisy labeled ones that are hard to be rejected by hard-selection. For the classifiers, the samples with the highest loss values in $\mathcal{R}^p$ are filtered out **mutually** with a reject ratio $R(t)$, thus more clean datasets can be constructed as:

$$\mathcal{R}^c_{i/\mathcal{I}} = \arg\min_{\mathcal{R}' : |\mathcal{R}'| \geq R(t)|\mathcal{R}^p|} L(\mathcal{R}'; \theta_i), \quad i \in \mathcal{I} \tag{8}$$

where the subscript "$i/\mathcal{I}$" indicates that two datasets are constructed in a "cross-selection" manner with the two classifiers $\theta_{\mathcal{I}}$, just as in the methods (Han et al., 2018; Yu et al., 2019) and is illustrated in the bottom part of Fig. 2. $R(t)$ is a function of updating step t, which changes linearly at first and then remains unchanged after a fixed number:

$$R(t) = \min\left\{\frac{t}{t_k} r, r\right\}, \tag{9}$$

where t_k is the step at which the reject ratio stops changing, and r is the final constant reject ratio. Compared to the conventional noisy data learning tasks, each SSFSC task includes rather fewer training samples in both the support set $\mathcal{S}$ and the pseudo labeled set $\mathcal{R}$, so filtering out estimated noisy labeled samples as in (Han et al., 2018; Yu et al., 2019; Wang et al., 2019; Huang et al., 2019) may waste some valuable samples. In order to solve this problem, we synthesize new datapoints to make up those rejected data. In specific, we use mixup (Zhang et al., 2018a) which can be easily applied to the samples with different categories. This method synthesizes new samples with the convex combinations of data pairs (x_i and x_j) as well as their corresponding labels (y_i and y_j):

$$x^s = \lambda x_i + (1-\lambda) x_j, \tag{10}$$

$$y^s = \lambda y_i + (1-\lambda) y_j, \tag{11}$$

where λ is a variable randomly sampled from the beta distribution $Beta(\alpha, \beta)$. This linear combination regularizes the network to prefer simple linear behavior between training instances, such as to reduce oscillations in the regions far from them. As indicated in Zhang et al. (2018a), applying this operation to noisy labeled data can improve the robustness of model training. However, we empirically find that it is not quite effective in the more challenging SSFSC tasks (with fewer training samples). Instead, we choose to synthesize new samples with the support set $\mathcal{S}$, and the obtained augmentation dataset is denoted as $\mathcal{D}^a$.

Mutual-teaching for the classifiers updating. After obtaining $\mathcal{R}^c_{\mathcal{I}}$ and $\mathcal{D}^a$, the double-branch classifiers learn to update their parameters in a mutual-teaching manner for each re-training step (as shown in the second part of Fig. 2). The classifiers' parameters are first initialized as $\theta_{\mathcal{I},0} \leftarrow \theta'_{\mathcal{I}}$, where $\theta'_{\mathcal{I}}$ ($\mathcal{I} = \{0,1\}$) are task-generic parameters meta-learned by the outer loop. We then update $\theta_{\mathcal{I},0}$ by performing gradient descent steps on three data splits: $\mathcal{R}^c_{\mathcal{I}}$, $\mathcal{S}$ as well as $\mathcal{D}^a$. Assuming there are total T steps for parameters updating, re-training takes the first $1 \sim m$ steps ($t \in \{1, \dots, m\}$), we have

$$\theta_{\mathcal{I},t} \leftarrow \theta_{\mathcal{I},t-1} - \alpha \nabla_{\theta_{\mathcal{I},t-1}} L_{re}\left(\mathcal{D}; \theta_{\mathcal{I},t-1}\right), \tag{12}$$

where α is the learning rate used for gradient descent, and $\mathcal{D} = \mathcal{S} \cup \mathcal{R}^c_{\mathcal{I}} \cup \mathcal{D}^a$. L_{re} denotes the classification loss of re-training steps computed as follows,

$$\begin{aligned} &L_{re}\left(\mathcal{D}; \theta_{\mathcal{I},t}\right) = \\ &\frac{1}{|\mathcal{S} \cup \mathcal{D}^a|} \sum_{(x_i,y_i)\in \mathcal{S}\cup\mathcal{D}^a} l_{ce}\left(f_{\theta_{\mathcal{I},t}}(x_i), y_i\right) \\ &+ \frac{1}{\left|\mathcal{R}^c_{\mathcal{I}}\right|} \sum_{(x_i,y_i)\in \mathcal{R}^c_{\mathcal{I}}} l_{ce}\left(\mathbf{w}_i \odot f_{\theta_{\mathcal{I},t}}(x_i), y_i\right). \end{aligned} \tag{13}$$

The conventional cross-entropy losses are computed for samples in $\mathcal{S} \cup \mathcal{D}^a$. And for samples in $\mathcal{R}^c_{\mathcal{I}}$, their predictions are weighted by $\mathbf{w}_i = \{w_{i,c}\}_{c=1}^{C}$ before going into the softmax layer. As $R(t)$ starts from 0 and increases linearly, more and more noisy labeled samples are filtered out mutually by the two classifiers, it seems that they teach each other how to learn with noisy labels. The process continues until at step t_k and their performance will become stable since then.

4.1.4. Fine-tuning classifiers with the support set

After m re-training steps, the classifiers are further fine-tuned with only labeled samples (from the support set $\mathcal{S}$) for the rest $T-m$ steps,

$$\theta_{\mathcal{I},t} \leftarrow \theta_{\mathcal{I},t-1} - \alpha \nabla_{\theta_{\mathcal{I},t-1}} L(\mathcal{S}; \theta_{\mathcal{I},t-1}), \tag{14}$$

where L is the conventional classification loss with the same computing process as Eq. (4). We find that the fine-tuning process is necessary to avoid the drifting problem and boost the final performance of our method.

