Learning Sample-Aware Threshold for Semi-Supervised Learning

Qi Wei, Lei Feng, Haoliang Sun*, Ren Wang, Rundong He, Yilong Yin*

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 (Meta-T), 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 demonstrate the effectiveness of our methods on both typical and imbalanced SSL tasks.

Index Terms—Semi-supervised learning, Confidence thresholds, Meta-learning, Bi-level

1 Introduction

Machine learning, especially deep learning, can achieve competitive performance against human beings on certain supervised learning tasks [1]. However, the performance heavily relies on the availability of a great amount of labeled training data. In many real-world applications, large-scale datasets with precise annotations are hard to obtain, as the acquisition of labeled data requires huge human labor and financial costs [2]. Meanwhile, acquiring abundant unlabeled data is cheap and effortless, which inspires researchers to leverage unlabeled data for enhancing model performance.

Semi-supervised learning (SSL) [3]–[9] 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 [8]. Most of the previous SSL studies heavily rely on the *pseudo-labeling* strategy [10], [11] that generates a hard label for unlabeled sample and trains the deep model on these pseudo-labels.

For pseudo-labeling methods [10]–[13], it is essential to set a proper threshold for selecting reliable pseudo-labels for unlabeled data. For example, FixMatch [11] selected high-confidence pseudo-labels via a fixed threshold (*e.g.*, 0.95 for CIFAR [14] and 0.65 for ImageNet [15]). However, as reported in Xu *et al.* [12], 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.

Qi Wei, Haoliang Sun, Ren Wang, Rundong He and Yilong Yin are with the School of Software, Shandong University, Jinan, China. (e-mail: {1998v7, xxlifelover}@gmail.com, {rundong_he}@mail.sdu.edu.cn, {ylyin, haolsun}@sdu.edu.cn).
Lei Feng is now with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. (e-mail: {feng0093@e.ntu.edu.sg}).
**Corresponding authors.*

To address this issue, subsequent works [12], [13], [16], [17] that dynamically generate the threshold to enable more robust SSL have been proposed. For instance, Xu et al. [12] 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. [13] leveraged the idea of curriculum learning [18] 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 [12], [13], [16] 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 [12], [13], [16] usually involve at least two hyper-parameters, which requires complex cross-validation phase and thus suffer from the scalability issue [19] 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 1 (a) 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.

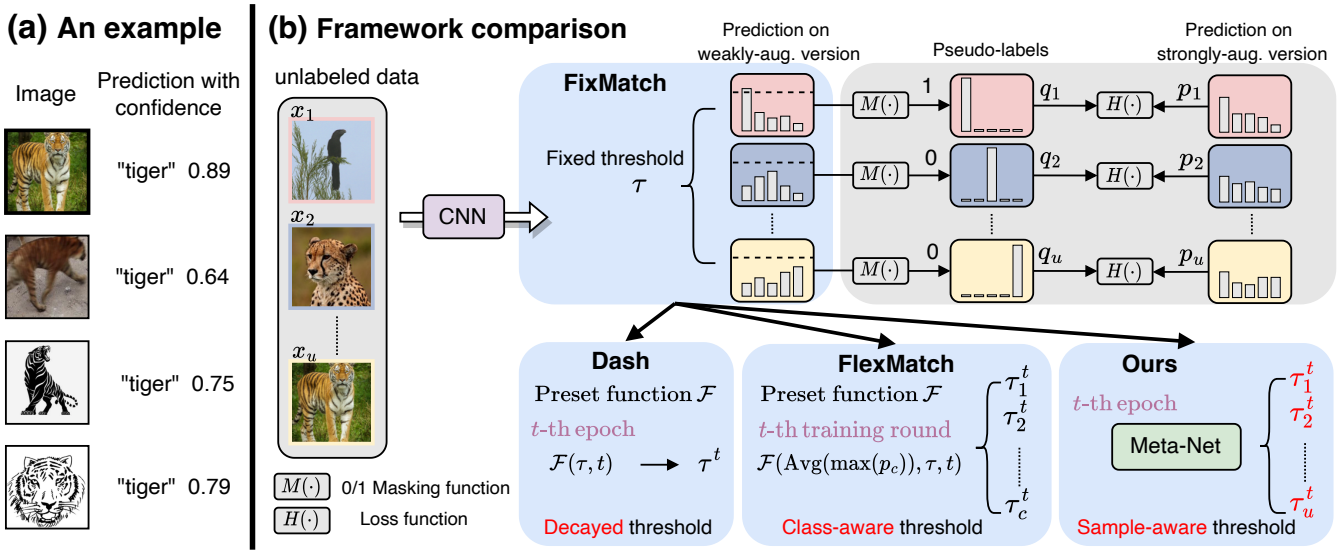


Fig. 1: **(a)** An example illustrates that deep models have different learning capabilities for different examples in class *tiger*. **(b)** Review of the pseudo-labeling training framework and comparison FixMatch [11] with three improved algorithms on the fixed confidence threshold. Compared to decayed and class-level thresholds in Dash [12] and FlexMatch [13], our method designs a meta-net which generates a more refined confidence threshold for each unlabeled example (*i.e.*, sample-level thresholds).

Specifically, we leverage the idea of meta-learning [20] 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 [21] of multilayer feedforward networks, our meta-net can be considered as a generalized version of the pre-defined functions mentioned above [12], [13]. 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 below:

- **Inner loop**. Generate instance-level thresholds for all unlabeled instances and utilizes the hard pseudo labels to train the classifier network.
- **Outer loop**. Update all parameters of the meta-net by a small scale of meta-data which are constructed on the given 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 a surrogate function to approximate the indication function, making the bi-level optimization problem reachable. In Figure 1 (b), we compare our method with vanilla FixMatch [11] and two improved methods [12], [13], 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:
  - Meta-T learns thresholds of unlabeled sample automatically through bi-level optimization, avoiding the the pathology of conventional threshold-based methods' reliance on strong prior knowledge on data.
  - 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 $\mathcal{O}(1/\epsilon^2)$, which is identical to that of the stochastic gradient descent (SGD).
- 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.

Extensive experiments on five image classification benchmarks and three text classification benchmarks demonstrate the effectiveness of our proposed Meta-T.

The rest of this paper is organized as follows. Section 2 introduces related works and discusses the relations to our work. Section 3 introduces the problem settings and the normal learning framework of pseudo-labeling methods. Section 4 first introduces technical details of the proposed methods for learning sample-aware thresholds. Then, it theoretically demonstrates the convergence rate of our learning algorithm and provides a green version of our method to reduce its training complexity. Section 5 reports experimental results on various datasets. Finally, Section 6 gives a conclusion and future outlook of this paper.

2 Related Work

In this section, we briefly review existing works on the following three related fields and highlight the novelty of our method compared with current methods.

2.1 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 [22]–[25]. Pseudo-labeling methods, which are also called self-learning in previous works, belong to an iterative mechanism [8] 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 [10]. Hybrid approaches [11]–[13] always integrate the above two methods with strong augmentation strategies (e.g., RandAugment [26] and CTAugment [27]). However, typical SSL methods require a strong assumption that labeled data and unlabeled data are sampled from the same data distribution. Compared to the performance of supervised learning methods trained with only labeled data, the worse performance of SSL algorithms might be obtained if this assumption cannot be satisfied [8], [28], encouraging more works to study more challenging data distribution, such as imbalanced SSL.

2.2 Imbalanced Semi-Supervised Learning

To improve the universality of SSL algorithms, some works [16], [29], [30] turn attention to more challenging settings like SSL under *class-imbalanced* label distribution. DARP [29] 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 [30] is based on a typical self-training strategy that adaptively adds pseudo-labeled data to the training set according to the label frequency. For example, the minority classes would be selected with a higher probability. Adsh [16] modified the fixed confidence threshold in FixMatch to be class-dependent and adaptive to class distributions.

2.3 Meta-Learning

Meta-Learning, also known as "learning to learn", aims to rapidly adapt to new tasks and has been widely applied to several weakly-supervised tasks, such as noisy labels learning [31], [32], out-of-distribution learning [33], and semi-supervised learning [34], [35]. In SSL fields, some works introduce the idea of meta-learning to learn a set of parameters. For example, Wang *et al.* [34] proposed a framework to learn sample weights for all unlabeled data, which aims to give high weights to more reliable pseudo-labels. [35] proposed to learn soft labels for unlabeled data while designing a one-order update strategy for bi-level training framework.

