



Learning sample-aware threshold for semi-supervised learning

Qi Wei^{1,2} · Lei Feng² · Haoliang Sun¹ · Ren Wang¹ · Rundong He¹ · Yilong Yin¹

Received: 31 May 2023 / Revised: 24 August 2023 / Accepted: 3 October 2023 /

Published online: 18 January 2024

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2024

Abstract

Pseudo-labeling methods are popular in semi-supervised learning (SSL). Their performance heavily relies on a proper *threshold* to generate hard labels for unlabeled data. To this end, most existing studies resort to a manually pre-specified function to adjust the threshold, which, however, requires prior knowledge and suffers from the scalability issue. In this paper, we propose a novel method named Meta-Threshold, which learns a dynamic confidence threshold for each unlabeled instance and does not require extra hyperparameters except a learning rate. Specifically, the instance-level confidence threshold is automatically learned by an extra network in a meta-learning manner. Considering limited labeled data as meta-data, the overall training objective of the classifier network and the meta-net can be formulated as a nested optimization problem that can be solved by a bi-level optimization scheme. Furthermore, by replacing the indicator function existed in the pseudo-labeling with a surrogate function, we theoretically provide the convergence of our training procedure, while discussing the training complexity and proposing a strategy to reduce its time cost. Extensive experiments and analyses demonstrate the effectiveness of our method on both typical and imbalanced SSL tasks.

Keywords Semi-supervised learning · Confidence thresholds · Meta-learning · Bi-level optimization

1 Introduction

Semi-supervised learning (SSL) (Zhu and Goldberg 2009; Sohn et al. 2020) aims to improve model performance by leveraging both abundant unlabeled data and limited labeled data. SSL algorithms provide a solution to explore the latent pattern underlying unlabeled data, which reduces requirements of a large amount of annotations (Sohn et al. 2020). Most of the previous SSL studies heavily rely on the *pseudo-labeling* strategy (Lee 2013; Sohn et al. 2020) that generates a hard label for unlabeled sample and trains the deep model on these pseudo-labels.

Editors: Vu Nguyen, Dani Yogatama.

Extended author information available on the last page of the article

For pseudo-labeling methods (Lee 2013; Sohn et al. 2020; Zhang et al. 2021; Xu et al. 2021), it is essential to set a proper threshold for selecting reliable pseudo-labels for unlabeled data. For example, FixMatch (Sohn et al. 2020) selected high-confidence pseudo-labels via a fixed threshold (e.g., 0.95 for CIFAR Krizhevsky and Hinton (2009) and 0.65 for ImageNet (Deng et al. 2009)). However, as reported in Xu et al. (2021), fixing the threshold in the entire training process could mitigate the learning efficiency and raise the error rate of pseudo-labels, especially in the early learning stage.

To address this issue, subsequent works (Xu et al. 2021; Guo and Li 2022; Zhang et al. 2021; Saito et al. 2021) that dynamically generate the threshold to enable more robust SSL have been proposed. For instance, Xu et al. (2021) translated the fixed threshold to a loss threshold and selected the unlabeled data whose loss values (evaluated on pseudo-labels) are smaller than the loss threshold. Then, these selected data are incorporated into the training set, while the loss threshold gradually decreases over training iterations. Zhang et al. (2021) leveraged the idea of curriculum learning (Bengio et al. 2009) to take into account the learning status of each class and flexibly adjusted thresholds for different classes at each time step via a preset function.

Despite the decent performance of the pseudo-labeling methods mentioned above, they share two common drawbacks. Firstly, they (Xu et al. 2021; Guo and Li 2022; Zhang et al. 2021) always resort to manually pre-specified functions to adjust the threshold. This tends to be infeasible when we know little knowledge of underlying datasets or when the label conditions are too complicated. Secondly, these methods (Xu et al. 2021; Guo and Li 2022; Zhang et al. 2021) usually involve at least two hyper-parameters, which requires complex cross-validation phase and thus suffer from the scalability issue (Franceschi et al. 2018) when we apply them to real-world application.

To address the two drawbacks mentioned above, this paper presents a simple yet effective strategy to automatically learn sample-aware confidence thresholds for each unlabeled data. In contrast with previous works, our method does not resort prior knowledge to pre-define a function for adjusting thresholds while including only one hyper-parameter. Besides, to the best of our knowledge, we for the first time introduce instance-level thresholds, which is inspired by that the deep model has different learning capabilities for different categories even for different examples. Figure 1a shows a practical example. Intuitively, setting instance-level thresholds is more logical and beneficial to generate more accurate pseudo-labels for unlabeled instances, further facilitating deep model's learning.

Specifically, we leverage the idea of meta-learning (Finn et al. 2017) to construct a lightweight meta-net (e.g., three-layer MLP) for explicitly modeling the instance-level thresholds (finally obtain a set of thresholds for all unlabeled data). Thanks to the universal approximation theorem (Hornik et al. 1989) of multilayer feedforward networks, our meta-net can be considered as a generalized version of the pre-defined functions mentioned above (Zhang et al. 2021; Xu et al. 2021). In this way, our framework contains a classifier network and a meta-net, where the training problem of two networks is in a nested optimization scheme. This optimization problem can be solved by a bi-level strategy, which is presented as 1) **Inner loop**. Generate instance-level thresholds for all unlabeled instances and utilizes the hard pseudo labels to train the classifier network, 2) **Outer loop**. Update all parameters of the meta-net by a small scale of meta-data which are constructed on the labeled data.

An appealing feature of this formulation is that the inner loop can be viewed as a mapping from the sample threshold space into the meta-net parameter space, and the outer loop performs the optimization on thresholds. Since the indicator function $\mathbb{1}(\cdot)$, which is non-differentiable, explicitly exists in the pseudo-labeling framework, we thus leverage

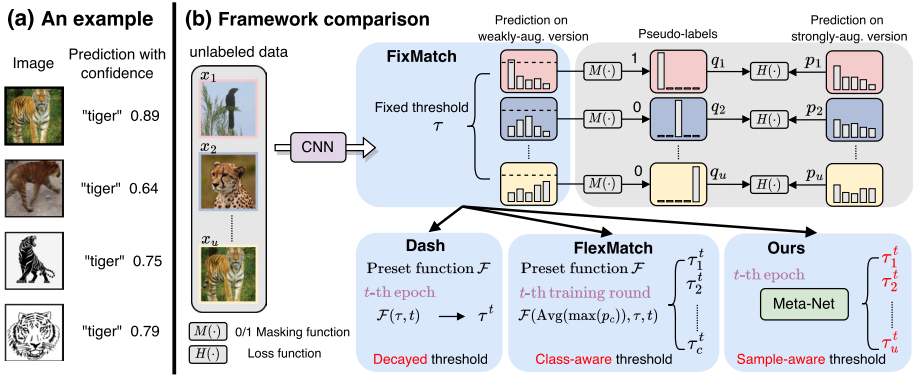


Fig. 1 a An example illustrates that deep models have different learning capabilities for different examples in class *tiger*. b Review of the pseudo-labeling training framework and comparison FixMatch (Sohn et al. 2020) with three improved algorithms on the fixed confidence threshold. Compared to decayed and class-level thresholds in Dash (Xu et al. 2021) and FlexMatch (Zhang et al. 2021), our method designs a meta-net which generates a more refined confidence threshold for each unlabeled example (i.e., sample-level thresholds)

a surrogate function to approximate it, making the bi-level optimization problem reachable. In Fig. 1b, we compare our method with vanilla FixMatch (Sohn et al. 2020) and two improved methods (Xu et al. 2021; Zhang et al. 2021), which highlights the merits of our method such as avoiding preset function and no prior knowledge is required.

Our contributions can be summarized as follows:

- We propose a simple yet effective training framework (named Meta-Threshold, **Meta-T**) based on bi-level optimization for threshold-based SSL, which enjoys the following benefits: 1) **Meta-T** learns thresholds of unlabeled sample automatically through bi-level optimization, avoiding the pathology of conventional threshold-based methods’ reliance on strong prior knowledge on data. 2) **Meta-T** only includes one extra hyper-parameter, i.e., the learning rate of the meta-net, which is not sensitive and thus does not require complex cross-validation.
- We introduce the surrogate function to replace the indicator function. Further, we theoretically provide the convergence of our framework and demonstrate that it enjoys a convergence rate of $\mathcal{O}(1/\epsilon^2)$.
- We integrate the proposed **Meta-T** into the framework of curriculum learning dubbed Green **Meta-T**, which significantly reduce the training cost of our learning algorithm with only slight loss of accuracy.
- Our method can be applied to solve both the conventional and imbalanced SSL tasks, exhibiting great potential in real-world applications.

2 Related work

Deep Semi-Supervised Learning As a common learning paradigm, deep SSL exhibits remarkable performance in leveraging a great deal of unlabeled data to train the deep model. Current deep SSL methods can be roughly divided into three categories: consistency-based methods, pseudo-labeling methods, and hybrid methods. The key idea

of consistency-based methods is that forcing the model's output of original unlabeled data and perturbed unlabeled data to keep the same (Laine and Aila 2016; Tarvainen and Valpola 2017; Xie et al. 2020). Pseudo-labeling methods, which are also called self-learning in previous works, belong to an iterative mechanism that uses limited labeled data to train the model to predict unlabeled data. Then, the generated labels of unlabeled data are introduced to train the model Lee (2013). Hybrid approaches (Sohn et al. 2020; Zhang et al. 2021; Xu et al. 2021) always integrate the above two methods with strong augmentation strategies (e.g., RandAugment (Cubuk et al. 2020) and CTAugment (Berthelot et al. 2019)).