Iterating "teach and learn" with fine-tuned models. Conventional self-training often follows an iterative procedure, aiming to obtain a gradually enlarged labeled set (Yarowsky, 1995; Triguero et al., 2015). Similarly, our method can be iterated once fine-tuned classifiers $\theta_{\mathcal{I},T}$ are obtained, i.e., using $\theta_{\mathcal{I},T}$ to predict more reliable pseudo labels for $\mathcal{R}$ and re-train the classifiers again. When the size of $\mathcal{R}$ is big enough, e.g., 100 samples per class, we can split $\mathcal{R}$ into multiple subsets (e.g., 10 subsets and each one has 10 samples) and perform the "teach and learn" process recursively each time on a new subset. We validate through experiments that first splitting $\mathcal{R}$ and then performing recursive training on subsets obtains better results than that using the large $\mathcal{R}$ and re-training only once.

4.2. Outer-loop: updating meta-learners $[\Phi_{swn}, \Phi_{ss}, \theta'_{\mathcal{I}}]$

As described by the **meta-train phase** (Eq. (2)) in Section 3, gradient descent methods typically use the final obtained models θ_T (in the inner-loop) to compute the validation loss on the query set $\mathcal{Q}$, which is used for optimizing the meta-learners (Sun et al., 2019b; Finn et al., 2017). In this paper, we have multiple meta-learners: Φ_{swn}, Φ_{ss} and $\theta'_{\mathcal{I}}$ (Φ_{ss} is originally used in MTL (Sun et al., 2019b)). We propose to update them with the validation losses calculated at different inner-loop updating steps, aiming to optimize them particularly towards specific purposes, as shown in Fig. 1. Specifically, $[\Phi_{ss}, \theta'_{\mathcal{I}}]$ work for both feature extraction and final classification, which affect the whole self-training process, so we choose to optimize them by using the average loss after the last updating steps T. While Φ_{swn} produces soft weights to refine the re-training process, and its quality should be evaluated by classifiers $\theta_{\mathcal{I},m}$ after the final re-training steps m. Two optimization functions are as follows,

$$\Phi_{swn} \leftarrow \Phi_{swn} - \beta_1 \nabla_{\Phi_{swn}} L^v_{avg}(\mathcal{Q}; \theta_{\mathcal{I},m}(\Phi_{swn}, \Phi_{ss}, \theta'_{\mathcal{I}})), \tag{15}$$

$$[\Phi_{ss}, \theta'_{\mathcal{I}}] \leftarrow [\Phi_{ss}, \theta'_{\mathcal{I}}] - \beta_2 \nabla_{[\Phi_{ss}, \theta'_{\mathcal{I}}]} L^v_{avg}(\mathcal{Q}; \theta_{\mathcal{I},T}(\Phi_{swn}, \Phi_{ss}, \theta'_{\mathcal{I}})). \tag{16}$$

where β_1 and β_2 are meta learning rates that are manually set in experiments. The average loss L^v_{avg} is computed as

$$L^v_{avg}(\mathcal{Q}; \theta_{\mathcal{I},t}(\Phi_{swn}, \Phi_{ss}, \theta'_{\mathcal{I}})) = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} L^v(\mathcal{Q}; \theta_{i,t}(\Phi_{swn}, \Phi_{ss}, \theta'_i)). \tag{17}$$

5. Experiments

We evaluate the proposed **LTTL** method in terms of few-shot image classification accuracy in semi-supervised settings. Below we describe the benchmarks we evaluate, details of settings, and ablation study.

5.1. Datasets and implementation details

Datasets. We conduct our experiments on two subsets of ImageNet (Russakovsky et al., 2015). **miniImageNet** was firstly proposed by Vinyals et al. (2016) and has been widely used in supervised FSC works (Finn et al., 2017; Ravi and Larochelle, 2017; Sun et al., 2019b; Rusu et al., 2019; Grant et al., 2018; Franceschi et al., 2018), as well as semi-supervised works (Liu et al., 2019; Ren et al., 2018a; Yu et al., 2020; Simon et al., 2020). In total, there are 100 classes with 600 samples of 84 × 84 color images per class. The 100 classes are divided into 64, 16, and 20 classes for meta-train, meta-validation, and meta-test respectively. **tieredImageNet** was proposed by Ren et al. (2018a). It includes a much larger number of 608 categories compared with miniImageNet. These classes are from 34 super-classes which are divided into 20 for meta-train (351 sub-classes), 6 for meta-validation (97 sub-classes), and 8 for meta-test (160 sub-classes). The average image number per class of tieredImageNet is 1281. All images are also resized to 84 × 84.

Network architectures. The backbone network architecture is adopted from MTL (Sun et al., 2019b), it contains 4 residual blocks and each block has 3 convolutional layers with 3 × 3 kernels. At the end of each block, a 2 × 2 max-pooling layer is applied. The number of filters are 64, 128, 256 and 512 for the 4 blocks respectively. Following residual blocks, a mean-pooling layer is applied to compress the feature maps to a 512-dimension embedding. The architecture of the proposed SWN consists of 2 convolutional layers with 3 × 3 kernels, followed by 2 fully-connected layers with dimensions 8 and 1, respectively.

Ablative settings. In order to show the effectiveness of our method, we design following settings belonging to two groups: with and without meta-training. Following are the detailed ablative settings. *no selection* denotes performing self-training without using any processing

Table 1
Classification accuracy (%) of different methods on miniImageNet and tieredImageNet datasets. There are three parts for each table: Part 1 represents different semi-supervised few-shot classification (SSFSC) methods, Part 2 represents the baseline method MTL and Part 3 represents our proposed methods. “1”: 4 CONV, “2”: ResNet-12, “3”: WideResNet-28.