Relations to ours, Two previous works L2R [36] and MW-Net [31] employed bi-level optimization to efficiently learn a set of hyper-parameter. Our work bears three critical differences, in terms of the problem setting, methodology and theory.

(1) *Problem setting*: [31], [36] 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*: [31], [36] 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 underlying the indicator function. For this, we propose a surrogate function to approximate the indicator function. Eventually, we joint the bi-level training framework with curriculum learning (called Green Meta-T), which significantly reduces the training cost.
(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. When we conduct multi-steps of inner loop in curriculum learning, the training complexity of Green Meta-T can be reduced to single-step update, which is remarkably lower than that of [31], [36].

3 Preliminaries

Problem setting. In a C-class classification task, we have a set of training data which contains N labeled examples $D^l = \{(\mathbf{x}_1^l, \mathbf{y}_1^l), \cdots, (\mathbf{x}_N^l, \mathbf{y}_N^l)\}$ and M unlabeled examples $D^u = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, 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 [11] 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

$$p_m = f(\mathcal{A}^w(\mathbf{x}_m); \mathbf{w}), \tag{1}$$

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. Secondly, given a strong augmented version of $\mathbf{x}_m$, the training objective can be summarised as

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

where $\mathbb{1}(\cdot)$ is an indicator function and denotes the selection of high-reliability of pseudo-label 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 discussed in the Introduction phase, many related works modified the fixed constant τ to improve the universality of pseudo-labeling algorithms. However, they [12], [13] 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. To avoid dependence of prior knowledge as well as fulfilling instance-level confidence thresholds, we propose a meta-net called threshold generation network (TGN). Based on this, we first rewrite the learning objective for threshold-based SSL methods and then present the design of TGN in Sec. 4.2. In Sec. 4.3, we solve this meta-optimization problem via bi-level training strategy which alternatively trains classifier network and TGN. Meanwhile, we provide the convergence analysis of our proposed algorithm in Sec. 4.4. Eventually, in Sec. 4.5, we analyze the training cost of our method and design a corresponding green version 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 Sec. 4.2. Then, the fixed constant τ in Eq. (2) can be replaced with a sample-level threshold τ_m and the loss of unlabeled data $\mathbf{x}_m$ is formulated as

$$\ell_{\mathbf{x}_m}(\mathbf{w}, \Theta) = \mathbb{1}(\max(p_m) > \mathcal{V}_m(\mathbf{w}, \Theta)) \cdot H(\hat{\mathbf{y}}_m, f(\mathcal{A}^s(\mathbf{x}_m); \mathbf{w})). \tag{3}$$

However, due to the non-differentiable property of the indicator function $\mathbb{1}(\cdot)$, computing partial derivative with respect to Θ in Eq. (3) 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 $\max(f(\mathcal{A}^w(\mathbf{x}_m); \mathbf{w})) - \mathcal{V}_m(\mathbf{w}, \Theta)$.

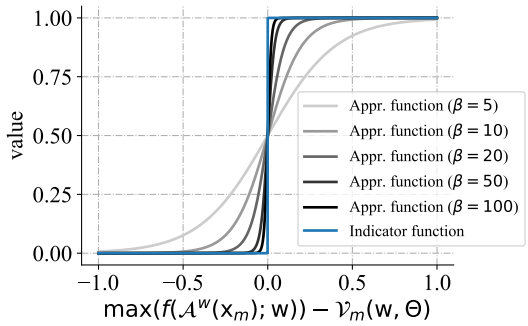


Fig. 2: Comparison results between the indicator function and the approximate function $\mathcal{S}(\cdot)$. In all experiments, we set the coefficient β in $\mathcal{S}(\cdot)$ as 100.

Discuss about the approximate function $\mathcal{S}(\cdot)$. In Figure 2, we compare the difference between the indicator function $\mathbb{1}(\cdot)$ and the suggorate function $\mathcal{S}(\cdot)$. We can observe that the input of function satisfies $\max(f(\mathcal{A}^w(\mathrm{x}_m); \mathbf{w}) - \mathcal{V}_m(\mathbf{w}, \Theta)) \in [-1, 1]$. Meanwhile, the first-order and second-order gradient of sigmoid function obviously exist, which makes backpropagation of the training loss in Eq. (3) possible.

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

$$\mathbf{w}^*(\Theta) = \arg\min_{\mathbf{w}} L_u = \frac{1}{M}\sum\nolimits_{\mathbf{x}_m \in D^u} \ell_{\mathbf{x}_m}(\mathbf{w}, \Theta). \tag{4}$$

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 [13], [16], 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.

Formally, given a weakly-augmented version of unlabeled data $\mathbf{x}_m$, the classifier network $f_{\mathbf{w}}$ 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 $\hat{\mathbf{y}}_m^t = \arg\max(g(p_m^t))$. Meanwhile, the average class confidence can be represented as

$$\overline{\mathrm{p}}_c^t = \mathrm{avg}(\sum\nolimits_{m=1}^{M} g(p_m^t | c = \hat{\mathbf{y}}_m^t)). \tag{5}$$

Note that $\overline{\mathrm{p}}_c^t$ 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

$$\tau_m^t = \mathcal{V}(g(f(\mathrm{x}_m; \mathbf{w})), \overline{\mathrm{p}}_c^t; \Theta). \tag{6}$$

As shown in Figure 3 (b), 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 $\overline{\mathrm{p}}_c^t$ (a C-dimension vector). Therefore, the input layer in TGN is $2C$ dimension.

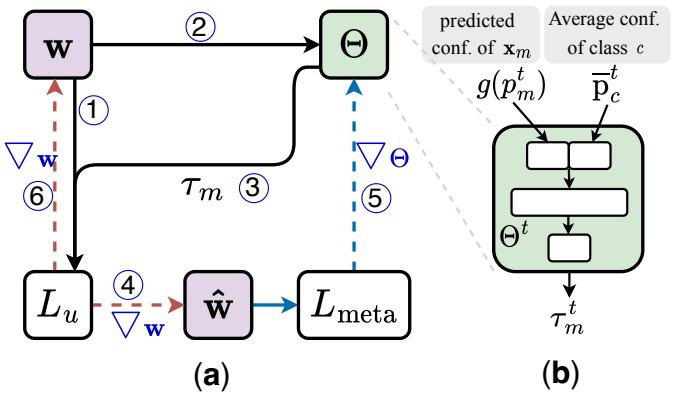


Fig. 3: (a) **Flowchart of our learning algorithm**. The solid and dashed lines represent forward and backward propagation, respectively. In each iteration, overall training process contains six phases. *Step 1*: feed weak-augmented images to the classifier network and attain pseudo-labels with prediction 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, including the predicted confidence $g(p_m^t)$ and $\overline{\mathrm{p}}_c^t$, the average confidence of class c.

Remark. (1. Training costs caused by TGN) In practice, the computation of average class confidence $\overline{\mathrm{p}}_c^t$ is conducted after each training epoch, which does not incur high computation cost. (2. Training efficiency of TGN) The performance of TGN relies on the scale of meta-data. However, the learnable parameters of TGN are slight. For example, the architecture of TGN for CIFAR-10 is $\{h_1, h_2, h_3\} = \{20, h_2, 1\}$ and the learnable parameters are $21 \times h_2$. We experimentally demonstrate that TGN can achieve remarkable performance with only 40 labeled meta-data.

4.3 Meta-Optimization Problem

There are two networks in our training framework, including a classification network $f_{\mathbf{w}}$ and a meta-net $\mathcal{V}_\Theta$. The parameters $\mathbf{w}$ and Θ can be optimized by the meta-learning idea [31], [37], [38]. Specifically, we require a small amount of meta-data set which can be sampled from labeled data in SSL task. Since some works [31], [32] 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

$$\Theta^* = \arg\min_{\Theta} L_{\text{meta}}(\mathbf{w}^*(\Theta)) = \frac{1}{N}\sum_{i=1}^{N} H_i(\mathbf{w}^*(\Theta)), \quad (7)$$

where $H_i(\mathbf{w}) = H(\mathbf{y}_i^l, f(\mathbf{x}_i^l; \mathbf{w}))$ denotes the cross-entropy loss computed on i-th meta-data.