Imbalanced Semi-Supervised Learning To improve the universality of SSL algorithms, some works (Kim et al. 2020; Wei et al. 2021; Guo and Li 2022) turn attention to more challenging settings like SSL under *class-imbalanced* label distribution. DARP (Kim et al. 2020) designed a distribution-aligning manner to modify biased pseudo-labels to match the true class distribution. However, this method requires prior knowledge about data distribution, which is hard to fulfill in real applications. For this, DARP manages to estimate the class distribution by a confusion matrix between labeled and unlabeled data. CReST (Wei et al. 2021) is based on a typical self-training strategy that adaptively adds pseudo-labeled data to the training set according to the label frequency.

Meta-Learning also known as “learning to learn”, has been widely applied to several weakly-supervised tasks, such as noisy labels learning (Shu et al. 2019; Sun et al. 2022), out-of-distribution learning (Guo et al. 2020), and semi-supervised learning (Wang et al. 2020; Xiao et al. 2021). In SSL fields, some works introduce the idea of meta-learning to learn a set of parameters. For example, Wang et al. (2020) proposed a framework to learn sample weights for all unlabeled data, which aims to give high weights to more reliable pseudo-labels. Xiao et al. (2021) proposed to learn soft labels for unlabeled data while designing a one-order update strategy for bi-level framework.

Relations Two works L2RW (Ren et al. 2018) and MW-Net (Shu et al. 2019) employed bi-level to efficiently learn a set of hyper-parameter. Our work bears three critical differences.

- (1) *Problem setting*: (Ren et al. 2018; Shu et al. 2019) focus on improving the generalization performance of deep models under noisy labels learning, while our work aims to enhance the quality of generated pseudo-labels for unlabeled data in semi-supervised learning.
- (2) *Methodology*: (Ren et al. 2018; Shu et al. 2019) learn a set of sample weights for training (label-corrupted) samples and then minimize the product of training loss and corresponding weight, while our framework generates thresholds which are used to select the high-reliability pseudo-labels instead of directly participating in model's training. Besides, our method obeys the framework of the pseudo-labeling method and thus suffers from the non-differentiable issue of the indicator function, which can be solved by a surrogate function. Eventually, we joint the bi-level training framework with curriculum learning, significantly reducing the cost of bi-level strategy.
- (3) *Theory*: We introduce a surrogate function to replace the indicator function and provide the convergence guarantee of our learning algorithm when the upper bound of the surrogate function is given. Besides, we simply give an analysis of training costs of both Meta-T and Green Meta-T.

3 Preliminaries

Problem setting. In a C -class classification task, we have a set of training data which contains N labeled examples $D^l = \{(\mathbf{x}'_1, \mathbf{y}'_1), \dots, (\mathbf{x}'_N, \mathbf{y}'_N)\}$ and M unlabeled examples $D^u = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ denotes the input d -dimensional feature vector and $\mathbf{y} \in \mathcal{Y}$ is one-hot label. Given a deep model f with learnable parameters \mathbf{w} and a classification loss function $H(\cdot)$ (e.g., cross-entropy loss), the training objective in typical supervised learning is $L_s = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim D^l} H(f(\mathbf{x}), \mathbf{y})$. To achieve higher performance, the training objective of SSL algorithms can be summarised as $L_s + \lambda_u L_u$, where L_u is constructed on D^u and the trade-off coefficient λ_u satisfies $\lambda_u > 0$.

3.1 Confidence thresholds in semi-supervised learning

Due to its simplicity yet great performance, we select FixMatch (Sohn et al. 2020) as an example to illustrate the usage of confidence threshold in pseudo-labeling methods.

The core idea of FixMatch is the introduction of confidence threshold and strong augmentation strategies. To train the classifier on unlabeled data, FixMatch first computes the pseudo-label on the weakly-augmented version of image. For each unlabeled data $\mathbf{x}_m \in D^u$, the prediction of classification network is $p_m = f(\mathcal{A}^w(\mathbf{x}_m); \mathbf{w})$, where \mathcal{A}^w denotes weak augmentation strategies, and the pseudo-label can be written as $\hat{\mathbf{y}}_m = \arg \max(p_m)$. Due to the property of function $\arg \max$, $\hat{\mathbf{y}}_m$ is a one-hot probability distribution. Then, the training loss of \mathbf{x}_m can be summarised as

$$\ell_{\mathbf{x}_m} = \mathbb{1}(\max(p_m) > \tau) \cdot H(\hat{\mathbf{y}}_m, f(\mathcal{A}^s(\mathbf{x}_m); \mathbf{w})), \quad (1)$$

where $\mathbb{1}(\cdot)$ is an indicator function and denotes the selection of high-reliability of pseudo-label, \mathcal{A}^s denotes strong augmentation strategies, and τ is a **fixed constant**. Eventually, the training objective of all unlabeled data is $L_u = \frac{1}{M} \sum_{\mathbf{x}_m \in D^u} \ell_{\mathbf{x}_m}$.

As mentioned before, many related works (Zhang et al. 2021; Xu et al. 2021) modified the fixed constant τ to improve the universality of pseudo-labeling algorithms. However, they always resort to prior knowledge and further design a task-specific function to adjust this value, limiting their application in practice. Thus, in the next section, we devise a framework that does not require pre-defined functions yet enables sample-aware confidence thresholds.

4 Proposed method

Overview. We construct a meta-net (threshold generation network, or TGN) for dynamically produce sample-level threshold. First, we rewrite the learning objective for threshold-based SSL methods. Second, we introduce the architecture of TGN. Then, we solve this meta-optimization problem via bi-level strategy which alternatively trains the classifier and TGN. Eventually, we analyse the convergence of our algorithm and provide a green version of our method which enjoys lower training time.

4.1 Learning with sample-level thresholds

To alleviate the aforementioned issues of previous methods, we want to construct a meta-learning framework that could generate a sample-level confidence threshold for all unlabeled data in each training step. To be specific, given a meta-net \mathcal{V} with parameters Θ , the confidence threshold of unlabeled data \mathbf{x}_m can be written as $\tau_m \leftarrow \mathcal{V}_m(\mathbf{w}, \Theta)$, while the architecture and input of \mathcal{V} is detailedly illustrated in Sect. 4.2. Then, the fixed constant τ in Eq. (1) can be replaced with a sample-level threshold τ_m and the loss of unlabeled data \mathbf{x}_m is formulated as

$$\ell_{\mathbf{x}_m}(\mathbf{w}, \Theta) = \mathbb{1}(\max(p_m) > \mathcal{V}_m(\mathbf{w}, \Theta)) \cdot H(\hat{\mathbf{y}}_m, f(\mathcal{A}^s(\mathbf{x}_m); \mathbf{w})). \tag{2}$$

However, due to the non-differentiable property of the indicator function $\mathbb{1}(\cdot)$, computing partial derivative with respect to Θ in Eq. (2) is infeasible. In the practical training phase, we introduce a modified sigmoid function to replace it, which can be written as $\mathcal{S}(x) = \frac{1}{1 + \exp^{-\beta x}}$ where the input is $\max(f(\mathcal{A}^w(\mathbf{x}_m); \mathbf{w})) - \mathcal{V}_m(\mathbf{w}, \Theta)$ and β is the slope parameter to control the shape of the function.

Discuss about the approximate function $\mathcal{S}(\cdot)$. In Fig. 2, we compare the difference between the indicator function $\mathbb{1}(\cdot)$ and the sigmoid function $\mathcal{S}(\cdot)$. We can observe that the input of function satisfies $\max(f(\mathcal{A}^w(\mathbf{x}_m); \mathbf{w})) - \mathcal{V}_m(\mathbf{w}, \Theta) \in [-1, 1]$. Meanwhile, the first-order and second-order gradient of sigmoid function obviously exist, making backpropagation of the training loss in Eq. (2) possible.

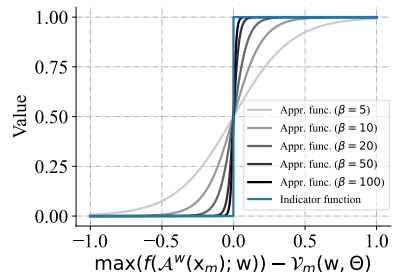
Eventually, the optimal classifier parameters \mathbf{w}^* can be calculated by minimizing the loss

$$\mathbf{w}^*(\Theta) = \arg \min_{\mathbf{w}} L_u = \frac{1}{M} \sum_{\mathbf{x}_m \in D^u} \ell_{\mathbf{x}_m}(\mathbf{w}, \Theta). \tag{3}$$

4.2 Threshold generation network TGN

In this subsection, we design a threshold generation network (TGN), serving as a *meta model*. By summarizing previous works (Zhang et al. 2021; Guo and Li 2022), we found that considering average class confidence provides more valuable information for generating threshold and improves the applicability of methods on extreme data distribution. Thus, we construct the meta-net which learns from instance confidence and average class confidence simultaneously and outputs sample-aware threshold for unlabeled data.