Method	miniImageNet	
	1-shot	5-shot
Masked Soft k-Means[1], Ren et al. (2018a)	50.4 ± 0.3	64.4 ± 0.2
TPN[1], Liu et al. (2019)	52.8 ± 0.3	66.4 ± 0.2
Semi DSN[1], Simon et al. (2020)	53.0 ± 0.82	69.1 ± 0.62
Masked Soft k-Means[2], Ren et al. (2018a)	62.1 ± 1.8	73.6 ± 0.8
TPN[2], Liu et al. (2019)	62.7 ± 1.8	74.2 ± 0.8
TransMatch[3], Yu et al. (2020)	63.0 ± 1.01	**81.2** ± 0.59
MTL[2], Sun et al. (2019b)	61.2 ± 1.8	75.5 ± 0.9
LTL[2] (Ours, LST (Li et al., 2019))	70.1 ± 0.56	78.3 ± 0.26
LTTL[2] (Ours, LTT(s))	**71.2** ± 0.59	78.5 ± 0.27
Method	tieredImageNet	
	1-shot	5-shot
Masked Soft k-Means[1], Ren et al. (2018a)	52.4 ± 0.4	69.9 ± 0.2
TPN[1], Liu et al. (2019)	55.7 ± 0.3	71.0 ± 0.2
Semi DSN[1], Simon et al. (2020)	54.1 ± 0.96	72.1 ± 0.69
Masked Soft k-Means[2], Ren et al. (2018a)	68.6 ± 0.8	81.0 ± 0.8
TPN[2], Liu et al. (2019)	72.1 ± 1.8	83.3 ± 0.8
MTL[2], Sun et al. (2019b)	65.6 ± 1.8	78.6 ± 0.9
LTL[2] (Ours, LST (Li et al., 2019))	79.0 ± 0.57	84.9 ± 0.28
LTTL[2] (Ours, LTT(s))	**80.3** ± 0.54	**85.9** ± 0.28

of pseudo labels. *hard* denotes performing hard-selection for pseudo-labeled data. *hard* (Φ_{ss}, θ') means using hard-selection and meta-training $[\Phi_{ss}, \theta_T]$. *soft* denotes assigning soft weights for selected pseudo-labeled data by the meta-trained SWN. *recursive* applies multiple iterations of self-training based on fine-tuned models. Noting that *recursive* is only used for the meta-test phase, as the meta-trained SWN can be repeatedly used. We also have a comparable setting to *recursive* called *mixing* in which the whole unlabeled set $\mathcal{R}$ is used and run only one re-training round (see Section 4.1.4).

Ablative settings for the cross-teaching. We further design several settings to show the effectiveness of the new proposed LTTL method. Specifically, **LTL** denotes our original method (LST) in the conference version (Li et al., 2019). **LTT** denotes that we use the cross-teaching framework without using the feature synthesizing method. **LTT(pl)** denotes that we use mixup to synthesize new instances from pseudo-labeled set in order to make up the rejected data after cross-selection. **LTT(s)** denotes that we use mixup to synthesize new instances from support set. **LTT(s)** is the final version of our method, so **LTTL** and **LTT(s)** are the same method in the following contents.

5.2. *Overview for two datasets with related methods*

In Table 1, we present our results compared with related methods. The upper table part presents the results on miniImageNet. We can see that the proposed LTTL achieves the best performance for the 1-shot (71.2%) and improves the accuracy by 1.1% compared with LTL (the method in the conference version). Compared with the baseline method MTL (Sun et al., 2019b), LTTL improves the performance by 10.0% and 3.0% respectively for 1-shot and 5-shot, which proves the efficiency of our proposed method with unlabeled data. As for other SSFSC methods, we can see that Masked Soft k-Means (Ren et al., 2018a), TPN (Liu et al., 2019) improve their performance by a large margin (more than 10% for 1-shot and 7% for 5-shot) when they are equipped with pre-trained deeper backbone and use more unlabeled samples (100 per class). Compared with them, our LTTL achieves more than 8.5% and 4.3% improvements respectively for 1-shot and 5-shot tasks with the same amount of unlabeled data on miniImageNet. Compared with the recently proposed method (Yu et al., 2020), LTTL surpasses TransMatch by 8.2% for 1-shot, they obtain a better 5-shot accuracy 81.2%, which may be due to the fact that their method uses a deeper backbone model.

The lower part of the table presents the results on tieredImageNet. Our proposed LTTL performs best for both 1-shot (80.3%) and 5-shot (85.9%), and it surpasses LTL by 1.3% and 1.0% for respectively 1-shot and 5-shot. Compared with MTL (Sun et al., 2019b), LTTL further improves the results by 14.7% and 7.3% for respectively 1-shot and 5-shot. Moreover, our method also surpasses TPN (ResNet-12) by 8.2% and 2.6% for 1-shot and 5-shot.

5.3. *Ablation study*

In Table 2, we provide experimental results for the ablative settings.

Hard-selection & Soft-weighting. Hard-selection operation often brings improvements. In specific, *hard* can improve the performance of 1-shot and 5-shot by 3.5% and 1.0% on miniImageNet, 2.3% and 0.5% on tieredImageNet, compared with *no selection*. Moreover, recursively repeating the operation (*recursive,hard*), when dividing a large number of unlabeled samples into several splits, brings more than 1% average gain in all the cases. As for soft-weighting, the meta-trained SWN can enhance the effect of more valuable samples, leading to better performance. When using SWN individually, *soft* has already achieved comparable results with previous SSFSC methods (Ren et al., 2018a; Liu et al., 2019). When using SWN in cooperation with hard-selection (*hard,soft*), the setting achieves 0.9% and 0.8% improvements on tieredImageNet for 1-shot and 5-shot respectively compared with *hard*(Φ_{ss}, θ'), which also shows that the two operations are complementary.