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

Require: Unlabeled data D^u, meta (limited labeled) data D^l, batch size n, a scaling 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 $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), ..., (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$ from D^l.
4: Random sample $\{\mathbf{x}_1, ..., \mathbf{x}_{(\mu\times n)}\}$ from D^u.
5: Formulate the classifier network learning function $\hat{\mathbf{w}}^{(t)}(\Theta)$. ▷ Eq. (8)
6: Update $\Theta^{(t+1)}$. ▷ Eq. (9)
7: Update $\mathbf{w}^{(t+1)}$. ▷ Eq. (10)
8: **end for**

Obtaining the optimal parameters $\mathbf{w}^*$ in Eq. (4) and Θ^* in Eq. (7) is a nested optimization problem. For this, we resort to bi-level training strategy as MAML [20] and update parameters of meta-net with online strategy. To be specific, the training loss of classifier network and meta-net (Eq. (4) and Eq. (7)) 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 $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), ..., (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$ and $\{\mathbf{x}_1, ..., \mathbf{x}_{(\mu\times n)}\}$, respectively. Note that since the performance of pseudo-labeling methods depends on a large batch size of unlabeled data, we obey this convention and introduce a parameter μ to control the batch size of unlabeled data. 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 these two networks can be summarized as the following three phases.

- **Formulating learning manner of classifier network**. Given the learning step with a size of α, the descent direction of the objective loss in Eq. (4) on a mini-batch unlabeled data is

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

where $\ell_{\mathbf{x}_i}$ is calculated by Eq. (3).
- **Updating parameters Θ of TGN**. As we obtain parameter $\hat{\mathbf{w}}^{(t)}(\Theta)$ with fixed Θ in Eq. (8), the update of our meta-net TGN can be achieved by a mini-batch of meta-data $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), ..., (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$. Specifically, $\Theta^{(t)}$ moves along the direction of direction of gradients *w.r.t.* the objective in Eq. (7)

$$\Theta^{(t+1)} = \Theta^{(t)} - \psi\frac{1}{n}\sum_{i=1}^{n}\nabla_{\Theta}H_i(\hat{\mathbf{w}}^{(t)}(\Theta)), \quad (9)$$

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

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \alpha\frac{1}{n\mu}\sum_{i=1}^{n\mu}\nabla_{\mathbf{w}}\ell_{\mathbf{x}_i}(\mathbf{w}^{(t)}, \Theta^{(t+1)}). \quad (10)$$

We illustrate the flowchart of our learning algorithm in Figure 3 (a), where *Step 4,5,6* represent Eq. (8), Eq. (9) and Eq. (10), respectively. Meanwhile, we summarize the overall updating steps in Algorithm 1. Compared to current SSL methods, the significant merit of our framework is that we do 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 that we cannot model the data distribution. Therefore, Meta-T has grater universality in real-world scenarios.

4.4 Convergence Analysis

We analyze the convergence of our training framework in Algorithm 1 and give a rigorously theoretical guarantee. Since the meta-net is vital in our framework, we prove that $\mathcal{V}(\Theta)$ can converge to the stationary point of the meta loss function under some mild conditions.

Lemma 1. *(Smoothness). Suppose the loss function H is L-Lipschitz and smooth, and the approximate function $\mathcal{S}$ 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 $\mathcal{S}$, the gradient of Θ w.r.t. the meta loss is Lipschitz continuous.*

Proof. See Appendix B.3. □

Lemma 1 implies that the meta loss *w.r.t.* the meta-network is smooth-bounded. We provide the convergence rate in Theorem 2 with the support of this essential property.

Theorem 2. *(Convergence rate) Based on the conditions in 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 β_t, $1 \leq t \leq T$ is a monotone descent sequence, $\beta_t = \min\{\frac{1}{L}, \frac{\mathrm{c}}{\sigma\sqrt{T}}\}$ for some $\mathrm{c} > 0$, such that $\frac{\sigma\sqrt{T}}{\mathrm{c}} \geq L$ and $\sum_{t=1}^{\infty} \beta_t \leq \infty$, $\sum_{t=1}^{\infty} \beta_t^2 \leq \infty$. Then we have*

$$\frac{1}{T}\sum\nolimits_{t=1}^{T} \mathbb{E}\left[\left\| \nabla L_{\mathrm{meta}}\left(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})\right) \right\|_2^2 \right] \leq \mathcal{O}(\frac{1}{\sqrt{T}}). \quad (11)$$

Proof. See Appendix B.4. □

To be specific, Theorem 2 means that the our algorithm can achieve $\mathbb{E}\left[\left\| \nabla L_{\mathrm{meta}}\left(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})\right) \right\|_2^2 \right] \leq \epsilon$ in $\mathcal{O}(1/\epsilon^2)$ steps, and our algorithm 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 to the traditional 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.* [39], the main cost of training time is caused by the backpropagation in updating the parameters Θ of the meta-net since the meta-gradient in Eq. (9) 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. Algorithm 2 shows the new training procedure. As follows, we give the training complexity of Green Meta-T.

Algorithm 2 Learning algorithm of **Green Meta-T**.

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

```
Initialize w^(0) for classifier network and Θ^(0) for TGN.
for t = 0 to T − 1 do
    Random sample {x_1, ..., x_(μ×n)} from D^u.
    if t % k = 0 then
        /* conduct Meta-T for classifier and TGN learning */
        Random sample {(x_1^l, y_1^l), ..., (x_n^l, y_n^l)} from D^l.
        Formulate the classifier network learning function
        ŵ^(t)(Θ).                                   ▷ Eq. (8)
        Update Θ^(t+1).                             ▷ Eq. (9)
        Update w^(t+1).                             ▷ Eq. (10)
    else
        /* only conduct classifier learning */
        Update w^(t+1).                             ▷ Eq. (10)
    end if
end for
```

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

Proposition 3 holds that the training complexity of Green Meta-T could gradually reduce to $\mathcal{T}$ as the value of k increases. However, reducing the training phases of TGN without limit would degrade the performance of TGN and further degenerate the quality of pseudo-labels. Therefore, selecting a suitable k is essential and require cross-validation. In Section 5.4, we experimentally analyze the sensitivity of hyper-parameter k.

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

- **CIFAR-10** [14], which consists of 60,000 images of 10 categories. We adopt the splitting strategy in FixMatch by randomly selecting {40, 250, 4000} samples from the training set to construct the labeled set and removing labels of reset samples to build the unlabeled set.
- **CIFAR-100** [14], which is more challenging than CIFAR-10 including 100 classes belonging to 20 superclasses where each category contains 600 images with the resolution of 32 × 32. Similar splitting manners as CIFAR-10 are employed and the number of labeled data belongs to {400, 2500, 10000}.

TABLE 1: Error rates (%) for previous state-of-the-art methods on CIFAR-10 and CIFAR-100 with varying size of labeled set. **Bold** indicates the best result and underline indicates the second-best result.

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**	37.81±2.21	**25.07±0.32**	**20.58±0.11**
threshold-based						
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	4.78±0.12	4.13±0.06	44.83±1.36	27.18±0.21	21.97±0.14
FlexMatch	4.97±0.06	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**	4.01±0.09	**36.17±1.40**	25.81±0.72	20.74±0.23
Full-Supervised		4.62±0.05			19.30±0.09	

- **SVHN** [40], which is collected from the street view house number and consists of 73527 digits for training and 26032 digits for testing. Its resolution is 32 × 32. We construct the labeled set on the training set with varying scale $\{40, 250\}$.
- **SLT-10** [41], which is a large-scale dataset and specifically designed for semi-supervised learning task. This dataset contains 10 classes with 500 labeled samples per class and 100,000 unlabeled images totally. The images in SLT-10 belongs to higher resolution, i.e., 96 x 96.
- **ImageNet** [15], which is the most popular test benchmark in image classification field. It contains 1000 categories with 100,0000 training sample in total. We only use a small-scale samples for constructing the labeled set and the ratio belongs to $\{1\%, 10\%\}$.
- **IMDb** [42], which contains 50,000 samples with two classes. We randomly sample 1000 per-class for constructing the labeled data and the reset samples are used as the unlaebeled data.
- **Amazon-5** [43], we randomly sample 5,000 data and 50,000 data per-class as validation set and training set respectively.
- **Yelp-5** [43], we keep the same setting as that in Amazon-5 and randomly sample 5,000 data and 50,000 data per-class as validation set and training set, respectively.

Implementation Details. Our code is implemented by Pytorch 1.9.0 with GTX 3090. We leverage a pytorch library called **Higher**[1] to implement our algorithm, which provides support for higher-order optimization. For all experiments, we repeat five times with different random seeds. Other settings for two networks are shown below

- For **the classifier**, more information about data preprocessing and training procedure can be found in Appendix A.
- 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.

1. https://github.com/facebookresearch/higher

5.2 Results on Typical SSL Task

To demonstrate the effectiveness of our method, we first conduct experiments on typical SSL tasks where the data distribution of both labeled and unlabeled data is balanced.