Fig. 2 Compare the indicator function with the approximate function $\mathcal{S}(\cdot)$ with varying β



Formally, given a weakly-augmented version of unlabeled data \mathbf{x}_m , the classifier network f_w gives the prediction result (a soft label) $g(p_m^t)$ in t -th iteration, where $g(\cdot)$ denotes Softmax function. Further, the pseudo-label is $\hat{y}_m^t = \arg \max(g(p_m^t))$. Meanwhile, the average class confidence can be represented as $\bar{p}_c^t = \frac{1}{M} \sum_{m=1}^M g(p_m^t | c = \hat{y}_m^t)$. Note that \bar{p}_c^t can be regarded as an average soft label of class c in time t . Therefore, for unlabeled data \mathbf{x}_m , the generated threshold in t -th iteration is

$$\tau_m^t = \mathcal{V}(g(f(x_m; \mathbf{w})), \bar{p}_c^t; \Theta). \tag{4}$$

As shown in Fig. 3b, we illustrate the architecture of proposed TGN, which belongs to a lightweight net (e.g., three full-connected layers). For \mathbf{x}_m , we connect its prediction result $g(p_m^t)$ (a C -dimension soft label) with the average class confidence \bar{p}_c^t (a C -dimension vector). Therefore, the input layer in TGN is $2C$ dimension.

4.3 Meta-optimization problem

There are two networks in our training framework, including a classification network f_w and a meta-net \mathcal{V}_Θ . The parameters \mathbf{w} and Θ can be optimized by the meta-learning idea (Andrychowicz et al. 2016; Shu et al. 2019). Specifically, we require a small amount of meta-data set which can be sampled from labeled data in SSL task. Since some works (Shu et al. 2019; Sun et al. 2022) proved that the generalization performance of the meta-model largely benefits from a large scale of meta-data, we straightforwardly represent this meta-data set as $D^{\text{meta}} = D^l = \{(\mathbf{x}_i^l, \mathbf{y}_i^l)\}_{i=1}^N$ (i.e., we use the total labeled data for constructing the meta-data set). The optimal parameters Θ^* can be obtained by minimizing the following loss

$$\Theta^* = \arg \min_{\Theta} L_{\text{meta}}(\mathbf{w}^*(\Theta)) = \frac{1}{N} \sum_{i=1}^N H_i(\mathbf{w}^*(\Theta)). \tag{5}$$

For clarity, we represent $H_i(\mathbf{w})$ as $H(\mathbf{y}_i^l, f(\mathbf{x}_i^l; \mathbf{w}))$.

Obtaining the optimal parameters \mathbf{w}^* in Eq. (3) and Θ^* in Eq. (5) is a nested optimization problem. For this, we resort to bi-level training strategy as MAML (Finn et al.

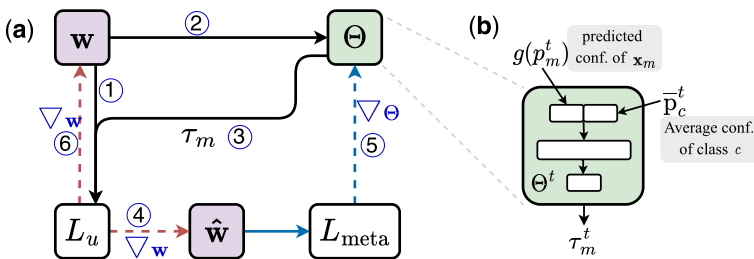


Fig. 3 **a Flowchart of our learning algorithm.** The solid and dashed lines represent forward and backward propagation, respectively. In each iteration, overall training process contains six phases. *Step 1:* feed weakly-augmented images to the classifier network and attain pseudo-labels with prediction confidence. *Step 2:* input a pair of average class confidence and predicted confidence into the meta-net TGN. *Step 3:* leverage generated sample-level threshold τ to select high-reliability data and compute the loss L_u . *Step 4:* update the classifier parameters while holding the computation graph for its gradient. *Step 5:* feed the meta-data into the meta-net, compute the loss L_{meta} and update Θ . *Step 6:* recompute the gradient of L_u w.r.t. \mathbf{w} and update \mathbf{w} . **b Architecture of TGN.** Given an unlabeled sample \mathbf{x}_m , TGN’s input consists of two parts

2017) and update parameters of meta-net with online strategy. To be specific, the training loss of classifier network and meta-net (Eq. (3) and Eq. (5)) can be optimized via the SGD optimizer. In each training iteration, given a mini-batch size number n , we have two batches of meta data and unlabeled data and represent them as $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), \dots, (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$ and $\{\mathbf{x}_1, \dots, \mathbf{x}_{(\mu \times n)}\}$, respectively. Note that we can increase μ to expand the size of unlabeled data in one iteration. In t -th iteration, we formulate the parameter of classifier network as $\mathbf{w}^{(t)}$ and the parameters of the meta-net as $\Theta^{(t)}$. The updates of two networks are as the following three phases.

Algorithm 1 Learning algorithm of **Meta-T**.

Require: Unlabeled/labeled data D^u/D^l , batch size n , a coefficient μ , max iterations T .
Ensure: Classifier network parameter $\mathbf{w}^{(T)}$.

- 1: Initialize $\mathbf{w}^{(0)}$ for classifier network and $\Theta^{(0)}$ for TGN.
- 2: **for** $t = 0$ **to** $T - 1$ **do**
- 3: Random sample $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), \dots, (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$ from D^l and $\{\mathbf{x}_1, \dots, \mathbf{x}_{(\mu \times n)}\}$ from D^u .
- 4: Calculate $\hat{\mathbf{w}}^{(t)}(\Theta)$. ▷ Eq. (6)
- 5: Update $\Theta^{(t+1)}$. ▷ Eq. (7)
- 6: Update $\mathbf{w}^{(t+1)}$. ▷ Eq. (8)
- 7: **end for**

- **Formulating learning manner of classifier network.** Given the learning step with a size of α , the descent direction of the objective loss in Eq. (3) on a mini-batch unlabeled data is

$$\hat{\mathbf{w}}^{(t)}(\Theta) = \mathbf{w}^{(t)} - \alpha \frac{1}{n\mu} \sum_{i=1}^{n\mu} \nabla_{\mathbf{w}} \ell_{\mathbf{x}_i}(\mathbf{w}^{(t)}, \Theta^{(t)}), \tag{6}$$

where $\ell_{\mathbf{x}_i}$ is calculated by Eq. (2).

- **Updating parameters Θ As we obtain parameter $\hat{\mathbf{w}}^{(t)}(\Theta)$ with fixed Θ in Eq. (6),** the update of our meta-net TGN can be achieved by a mini-batch of meta-data $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), \dots, (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$. Specifically, $\Theta^{(t)}$ moves along the direction of direction of gradients *w.r.t.* the objective in Eq. (5)

$$\Theta^{(t+1)} = \Theta^{(t)} - \psi \frac{1}{n} \sum_{i=1}^n \nabla_{\Theta} H_i(\hat{\mathbf{w}}^{(t)}(\Theta)), \tag{7}$$

where ψ denotes the learning step of the SGD optimizer. Note that Θ in this equation is a variable, which enables gradient computation of $\frac{\partial \hat{\mathbf{w}}^{(t)}(\Theta)}{\partial \Theta}$.

- **Updating parameters \mathbf{w} of classifier network.** Eventually, we utilize the updated TGN $\Theta^{(t+1)}$ to regenerate confidence threshold for unlabeled data and update the parameters \mathbf{w} of classifier network

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha \frac{1}{n\mu} \sum_{i=1}^{n\mu} \nabla_{\mathbf{w}} \ell_{\mathbf{x}_i}(\mathbf{w}^{(t)}, \Theta^{(t+1)}). \tag{8}$$

We illustrate the flowchart of our learning algorithm in Fig. 3a, where *Step 4,5,6* represent Eqs. (6), (7) and (8), respectively. Meanwhile, we summarize the overall updating steps in Algorithm 1. Compared to current SSL methods, Meta-T does not rely on any

prior knowledge to predefine the function for adjusting the threshold. We believe that this merit would expand applicability of our method in certain environments where we cannot model the data distribution.

4.4 Convergence analysis

We analyze the convergence of Meta-T and give a rigorously theoretical guarantee.

Lemma 1 (Smoothness). *Suppose the loss function H is L -Lipschitz and smooth, and the approximate function S is ζ -Lipschitz, and $\mathcal{V}(\cdot)$ is differential with δ -bounded gradient and twice differential with \mathcal{B} -bounded Hessian, and the loss function H have ρ -bounded gradients w.r.t. training/meta data and has upper bound with ϕ . Replacing indicator function with S , the gradient of Θ w.r.t. the meta loss is Lipschitz continuous.*

The Proof is shown in Appendix A.1 and Lemma 1 implies that the meta loss w.r.t. the meta-network is smooth-bounded.

Theorem 1 (Convergence) *Based on Lemma 1, let the learning rate α_t satisfies $\alpha_t = \min\{1, \frac{k}{T}\}$, for some $k > 0$, such that $\frac{k}{T} < 1$, and ψ_t , $1 \leq t \leq T$ is a monotone descent sequence, $\psi_t = \min\{\frac{1}{L}, \frac{c}{\sigma\sqrt{T}}\}$ for some $c > 0$, such that $\frac{\sigma\sqrt{T}}{c} \geq L$ and $\sum_{t=1}^{\infty} \psi_t \leq \infty$, $\sum_{t=1}^{\infty} \psi_t^2 \leq \infty$. Then we have $\frac{1}{T} \sum_{t=1}^T \mathbb{E} \left[\left\| \nabla_{L_{\text{meta}}} \left(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}) \right) \right\|_2^2 \right] \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$.*

The Proof is shown in Appendix A.2. To be specific, Theorem 1 means that the our algorithm can achieve $\mathbb{E} \left[\left\| \nabla_{L_{\text{meta}}} \left(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}) \right) \right\|_2^2 \right] \leq \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps, and would eventually convergence to a stationary point with the training iteration step increases.