Recursive setting. Comparing the results of *recursive,hard* with *hard*, we can see that by doing recursive self-training when updating θ, the results are improved in both “meta” and “no meta” scenarios. E.g., it boosts the results by 5.0% when applying recursive training to *hard,soft* for miniImageNet 1-shot. However, when using *mixing,hard,soft* that learns all unlabeled data without *recursive*, the improvement reduces by 3.8%. These observations show that recursive self-training can successfully leverage unlabeled samples. However, *recursive* sometimes brings undesirable results in the cases with “distractors”. E.g., compared with *hard*, the *recursive,hard* brings reduction for 1-shot on both two datasets, which might be due to the fact that disturbances caused by distracting classes in early recursive stages propagate to later stages. Even though our method is slightly more sensitive when adding out-of-distribution samples to the unlabeled dataset, we still obtain the best results compared with other two methods (Ren et al., 2018a; Liu et al., 2019). When using the newly proposed cross-teaching (*r,h,s*+LTT(s)), consistent improvements are achieved, e.g., 1.3% and 1.0% for 1-shot and 5-shot respectively on tieredImageNet.

In Table 3, we further provide ablative settings for the proposed cross-teaching process with three main settings: (1) *soft*, (2) *hard,soft*, which use a small number of unlabeled data (30 per class for 1-shot and 50 per class for 5-shot) as well as (3) *recursive,hard,soft* (*r,h,s*), which uses a larger number of unlabeled data (100 per class for both 1-shot and 5-shot).

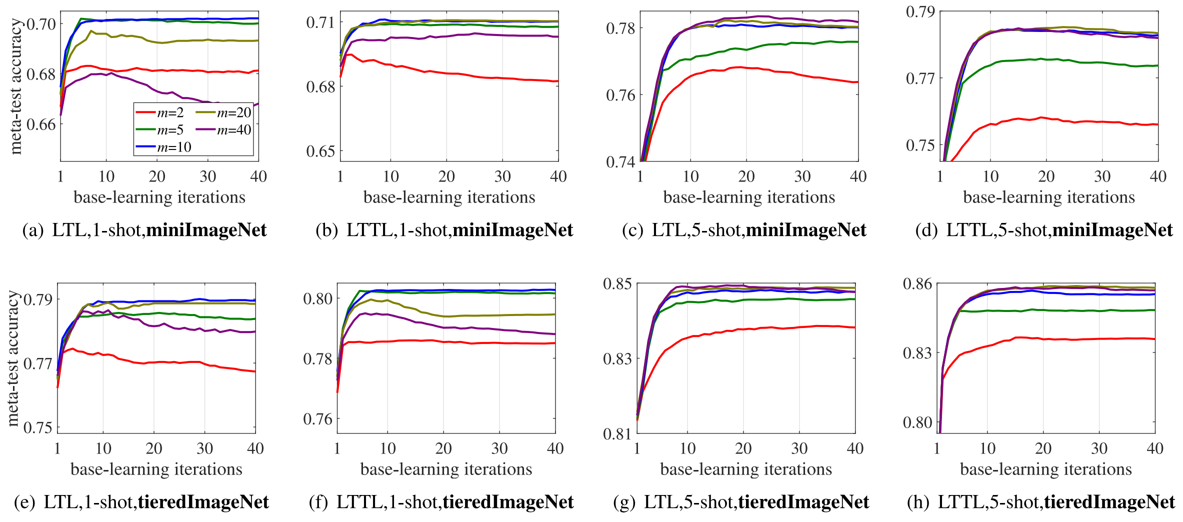
Double-branch classifiers. According to the table, we can see that using double-branch classifiers and let them teach each other in a cross-teaching manner often brings improvements. For example, compared with LTL (the conference version method LST), LTT achieves 0.7% and 0.6% improvements in 1-shot with *hard,soft* setting with a small number of unlabeled data. The performance proves that filtering out high-loss instances can indeed reject some noisy labeled data in few-shot tasks. However, the improvements are often minor in w/$\mathcal{D}$ cases, especially the hard selection operation is used (*hard, soft* as well as *r,h,s*), for example, LTT achieves the same result 63.7% as LTL with *hard, soft* setting (1-shot on miniImageNet). One reason may be that this kind of sample selection method is not quite effective when there are out-of-distribution categories.

Table 2
Classification accuracy (%) in ablative settings (middle blocks) and related SSFSC works (bottom block), on miniImageNet and tieredImageNet. "fully supervised" means the labels of unlabeled data are used. "no meta" means SWN is not used in the framework while "meta" means SWN are used and updated by meta-learning. In the bottom rows, we also list the results of some related methods (Ren et al., 2018a; Liu et al., 2019) "w/$\mathcal{D}$" means using unlabeled data from 3 distracting classes that are **excluded** in the support set. "*r,h,s*" is the abbreviation for *recursive,hard,soft*. ResNet-12 (denoted as "2") is used as the backbone.