Comparison methods. We roughly categorize compared methods into two types. 1) Threshold-based methods, including Pseudo-Labeling (PL) [10], FixMatch [11], FlexMatch [13] and Dash [12]. 2) others, including Π-Model [22], MixMatch [44], UDA [25], CoMatch [45] and SimMatch [46]. Note that both CoMatch and SimMatch leverage representation learning such as contrastive learning technique and thus achieve greater performance.

Results on four image datasets. We conduct experiments on CIFAR-10, CIFAR-100, SVHN, and SLT-10. The results are shown in Table 1 and Table 2 (top). On CIFAR-10 & 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. Meanwhile, our method slightly underperforms a previous state-of-the-art method SimMatch with abundant unlabeled data. Under three settings, the average accuracy gap is smaller than 0.3%.

Compared with threshold-based methods (Pseudo-label, FixMatch, Dash, and FlexMatch), the improvement of our method is significant. On all tested benchmarks, Meta-T constantly outperforms them. For example, compared to Dash, our method achieves top-1 accuracy improvement of 5.21% on CIFAR-10 with 40 labeled data and 8.66% on CIFAR-100 with 400 labeled data.

Results on ImageNet. We select ImageNet to validate the effectiveness of Meta-T on real-world applications. For a fair comparison, we do not introduce and load pre-trained parameters for the classifier network. We report the results that are achieved by training the methods on only 1% & 10% labeled data and extensive unlabeled data. As shown in Table 2 (medium), our method outperforms previous methods in two settings. 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. We believe

TABLE 2: (**Top**) Error rates (%) for previous state-of-the-art methods on SVHN and STL-10 with varying size of labeled set. (**Medium**) Error rates (%) for previous state-of-the-art methods on three text datasets. (**Bottom**) Top-1 and Top-5 accuracy (%) on ImageNet test set with varying ratio of labeled samples. **Bold** indicates the best result and underline indicates the second-best result.

Error rates (%) ↓			
	SVHN		STL-10
Methods	40 labels	250 labels	1000 labels
Π-Model	-	18.96±1.92	26.23±0.82
VAT	74.75±3.38	4.33±0.12	37.95±1.12
MixMatch	42.55±14.53	3.98±0.23	10.41±0.61
UDA	52.63±20.51	5.69±2.76	7.66±0.56
ReMixMatch	3.34±0.20	2.92±0.48	5.23±0.45
PL	-	20.21±1.09	27.99±0.83
FixMatch	3.14±1.60	2.64±0.64	5.17±0.63
Dash	3.03±1.59	**2.17±0.10**	3.96±0.25
FlexMatch	8.19±3.20	-	5.77±0.18
Meta-T (ours)	**2.89±0.92**	2.29±0.51	**3.51±0.34**

Top-1 / Top-5 accuracy (%) ↑			
	ImageNet		
	1%	10%	100%
Sup. baseline	25.4 / 48.4	56.4 / 80.4	
FixMatch	53.4 / 74.4	70.8 / 89.0	
CoMatch	66.0 / 86.4	73.6 / 91.6	80.4 / 94.6
SimMatch	67.2 / 87.1	74.4 / 91.6	
Meta-T (ours)	**67.7 / 87.9**	**75.0 / 91.7**	

Error rates (%) ↓			
	IMDb	Amazon-5	Yelp-5
UAD	18.33±0.61	50.29±4.6	47.49±6.83
FixMatch	7.59±0.28	42.70±0.53	39.56±0.70
FlexMatch	7.80±0.23	42.34±0.62	39.01±0.17
SoftMatch	7.48±0.12	**42.14±0.92**	39.31±0.45
Meta-T(ours)	**7.20±0.20**	42.60±0.41	**38.44±0.37**

the superiority of our method on ImageNet can already demonstrate its effectiveness on real-world SSL tasks.

Results on three text datasets. We also test the performance of Meta-T on text classification tasks. 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 evaluate Meta-T with compared methods on the CIFAR-10 dataset under various levels of imbalanced ratio and different numbers of labeled examples.

Comparison methods. We roughly categorized compared methods into two parts. 1) Threshold-based methods, FixMatch [11], Dash [12] and FlexMatch [13]. 2) Others, cRT [47], LDAM, MixMatch [44], ReMixMatch [27], DARP [29], CReST [30] and Adsh [16].

Synthetic imbalanced datasets. Referring to [16], 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., $\gamma_l = \frac{N_1}{N_C}, \gamma_2 = \frac{M_1}{M_C}$. 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 $\gamma_1, \gamma_2 \in [50, 100, 150]$.

Results on imbalanced CIFAR-10. In Table 3, 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

Meta-T can reduce the error rate of produced pseudo-labels. We plot two types of pseudo-labels in the selected set in Figure 4. Due to the high training complexity of the bi-level strategy, we keep a similar training time as FixMatch, and we set the training epoch as 300. Besides, we also keep the same training strategy as FixMatch / FlexMatch for a fair comparison.

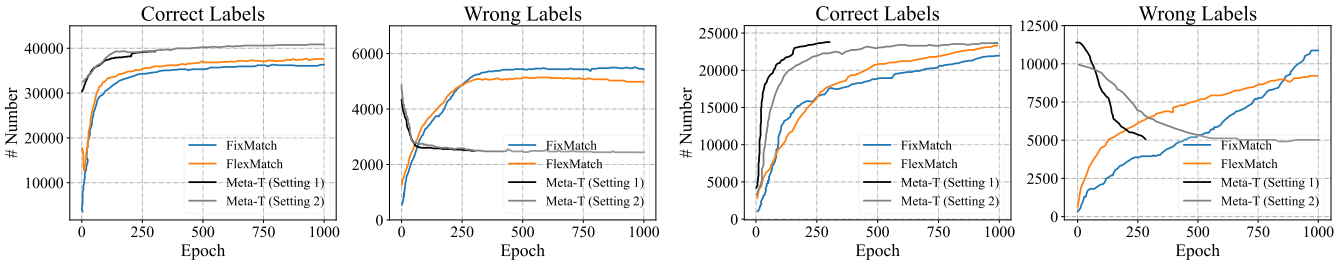
In Figure 4 (a,b) *left*, we exhibit results of the number of correct labels. Meta-T shows greater performance in generating correct pseudo-labels, which benefits from the higher quality of thresholds produced by TGN. In the early learning stage, the number of correct labels in our method is remarkably higher than that in FixMatch, reflecting the superiority of sample-level thresholds. In Figure 4 (a,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 FixMatch and FlexMatch. Fortunately, the number of wrong labels decrease with the learning process and is lastly lower than that of FixMatch.

Meta-T can generate robust sample-level thresholds. To show the competence of our method in generating competitive thresholds, we visualize the learned thresholds from the following three perspectives.

- **Generated sample-level thresholds**. Figure 6 (a) 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.
- **Robust thresholds under imbalanced SSL**. Figure 6 (b) visualize the produced thresholds and test accuracy (%)

TABLE 3: Top-1 test accuracy (%) on imbalanced CIFAR-10 under three imbalanced ratio: $\gamma \in \{50, 100, 150\}$ and two differenet size of labeled set: $N_1 = 1500, M_1 = 3000$ and $N_1 = 500, M_1 = 4000$. The backbone is Wide ResNet-28-2. **Bold** indicates the best result and underline indicates the second-best result.

Methods	$N_1 = 1500, M_1 = 3000$			$N_1 = 500, M_1 = 4000$		
	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$	$\gamma = 50$	$\gamma = 100$	$\gamma = 150$
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
threshold-based						
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	72.29±0.42	77.90±0.39	70.41±0.27	62.11±0.32
FlexMatch	82.86±0.25	75.47±0.41	70.62±0.30	78.69±0.50	71.80±0.29	62.85±0.39
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**



(a) CIFAR-10 w/ 250 labeled samples

(b) CIFAR-100 w/ 400 labeled samples

Fig. 4: Visualization of the curve of correct and error pseudo-labels in the selected set with varying training epochs. We adopt two training procedures, including **Setting 1**: keep the identical training time (FixMatch / FlexMatch: 1000 epochs, Ours: 300 epochs), **Setting 2**: keep the same training procedure as FixMatch (1000 epochs with cosine learning rate decay).

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. Besides, FlexMatch, a class-aware thresholds generation method, also learns class-discriminative thresholds. Compared with other threshold-based methods, our method achieves state-of-the-art performance on the biased SSL setting.