4.5 Green meta-T: training with lower complexity

Training complexity analysis. Compared with the single-step training procedure, the training process of Meta-T can be divided into three parts, (1) forward and backward passes of the classifier network for computing $\hat{\mathbf{w}}(\Theta)$; (2) forward and backward passes of TGN for updating Θ ; (3) forward and backward passes of classifier network for updating \mathbf{w} . Hence, compared with FixMatch, which only involves one forward and backward pass, Meta-T requires approximately three times of training time.

As summarized by Xu et al. (2021), the main cost of training time is caused by the back-propagation in updating the parameters Θ of the meta-net since the meta-gradient in Eq. (7) needs to compute the similarity between each meta-data and unlabeled data. Therefore, reducing the computation of $\hat{\mathbf{w}}(\Theta)$ would significantly decrease training time. To this end, we change the training procedure that integrates our proposed Meta-T algorithm with curriculum learning and name it **Green Meta-T**. Specifically, we conduct the bi-level strategy (i.e. Meta-T) once for learning the classifier network and TGN, and then continuously do k -step classifier learning. Then, we give the training complexity of Green Meta-T as follows.

Proposition 1 *Suppose a fixed training iteration T , the training time of FixMatch and Meta-T can be represented as T and $3T$, respectively. Given a hyper-parameter k , the training time of Green Meta-T is $\frac{k+2}{k}T$.*

Proposition 1 means that the training complexity of Green Meta-T could gradually reduce to \mathcal{T} with the value of k increases.

5 Experiments

5.1 Experimental settings

Datasets. We select five image classification datasets and three text classification datasets to evaluate the effectiveness of **Meta-T**, including five image benchmarks CIFAR-10 (Krizhevsky and Hinton 2009), CIFAR-100 (Krizhevsky and Hinton 2009), SVHN (Coates et al. 2011), SLT-10 (Netzer et al. 2011), and ImageNet (Deng et al. 2009), three text benchmarks IMDb (Maas et al. 2011), Amazon-5 (Zhang et al. 2015) and Yelp-5 (Zhang et al. 2015). Detailed statistics of these datasets are shown in Table 1.

Implementation Details. Our code is implemented by Pytorch 1.9.0 with GTX 3090. We leverage a pytorch library called **Higher** (Grefenstette et al. 2019) to implement our algorithm, which provides support for higher-order optimization. For all experiments, we repeat five times with different random seeds. Others for two networks are shown below

- For **the classifier**, more information about data preprocessing and training procedure can be found in Table 2.
- For **TGN**, we set the size of meta-data as 32 and utilize Adam optimizer with $1e-3$ learning rate for all training epoches. We construct the three-layers fully-connected MLP for TGN, whose structure is $\{2\mathcal{C}, h, 1\}$. Notably, h is set as 100 for all image datasets and 1000 for all text datasets and \mathcal{C} is the number of categories.

5.2 Results on typical SSL task

Baselines. We categorize compared methods into two types. 1) Threshold-based methods, including Pseudo-Labeling (PL) Lee (2013), FixMatch (Sohn et al. 2020), FlexMatch (Zhang et al. 2021) and Dash (Xu et al. 2021). 2) others, including Π -Model (Sajjadi et al. 2016), MixMatch (Berthelot et al. 2019), UDA (Xie et al. 2020), CoMatch (Li et al. 2021) and SimMatch (Zheng et al. 2022).

Results on four image datasets. We conduct experiments on CIFAR-10, CIFAR-100, SVHN, SLT-10 and ImageNet. The results are shown in Tables 3 and 4. On CIFAR-10 &

Table 1 Details about five tested benchmarks

	Image Datasets					Text Datasets		
	CIFAR-10	CIFAR-100	SVHN	SLT-10	ImageNet	IMDb	Amazon-5	Yelp-5
Classes	10	100	10	10	1000	2	5	5
Labeled data	50000	50000	73257	5000	1200000	50000	250000	250000
Unlabeled data	–	–	–	100000	–	2000	50000	50000
Test data	10000	10000	26032	8000	15000	25000	5000	5000
Image size	32×32	32×32	32×32	96×96	224×224	–	–	–

Table 2 Detailed settings about training procedure of the backbone (the classifier network)

	CIFAR-10	CIFAR-100	SVHN	SLT-10	ImageNet	IMDb	Amazon-5	Yelp-5
Bs (labeled/Meta)	32	32	32	32	32	32	32	32
Bs (unlabeled)	192	192	192	128	64	256	256	256
Model	WRResNet-28-2			WRResNet-37-2	ResNet-50	Pre-trained BERT-Base		
Optimizer	SGD					AdamW		
Epoch	300	300	300	300	300	200	200	200
Learning rate	0.03	0.03	0.03	0.03	0.03	1e-5		
Weight decay	5e-4	1e-3	5e-4	5e-4	5e-4	–	–	–
Momentum	0.9	0.9	0.9	0.9	0.9	–	–	–
Lr scheduler	divided by 10 at [100,200] epoch					divided by 10 at [100,150] epoch		

Table 3 Error rates (%) for previous SOTA methods on CIFAR-10 and CIFAR-100 with varying size of labeled set

Methods	CIFAR-10 (Wide ResNet-28-2)			CIFAR-100 (Wide ResNet-28-8)		
	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels
Π-Model	–	54.26±3.97	14.01±0.38	–	57.25±0.48	37.88±0.11
VAT	74.66±2.12	41.03±1.79	10.51±0.12	85.20±1.40	46.84±0.79	32.14±0.19
MixMatch	47.54±11.50	11.05±0.86	6.42±0.10	67.61±1.32	39.94±0.37	28.31±0.33
UDA	29.05±5.93	8.82±1.08	4.88±0.18	59.28±0.88	33.13±0.22	24.50±0.25
CoMatch	6.91±1.39	4.91±0.33	–	–	–	–
SimMatch	5.60±1.37	4.84±0.39	3.96±0.01	<u>37.81±2.21</u>	25.07±0.32	20.58±0.11
Pseudo-labeling	–	49.78±0.43	16.09±0.28	–	57.38±0.46	36.21±0.19
FixMatch	11.39±3.37	5.07±0.65	4.26±0.05	48.85±1.75	28.29±0.11	22.60±0.12
Dash	9.16±4.31	<u>4.78±0.12</u>	4.13±0.06	44.83±1.36	27.18±0.21	21.97±0.14
FlexMatch	<u>4.97±0.06</u>	4.98±0.09	4.19±0.01	39.94±1.62	26.49±0.20	21.90±0.15
Meta-T (ours)	4.39±0.28	4.10±0.20	<u>4.01±0.09</u>	36.17±1.40	<u>25.81±0.72</u>	<u>20.74±0.23</u>

The best and the second best performance are highlighted by bold and underline, respectively

Table 4 (Left) Error rates (%) for previous methods on SVHN and STL-10 with varying size of labeled set

Error rates (%) ↓	SVHN			STL-10			Top-1 / Top-5 accuracy (%) ↑		
	40 labels	250 labels	1000 labels	1%	10%	100%	ImageNet		
Methods	40 labels	250 labels	1000 labels	1%	10%	100%			
Π-Model	–	18.96±1.92	26.23±0.82	Sup. base-line	25.4 / 48.4	56.4 / 80.4			
VAT	74.75±3.38	4.33±0.12	37.95±1.12	FixMatch	53.4 / 74.4	70.8 / 89.0			
MixMatch	42.55±14.53	3.98±0.23	10.41±0.61	CoMatch	66.0 / 86.4	73.6 / 91.6	80.4 / 94.6		
UDA	52.63±20.51	5.69±2.76	7.66±0.56	SimMatch	<u>67.2 / 87.1</u>	<u>74.4 / 91.6</u>			
ReMixMatch	3.34±0.20	2.92±0.48	5.23±0.45	Meta-T (ours)	67.7 / 87.9	75.0 / 91.7			
				Error rates (%) ↓			IMDb	Amazon-5	Yelp-5
PL	–	20.21±1.09	27.99±0.83	UAD	18.33±0.61	50.29±4.6	47.49±6.83		
FixMatch	3.14±1.60	2.64±0.64	5.17±0.63	FixMatch	7.59±0.28	42.70±0.53	39.56±0.70		
Dash	<u>3.03±1.59</u>	2.17±0.10	<u>3.96±0.25</u>	FlexMatch	7.80±0.23	<u>42.34±0.62</u>	<u>39.01±0.17</u>		
FlexMatch	8.19±3.20	–	5.77±0.18	SoftMatch	<u>7.48±0.12</u>	42.14±0.92	39.31±0.45		
Meta-T (ours)	2.89±0.92	<u>2.29±0.51</u>	3.51±0.34	Meta-T (ours)	7.20±0.20	42.60±0.41	38.44±0.37		

(Right top) Top-1 and Top-5 accuracy (%) on ImageNet test set with varying ratio of labeled samples.