		miniImagenet		tieredImagenet		miniImagenet w/$\mathcal{D}$		tieredImagenet w/$\mathcal{D}$	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
fully supervised		80.5± 0.54	83.6± 0.24	86.6± 0.55	88.7± 0.26	–	–	–	–
no meta	*no selection*	59.6± 0.54	75.2± 0.23	67.4± 0.54	81.2± 0.26	54.8± 0.54	73.2± 0.24	66.0± 0.54	79.4± 0.26
	hard	63.1± 0.54	76.2± 0.24	69.7± 0.54	81.7± 0.26	61.2± 0.54	75.1± 0.24	69.0± 0.54	81.0± 0.26
	recursive,hard	64.6± 0.53	76.9± 0.24	72.2± 0.56	83.3± 0.27	61.0± 0.57	75.7± 0.26	68.8± 0.57	81.1± 0.28
meta	*hard* (Φ_{ss}, θ')	64.2± 0.52	76.9± 0.25	74.7± 0.54	83.2± 0.26	62.9± 0.54	75.6± 0.24	73.4± 0.55	82.4± 0.26
	soft	63.0± 0.54	76.3± 0.24	73.0± 0.54	83.5± 0.26	61.5± 0.54	75.2± 0.24	71.1± 0.54	82.2± 0.25
	hard,soft	65.1± 0.54	77.2± 0.24	75.6± 0.53	84.0± 0.26	63.7± 0.53	76.1± 0.23	74.1± 0.54	83.1± 0.26
	mixing,hard,soft	66.3± 0.54	77.6± 0.24	76.7± 0.54	84.2± 0.26	64.7± 0.54	76.6± 0.24	**74.9**± 0.54	83.3± 0.26
	recursive,hard,soft	70.1± 0.56	78.3± 0.26	79.0± 0.57	84.9± 0.28	64.2± 0.56	76.8± 0.27	74.0± 0.58	83.2± 0.28
	r,h,s + LTT(s)	**71.2**± 0.59	**78.5**± 0.27	**80.3**± 0.54	**85.9**± 0.28	**64.8**± 0.56	**76.9**± 0.27	74.0± 0.58	**83.6**± 0.28
Masked Soft k-Means^2		62.1± 1.8	73.6± 0.8	68.6± 1.7	81.0± 0.7	61.0± 1.7	72.0± 0.8	66.9± 1.8	80.2± 0.9
TPN^2		62.7± 1.7	74.2± 0.8	72.1± 1.8	83.3± 0.8	61.3± 1.7	72.4± 0.8	71.5± 1.7	82.7± 0.9
Masked Soft k-Means		50.4± 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
TPN		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

Table 3
Classification accuracy (%) in ablative settings for the cross-teaching framework, on miniImageNet and tieredImageNet. We use three different settings for evaluating the new proposed framework: "*soft*" means only SWN is used to process the pseudo-labeled samples; "*hard,soft*" means both hard-selection and SWN are used; "*r,h,s*" is the final setting in which recursive self-training is used to exploit large amount of unlabeled data. "w/$\mathcal{D}$" means using unlabeled data from 3 distracting classes that are **excluded** in the support set. "*r,h,s*" is the abbreviation for the *recursive,hard,soft* setting. ResNet-12 is used as the backbone.

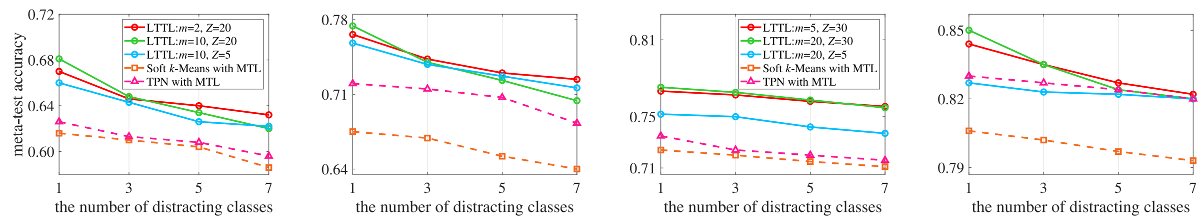
		miniImagenet		tieredImagenet		miniImagenet w/$\mathcal{D}$		tieredImagenet w/$\mathcal{D}$	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
soft	LTL	63.0± 0.54	76.3± 0.24	73.0± 0.54	83.5± 0.26	61.5± 0.54	75.2± 0.24	71.1± 0.54	82.2± 0.25
	LTT	63.5± 0.54	76.3± 0.23	73.4± 0.54	83.9± 0.24	61.9± 0.54	75.0± 0.24	71.5± 0.54	82.7± 0.24
	LTT(pl)	62.6± 0.54	76.0± 0.24	72.5± 0.53	83.5± 0.25	61.2± 0.54	74.6± 0.24	71.1± 0.53	83.0± 0.24
	LTT(s)	**64.0**± 0.54	**76.5**± 0.24	**74.2**± 0.54	**84.4**± 0.24	**62.2**± 0.55	75.2± 0.24	**72.1**± 0.54	**83.1**± 0.24
hard,soft	LTL	65.1± 0.54	77.2± 0.24	75.6± 0.53	84.0± 0.26	63.7± 0.53	76.1± 0.23	74.1± 0.54	83.1± 0.26
	LTT	65.8± 0.55	77.0± 0.25	76.2± 0.54	84.5± 0.26	63.7± 0.54	75.9± 0.24	74.3± 0.55	83.5± 0.24
	LTT(pl)	64.9± 0.54	77.2± 0.24	75.6± 0.54	84.6± 0.25	63.4± 0.55	76.1± 0.24	74.0± 0.53	83.9± 0.24
	LTT(s)	**66.2**± 0.54	77.2± 0.23	**76.7**± 0.54	**84.8**± 0.25	**63.9**± 0.54	**76.2**± 0.24	**74.8**± 0.54	83.9± 0.24
r,h,s	LTL	70.1± 0.56	78.3± 0.26	79.0± 0.57	84.9± 0.28	64.2± 0.56	76.8± 0.27	74.0± 0.58	83.2± 0.28
	LTT	70.3± 0.56	78.1± 0.26	79.3± 0.57	85.5± 0.28	64.5± 0.56	76.9± 0.27	73.9± 0.57	83.6± 0.27
	LTT(pl)	70.5± 0.57	78.4± 0.26	79.6± 0.57	85.4± 0.28	64.8± 0.57	76.7± 0.28	73.8± 0.57	83.5± 0.28
	LTT(s)	**71.2**± 0.59	**78.5**± 0.27	**80.3**± 0.54	**85.9**± 0.28	64.8± 0.56	76.9± 0.27	74.0± 0.57	83.6± 0.28