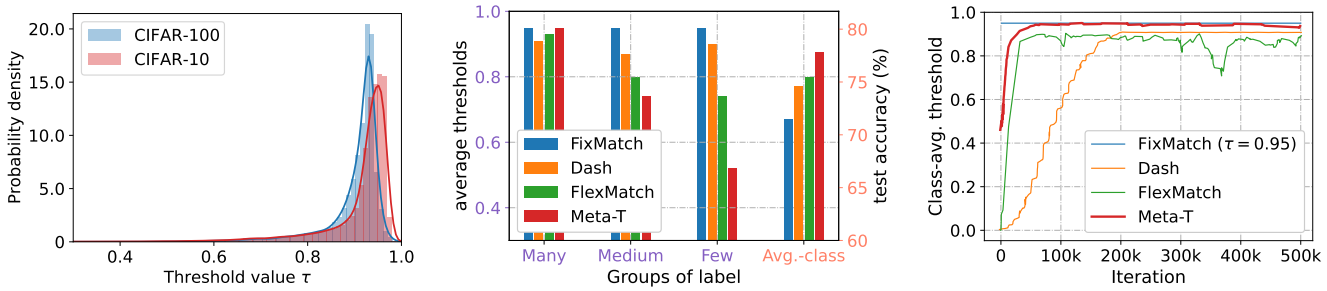
- **Dynamic learning process**. Figure 6 (c) shows the comparison results from dynamic threshold generation. In the beginning, Meta-T tends to initialize thresholds of all unlabeled data as 0.5 and then immediately grow up to 0.95, which is identical to the setting in FixMatch. This result demonstrates thresholds learned by Meta-T are close to the optimal thresholds. Besides, compared with FlexMatch, our method exhibits greater stability after achieving the highest thresholds (iteration: from 300k to 500k). Compared with Dash, Meta shows a faster convergence rate, while the higher threshold is beneficial to select high-reliability pseudo-labels and improves model learning.

Meta-T well considers tailed classes in imbalanced data. To show the effectiveness of our proposal on tackling imbalanced SSL tasks, we conduct experiments from the perspective of the confusion matrix on unlabeled data. Figure 6 (b) and (c) exhibit the comparison results between FixMatch and Meta-T. 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. For example, only 61% of the samples, which are labeled as class 9, are correct. However, Meta-T achieves significant results on tailed classes and consistently attains more than 80% accuracy on all classes, verifying the effectiveness of Meta-T on imbalanced SSL tasks.

5.5 Ablation Studies

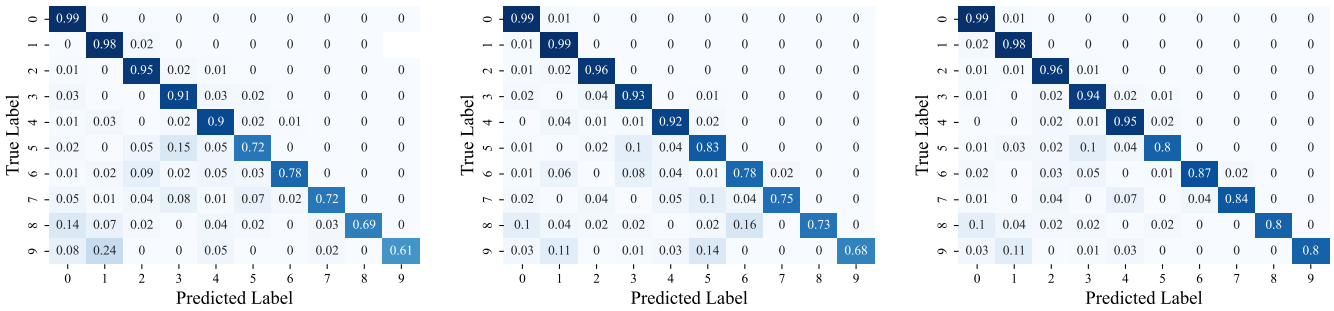
Although our learning framework does not contain massive hyper-parameters, we consider there still exist two factors that influence the performance of our method, including the **architecture of TGN** and the **training strategy of TGN**. In this subsection, we conduct experiments to evaluate them.

Firstly, 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 4 *left*. It can be seen that varying (five) MLP settings have unsubstantial effects on the final result. Therefore, we prefer to



(a) Generated sample-level thresholds (b) Threshold generation under biased SSL (c) Dynamic threshold generation

Fig. 5: 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 $N_1 = 1500, M_1 = 3000, \gamma_1 = \gamma_2 = 100$). Note that "Many", "Medium", and "Few" denote class index of "0-3", "4-6" and "7-9", respectively. (**c**) Visualization of class-average confidence threshold *v.s.* learning processes. We compare Meta-T with others under balanced SLT-10 w/ 40 labels.



(a) Confusion Matrix of *FixMatch* (b) Confusion Matrix of *FlexMatch* (c) Confusion Matrix of *Meta-T*

Fig. 6: From the perspective of the confusion matrix, we compare Meta-T with FixMatch and FlexMatch under CIFAR-10 with $\gamma = \gamma_l = \gamma_u = 100, N_1 = 1500, M_1 = 3000$.

TABLE 4: Ablation studies of different settings of the meta-net TGN.

Architecture of TGN

$\{2\mathcal{C}, h_1, ..., h_n, 1\}$	CIFAR-10 40 Labels	CIFAR-10 250 Labels	CIFAR-100 400 Labels	CIFAR-100 2500 Labels
$2\mathcal{C}$ - 10 - 1	4.39	4.26	**36.17**	25.81
$2\mathcal{C}$ - 100 - 1 (Ours)	**4.21**	**4.10**	36.98	26.27
$2\mathcal{C}$ - 1000 - 1	4.49	4.29	37.02	26.19
$2\mathcal{C}$ - 10 - 10 - 1	4.78	4.55	37.11	**25.42**
$2\mathcal{C}$ - 100 - 100 - 1	4.91	4.49	37.04	26.98

Training strategies of TGN

Learning strategy	CIFAR-10 250 labels	CIFAR-100 2500 labels
Adam w/ 1e-3 (Ours)	4.10	**25.81**
Adam w/ 5e-4	4.21	26.50
SGD w/ CosineAnnealingLR [1e-3, 1e-4]	**4.02**	26.14
SGD w/ MultiSteps [1e-3, 5e-4, 1e-4]	4.39	26.42
keep same as FlexMatch	4.17	26.03

adopt the simple yet effective one, i.e., $\{2\mathcal{C}, 100, 1\}$, 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.

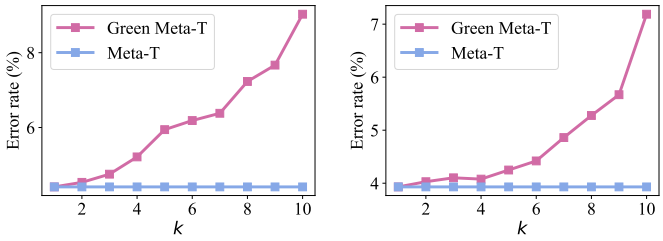
Secondly, compared with existing methods [11]–[13], our framework indeed introduces one hyper-parameter (i.e., the learning rate of meta-net β), which does not require complex cross-validation process. Experimentally, we conduct ablation studies and show results with different settings of optimization for TGN (the meta-net) in Table 4 *right*. We can conclude that our algorithm is insensitive to the hyperparameter β. Thus, we select a normal setting, *i.e.*, Adam optimizer with 1e-3 learning rate.

5.6 Analysis of Green Meta-T

We analyze the green version of the proposed Meta-T from two perspectives, including the hyperparameter selection and training time analysis.

Hyperparameter selection of k. The hyperparameter k in Green Meta-T framework decides the eventual training cost. Here, we make ablation studies on two SSL settings to select a suitable value of k, where $k \in \{1, 2, ..., 10\}$.

In Figure 7, 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 (in Algorithm 2 Line 12). Further, the performance of Green Meta-T would decrease since the lower quality of pseudo-labels generated by a poor performance TGN. (2) A relatively large k might not degrade the performance of Green Meta-T under a mild SSL setting (*i.e.*, moderate amount of labeled data). For example, we set $k = 4$ on CIFAR-10 with 4000 labeled data, and the error rate of the classifier slightly increased compared with $k = 0$ and $k = 1$. Therefore, we recommend a relatively large k under some mild SSL



(a) CIFAR-10 w/ 40 labels (b) CIFAR-10 w/ 4000 labels

Fig. 7: Sensitivity analysis of k in Green Meta-T.