(Right bottom) Error rates (%) for previous methods on three text datasets

The best and the second best performance are highlighted by bold and underline, respectively

100, Meta-T outperforms previous methods in the majority of settings. Under an extremely small size of the labeled set, the superiority of our method is significant. For example, we achieve 1.64% Top-1 accuracy improvements on CIFAR-100 with only 4 samples per class. Compared with threshold-based methods (Lee 2013; Sohn et al. 2020; Zhang et al. 2021; Xu et al. 2021), the improvement of our method is significant. On all settings, Meta-T

constantly outperforms their performance. Eventually, our method also achieved the SOTA performance on ImageNet. By leveraging only 1% labeled data, Meta-T attains 67.7% top-1 accuracy on the test set. Compared to the previous state-of-the-art method SimMatch, the obtained improvement of 0.5% is significant in ImageNet. The superiority of Meta-T on ImageNet can already demonstrate its effectiveness on real-world SSL tasks.

Results on three text datasets. For a fair comparison, we keep the same training procedure with SoftMatch. Under two text benchmarks, including IMBb and Yelp-5, our method consistently achieves the best top-1 accuracy. Especially in Yelp-5 dataset, Meta-T outperforms the second-best method FlexMatch with 0.57% accuracy, which is a huge improvement in such a large-scale dataset.

5.3 Results on imbalanced SSL task

We categorized compared methods into two parts. 1) Threshold-based methods, FixMatch (Sohn et al. 2020, Dash Xu et al. 2021) and FlexMatch (Zhang et al. 2021). 2) Others, cRT (Kang et al. 2019), LDAM, MixMatch (Berthelot et al. 2019), ReMixMatch (Berthelot et al. 2019), DARP (Kim et al. 2020), CReST (Wei et al. 2021) and Adsh (Guo and Li 2022). For constructing imbalanced datasets, we refer to Guo and Li (2022). Specifically, we write the size of two training sets as $N = \sum_{c=1}^C N_c$ and $M = \sum_{c=1}^C M_c$. To construct imbalanced datasets, two parameters (*imbalance ratio*) γ_l, γ_u is introduced, i.e., $\gamma_l = \frac{N_l}{N_c}, \gamma_u = \frac{M_l}{M_c}$. Once γ_l, γ_u and N_1, M_1 are given, we set $N_c = N_1 \cdot \gamma_l^{-\frac{c-1}{c-1}}, M_c = M_1 \cdot \gamma_u^{-\frac{c-1}{c-1}}$ for $1 < c \leq C$. We conduct experiments on two settings, i.e., $N_1 = 500, M_1 = 4000$ and $N_1 = 1500, M_1 = 3000$ with varying imbalanced ratios $\gamma_1, \gamma_2 \in [50, 100, 150]$.

In Table 5, we conduct the comparison experiments on the settings $\gamma = \gamma_1 = \gamma_2$ and report the results. From the results, we can see that (1) our proposed Meta-T achieves the state-of-the-art performance in most cases, showing its robustness in such a data-imbalanced case; (2) with the imbalanced ratio increasing, the performance of our algorithm becomes more significant. Compared to the second best performance (i.e., Adsh), we achieve 1.43% top-1 accuracy improvements under $\gamma = 100$ and 2.42% improvements under $\gamma = 150$. The performance of Meta-T is slightly lower than that of Adsh on the case $N_1 = 500, M_1 = 4000, \gamma = 50$.

5.4 Effectiveness analysis

Pseudo-labels. We verify the quality of produced pseudo-labels on both typical and imbalanced SSL settings.

- **Typical SSL.** In Fig. 4a, b *left*, Meta-T shows greater performance in generating correct pseudo-labels, which benefits from the higher quality of thresholds produced by TGN. In the early learning stage, the number of correct labels in our method is remarkably higher than that in FixMatch, reflecting the superiority of sample-level thresholds. In Fig. 4a, b *right*, we exhibit the results of the number of wrong labels. Due to the poor performance of TGN in the early learning stage, some thresholds with low quality are produced, causing a greater number of wrong pseudo-labels compared with deterministic methods such as FlexMatch. Fortunately, the number of wrong labels decrease with the learning process and is lastly lower than that of FixMatch.

Table 5 Top-1 test accuracy (%) on imbalanced CIFAR-10 under three imbalanced ratio and two different size of labeled set. The backbone is Wide ResNet-28-2

Methods	$N_l = 1500, M_l = 3000$			$N_l = 500, M_l = 4000$		
	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$
Supervised	65.23±0.05	58.94±0.13	55.63±0.38	51.31±0.34	45.82±0.41	40.90±0.39
cRT	67.82±0.14	63.43±0.45	59.56±0.44	56.28±1.45	48.11±0.79	45.02±1.08
LDAM	68.91±0.10	63.15±0.24	58.68±0.30	56.41±0.92	49.27±0.88	45.10±0.75
MixMatch	73.59±0.46	65.03±0.26	62.71±0.29	65.32±1.20	56.41±1.96	52.38±1.88
ReMixMatch	78.96±0.29	72.88±0.12	68.61±0.40	76.83±0.98	70.12±1.23	59.58±1.30
DARP	81.60±0.31	75.23±0.14	69.31±0.26	76.72±0.46	69.41±0.50	61.23±0.31
CReST	82.03±0.26	75.08±0.41	69.84±0.39	76.18±0.36	69.50±0.70	60.81±0.55
Adsh	83.38±0.06	76.52±0.35	71.49±0.30	79.27±0.38	70.97±0.46	62.04±0.51
FixMatch	79.10±0.14	71.50±0.31	68.47±0.15	77.34±0.96	68.45±0.94	60.10±0.82
Dash	81.93±0.10	74.62±0.26	<u>72.29±0.42</u>	77.90±0.39	70.41±0.27	62.11±0.32
FlexMatch	<u>82.86±0.25</u>	<u>75.47±0.41</u>	70.62±0.30	<u>78.69±0.50</u>	<u>71.80±0.29</u>	<u>62.85±0.39</u>
Meta-T (ours)	83.94±0.12	77.80±0.39	73.07±0.58	78.41±0.22	72.40±0.42	64.46±0.60

The best and the second best performance are highlighted by bold and underline, respectively

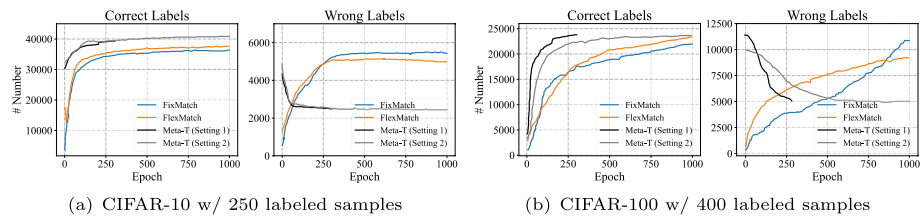


Fig. 4 Visualization of the curve of correct and error pseudo-labels in the selected set with varying training epochs. Note that **Setting 1**: keep the identical training time (FixMatch / FlexMatch: 1000 epochs, Ours: 300 epochs), **Setting 2**: keep the same training epochs as FixMatch

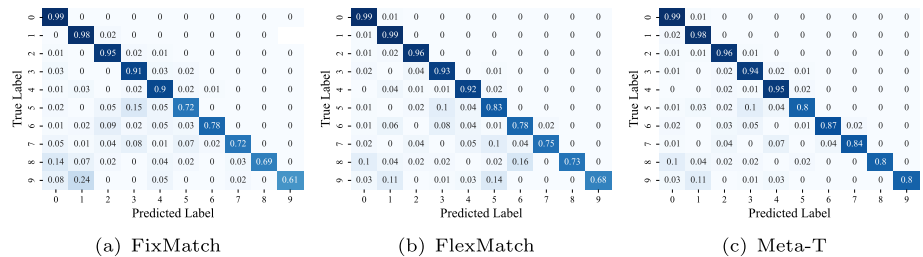


Fig. 5 From the perspective of the confusion matrix, we compare Meta-T with FixMatch and FlexMatch under CIFAR-10 with $\gamma = \gamma_l = \gamma_u = 100, N_l = 1500, M_l = 3000$

- Imbalanced SSL.** We conduct experiments from the perspective of the confusion matrix on unlabeled data and show results in Fig. 5. Thanks to the average class confidence, which is input into the TGN, we believe that TGN can learn the classifier confidence scores regarding varying categories under imbalanced settings and thus adaptively generate class-balanced confidence thresholds. Experimentally, FixMatch focuses

on the studies of majority categories and thus produces unreliable pseudo-labels for minority classes. However, Meta-T achieves significant results on tailed classes and attains more than 80% accuracy on all classes.

Sample-level thresholds. We show the learned thresholds from three aspects to demonstrate the effectiveness of Meta-T.

- **Accuracy.** Figure 6a shows the learned confidence thresholds on CIFAR-10 and CIFAR-100. We can observe that (1) the main learned sample-level thresholds are in the interval of $[0.9, 1.0]$, supporting the prior knowledge that the confidence threshold should be set as 0.95 for CIFAR. The results verify that competitive sample-level thresholds can be learned by TGN; (2) some thresholds less than 0.95 are learned by our algorithm, where the samples can be regarded as hard (or boundary) samples. For this, it is reasonable that TGN gives them relatively low thresholds, which benefits the model's learning for these samples.
- **Robustness.** Figure 6b visualize the produced thresholds and test accuracy (%) under long-tail semi-supervised learning. We can see that our proposed Meta-T learns lower thresholds for tailed classes while keeping high thresholds for many-shot classes. Since a small number of tailed classes, the classifier has moderate or low confidence for these samples. For this, Meta-T produces relatively small thresholds (around 0.5) and thus enables the classifier to learn from more long-tailed unlabeled samples.
- **Stability.** Figure 6c shows the comparison results from dynamic threshold generation. In the beginning, Meta-T tends to initialize thresholds of all unlabeled data as 0.5 and then immediately grow up to 0.95, which is identical to the setting in FixMatch. This result demonstrates thresholds learned by Meta-T are close to the optimal thresholds.