(a) LTL,1-shot,**miniImageNet** (b) LTTL,1-shot,**miniImageNet** (c) LTL,5-shot,**miniImageNet** (d) LTTL,5-shot,**miniImageNet**
(e) LTL,1-shot,**tieredImageNet** (f) LTTL,1-shot,**tieredImageNet** (g) LTL,5-shot,**tieredImageNet** (h) LTTL,5-shot,**tieredImageNet**

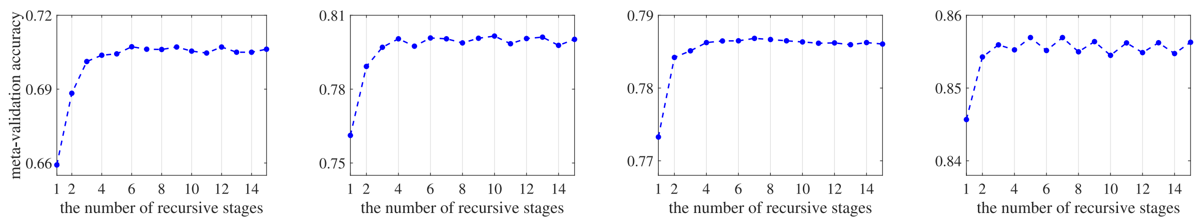
Fig. 3. Classification accuracy using different numbers of re-training steps, e.g. $m = 2$ means using 2 steps for re-training and 38 steps (40 steps in total) for fine-tuning at every recursive stage. Each curve shows the results obtained at the final recursive stage. LTL: our method (LST) in the conference version (Li et al., 2019); LTTL: the new proposed teaching and learning framework.

Applying feature synthesizing to support set *vs.* pseudo-labeled set. Compared with LTL and LTT, we can see that using feature synthesizing to augment the support set can further improve the proposed algorithm's performance. When using small amount of unlabeled samples with *soft* and *hard,soft* settings, LTT(s) achieves more than 1% and 0.5% improvements compared with LTL and LTT for 1-shot, moreover, it obtains best or comparable performance in *recursive,hard,soft* (*r,h,s*)



(a) 1-shot,**miniImageNet** (b) 1-shot,**tieredImageNet** (c) 5-shot,**miniImageNet** (d) 5-shot,**tieredImageNet**

Fig. 4. Classification accuracy with different numbers of distracting classes on the miniImageNet and tieredImageNet. We evaluate our proposed LTTL by using different values for re-training steps m and hard-selection number Z. Noting that (a) and (b) share legend, and (c) and (d) share legend.



(a) 1-shot,**miniImageNet** (b) 1-shot,**tieredImageNet** (c) 5-shot,**miniImageNet** (d) 5-shot,**tieredImageNet**

Fig. 5. Meta-validation accuracy achieved at different numbers (1–15) of recursive stages, on the miniImageNet and tieredImageNet datasets.

for both 1-shot and 5-shot tasks. On the contrary, applying the mixup operation to pseudo-labeled set often performs poorly, e.g., 62.6% with *soft* and 64.9% with *hard,soft* for 1-shot, which are even worse than the single-branch LTL. It proves that mixup is not quite effective to deal with noisy labeled data predicted by few-shot classifiers and using more noisy labeled samples harms the performance.

In the following, we present the experimental results to evaluate the key components of LTTL including re-training steps, distracting classes, pseudo-labeling accuracy, and the effect of recursive stages.

Quantitative analyses on the number of re-training steps. In Fig. 3, we present the results for different re-training steps (2, 5, 10, 20 and 40) on miniImageNet and tieredImageNet. The figure shows two different settings respectively: LTL is our method proposed in the conference version; LTTL that uses new proposed cross-teaching to learn the classifiers.

In 1-shot cases, the results illustrate that too many re-training steps may lead to the drifting problem which causes side effects on performance, and fine-tuning the classifiers with a few clean (labeled) data after re-training steps can effectively relieve the problem. Specifically, both LTL and LTTL achieve their best performance with 10 re-training steps, while using 5 steps obtains close results. When we do not use the fine-tuning operation (m = 40), the performance always declines obviously. Moreover, compared with LTL, using cross-teaching can further reduce the effect of noisy labeled data and perform better. In 5-shot cases, the four figures show similar phenomenon: best performance are often achieved with more re-training steps (at 20 steps or 40 steps) for both settings. Drifting problems are not quite obvious for 5-shot, which may be due to the fact that containing more labeled samples in the support set leads to more stable self-training (cross-teaching) process with pseudo-labeled data.

Quantitative analyses on the number of distracting classes. In Fig. 4, we show the effects of distracting classes on the LTTL and other related methods (improved versions *with* MTL) (Ren et al., 2018a; Liu et al., 2019). We evaluate the proposed method by assigning different values for re-training steps m and hard-selection number (per class) Z. Specifically, we set 2,10 for m and 5,20 for Z in 1-shot cases. For 5-shot cases, we use relatively larger values 5,30 for m and 5,30 for Z. It is clear to see that more distracting classes cause more performance deduction for all methods, e.g., about 6% reduction for 1-shot and 3% reduction for 5-shot when increasing the number of distracting classes from 1 to 7.

Table 4
Pseudo-labeling accuracies (%) during the meta-training process, on miniImageNet (*mini*) and tieredImagenet (*tiered*).