TABLE 5: Comparison results with different training cost (hours). "Last" denote that we train the classifier until total convergence. We set $k = 2$ in Green Meta-T.

	1 h	5 h	10 h	20 h	Last
Error rate (%) ↓					
Setting 1: *CIFAR-10 w/ 250 labels*					
FixMatch	**7.95**	6.44	5.97	5.39	5.07
Meta-T	8.44	6.91	5.04	**4.31**	**4.10**
Green Meta-T	8.13	**6.16**	**4.61**	4.46	4.32
Setting 2: *CIFAR-100 w/ 400 labels*					
FixMatch	44.19	40.38	39.29	39.02	38.85
Meta-T	45.49	41.86	38.47	**36.29**	**36.17**
Green Meta-T	**42.91**	**39.63**	**38.14**	38.03	37.90
Top-1 accuracy (%) ↑					
Setting 3: *CIFAR-10* N_1=1500, M_1=3000, γ_1=γ_2=100					
FixMatch	59.49	65.71	69.42	71.18	71.50
Meta-T	57.50	68.19	73.92	**76.48**	**77.80**
Green Meta-T	**60.28**	**70.52**	**74.11**	75.08	75.49

conditions, which is able to keep the balance between the training cost and the model performance.

Training time. To detailedly analyze the training time of our framework Meta-T and its green version Green Meta-T, we conduct experiments on three SSL settings to show the algorithm performance on different training time.

The results are reported in Table 5. We can see that Green Meta-T shows faster convergence on the early learning compared with Meta-T. Since more classifier learning rounds performed in Green Meta-T in the same training time, the performance of Green Meta-T consistently outperform that of Meta-T under all settings before the first 10 hours of training. With the training time increases, the performance of Meta-T further improves and achieves SOTA results. Therefore, we can apply Green Meta-T on certain circumstance, which sacrifices some accuracy to speed up training.

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 preset 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 the effectiveness of our method.

Future work. In this paper, second-order gradient computation in a bi-level strategy causes the huge training time of our proposed Meta-T. For this, we integrate our learning algorithm Meta-T into the curriculum learning framework to decrease the training time. In future work, we consider that replacing the second-order gradient with a first-order approximation would significantly reduce the training cost.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[3] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[4] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[5] W. He and Z. Jiang, "Semi-supervised learning with the em algorithm: A comparative study between unstructured and structured prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2912–2920, 2020.

[6] D. Shi, L. Zhu, J. Li, Z. Cheng, and Z. Liu, "Binary label learning for semi-supervised feature selection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[7] Z. Zhang, M. Zhao, and T. W. Chow, "Graph based constrained semi-supervised learning framework via label propagation over adaptive neighborhood," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2362–2376, 2013.

[8] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[9] Z. Wang, L. Zhang, R. Wang, F. Nie, and X. Li, "Semi-supervised learning via bipartite graph construction with adaptive neighbors," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[10] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop*, vol. 3, no. 2, 2013, p. 896.

[11] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *in NIPS*, vol. 33, pp. 596–608, 2020.

[12] Y. Xu, L. Shang, J. Ye, Q. Qian, Y.-F. Li, B. Sun, H. Li, and R. Jin, "Dash: Semi-supervised learning with dynamic thresholding," in *ICLR*, 2021, pp. 11 525–11 536.

[13] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," *in NIPS*, vol. 34, pp. 18 408–18 419, 2021.

[14] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on CVPR*. Ieee, 2009, pp. 248–255.

[16] L.-Z. Guo and Y.-F. Li, "Class-imbalanced semi-supervised learning with adaptive thresholding," in *ICLR*, 2022, pp. 8082–8094.

[17] K. Saito, D. Kim, and K. Saenko, "Openmatch: Open-set consistency regularization for semi-supervised learning with outliers," *in NIPS*, 2021.

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

[19] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *ICML*. PMLR, 2018, pp. 1568–1577.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICLR*. PMLR, 2017, pp. 1126–1135.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[22] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," *in NIPS*, vol. 29, 2016.

[23] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[24] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *in NIPS*, vol. 30, 2017.

[25] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," *in NIPS*, vol. 33, pp. 6256–6268, 2020.

[26] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *CVPRW*, 2020, pp. 702–703.

[27] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *arXiv preprint arXiv:1911.09785*, 2019.

[28] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *in NIPS*, vol. 31, 2018.

[29] J. Kim, Y. Hur, S. Park, E. Yang, S. J. Hwang, and J. Shin, "Distribution aligning refinery of pseudo-label for imbalanced semi-supervised learning," *in NIPS*, vol. 33, pp. 14 567–14 579, 2020.

[30] C. Wei, K. Sohn, C. Mellina, A. Yuille, and F. Yang, "Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning," in *CVPR*, 2021, pp. 10 857–10 866.

[31] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," *in NIPS*, vol. 32, 2019.

[32] H. Sun, C. Guo, Q. Wei, Z. Han, and Y. Yin, "Learning to rectify for robust learning with noisy labels," *Pattern Recognition*, vol. 124, p. 108467, 2022.

[33] L.-Z. Guo, Z.-Y. Zhang, Y. Jiang, Y.-F. Li, and Z.-H. Zhou, "Safe deep semi-supervised learning for unseen-class unlabeled data," in *ICLR*. PMLR, 2020, pp. 3897–3906.

[34] Y. Wang, J. Guo, S. Song, and G. Huang, "Meta-semi: A meta-learning approach for semi-supervised learning," *arXiv preprint arXiv:2007.02394*, 2020.

[35] T. Xiao, X.-Y. Zhang, H. Jia, M.-M. Cheng, and M.-H. Yang, "Semi-supervised learning with meta-gradient," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 73–81.

[36] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018, pp. 4334–4343.

[37] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," *in NIPS*, vol. 29, 2016.

[38] L. Wu, F. Tian, Y. Xia, Y. Fan, T. Qin, L. Jian-Huang, and T.-Y. Liu, "Learning to teach with dynamic loss functions," *in NIPS*, vol. 31, 2018.

[39] Y. Xu, L. Zhu, L. Jiang, and Y. Yang, "Faster meta update strategy for noise-robust deep learning," in *CVPR*, 2021, pp. 144–153.

[40] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011, pp. 215–223.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[42] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[43] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *in NIPS*, vol. 28, 2015.

[44] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *in NIPS*, vol. 32, 2019.

[45] J. Li, C. Xiong, and S. C. Hoi, "Comatch: Semi-supervised learning with contrastive graph regularization," in *ICCV*, 2021, pp. 9475–9484.

[46] M. Zheng, S. You, L. Huang, F. Wang, C. Qian, and C. Xu, "Simmatch: Semi-supervised learning with similarity matching," in *CVPR*, 2022, pp. 14 471–14 481.

[47] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," *arXiv preprint arXiv:1910.09217*, 2019.

TABLE 6: Detailed settings about training procedure of the backbone (the classifier network). Note that one training epoch contains 1024 iterations.

	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	WResNet-28-2			WResNet-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		

APPENDIX A
MORE EXPERIMENTAL DETAILS

A.1 Data Preprocessing

For those datasets that only provide labeled data, we only keep small-scale example as the labeled set and remove the labels of rest examples and translate them to the unlabeled set.

The augmentation strategy for different datasets can be divided into two parts:

- For small or medium scale of datasets (e.g., CIFAR, SVHN and STL-10), we follow FixMatch [11] and apply RandAugment [26] and CTAugment [44] on both labeled and unlabeled data. In our paper, we report the results on RandAugment which provides a greater performance for our proposed Meta-T.
- For large-scale dataset (i.e., ImageNet), we obey the convention in FixMatch [11]. The augmentation strategy include resizing, cropping (from 256×256 to 224×224) and horizontal flip.

A.2 Training Procedure

Detailed parameters of training strategies can be found in Table 6.

APPENDIX B
CONVERGENCE PROOF OF OUR METHOD

We give the theoretical guarantee of convergence for our learning algorithm.