5.5 Sensitivity analysis

We conduct experiments to analyse the sensitivity of Meta-T in three aspects.

The architecture of TGN. To exhibit the impact of the architecture of TGN, we try different MLP architecture settings with different depths and widths and show the results in Table 6 left. It can be seen that varying (five) MLP settings have unsubstantial effects on the final result. Therefore, we prefer to adopt the simple yet effective one, i.e., $\{2C, 100, 1\}$,

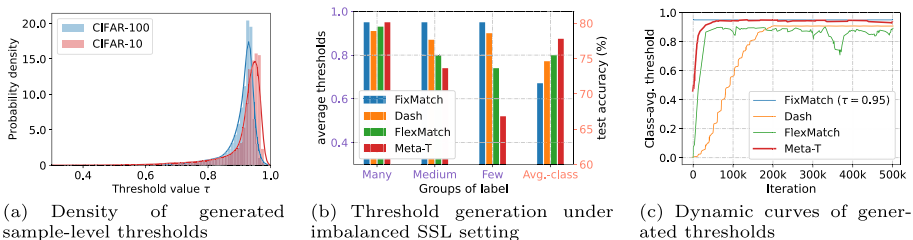


Fig. 6 Results about learned confidence thresholds from three aspects. **(a)** Visualization of generated sample-level thresholds τ for all unlabeled data on balanced CIFAR-10 (250 labels) and CIFAR-100 (2500 labels). **(b)** Visualization of generated thresholds under imbalanced SSL (CIFAR-10 with $N_1 = 1500, M_1 = 3000, \gamma_1 = \gamma_2 = 100$). **(c)** Visualization of class-average confidence threshold v.s. learning processes. We compare Meta-T with others under balanced SLT-10 w/ 40 labels

Table 6 Ablation studies of different settings of the meta-net TGN

$\{2C, h_1, \dots, h_n, 1\}$	CIFAR-10		CIFAR-100		Strategy	CIFAR-10	CIFAR-100
	# 40	# 250	# 400	# 2500		# 250	# 2500
2C - 10 - 1	4.39	4.26	36.17	25.81	Setting 1	4.10	25.81
2C - 100 - 1 (Ours)	4.21	4.10	36.98	26.27	Setting 2	4.21	26.50
2C - 1000 - 1	4.49	4.29	37.02	26.19	Setting 3	4.02	26.14
2C - 10 - 10 - 1	4.78	4.55	37.11	25.42	Setting 4	4.39	26.42
2C - 100 - 100 - 1	4.91	4.49	37.04	26.98	Setting 5	4.17	26.03

(Left): Architecture of TGN. (Right): Training strategies of TGN. Note that “Setting 1”: Adam w/ 1e-3 (Ours), “Setting 2”: Adam w/ 5e-4, “Setting 3”: SGD w/ CosineAnnealingLR [1e-3, 1e-4], “Setting 4”: ASGD w/ MultiSteps [1e-3, 5e-4, 1e-4], “Setting 5”: keep the same as FixMatch

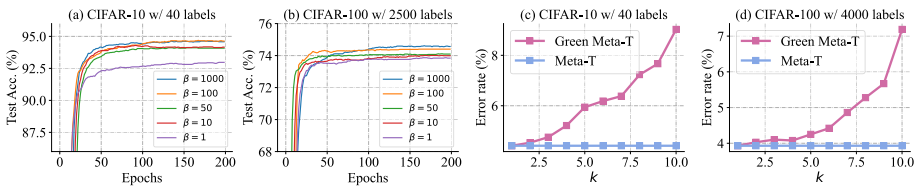


Fig. 7 Sensitivity analysis of the slope parameter β in the surrogate function (a, b) and the step number k of Green Meta-T (c, d)

for all datasets. Meanwhile, we consider that TGN can attain great performance even under a small-scale meta-data due to its tiny number of parameters.

The learning rate ψ w.r.t. the meta-net. Compared with existing methods, our framework introduces an extra hyper-parameter (i.e., the learning rate of meta-net ψ), which does not require complex cross-validation process. Experimentally, we conduct ablation studies and show results with different settings of optimization for TGN in Table 6 right. We can conclude that our algorithm is insensitive to the hyperparameter ψ . Thus, we select a normal setting, i.e., Adam optimizer with 1e-3 learning rate.

The slope parameter β . We conduct experiments with varying settings, $\beta \in \{1, 10, 50, 100, 1000\}$. As shown in Figure 7a, b, the generalization performance improves as β increases at the beginning. When β exceeds 100, the improvement of the performance can be trivial. We thus set $\beta = 100$ for all experiments.

5.6 Efficiency analysis

The step number k of Green Meta-T. We make ablation studies on two SSL settings with $k \in \{1, 2, \dots, 10\}$. In Fig. 7c, d, we can observe that (1) with k increases, the error rate of Green Meta-T gradually increases compared to Meta-T. It is reasonable that the learning of TGN would significantly decrease when conducting more rounds of classifier learning in the outer loop of curriculum learning. (2) A relatively large k might not degrade the performance of Green Meta-T under a mild SSL setting.

To demonstrate efficiency of Green Meta-T, we plot learning curves whose abscissa is the number of accumulative **floating point operations (FLOPs)**. FLOPs are from both the

forward and backward propagation. To show the efficiency of Green Meta-T, we plot train loss, train accuracy, test loss, test accuracy with identical numbers of FLOPs for two learning algorithms in Figure 8. Since the number of epoch for two algorithms is identical, the learning process of Green Meta-T ends after approximately 240k FLOPs. We highlight that Green Meta-T achieves faster convergence than Meta-T when accumulative FLOPs are identical and reduces the computation cost from the second-order derivative at the meta-learning phase.

6 Conclusion

In this paper, we consider sample-level thresholds for pseudo-labeling methods in semi-supervised learning while a simple yet effective framework Meta-T is proposed. Compared with previous methods, Meta-T only contains one hyperparameter and does not rely on pre-set adjustment functions. By constructing a lightweight meta-net, the sample-aware thresholds can be automatically generated by this network. The update of the classifier network and meta-network can be achieved via bi-level strategy. We also design a surrogate function to replace the indicator function in typical pseudo-labeling methods. Further, we theoretically analyze the convergence of Meta-T and provide a solution to reduce training complexity, called Green Meta-T. Extensive experiments on typical and imbalanced SSL demonstrate its effectiveness.

Appendix A: Theoretical proof of our method

A.1 Proofs of smoothness

Given a small amount of meta dataset with n samples $\{(\mathbf{x}'_1, \mathbf{y}'_1), \dots, (\mathbf{x}'_n, \mathbf{y}'_n)\}$ and another unlabeled data $\{\mathbf{x}_1, \dots, \mathbf{x}_{(\mu \times n)}\}$ with size of $\mu \times n$. By replacing the indicator function with the approximate function, the meta loss is $L_{\text{meta}}(\mathbf{w}^*(\Theta)) = \frac{1}{n} \sum_{i=1}^n H(\mathbf{y}'_i, f(\mathbf{x}'_i; \mathbf{w}^*(\Theta)))$ and the training loss is

$$L_{\text{train}}(\mathbf{w}, \Theta) = \frac{1}{n\mu} \sum_{i=1}^{n\mu} \mathbb{1}(\max(f(\mathcal{A}^w(\mathbf{x}_i; \mathbf{w})) > \mathcal{V}_i(\mathbf{w}, \Theta)) \cdot H(\hat{\mathbf{y}}_i, f(\mathcal{A}^s(\mathbf{x}_i; \mathbf{w}))), \quad (\text{A1})$$

where $\mathcal{S}_i(\mathbf{w}, \Theta) = \mathcal{S}(\max(f(\mathcal{A}^w(\mathbf{x}_i; \mathbf{w})) - \mathcal{V}_i(\mathbf{w}, \Theta))$.