Iteration		0	0.2k	0.5k	2k	5k	10k	15k
mini	1-shot	60.0	64.2	66.7	68.3	72.3	73.3	74.0
	5-shot	80.0	80.9	82.5	82.8	84.8	84.3	84.0
tiered	1-shot	61.0	72.3	72.0	73.1	77.8	77.7	78.3
	5-shot	69.0	86.4	86.3	87.2	88.3	88.5	88.6

Table 5
Pseudo-labeling accuracies (%) at six recursive stages of meta-test, on miniImageNet (*mini*) and tieredImagenet (*tiered*). Stage-1 is initialization.

Stage		1	2	3	4	5	6
mini	1-shot	55.8	63.7	66.8	67.9	68.3	68.6
	5-shot	71.3	76.7	77.9	78.1	78.2	78.3
tiered	1-shot	64.8	74.9	77.2	77.9	78.3	78.3
	5-shot	80.7	84.5	85.3	85.5	85.5	85.6

Quantitative analyses on the number of recursive stages. During meta-validation, we search the best values for the number of recursive stages in 1-shot and 5-shot cases with *recursive,hard,soft* setting, and the results are shown in Fig. 5. We observe that our method's performance is always saturated after running 6 stages for both 1-shot and 5-shot. In experiments, we split 100 samples (per class) as the unlabeled set. At each recursive stage, we sample a subset, i.e., 30 for 1-shot and 50 for 5-shot. After a few stages, the model has sampled and learned all unlabeled samples, therefore, its performance gets saturated. We choose the peak values: 6 stages for 1-shot and 5 stages for 5-shot during meta-test, on both datasets.

The performance of pseudo-labeling. For miniImageNet and tieredImageNet, we record the accuracies of pseudo-labeling for meta-training and meta-test (based on the setting *recursive,hard,soft*), in Tables 4 and 5, respectively. In meta-training, we run the training process with 15k meta iterations, it is clear that the accuracy of pseudo-labeling often grows rapidly from iter = 0 to iter = 5k, and then they reach saturation. We use 6 recursive stages during the meta-test phase according to the results from Fig. 5. The accuracy often grows rapidly for the first three stages. Taking 1-shot case on miniImageNet as an example, from stage-1 to stage-3, the average accuracy of 6,000 meta-test tasks increases from 55.8% to 66.8%, and then it only achieves 1.8% gain after the rest three stages.

6. Conclusion

We propose a novel learning to teach and learn (LTTL) approach for semi-supervised few-shot classification. For "learn", we introduce a recursive-learning-based self-training strategy for the inner loop and meta-learn a soft-weighting network (SWN) to select more informative samples from the pseudo-labeled set in the outer loop. For "teach", we leverage the cross-teaching for double-branch classifiers, which allow one classifier to teach the other one to obstruct the noisy labels according to its learning "experiences". From the model evaluation on SSFSC tasks, we found our approach can achieve consistent improvements over many few-shot learning methods.

CRediT authorship contribution statement

Xinzhe Li: Methodology, Software, Investigation, Writing – original draft, Visualization. **Jianqiang Huang:** Writing – review & editing, Supervision. **Yaoyao Liu:** Software, Writing – review & editing. **Qin Zhou:** Writing – review & editing. **Shibao Zheng:** Supervision, Funding acquisition. **Bernt Schiele:** Conceptualization. **Qianru Sun:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by A*STAR, Singapore under its AME YIRG Grant (Project No. A20E6c0101). It was also partially supported by German Research Foundation (DFG CRC 1223), National Natural Science Foundation of China (62071292, 61771303) and Science and Technology Commission of Shanghai Municipality, China (18DZ2270700).