Given a small amount of meta dataset with n samples $\{(\mathbf{x}_1^l, \mathbf{y}_1^l), ..., (\mathbf{x}_n^l, \mathbf{y}_n^l)\}$, the overall meta loss is,

$$L_{\text{meta}}(\mathbf{w}^*(\Theta)) = \frac{1}{n}\sum_{i=1}^{n} H(\mathbf{y}_i^l, f(\mathbf{x}_i^l; \mathbf{w}^*(\Theta))), \tag{12}$$

where $\mathbf{w}$ is the parameter of the classifier network and Θ is the parameter of our proposed TGN. Suppose that we have another unlabeled data $\{\mathbf{x}_1, ..., \mathbf{x}_{(\mu\times n)}\}$ with size of $\mu \times n$, the overall training loss is

$$\begin{aligned} L_{train}(\mathbf{w}, \Theta) = L_u(\mathbf{w}, \Theta) &= \frac{1}{n\mu}\sum_{i=1}^{n\mu} \ell_{\mathbf{x}_i}(\mathbf{w}, \Theta) \\ &= \frac{1}{n\mu}\sum_{i=1}^{n\mu} \mathbb{1}(\max(f(\mathcal{A}^w(\mathbf{x}_i); \mathbf{w})) > \mathcal{V}_i(\mathbf{w}, \Theta)) \cdot H(\hat{\mathbf{y}}_i, f(\mathcal{A}^s(\mathbf{x}_i); \mathbf{w})), \end{aligned} \tag{13}$$

where $\hat{\mathbf{y}}_i$ is the predicted result on weakly-augmented version of data $\mathbf{x}_i$ and $\mathcal{V}_i(\mathbf{w}, \Theta) = \mathcal{V}(g(f(\mathbf{x}_i; \mathbf{w})), \overline{\mathrm{p}}_c^t; \Theta)$.

B.1 Discuss about the Approximate Function $\mathcal{S}$

We replace the non-differential indicator function in Eq. 13 with a modified sigmoid function $\mathcal{S}(x) = \frac{1}{1+\exp^{-\beta x}}$. In Figure 2, we compare the difference between these two functions. We can see that the input of function satisfies $\max(f(\mathcal{A}^w(\mathrm{x}_m); \mathbf{w}) - \mathcal{V}_m(\mathbf{w}, \Theta)) \in [-1, 1]$. Meanwhile, the first-order and second-order gradient of sigmoid function obviously exist, which makes backpropagation of the training loss in Eq. 13 possible. Therefore, in practice, the training loss for unlabeled data can be written as

$$\ell_{\mathrm{x}_i}(\mathbf{w}, \Theta) = \mathcal{S}_i(\mathbf{w}, \Theta) \cdot H(\hat{\mathbf{y}}_i, f(\mathcal{A}^s(\mathbf{x}_i); \mathbf{w})), \tag{14}$$

where $\mathcal{S}_i(\mathbf{w},\Theta)=\mathcal{S}(\max(f(\mathcal{A}^w(\mathbf{x}_i;\mathbf{w})))-\mathcal{V}_i(\mathbf{w},\Theta))$ and the first term $\max(f(\mathcal{A}^w(\mathbf{x}_i;\mathbf{w})))$ is a constant. Meanwhile, the overall training loss of a mini-batch unlabeled data in Eq. (13) can be written as

$$L_{train}(\mathbf{w},\Theta)=\frac{1}{n\mu}\sum_{i=1}^{n\mu}\ell_{\mathrm{x}_i}(\mathbf{w},\Theta)=\frac{1}{n\mu}\sum_{i=1}^{n\mu}\mathcal{S}_i(\mathbf{w},\Theta)\cdot H(\hat{\mathbf{y}}_i,f(^s(\mathbf{x}_i);\mathbf{w})). \tag{15}$$

Note that the following theoretical verification is based on Eq. (15).

B.2 Derivation of the Generated Threshold in TGN

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

$$\Theta^{(t+1)}=\Theta^{(t)}-\beta\frac{1}{n}\sum_{i=1}^{n}\nabla_\Theta H(\mathbf{y}_i^l,f(\mathbf{x}_i^l;\hat{\mathbf{w}}^{(t)}(\Theta))). \tag{16}$$

To be concise, we formulate $H(\mathbf{y}_i^l,f(\mathbf{x}_i^l;\hat{\mathbf{w}}^{(t)}(\Theta)))$ as $H_i^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))$. Then, the computation of backpropagation for the above equation can be written as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\nabla_\Theta H_i^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}\\
=&\frac{1}{n}\sum_{i=1}^{n}\frac{\partial H_i^{\mathrm{meta}}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}\sum_{j=1}^{n\mu}\frac{\partial\hat{\mathbf{w}}^{(t)}(\Theta)}{\partial\mathcal{S}_j(\mathbf{w}^{(t)};\Theta)}\frac{\partial\mathcal{S}_j(\mathbf{w}^{(t)};\Theta)}{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}\frac{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}\\
=&\frac{-\alpha}{n^2\mu}\sum_{i=1}^{n}\frac{\partial H_i^{\mathrm{meta}}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}\sum_{j=1}^{n\mu}\frac{\partial\ell_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial\mathcal{S}_j(\mathbf{w})}\frac{\partial\mathcal{S}_j(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}\frac{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}\\
=&\frac{-\alpha}{n\mu}\sum_{j=1}^{n\mu}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\partial H_i^{\mathrm{meta}}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T}\frac{\partial\ell_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial\mathcal{S}_j(\mathbf{w})}\frac{\partial\mathcal{S}_j(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}\right)\frac{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}.
\end{aligned} \tag{17}$$

Let $G_{ij}=\frac{\partial H_i^{\mathrm{meta}}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T}\frac{\partial\ell_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial\mathcal{S}_j(\mathbf{w})}\frac{\partial\mathcal{S}_j(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}$ and substitute G_{ij} into Eq. (17), then we get:

$$\Theta^{(t+1)}=\Theta^{(t)}+\frac{\alpha\beta}{n\mu}\sum_{j=1}^{n\mu}\left(\frac{1}{n}\sum_{i=1}^{n}G_{ij}\right)\frac{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}. \tag{18}$$

B.3 Proofs of Smoothness

Lemma 4. *(Smoothness). Suppose the loss function H is L-Lipschitz and smooth, and the approximate function $\mathcal{S}$ 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 $\mathcal{S}$, the gradient of Θ w.r.t. the meta loss is Lipschitz continuous.*

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

$$\begin{aligned}
&\nabla_\Theta H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}\\
=&-\frac{\alpha}{n\mu}\sum_{j=1}^{n\mu}\left(\frac{\partial H^{\mathrm{meta}}(\hat{\mathbf{w}})}{\partial\hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T}\frac{\partial\ell_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial\mathcal{S}_j(\mathbf{w})}\frac{\partial\mathcal{S}_j(\mathbf{w})}{\partial\mathbf{w}}\Big|_{\mathbf{w}^{(t)}}\right)\frac{\partial\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}.
\end{aligned} \tag{19}$$

Let $\mathcal{V}_j(\Theta)=\mathcal{V}_j(\mathbf{w}^{(t)};\Theta)$ and introduce G_{ij} which is defined in Eq. (18). Taking the gradient of Θ on both side of Eq. (19), we attain

$$\begin{aligned}
&\nabla^2_{\Theta^2}H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}\\
=&-\frac{\alpha}{n\mu}\sum_{j=1}^{n\mu}\left[\frac{\partial}{\partial\Theta}(G_{ij})\Big|_{\Theta^{(t)}}\frac{\partial\mathcal{V}_j(\Theta)}{\partial\Theta}\Big|_{\Theta^{(t)}}+(G_{ij})\frac{\partial^2\mathcal{V}_j(\Theta)}{\partial^2\Theta}\Big|_{\Theta^{(t)}}\right].
\end{aligned} \tag{20}$$

The first term in Eq. (20) 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_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{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_{\mathbf{x}_k}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{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_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{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_{\mathbf{x}_k}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{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_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{S}_j(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \right\| \\
\leq & \alpha L \delta^2 \phi^2 \zeta^2,
\end{aligned} \tag{21}$$

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

The second term in Eq. (20) right hand side can be summarized as

$$\begin{aligned}
\left\| (G_{ij}) \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta}\Big|_{\Theta^{(t)}} \right\| &= \left\| \frac{\partial H^{\text{meta}}(\hat{\mathbf{w}})}{\partial \hat{\mathbf{w}}}\Big|_{\hat{\mathbf{w}}^{(t)}}^{T} \frac{\partial \ell_{\mathbf{x}_j}(\mathcal{S}_j(\mathbf{w}))}{\partial \mathcal{S}_j(\mathbf{w})} \frac{\partial \mathcal{S}_j(\mathbf{w})}{\partial \mathbf{w}}\Big|_{\mathbf{w}^{(t)}} \frac{\partial^2 \mathcal{V}_j(\Theta)}{\partial^2 \Theta}\Big|_{\Theta^{(t)}} \right\| \\
&\leq \rho \phi \zeta \mathcal{B}.
\end{aligned} \tag{22}$$