Firstly, we recall the update equation of the parameters of TGN as follows:

$$\Theta^{(t+1)} = \Theta^{(t)} - \psi \frac{1}{n} \sum_{i=1}^n \nabla_{\Theta} H(\mathbf{y}'_i, f(\mathbf{x}'_i; \hat{\mathbf{w}}^{(t)}(\Theta))). \quad (\text{A2})$$

To be concise, we formulate $H(\mathbf{y}'_i, f(\mathbf{x}'_i; \hat{\mathbf{w}}^{(t)}(\Theta)))$ as $H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))$. Then, the computation of backpropagation for the above equation can be written as

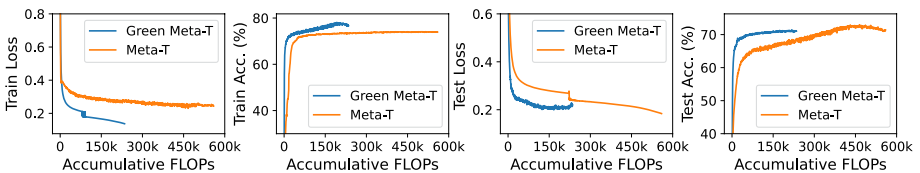


Fig. 8 Results w.r.t. accumulative FLOPs on CIFAR-100 with 1000 labels

$$\begin{aligned}
 & \frac{1}{n} \sum_{i=1}^n \nabla_{\Theta} H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}} = \frac{1}{n} \sum_{i=1}^n \frac{\partial H_i^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}} \sum_{j=1}^{n\mu} \frac{\partial \hat{\mathbf{w}}^{(t)}(\Theta)}{\partial S_j(\mathbf{w}^{(t)}; \Theta)} \frac{\partial S_j(\mathbf{w}^{(t)}; \Theta)}{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)} \frac{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} \\
 & = \frac{-\alpha}{n^2 \mu} \sum_{i=1}^n \frac{\partial H_i^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}} \sum_{j=1}^{n\mu} \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} \\
 & = \frac{-\alpha}{n\mu} \sum_{j=1}^{n\mu} \left(\frac{1}{n} \sum_{i=1}^n \frac{\partial H_i^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}} \right)^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}}.
 \end{aligned} \tag{A3}$$

Let $G_{ij} = \frac{\partial H_i^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}}$ and substitute G_{ij} into Eq. (A3), then

$$\Theta^{(t+1)} = \Theta^{(t)} + \frac{\alpha \psi}{n\mu} \sum_{j=1}^{n\mu} \left(\frac{1}{n} \sum_{i=1}^n G_{ij} \right) \frac{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}}. \tag{A4}$$

Proof The gradient of Θ w.r.t. meta loss can be formulated as:

$$\nabla_{\Theta} H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}} = -\frac{\alpha}{n\mu} \sum_{j=1}^{n\mu} \left(\frac{\partial H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \right) \frac{\partial \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}}. \tag{A5}$$

Let $\mathcal{V}_j(\Theta) = \mathcal{V}_j(\mathbf{w}^{(t)}; \Theta)$ and introduce G_{ij} which is defined in Eq. (A4). Taking the gradient of Θ on both side of Eq. (A5), we attain

$$\nabla_{\Theta^2}^2 H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}} = -\frac{\alpha}{n\mu} \sum_{j=1}^{n\mu} \left[\frac{\partial}{\partial \Theta} (G_{ij}) \Big|_{\Theta^{(t)}} \frac{\partial \mathcal{V}_j(\Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} + (G_{ij}) \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta} \Big|_{\Theta^{(t)}} \right]. \tag{A6}$$

The first term in Eq. (A6) right hand side can be summarized as

$$\begin{aligned}
 & \left\| \frac{\partial}{\partial \Theta} (G_{ij}) \Big|_{\Theta^{(t)}} \frac{\partial \mathcal{V}_j(\Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} \right\| \leq \delta \left\| \frac{\partial}{\partial \hat{\mathbf{w}}} \left(\frac{\partial H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \Theta} \Big|_{\Theta^{(t)}} \right) \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \right\| \\
 & = \delta \left\| \frac{\partial}{\partial \hat{\mathbf{w}}} \left(\frac{\partial H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}} \frac{-\alpha}{n\mu} \sum_{k=1}^{n\mu} \frac{\partial \ell_{x_k}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_k(\Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} \right) \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \right\| \\
 & = \delta \left\| \left(\frac{\partial^2 H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}^2} \Big|_{\hat{\mathbf{w}}^{(t)}} \frac{-\alpha}{n\mu} \sum_{k=1}^{n\mu} \frac{\partial \ell_{x_k}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \frac{\partial \mathcal{V}_k(\Theta)}{\partial \Theta} \Big|_{\Theta^{(t)}} \right) \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \right\| \\
 & \leq \alpha L \delta^2 \phi^2 \zeta^2,
 \end{aligned} \tag{A7}$$

since $\left\| \frac{\partial H(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}}^T \right\| \leq \rho$, $\left\| \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \right\| \leq \phi$, $\left\| \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \right\| \leq \zeta$, $\left\| \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta} \Big|_{\Theta^{(t)}} \right\| \leq \mathcal{B}$.

The second term in Eq. (A6) right hand side can be summarized as

$$\left\| (G_{ij}) \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta} \Big|_{\Theta^{(t)}} \right\| = \left\| \frac{\partial H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}}^T \frac{\partial \ell_{x_j}(S_j(\mathbf{w}))}{\partial S_j(\mathbf{w})} \frac{\partial S_j(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\mathbf{w}^{(t)}} \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta} \Big|_{\Theta^{(t)}} \right\| \leq \rho \phi \zeta \mathcal{B}. \tag{A8}$$

Combining the results in Eq. (A7) and Eq. (A8), we have $\left\| \nabla_{\Theta^2}^2 H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}} \right\| \leq \phi \zeta (\alpha L \delta^2 \phi \zeta + \rho \mathcal{B})$. Define $\hat{L} = \phi \zeta (\alpha L \delta^2 \phi \zeta + \rho \mathcal{B})$, based on the Lagrange mean value theorem, we have:

$$\left\| \nabla L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta_1)) - \nabla L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta_2)) \right\| \leq \hat{L} \|\Theta_1 - \Theta_2\|, \text{ for all } \Theta_1, \Theta_2, \tag{A9}$$

where $\nabla L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta_1)) = \nabla_{\Theta} L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta_1}$. □

A.2 Proofs of convergence

Proof The update of parameters Θ in t -th iteration can be written as $\Theta^{(t+1)} = \Theta^{(t)} - \psi_n \frac{1}{n} \sum_{i=1}^n \nabla_{\Theta} H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{\Theta^{(t)}}$. Training with a mini-batch of meat-data B_t that is uniformly drawn from the data set, we rewrite the equation above as:

$$\Theta^{(t+1)} = \Theta^{(t)} - \psi_t \left[\sum_{i=1}^n \nabla_{\Theta} H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) + \varepsilon^{(t)} \right], \tag{A10}$$

where $\varepsilon^{(t)} = \nabla_{\Theta} H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) \Big|_{B_t} - \nabla_{\Theta} H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))$. Note that the expectation of $\varepsilon^{(t)}$ obeys $\mathbb{E}[\varepsilon^{(t)}] = 0$ and its variance is finite. Consider that

$$\begin{aligned} & H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ &= \underbrace{H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}))}_{\text{term 1}} + \underbrace{H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))}_{\text{term 2}}. \end{aligned} \tag{A11}$$

For term 1, by Lipschitz smoothness of the meta loss function for Θ , we have

$$\begin{aligned} & H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) \\ & \leq \left\langle \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})), \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) \right\rangle + \frac{L}{2} \left\| \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) \right\|_2^2. \end{aligned}$$

According to Eq. (6) (8) (A1), then we have

$$\left\| H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) \right\| \leq \alpha_t \rho^2 + \frac{1}{2} L \alpha_t \rho^2 = \alpha \rho^2 \left(1 + \frac{\alpha_t L}{2} \right) \tag{A12}$$

since $\left\| \frac{\partial H_i(\mathbf{w})}{\partial \mathbf{w}} \Big|_{\hat{\mathbf{w}}^{(t)}} \right\| \leq \rho$, $\left\| \frac{\partial H_i^{\text{meta}}(\mathbf{w})}{\partial \hat{\mathbf{w}}} \Big|_{\hat{\mathbf{w}}^{(t)}} \right\| \leq \rho$.

For term2, considering Lipschitz continuity of $\nabla H_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))$ demonstrated in Lemma 1, we can obtain the following:

$$\begin{aligned} & H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ & \leq \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \Theta^{(t+1)} - \Theta^{(t)} \right\rangle + \frac{L}{2} \left\| \Theta^{(t+1)} - \Theta^{(t)} \right\|_2^2 \\ & = - \left(\psi_t - \frac{L\psi_t^2}{2} \right) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\psi_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 - (\psi_t - L\psi_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle. \end{aligned} \tag{A13}$$

Summing up the Eq. (A12) (A13), the Eq. (A11) can be summarized as

$$\begin{aligned} & H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\ & \leq \alpha \rho^2 \left(1 + \frac{\alpha_t L}{2} \right) - \left(\psi_t - \frac{L\psi_t^2}{2} \right) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\psi_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 - (\psi_t - L\psi_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle. \end{aligned}$$

Rearranging the terms, we can obtain

$$\begin{aligned} & \left(\psi_t - \frac{L\psi_t^2}{2} \right) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \\ & \leq \alpha\rho^2 \left(1 + \frac{\alpha_t L}{2} \right) - \left(\psi_t - \frac{L\psi_t^2}{2} \right) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\psi_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 \\ & \quad - \left(\psi_t - L\psi_t^2 \right) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle. \end{aligned}$$

Summing up the above inequalities and rearranging the terms, we can obtain

$$\begin{aligned} & \sum_{t=1}^T \left(\psi_t - \frac{L\psi_t^2}{2} \right) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \\ & \leq H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(T)}(\Theta^{(T)})) + \\ & \quad \sum_{t=1}^T \alpha\rho^2 \left(1 + \frac{\alpha_t L}{2} \right) - \sum_{t=1}^T \left(\psi_t - L\psi_t^2 \right) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle + \frac{L}{2} \sum_{t=1}^T \left\| \varepsilon^{(t)} \right\|_2^2 \\ & \leq H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^T \alpha\rho^2 \left(1 + \frac{\alpha_t L}{2} \right) \\ & \quad - \sum_{t=1}^T \left(\psi_t - L\psi_t^2 \right) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle + \frac{L}{2} \sum_{t=1}^T \left\| \varepsilon^{(t)} \right\|_2^2. \end{aligned} \tag{A14}$$