References

Antoniou, A., Edwards, H., Storkey, A., 2019. How to train your maml. In: ICLR.
Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K., 2019. Unsupervised label noise modeling and loss correction. In: ICML.
Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A.C., Bengio, Y., Lacoste-Julien, S., 2017. A closer look at memorization in deep networks. In: ICML.
Berthelot, D., Carlini, N., Goodfellow, I.J., Papernot, N., Oliver, A., Raffel, C., 2019. MixMatch: A holistic approach to semi-supervised learning. In: NeurIPS.
Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C., Huang, J.-B., 2019. A closer look at few-shot classification. In: ICLR.
Dong-Hyun, L., 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: ICML Workshops.
Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML.
Finn, C., Xu, K., Levine, S., 2018. Probabilistic model-agnostic meta-learning. In: NeurIPS.
Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M., 2018. Bilevel programming for hyperparameter optimization and meta-learning. In: ICML.
Grant, E., Finn, C., Levine, S., Darrell, T., Griffiths, T.L., 2018. Recasting gradient-based meta-learning as hierarchical Bayes. In: ICLR.
Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I.W., Sugiyama, M., 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In: NeurIPS.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: CVPR.
Hou, R., Chang, H., Bingpeng, M., Shan, S., Chen, X., 2019. Cross attention network for few-shot classification. In: NeurIPS.
Huang, J., Qu, L., Jia, R., Zhao, B., 2019. O2U-net: A simple noisy label detection approach for deep neural networks. In: ICCV.
Laine, S., Aila, T., 2017. Temporal ensembling for semi-supervised learning. In: ICLR.
Li, J., Socher, R., Hoi, S.C., 2020. DivideMix: Learning with noisy labels as semi-supervised learning. In: ICLR.
Li, X., Sun, Q., Liu, Y., Zheng, S., Zhou, Q., Chua, T.-S., Schiele, B., 2019. Learning to self-train for semi-supervised few-shot classification. In: NeurIPS.
Liu, Y., Lee, J., Park, M., Kim, S., Yang, Y., 2019. Transductive propagation network for few-shot learning. In: ICLR.
Liu, Y., Schiele, B., Sun, Q., 2020. An Ensemble of Epoch-Wise Empirical Bayes for Few-Shot Learning. In: ECCV.
Miyato, T., Dai, A.M., Goodfellow, I.J., 2016. Virtual adversarial training for semi-supervised text classification. ArXiv 1605.07725.
Ravi, S., Larochelle, H., 2017. Optimization as a model for few-shot learning. In: ICLR.
Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J.B., Larochelle, H., Zemel, R.S., 2018. Meta-learning for semi-supervised few-shot classification. In: ICLR.
Ren, M., Zeng, W., Yang, B., Urtasun, R., 2018. Learning to reweight examples for robust deep learning. In: ICML.
Rohrbach, M., Ebert, S., Schiele, B., 2013. Transfer learning in a transductive setting. In: NIPS.
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large scale visual recognition challenge. Int. J. Comput. Vis. 115 (3), 211–252.
Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R., 2019. Meta-learning with latent embedding optimization. In: ICLR.
Shelhamer, E., Long, J., Darrell, T., 2017. Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39 (4), 640–651.
Simon, C., Koniusz, P., Nock, R., Harandi, M., 2020. Adaptive subspaces for few-shot learning. In: CVPR.
Snell, J., Swersky, K., Zemel, R.S., 2017. Prototypical networks for few-shot learning. In: NIPS.
Sun, Q., Liu, Y., Chen, Z., Chua, T.-S., Schiele, B., 2019a. Meta-transfer learning through hard tasks. ArXiv 1910.03648.
Sun, Q., Liu, Y., Chua, T.-S., Schiele, B., 2019b. Meta-transfer learning for few-shot learning. In: CVPR.
Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H.S., Hospedales, T.M., 2018. Learning to compare: relation network for few-shot learning. In: CVPR.
Triguero, I., García, S., Herrera, F., 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowl. Inf. Syst. 42 (2), 245–284.
Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., Wierstra, D., 2016. Matching networks for one shot learning. In: NIPS.
Wang, Y., Girshick, R.B., Hebert, M., Hariharan, B., 2018. Low-shot learning from imaginary data. In: CVPR.
Wang, X., Wang, S., Shi, H., Wang, J., Mei, T., 2019. Co-mining: Deep face recognition with noisy labels. In: ICCV.
Xian, Y., Sharma, S., Schiele, B., Akata, Z., 2019. f-VAEGAN-D2: A feature generating framework for any-shot learning. In: CVPR.
Yann, L., Yoshua, B., Geoffrey, H., 2015. Deep learning. Nature 521 (7553), 436.
Yarowsky, D., 1995. Unsupervised word sense disambiguation rivaling supervised methods. In: ACL.
Yu, Z., Chen, L., Cheng, Z., Luo, J., 2020. TransMatch: A transfer-learning scheme for semi-supervised few-shot learning. In: CVPR.
Yu, X., Han, B., Yao, J., Niu, G., Tsang, I.W., Sugiyama, M., 2019. How does disagreement help generalization against label corruption? In: ICML.
Zhang, R., Che, T., Grahahramani, Z., Bengio, Y., Song, Y., 2018b. MetaGAN: An adversarial approach to few-shot learning. In: NeurIPS.
Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D., 2018a. mixup: Beyond empirical risk minimization. In: ICLR.

Xinzhe Li received his B.S. degree in electronic information engineering from Dalian University of Technology, Dalian, China, in 2015. He is current a Ph.D. student at the Department of Electronic Engineering, Shanghai Jiao Tong University (SJTU), Shanghai, China. His current research interests include few-shot learning, long-tailed recognition.

Jianqiang Huang is with Alibaba DAMO Academy as well as Nanyang Technological University. He received the second prize of National Science and Technology Progress Award in 2010 and the first prize of Scientific and Technological Award of Zhejiang Province in 2018. His research interests focus on computer vision and machine learning in urban computing.

Yaoyao Liu is currently a Ph.D. student at Max Planck Institute for Informatics, Saarland Informatics Campus, Germany. From June 2018 to August 2019, he was a research intern at the School of Computing, National University of Singapore. Before this, he obtained his bachelor's degree at Qiushi Honors College, Tianjin University. His research interests include few-shot learning, meta learning, incremental learning and image generation.

Qin Zhou received her Ph.D. degree in Information and Communication Engineering from Shanghai Jiao Tong University in March, 2019. Before that, she was a visiting student at Professor Haibin Ling's lab from Oct, 2016 to July, 2018. She is currently working in the Alibaba DAMO academy as a senior algorithm engineer. Her research interests include computer vision, machine learning, convex optimization.

Shibao Zheng received his B.S. degree in communication engineering from Xidian University, Xi'an and M.S.degree in the signal and information processing from the 54th institute of CETC, Shijiazhuang, China, in 1983 and 1986, respectively. He is currently a professor at the Department of Electronic Engineering department and vice director of Elderly Health Information and Technology Institute, Shanghai Jiao Tong University (SJTU), Shanghai, China. His current research interests include urban video surveillance system, intelligent video analysis, and elderly health technology, etc.

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 "iplome 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 Ph.D. 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.

Qianru Sun is a Tenure-Track Assistant Professor in the School of Information Systems, Singapore Management University. From 2018 to 2019, she was a Joint Research Fellow at the National University of Singapore and MPI for Informatics. From 2016 to 2018, she held the Lise Meitner Award Fellowship of MPI for Informatics. In 2016, she got her Ph.D. degree from Peking University. Her research interests are computer vision and machine learning that aim to develop efficient algorithms and systems for visual understanding. In particular, her research topics include object recognition, semantic segmentation, person reidentification, image generation, meta-learning, semi-supervised learning, self-supervised learning, and incremental learning.