Combining the results in Eq. (21) and Eq. (22), we have

$$\left\| \nabla^2_{\Theta^2} H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}} \right\| \leq \phi\zeta(\alpha L \delta^2 \phi \zeta + \rho \mathcal{B}). \tag{23}$$

Define $\hat{L} = \phi\zeta(\alpha L \delta^2 \phi \zeta + \rho \mathcal{B})$, based on the Lagrange mean value theorem, we have:

$$\left\| \nabla L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta_1)) - \nabla L_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta_2)) \right\| \leq \hat{L} \left\| \Theta_1 - \Theta_2 \right\|, \text{ for all } \Theta_1, \Theta_2, \tag{24}$$

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

□

B.4 Proofs of Convergence

Theorem 5. *Following 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 η_t, $1 \leq t \leq T$ is a monotone descent sequence, $\eta_t = \min\{\frac{1}{L}, \frac{\mathrm{c}}{\sigma\sqrt{T}}\}$ for some $\mathrm{c} > 0$, such that $\frac{\sigma\sqrt{T}}{\mathrm{c}} \geq \hat{L}$ and $\sum_{t=1}^{\infty} \eta_t \leq \infty, \sum_{t=1}^{\infty} \eta_t^2 \leq \infty$. Then we have*

$$\frac{1}{T} \sum\nolimits_{t=1}^{T} \mathbb{E}\Big[\Big\| \nabla L_{\text{meta}}\Big(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}) \Big) \Big\|_2^2 \Big] \leq \mathcal{O}(\frac{1}{\sqrt{T}}). \tag{25}$$

Proof. The update of parameters Θ in t-th iteration can be written as

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{n} \sum_{i=1}^{n} \nabla_\Theta H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))\Big|_{\Theta^{(t)}}. \tag{26}$$

Training with a mini-batch of meat-data B_t that is uniformly drawn from the data set, we rewrite the equation above as:

$$\Theta^{(t+1)} = \Theta^{(t)} - \beta_t \Big[\sum_{i=1}^{n} \nabla_\Theta H_i^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta)) + \varepsilon^{(t)} \Big], \tag{27}$$

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

Consider that

$$\begin{aligned}
& H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\
= & \underbrace{H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}))}_{\text{term1}} + \underbrace{H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))}_{\text{term2}}.
\end{aligned} \tag{28}$$

For $\mathrm{term1}$, by Lipschitz smoothness of the meta loss function for Θ, we have

$$\begin{aligned}
& H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) \\
\leq & \Big\langle \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})), \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) \Big\rangle + \frac{L}{2} \Big\| \hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) \Big\|_2^2.
\end{aligned} \tag{29}$$

According to Eq. 8,10,15, we write $\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)}) - \hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)}) = -\alpha_t \frac{1}{n}\sum_{i=1}^{n} \mathcal{S}(\mathbf{w}^{(t+1)}; \Theta^{(t+1)}) \nabla_{\mathbf{w}} H(\mathbf{w})\Big|_{\mathbf{w}}^{(t+1)}$ and then we have

$$\left\| H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) \right\| \leq \alpha_t \rho^2 + \frac{1}{2} L \alpha_t \rho^2 = \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) \tag{30}$$

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

For term2, considering Lipschitz continuity of $\nabla H_{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta))$ demonstrated in Lemma 4, we can obtain the following:

$$\begin{aligned}
&H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\
\leq& \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \Theta^{(t+1)} - \Theta^{(t)} \right\rangle + \frac{L}{2} \left\| \Theta^{(t+1)} - \Theta^{(t)} \right\|_2^2 \\
=& \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), -\beta_t \Big[H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \varepsilon^{(t)} \Big] \right\rangle + \frac{L\beta_t^2}{2} \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \varepsilon^{(t)} \right\|_2^2 \\
=& -(\beta_t - \frac{L\beta_t^2}{2}) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\beta_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 - (\beta_t - L\beta_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle.
\end{aligned} \tag{31}$$

Summing up the Eq. (30), (31), the Eq. (28) can be summarized as

$$\begin{aligned}
&H^{\text{meta}}(\hat{\mathbf{w}}^{(t+1)}(\Theta^{(t+1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \\
\leq& \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) - (\beta_t - \frac{L\beta_t^2}{2}) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\beta_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 - (\beta_t - L\beta_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle.
\end{aligned} \tag{32}$$

Rearranging the terms, we can obtain

$$\begin{aligned}
&(\beta_t - \frac{L\beta_t^2}{2}) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \\
\leq& \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) - (\beta_t - \frac{L\beta_t^2}{2}) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 + \frac{L\beta_t^2}{2} \left\| \varepsilon^{(t)} \right\|_2^2 - (\beta_t - L\beta_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle.
\end{aligned} \tag{33}$$

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

$$\begin{aligned}
&\sum_{t=1}^{T} (\beta_t - \frac{L\beta_t^2}{2}) \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \\
\leq& H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) - H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) + \\
&\sum_{t=1}^{T} \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) - \sum_{t=1}^{T} (\beta_t - L\beta_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle + \frac{L}{2} \sum_{t=1}^{T} \left\| \varepsilon^{(t)} \right\|_2^2 \\
\leq& H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T} \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) - \sum_{t=1}^{T} (\beta_t - L\beta_t^2) \left\langle H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})), \varepsilon^{(t)} \right\rangle + \frac{L}{2} \sum_{t=1}^{T} \left\| \varepsilon^{(t)} \right\|_2^2.
\end{aligned} \tag{34}$$

We take the expectations *w.r.t.* $\varepsilon^{(N)}$ on both size of Eq. (34), then we have:

$$\sum_{t=1}^{T} (\beta_t - \frac{L\beta_t^2}{2}) \mathop{\mathbb{E}}_{\varepsilon^{(N)}} \left\| \nabla H^{\text{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)})) \right\|_2^2 \leq H^{\text{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T} \alpha \rho^2 (1 + \frac{\alpha_t L}{2}) + \frac{L\sigma^2}{2} \sum_{t=1}^{T} \beta_t^2, \tag{35}$$

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

that

$$
\begin{aligned}
\min_t \mathbb{E}\Big[\Big\|\nabla H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\Big\|_2^2\Big] &\leq \frac{\sum_{t=1}^{T}(\beta_t - \frac{L\beta_t^2}{2}) \mathop{\mathbb{E}}_{\varepsilon^{(N)}} \Big\|\nabla H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(t)}(\Theta^{(t)}))\Big\|_2^2}{\sum_{t=1}^{T}(\beta_t - \frac{L\beta_t^2}{2})} \\
&\leq \frac{1}{\sum_{t=1}^{T}(2\beta_t - L\beta_t^2)}\Big[2H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T}\alpha\rho^2(2+\alpha_t L) + L\sigma^2\sum_{t=1}^{T}\beta_t^2\Big] \\
&\leq \frac{1}{\sum_{t=1}^{T}\beta_t}\Big[2H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \sum_{t=1}^{T}\alpha\rho^2(2+\alpha_t L) + L\sigma^2\sum_{t=1}^{T}\beta_t^2\Big] \\
&\leq \frac{1}{T\beta_t}\Big[2H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)})) + \alpha_1\rho^2 T(2+L) + L\sigma^2\sum_{t=1}^{T}\beta_t^2\Big] \\
&= \frac{2H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T}\frac{1}{\beta_t} + \frac{2\alpha_1\rho^2(2+L)}{\beta_t} + \frac{L\sigma^2}{T}\sum_{t=1}^{T}\beta_t \\
&\leq \frac{2H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T}\frac{1}{\beta_t} + \frac{2\alpha_1\rho^2(2+L)}{\beta_t} + L\sigma^2\beta_t \\
&= \frac{H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{T}\max\{L, \frac{\sigma\sqrt{T}}{\mathrm{c}}\} + \min\{1, \frac{k}{T}\}\max\{L, \frac{\sigma\sqrt{T}}{\mathrm{c}}\}\rho^2(2+L) + L\sigma^2\min\{\frac{1}{L}, \frac{\mathrm{c}}{\sigma\sqrt{T}}\} \\
&\leq \frac{\sigma H^{\mathrm{meta}}(\hat{\mathbf{w}}^{(1)}(\Theta^{(1)}))}{\mathrm{c}\sqrt{T}} + \frac{k\sigma\rho^2(2+L)}{\mathrm{c}\sqrt{T}} + \frac{L\sigma\mathrm{c}}{\sqrt{T}} = \mathcal{O}(\frac{1}{\sqrt{T}}).
\end{aligned}
\tag{36}
$$

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