We take the expectations *w.r.t.* $\varepsilon^{(N)}$ on both size of Eq. (A14), then we have:

$$\begin{aligned} & \sum_{t=1}^T \left(\psi_t - \frac{L\psi_t^2}{2} \right) \mathbb{E}_{\varepsilon^{(N)}} \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \leq H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) \\ & \quad + \sum_{t=1}^T \alpha\rho^2 \left(1 + \frac{\alpha_t L}{2} \right) + \frac{L\sigma^2}{2} \sum_{t=1}^T \psi_t^2, \end{aligned}$$

since $\mathbb{E}_{\varepsilon^{(N)}} \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle = 0$ and $\left\| \varepsilon^{(t)} \right\|_2^2 \leq \sigma^2$, where σ^2 represents the variance of $\varepsilon^{(t)}$. Eventually, we deduce that

$$\begin{aligned} \min_t \mathbb{E} \left[\left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \right] & \leq \frac{\sum_{t=1}^T \left(\psi_t - \frac{L\psi_t^2}{2} \right) \mathbb{E}_{\varepsilon^{(N)}} \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2}{\sum_{t=1}^T \left(\psi_t - \frac{L\psi_t^2}{2} \right)} \\ & \leq \frac{1}{\sum_{t=1}^T (2\psi_t - L\psi_t^2)} \left[2H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^T \alpha\rho^2(2 + \alpha_t L) + L\sigma^2 \sum_{t=1}^T \psi_t^2 \right] \\ & \leq \frac{1}{\sum_{t=1}^T \psi_t} \left[2H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^T \alpha\rho^2(2 + \alpha_t L) + L\sigma^2 \sum_{t=1}^T \psi_t^2 \right] \\ & \leq \frac{1}{T\psi_t} \left[2H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \alpha_1 \rho^2 T(2 + L) + L\sigma^2 \sum_{t=1}^T \psi_t^2 \right] \\ & \leq \frac{2H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T} \frac{1}{\psi_t} + \frac{2\alpha_1 \rho^2(2 + L)}{\psi_t} + L\sigma^2 \psi_t \\ & = \frac{H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T} \max \left\{ L, \frac{\sigma\sqrt{T}}{c} \right\} + \min \left\{ 1, \frac{k}{T} \right\} \max \left\{ L, \frac{\sigma\sqrt{T}}{c} \right\} \rho^2(2 + L) + L\sigma^2 \min \left\{ \frac{1}{L}, \frac{c}{\sigma\sqrt{T}} \right\} \\ & \leq \frac{\sigma H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{c\sqrt{T}} + \frac{k\sigma\rho^2(2 + L)}{c\sqrt{T}} + \frac{L\sigma c}{\sqrt{T}} = \mathcal{O} \left(\frac{1}{\sqrt{T}} \right). \end{aligned}$$

Therefore, we can conclude that under some mild conditions, our algorithm can always achieve $\min_{0 \leq t \leq T} \mathbb{E} \left[\left\| \nabla H^{\text{meta}}(\Theta^{(t)}) \right\|_2^2 \right] \leq \mathcal{O} \left(\frac{1}{\sqrt{T}} \right)$ in T steps. □

Author Contributions Conceptualization: W-Q; Methodology: W-Q; Theoretical analysis: F-L; Writing-original draft preparation: W-Q, S-HL; Writing-review and editing: W-R, H-RD; Funding acquisition: S-HL, F-L, Y-YL.

Funding This research was supported by Natural Science Foundation of China(No. 62106129, 62176139, 62106028), Natural Science Foundation of Shandong Province (No. ZR2021QF053, ZR2021ZD15) and Chongqing Overseas Chinese Entrepreneurship and Innovation Support Program, and CAAI-Huawei MindSpore Open Fund.

Availability of data and materials Not applicable.

Declarations

Conflict of interest The author declares that he has no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Code availability Not applicable.

References


- Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., & De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. in *NIPS* **29**
- Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009) Curriculum learning. In: *ICML*, pp. 41–48
- Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., & Raffel, C. (2019) Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. arXiv preprint [arXiv:1911.09785](https://arxiv.org/abs/1911.09785)
- Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C.A. (2019) Mixmatch: A holistic approach to semi-supervised learning. in *NIPS* **32**
- Coates, A., Ng, A., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In: *AISTATS*, pp. 215–223
- Cubuk, E.D., Zoph, B., Shlens, J., & Le, Q.V. (2020) Randaugment: Practical automated data augmentation with a reduced search space. In: *CVPRW*, pp. 702–703.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In: *CVPR*, pp. 248–255 IEEE
- Finn, C., Abbeel, P., & Levine, S. (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: *ICLR (PMLR)*, pp. 1126–1135.
- Franceschi, L., Frascioni, P., Salzo, S., Grazi, R., & Pontil, M. (2018) Bilevel programming for hyperparameter optimization and meta-learning. In: *ICML*
- Grefenstette, E., Amos, B., Yarats, D., Htut, P.M., Molchanov, A., Meier, F., Kiela, D., Cho, K., & Chintala, S. (2019) Generalized inner loop meta-learning. arXiv preprint [arXiv:1910.01727](https://arxiv.org/abs/1910.01727)
- Guo, L.-Z., & Li, Y.-F. (2022) Class-imbalanced semi-supervised learning with adaptive thresholding. In: *ICLR*, pp. 8082–8094
- Guo, L.-Z., Zhang, Z.-Y., Jiang, Y., Li, Y.-F., & Zhou, Z.-H. (2020) Safe deep semi-supervised learning for unseen-class unlabeled data, in: *ICLR*
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.
- Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., & Kalantidis, Y. (2019) Decoupling representation and classifier for long-tailed recognition. arXiv preprint [arXiv:1910.09217](https://arxiv.org/abs/1910.09217)
- Kim, J., Hur, Y., Park, S., Yang, E., Hwang, S. J., & Shin, J. (2020). Distribution aligning refinery of pseudo-label for imbalanced semi-supervised learning. *NIPS*, *33*, 14567–14579.

- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images
- Laine, S., & Aila, T. (2016) Temporal ensembling for semi-supervised learning. arXiv preprint [arXiv:1610.02242](https://arxiv.org/abs/1610.02242)
- Lee, D.-H. (2013) Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: ICML Workshop. vol 3, p. 896
- Li, J., Xiong, C., & Hoi, S.C. (2021) Comatch: Semi-supervised learning with contrastive graph regularization. In: ICCV, pp. 9475–9484
- Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., & Potts, C. (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pp. 142–150
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A.Y. (2011) Reading digits in natural images with unsupervised feature learning
- Ren, M., Zeng, W., Yang, B., & Urtasun, R. (2018) Learning to reweight examples for robust deep learning. In: ICML, pp 4334–4343
- Saito, K., Kim, D., & Saenko, K. (2021) Openmatch: Open-set consistency regularization for semi-supervised learning with outliers. in NIPS
- Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016) Regularization with stochastic transformations and perturbations for deep semi-supervised learning. in NIPS **29**
- Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., & Meng, D. (2019). Meta-weight-net: Learning an explicit mapping for sample weighting. In *NIPS*, 32, 19.
- Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., & Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NIPS*, 33, 596–608.
- Sun, H., Guo, C., Wei, Q., Han, Z., & Yin, Y. (2022). Learning to rectify for robust learning with noisy labels. *Pattern Recognition*, 124, 108467.
- Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 30, 17.
- Wang, Y., Guo, J., Song, S., & Huang, G. (2020). Meta-semi: A meta-learning approach for semi-supervised learning. arXiv preprint [arXiv:2007.02394](https://arxiv.org/abs/2007.02394)
- Wei, C., Sohn, K., Mellina, C., Yuille, A., & Yang, F. (2021) Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In: CVPR, pp. 10857–10866
- Xiao, T., Zhang, X.-Y., Jia, H., Cheng, M.-M., & Yang, M.-H. (2021). Semi-supervised learning with meta-gradient. In: International Conference on Artificial Intelligence and Statistics, pp. 73–81
- Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. *NIPS*, 33, 6256–6268.
- Xu, Y., Shang, L., Ye, J., Qian, Q., Li, Y.-F., Sun, B., Li, H., & Jin, R. (2021). Dash: Semi-supervised learning with dynamic thresholding. In: ICLR, pp. 11525–11536
- Xu, Y., Zhu, L., Jiang, L., & Yang, Y. (2021) Faster meta update strategy for noise-robust deep learning. In: CVPR, pp. 144–153
- Zhang, X., Zhao, J., & LeCun, Y. (2015) Character-level convolutional networks for text classification. in NIPS **28**
- Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., & Shinozaki, T. (2021). Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *NIPS*, 34, 18408–18419.
- Zheng, M., You, S., Huang, L., Wang, F., Qian, C., & Xu, C. (2022) Simmatch: Semi-supervised learning with similarity matching. In: CVPR, pp. 14471–14481
- Zhu, X., & Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1), 1–130.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Qi Wei^{1,2}  · Lei Feng² · Haoliang Sun¹ · Ren Wang¹ · Rundong He¹ · Yilong Yin¹

✉ Haoliang Sun
haolsun@sdu.edu.cn

✉ Yilong Yin
ylyin@sdu.edu.cn

Qi Wei
1998v7@gmail.com

Lei Feng
lfengqaq@gmail.com

Ren Wang
xxlifelover@gmail.com

Rundong He
rundong_he@mail.sdu.edu.cn

¹ School of Software, Shandong University, Jinan, China

² School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore