Rethinking Confidence Scores and Thresholds in Pseudolabeling-based SSL

Anonymous Authors[1]

Abstract

Modern semi-supervised learning (SSL) methods rely on pseudolabeling and consistency regularization. Pseudolabeling is typically performed by comparing the model's confidence scores and a predefined threshold. While several heuristics have been proposed to improve threshold selection, the underlying issues of overconfidence and miscalibration in confidence scores remain largely unaddressed, leading to inaccurate pseudolabels, degraded test accuracy, and prolonged training. We take a first-principles approach to learn confidence scores and thresholds with an explicit knob for error. This flexible framework addresses the fundamental question of optimal scores and threshold selection in pseudolabeling. Moreover, it gives practitioners a principled way to control the quality and quantity of pseudolabels. Such control is vital in SSL, where balancing pseudolabel quality and quantity directly affects model performance and training efficiency. Our experiments show that, by integrating this framework with modern SSL methods, we achieve significant improvements in accuracy and training efficiency. In addition, we provide novel insights on the tradeoffs between the choices of the error parameter and the end model's performance.

1. Introduction

The lack of high-quality labeled data is a major bottleneck in training high-accuracy models. The semi-supervised learning (SSL) paradigm tackles this problem by leveraging abundant unlabeled data alongside a limited set of labeled examples (Chapelle et al., 2006; Zhu, 2005; van Engelen & Hoos, 2019). While SSL dates back decades and includes a wide variety of approaches, modern SSL methods frequently rely on a pair of ideas: *self-training or pseudolabeling* – where model generated labels are assigned to unlabeled data for further training (McLachlan, 1975; Amini et al., 2023; Rosenberg et al., 2005; Lee, 2013; Rizve et al., 2021) – and *consistency regularization* to enforce stability in predictions across perturbed inputs (Laine & Aila, 2017; Bachman et al., 2014; Sajjadi et al., 2016; Fan et al., 2021; Kukačka et al., 2017). SSL techniques with these ideas have strongly performed on several benchmark datasets.

While pseudolabeling is a powerful technique, its effectiveness hinges on a fundamental question: *which points should be labeled using the model's predictions?* Since pseudolabels are derived from a model being trained, they can be highly unreliable. A naive approach that assigns pseudolabels too liberally risks injecting noisy labels into training, amplifying the model's existing errors – a phenomenon known as confirmation bias. Conversely, an overly conservative approach that selects only the correct predictions severely limits the amount of useful training data. Both extremes can degrade SSL performance, leading to either poor generalization or slow convergence. To fully harness the potential of pseudolabeling, we need a principled approach for pseudolabeling with explicit control on these trade-offs.

A widely used strategy pseudolabels points on which the model's confidence score exceeds a threshold. This approach provides a simple mechanism for selecting points while controlling the quality and quantity of pseudolabels via thresholds. However, the prior works based on this approach suffer from two key limitations that restrict its effectiveness. First, thresholding techniques are often heuristic-driven, lacking a precise control for a target error level (Sohn et al., 2020; Zhang et al., 2021; Wang et al., 2023; Xu et al., 2021; Chen et al., 2023; Li et al., 2023). Second, commonly used scores, such as the model's softmax outputs, tend to be unreliable. Recent studies in this vein (Loh et al., 2022; Mishra et al., 2024; Rizve et al., 2021) have highlighted issues of overconfidence and miscalibration in these scores, leading to inefficiencies in pseudolabeling.

In addition to these problems with the choices of scores and thresholds, an equally important question remains: how much error should be allowed in pseudolabeling? As discussed earlier, a very low tolerance may pseudolabel too few points and conversely a high tolerance may allow for large errors, hurting the efficiency of pseudolabeling in both cases. To solve these issues, we seek *a principled solution*

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

to get confidence scores and thresholds for pseudolabeling at a given target level of error tolerance.

We take a first principles approach to formalize the pseudolabeling objective: maximize the number of pseudolabeled points while adhering to the given error tolerance. We formulate this objective as an optimization problem over a flexible space of confidence functions and thresholds. By solving this optimization problem we obtain confidence scores tailored to our objective of psuedolabeling. To ensure the pseudolabeling error constraint is strictly followed, we use a separate procedure to estimate thresholds on these scores using part of the validation data. Pseudolabeling with these scores and thresholds ensures we are pseudolabeling a maximal set of points that can be pseudolabeled at the given error tolerance level.

We integrate this approach into popular pseudolabeling-based SSL methods, providing two benefits. First, it provides a principled way to derive confidence scores and thresholds for any given error tolerance. Second, by enabling more precise pseudolabeling it improves the utilization of the unlabeled data and is expected to yield an end model with higher test accuracy compared to the ad-hoc choices of scores and thresholds.

Our main contributions are summarized as follows,

1. Our work settles the question of the right choices of confidence scores and thresholds in pseudolabeling-based SSL methods by introducing a framework for learning confidence scores and thresholds. Departing from heuristic-driven or ad-hoc and unreliable choices of scores and thresholds, this framework provides principled choices of scores and thresholds for pseudolabeling with any target error tolerance.

2. We show how this framework for learning confidence scores and thresholds can work in concert with popular SSL methods such as Fixmatch (Sohn et al., 2020), Freematch (Wang et al., 2023), etc., and conduct an extensive empirical evaluation demonstrating that by pseudolabeling using confidence scores and thresholds learned from our method can yield significant improvements in the test accuracy.

3. Leveraging our framework's ability to pseudolabel at any target error level, we study the impact of varying pseudolabeling error levels—from fixed to dynamic tolerance throughout training. Our results confirm the intuition that lower pseudolabeling errors lead to better end models compared to higher errors. Moreover, among dynamic schedules, it is better to use a decreasing schedule of error tolerances.

2. Background and Problem Setup

We begin with notation, then provide useful background and a statement of our goal.

Notation. Consider a feature space $\mathcal{X}$ and label space $\mathcal{Y} = \{1, \ldots, k\}$ in a k-class classification task. As usual in semi-supervised learning, we have access to a set $X_u = \{\mathbf{x}_u\}_{u=1}^{n_u}$ of unlabeled data drawn from the distribution P_x over $\mathcal{X}$. We also have access to $D_l = \{(\mathbf{x}_l, y_l)\}_{l=1}^{N_l}$, a set of labeled data points drawn from the joint distribution P_{xy}, with $N_l \ll n_u$. Let $h : \mathcal{X} \to \mathcal{Y}$ denote a model and $g : \mathcal{X} \to T^k \subseteq \mathbb{R}^k$ be an associated confidence function giving a score $g(\mathbf{x})$ indicating the confidence of h on its prediction for any data point $\mathbf{x}$. For any $\mathbf{x}$ the hard label prediction is $\widehat{y} := h(\mathbf{x})$. When the prediction $\widehat{y}$ is used as a pseudolabel we denote it as $\tilde{y}$. In general, for a vector $\mathbf{v} \in \mathbb{R}^d$, $\mathbf{v}[i]$ denotes its $i-$th component. The vector $\mathbf{t}$ denotes thresholds over the scores k-classes, and $\mathbf{t}[y]$ is its $y-$th entry, i.e., the threshold for class y.

2.1. Pseudolabeling-based Semi-Supervised Learning

Given, as above, a large collection of unlabeled data X_u and a small set of labeled points D_l, inductive semi-supervised learning (SSL) seeks to learn a classifier $\widehat{h}_{\text{ssl}}$ from the model class $\mathcal{H}$. The promise of SSL is that by effectively using X_u in the learning process it can learn a better classifier than its supervised counterpart, which learns only from D_l.

In many recent pseudolabeling-based SSL techniques, in each iteration of training, a batch of labeled and unlabeled data is obtained, then the sum of the losses $\widehat{\mathcal{L}} = \widehat{\mathcal{L}}_s + \lambda_u \widehat{\mathcal{L}}_u + \lambda_r \widehat{\mathcal{L}}_r$ is minimized w.r.t to the model h. Here $\widehat{\mathcal{L}}_s$ is the supervised loss, $\widehat{\mathcal{L}}_u$ unsupervised loss, and $\widehat{\mathcal{L}}_r$ is (the sum of) regularization term(s). The constants λ_u, λ_r are hyperparameters controlling the relative importance of the corresponding terms.

Supervised loss. Given a batch of labeled data, D_l^b the supervised loss is computed as follows, $\widehat{\mathcal{L}}_s(h \mid D_l^b) = \frac{1}{|D_l^b|} \sum_{(\mathbf{x},y) \in D_l^b} H(y, h, \mathbf{x})$. Here $H(y, h, \mathbf{x})$ is the standard cross-entropy loss between the 1-hot representation of y and the softmax output of h on input $\mathbf{x}$.

Unsupervised loss and consistency regularization. For the unlabeled batch X_u^b, pseudolabels $\widetilde{y} = h(\mathbf{x})$ are computed for each $\mathbf{x} \in X_u^b$. Then, a pseudolabeling mask, $S(\mathbf{x}, g, \mathbf{t} \mid h) = \mathbb{1}(g(\mathbf{x})[\widetilde{y}] \geq \mathbf{t}[\tilde{y}])$ is computed. It is 1 for points having confidence score $g(\mathbf{x})[\tilde{y}]$, bigger than predetermined threshold $\mathbf{t}[\tilde{y}]$, corresponding to the predicted class $\tilde{y}$. Recent methods, couple this loss and consistency regularization together by doing pseudolabeling on weakly augmented data using weak transform $\omega : \mathcal{X} \mapsto \mathcal{X}$ and then defining the cross-entropy loss on the strongly augmented

data using strong transformation $\Omega : \mathcal{X} \mapsto \mathcal{X}$. The loss is,

$$\widehat{\mathcal{L}}_u := \frac{1}{|\widetilde{D}_u^b|} \sum_{(x,\tilde{y}) \in \tilde{D}_u^b} S(\omega(\mathbf{x}), g, \mathbf{t}|h) \cdot H(\tilde{y}, h, \Omega(\mathbf{x})).$$

Regularization. A regularization term (or a sum of multiple regularizers) is often included along with the above two losses to encourage desired behavior(s). For instance, Freematch (Wang et al., 2023) adds a self-adaptive class fairness regularizer to encourage diverse predictions during the initial training phase. Similarly, a regularizer is introduced in (Mishra et al., 2024) to encourage calibration in the model's confidence scores. Including such regularizers has been fruitful in pseudolabeling-based SSL.

2.2. Quality and Quantity of Pseudolabels

Given a classifier h, the quality and quantity of the pseudolabels w.r.t. to score function g and thresholds $\mathbf{t}$, are:

Pseudolabeling coverage (quantity). Given a set of points X, the pseudolabeling coverage is the fraction of points that are pseudolabeled using h, g and $\mathbf{t}$. This measurement captures the quantity of pseudolabels and is defined as

$$\widehat{\mathcal{P}}(g, \mathbf{t} \mid h, X) := \frac{1}{|X|} \sum_{(\mathbf{x}) \in X} S(\mathbf{x}, g, \mathbf{t} \mid h) \tag{1}$$

$$\mathcal{P}(g, \mathbf{t} \mid h) := \mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h)]. \tag{2}$$

Pseudolabeling error (quality). This is the fraction of pseudolabeled points that got incorrect labels. This metric captures the quality of pseudolabels:

$$\widehat{\mathcal{E}}(g, \mathbf{t} \mid h, D) := \frac{\sum_{(\mathbf{x},y,\tilde{y}) \in D} S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)}{\sum_{(\mathbf{x},y,\tilde{y}) \in D} S(\mathbf{x}, g, \mathbf{t} \mid h)}, \tag{3}$$

$$\mathcal{E}(g, \mathbf{t} \mid h) = \frac{\mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)]}{\mathcal{P}(g, \mathbf{t}|h)}. \tag{4}$$

2.3. Our Goals

Pseudolabeling-based SSL aims to learn a classifier $\widehat{h}_{\text{ssl}}$ that generalizes well on the unseen data, i.e., has high test accuracy. This is typically achieved by pseudolabeling points using confidence scores and thresholds and incorporating them into training. However, existing choices of scores and thresholding strategies are often ad-hoc and unreliable, limiting their effectiveness. Departing from these unreliable approaches, our goal is to:

(i) Design principled solutions for confidence scores and thresholding to maximize the number of pseudolabeled points while ensuring pseudolabeling error is at most ϵ.

(ii) Incorporate these in the existing pseudolabeling-based SSL methods and assess whether this gives a better end model $\widehat{h}_{\text{ssl}}$.

(iii) Study the sensitivity of the SSL pipeline to pseudolabeling errors by leveraging the ability of our approach to explicitly ensure the pseudolabeling error remains below ϵ.

3. Methodology

In this section, we discuss our principled framework to learn scores and thresholds with explicit control of the pseudolabeling errors and use them in pseudolabeling-based SSL.

3.1. Pseudolabeling Optimization Framework

Given current model $\widehat{h}_i$ in the ith iteration, can we obtain confidence scores and thresholds using which we can identify a maximal set of points that can be pseudolabeled using $\widehat{h}_i$ with at most ϵ error? We begin with a theoretical formulation to learn such scores and thresholds and then introduce its practical version.

Theoretical framework. Instead of improving calibration or heuristics for thresholding, we propose to express the objective of pseudolabeling as an optimization problem over the space of confidence functions and thresholds. The objective is to maximize the quantity, i.e., the pseudolabeling coverage (eq. (2)) while keeping the pseudolabeling error (eq. (4)) below a tolerance level $\epsilon \in (0, 1)$. More specifically, given the classifier $\widehat{h}_i$ in any iteration i of SSL,

$$g_i^\star, \mathbf{t}_i^* \in \underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\max}\, \mathcal{P}(g, \mathbf{t} \mid \widehat{h}_i) \text{ s.t. } \mathcal{E}(g, \mathbf{t} \mid \widehat{h}_i) \leq \epsilon, \tag{5}$$

are the optimal confidence functions and thresholds for pseudolabeling using $\widehat{h}_i$'s predictions such that the pseudolabeling error is bounded by ϵ. This frees us from arbitrary choices of confidence scores, calibration techniques, and thresholding heuristics. Instead, solving the optimization problem over a flexible enough space will subsume specific strategies. Next, we discuss how to make this framework tractable to obtain scores and thresholds in practice.

Practical version. The optimization problem discussed earlier involves population-level quantities which are usually not accessible in practice. Thus we have to use their finite sample estimates and smooth variations to make the optimization problem tractable. Specifically, the coverage and error are estimated using a small amount of held-out labeled data (called calibration data D_{cal}) curated from the validation data. Then differentiable surrogates for the 0-1 variables are introduced. Let $\sigma(\alpha, z) := 1/(1 + \exp(-\alpha z))$ denote the sigmoid function on $\mathbb{R}$ with scale parameter

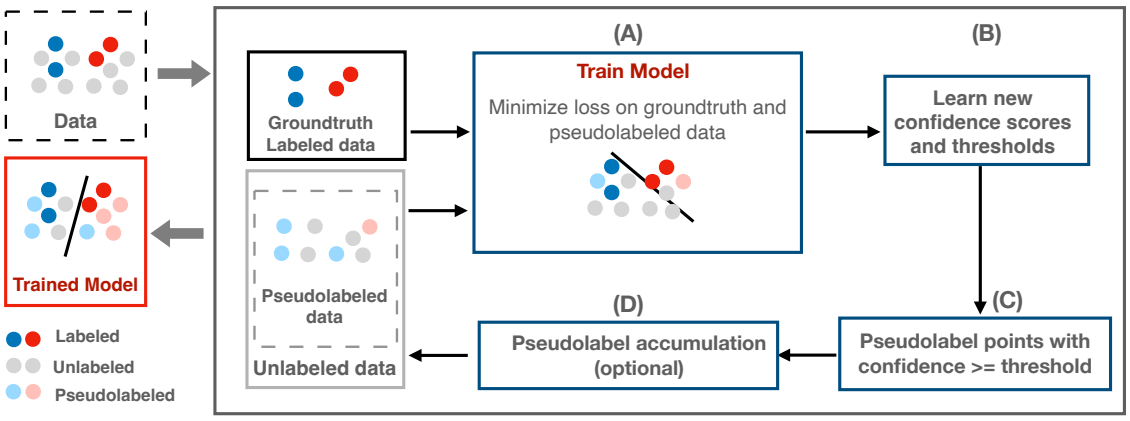

Figure 1. Workflow of pseudolabeling-based SSL with `PabLO` (A): Train model with standard supervised loss, consistency regularization, and other regularizers (B): Learn new confidence scores and thresholds (C): Pseudolabel points with scores greater than the estimated thresholds (D): An optional pseudolabeling accumulation to use previous pseudolabels for points that are not pseudolabeled in current round. Note, that the pseudolabels can be noisy (incorrect). The training and pseudolabeling loop continues until a pre-specified number of iterations. In the end, it outputs a model $\widehat{h}_{\text{ssl}}$ that is expected to have higher test accuracy than the model trained only on the given groundtruth training data. Note that in the end there might be points left unlabeled and the pseudolabels might be noisy.

$\alpha \in \mathbb{R}$. The surrogates are as follows,

$$\widetilde{\mathcal{P}}(g, \mathbf{t} \mid h, D_{\text{cal}}) := \frac{1}{|D_{\text{cal}}|} \sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big), \tag{6}$$

$$\widetilde{\mathcal{E}}(g, \mathbf{t} \mid h, D_{\text{cal}}) := \frac{\sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \mathbb{1}\big(y \neq \widetilde{y}\big)\, \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big)}{\sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big)} \tag{7}$$

Using these surrogates, the following practical optimization problem is obtained. It is also converted into unconstrained formulation by introducing the penalty term $\lambda \in \mathbb{R}^+$ controlling the relative importance of the error and coverage.

$$\widehat{g}_i, \widehat{\mathbf{t}}'_i \in \underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\min} \; -\widetilde{\mathcal{P}}(g, \mathbf{t} \mid \widehat{h}_i, D_{\text{cal}}) + \lambda\, \widetilde{\mathcal{E}}(g, \mathbf{t} \mid \widehat{h}_i, D_{\text{cal}}) \tag{P1}$$

We use 2-layer neural nets as a choice of $\mathcal{G}$ and $T^k = [0, 1]^k$. The optimization problem (P1) is non-convex but differentiable and we solve it using Stochastic Gradient Descent (SGD). See Appendix C for more details on our choice of $\mathcal{G}$ and training details and hyperparameters.

3.2. Threshold Estimation

While we can obtain both the confidence scores and thresholds by solving (P1), we propose to estimate thresholds separately on a held-out part of the validation data to avoid potential generalization issues due to learning them simultaneously from the same data D_{cal} and ensure that the pseudolabeling error constraint is strictly adhered to.

When dealing with datasets containing many classes there may not be enough samples per class to estimate reliable thresholds. Thus, to accommodate these possibilities we consider two variations of the threshold estimation procedure, (i) estimate a common (joint) threshold for all classes and (ii) estimate separate (classwise) thresholds for each class. The procedures are outlined in Algorithm 2 and Algorithm 1 in the Appendix. We discuss them briefly here.

The procedure takes in a confidence function $\widehat{g}_i$ and part of the held-out validation data referred to as D_{th}. The idea is to estimate errors at several thresholds on this data and then pick the smallest threshold. This can be done separately for each class to obtain classwise thresholds or a common threshold for all classes. We discuss classwise thresholding here. First the data D_{th} is partitioned into k subsets $D_{\text{th}}^{(y)}$ corresponding to each class $y \in \mathcal{Y}$. Here, we slightly abuse notation: instead of $\mathbf{t} \in T^k$, we use $t \in T$ in the estimate of pseudolabeling error at threshold t for class y. To obtain threshold $\widetilde{\mathbf{t}}[y]$ for class y, the procedure finds the smallest $t \in T$ such that $\widehat{\mathcal{E}}(\widehat{g}_i, t \mid h, D_{\text{th}}^{(y)}) + C_1 \widehat{\sigma}(\widehat{\mathcal{E}}) \leq \epsilon$. Here C_1 is a constant (we use $C_1 = 0.25$) and $\widehat{\sigma}(z) = \sqrt{z \cdot (1 - z)}$ and $\widehat{\mathcal{E}}$ is used for brevity in place of $\widehat{\mathcal{E}}(\widehat{g}_i, t \mid h, D_{\text{th}}^{(y)})$. The same process is used for joint threshold estimation where a single threshold t is estimated using entire D_{th} and the same t is used for all classes. Using the thresholds found using these procedures ensures pseudolabeling error remains below (or close to) the tolerance level ϵ.

Remarks. Departing from fixed thresholds as in (Sohn et al., 2020), prior works have proposed adaptive and class-wise heuristic thresholding schemes based on the model's learning status, such as in (Djurisic et al., 2023; Zhang et al., 2021; Wang et al., 2023) and others. In contrast, our approach is a principled way to estimate adaptive and class-wise pseudolabeling thresholds while providing strict con-

Dataset	Backbone Model h	k	n_u	N_{val}	N_{test}	N_l	N_{cal}	N_{th}	Augmentation
CIFAR-10	WRN-28-2	10	50K	6K	4K	250	1K	1K	Weak, Strong
CIFAR-100	WRN-28-2	100	50K	6K	4K	2500	3K	3K	Weak, Strong
SVHN	WRN-28-2	10	604,388	15,620	10,412	250	3K	3K	Weak, Strong

Table 1. Details of the dataset we use in our experiments. k is the number of classes. N_l is the number of labeled data points used for training the backbone model h. n_u is the number of unlabelled data points used for consistency regularization and pseudolabeling for all the methods. N_{val} is the number of points used for model selection in all methods. N_{test} is the number of test data points. N_{cal} is the number of points used for learning the g function. N_{th} is the number of data points used for threshold estimation.

trol over the quality of pseudolabels. Similar procedures have been used in the context of creating reliable datasets and are backed by theoretical guarantees for the quality of pseudolabels produced (Vishwakarma et al., 2023).

3.3. Pseudolabeling and Accumulation

In the usual pseudolabeling-based SSL setups, the pseudolabels inferred by the model for a mini-batch are discarded after each iteration. Moreover, it is not guaranteed that a previously pseudolabeled point will get pseudolabled in the current iteration as well. Given the quality of pseudolabels is high, it is appealing to reuse the past pseudolabel for a point that did not get pseudolabeled in the current iteration. We propose to do so for techniques where the quality of pseudolabels is assured. We refer to this as "pseudolabel accumulation".

Mathematically, let $\widetilde{y}_j^{(i-1)} = \widetilde{Y}_u^{(i-1)}[j]$ and $\widetilde{y}_j^{(i)} = \widetilde{Y}_u^{(i)}[j]$ be the previous and current (fresh) pseudolabels for jth unlabeled point. Let the corresponding masks (indicating whether the score is above the threshold) for these psuedolabels be $S_u^{(i-1)}[j]$ and $S_u^{(i)}[j] = \mathbb{1}(\widehat{g}_i(\mathbf{x}_j)[\widetilde{y}_j^{(i)}] \geq \widehat{\mathbf{t}}_i[\widetilde{y}_j^{(i)}])$. Then with accumulation,

$$\widetilde{Y}_u^{(i)}[j] \leftarrow S_u^{(i)}[j]\widetilde{Y}_u^{(i)}[j] + (1 - S_u^{(i)}[j])\widetilde{Y}_u^{(i-1)}[j],$$
$$S_u^{(i)}[j] \leftarrow S_u^{(i)}[j] \vee S_u^{(i-1)}[j].$$

Here $\vee$ is the boolean `or` operation and the steps are executed in the order. In words, if the point is pseudolabeled in the current iteration (i.e. its current mask is 1), then it will use the current pseudolabel o.w. if the point was pseudolabeled in earlier iteration(s) it will use the pseudolabel from that iteration and mark the point as pseudolabeld. In case the point is not pseudolabeled in this iteration or any other iteration in the past it will remain unlabeled. While it is appealing to use this trick, its use is only warranted in settings ensuring high-quality pseudolabels. We try to understand the consequences of the inclusion and exclusion of this trick in pseudolabeling-based SSL via experiments discussed in the next section.

We put together the steps for learning scores, thresholds, and performing (optional) accumulation in a common template of pseudolabeling-based SSL algorithms. We refer to this adapted method (Algorithm 3 in Appendix B) as `PabLO`. The high-level steps are also illustrated in Figure 1. Next, we discuss the empirical evaluation of `PabLO` and baselines.

4. Experiments

We conduct empirical evaluation over several settings to,

C1. Verify that the adaptations of popular pseudolabeling-based SSL methods with `PabLO` output models with better test accuracy.

C2. Study the effects of choice of error tolerance ϵ on test accuracy of the final model.

C3. Understand the role of pseudolabel accumulation in our method and baselines.

4.1. Experimental Setup

First, we briefly describe the experimental setup, with details deferred to Appendix C.

Methods. We use two simple base methods that capture the core ideas of pseudolabeling (PL) and consistency regularization (CR). The first is *Fixmatch* (Sohn et al., 2020), which uses fixed thresholds on (maximum softmax probability) MSP scores for PL and CR. *Freematch* (Wang et al., 2023) improves upon it by using adaptive, class-wise thresholds and class fairness regularization (CFR) along with CR, and is a promising method among others using dynamic thresholds for PL. We include their combinations with recently proposed *Bayesian Model Averaging (BAM)* (Loh et al., 2023) and *Margin Regularization (MR)* [1] (Mishra et al., 2024) to improve calibration in SSL. We replace the pseudolabeling component by our method `PabLO` to obtain *Fixmatch + Ours* (a combination of `PabLO` and CR) and *Freematch + Ours* (a combination of `PabLO`, CR, and CFR). We provide implementations of these in the code submitted along with the paper.

Datasets. We experiment with three datasets: *CIFAR-10* (Krizhevsky et al., 2009) is an image dataset with 10 classes. *CIFAR-100* (Krizhevsky et al., 2009) is an extended version

[1] We assign this name for convenience.

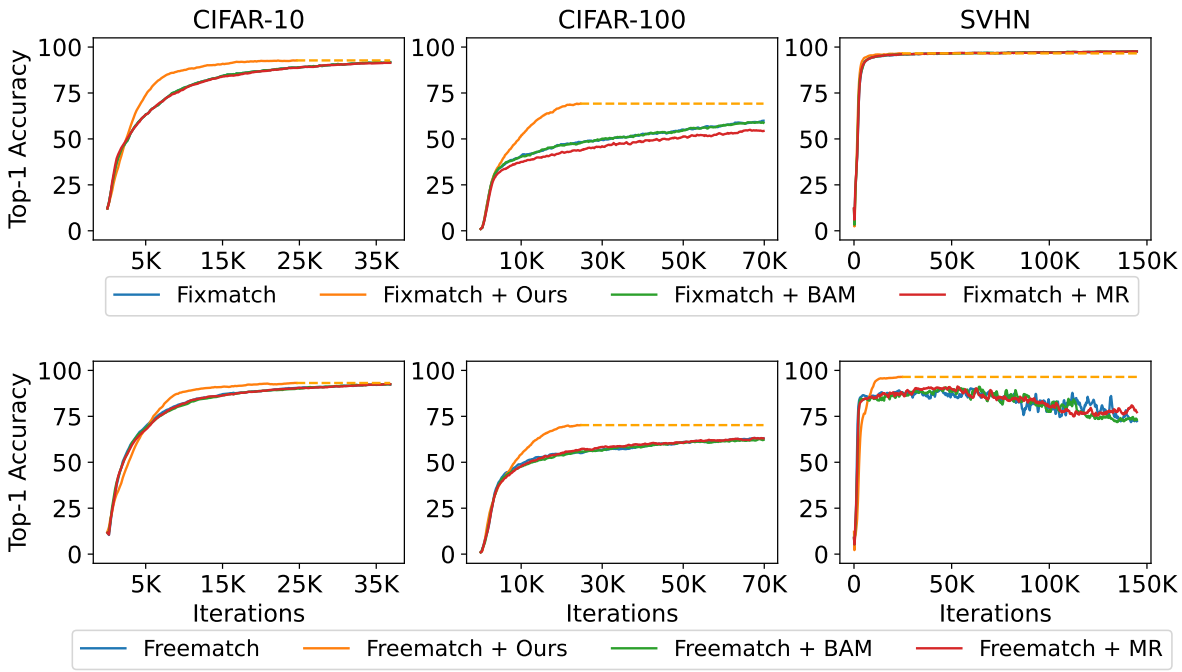

Figure 2. Top-1 accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

of CIFAR-10 with 100 classes. *SVHN* (Netzer et al., 2011) is a 10-class image dataset of digits from Google Street View. More details are summarized in Table 1. We use a portion of the validation data (D_{val}) for our method, split into D_{cal}, used to learn the function g, and D_{th}, used to estimate the threshold.

Adjusted iterations for baselines. Empirically, our method requires more time to run compared to base SSL techniques. Therefore, we adopt the following strategy to ensure a fair comparison between the baselines and our method: First, we train our method for 25K iterations and obtain the average per iteration time, denoted as α_o. Then, we train each baseline method b for 5K iterations and obtain the average per iteration time, denoted as α_b. Using these two values, we obtain the adjusted number of iterations, $\frac{\alpha_o}{\alpha_b} \times 25000$, for baseline method b. Coincidently, baselines under the same dataset have similar runtime. We, therefore, set the adjusted number of iterations on a dataset level. For CIFAR-10, the adjusted number of iterations for baselines is 37,000. For CIFAR-100, the adjusted number of iterations for baselines is 70,000. For SVHN, it is 145,000.

Models and training. The backbone encoder is a Wide ResNet-28-2 for all the datasets. We use the default hyperparameters and dataset-specific settings (learning rates, batch size, optimizers, and schedulers) following previous baseline recommendations (Wang et al., 2022). For confidence functions class $\mathcal{G}$, we use a class of 2-layer neural nets and provide the last two layers representations from h as input. We train it using SGD. The hyperparameters are deferred to Appendix C. Unless otherwise specified, our method uses pseudolabeling error tolerance $\epsilon = 5\%$.

4.2. Results and Discussion

To verify our main claims, we compare the baselines, their combinations with our method, and methods that induce calibrated scores in SSL. We run all methods with three random seeds and report (in Table 2) the mean and standard deviation of accuracy across three runs.

C1. Test accuracy improvements. Since our method maximizes the pseudolabeling coverage and accuracy, it provides more accurate pseudolabels for model training. Therefore, we expect it to yield a model with better test accuracy than the baselines. We report the test accuracies at the end of 25K iterations in Table 2 for our methods. For the baselines, we report the test accuracies at the end of the corresponding adjusted number of iterations (well above 25K). Figure 2 illustrates how the top-1 accuracy evolves during the SSL. Similarly, Figure 4 and 5 show how batch pseudolabeling accuracy and batch pseudolabeling coverage change.

First, as expected, integrating our method into the base methods improves test accuracy across all settings. For CIFAR-10, using it with Fixmatch provides almost 2% improvement over Fixmatch alone, and using it with Freematch yields 1% improvement over Freematch. Much more significant improvements are observed in the much harder setting of CIFAR-100: a nearly 10% improvement in top-1 accuracy over Fixmatch and around 5% improvement over

Dataset	CIFAR-10	CIFAR-100	SVHN
# Labels	250	2500	250
Fixmatch	90.8 ± 0.78	59.09 ± 1.10	$\mathbf{97.57 \pm 0.08}$
Fixmatch + MR	90.41 ± 0.83	54.16 ± 0.18	97.55 ± 0.08
Fixmatch + BaM	90.67 ± 0.90	56.60 ± 2.45	97.51 ± 0.13
Fixmatch + Ours	$\mathbf{92.69 \pm 0.74}$	$\mathbf{69.10 \pm 0.45}$	96.54 ± 0.13
Freematch	92.26 ± 0.18	63.13 ± 0.46	92.90 ± 2.76
Freematch + MR	92.17 ± 0.36	62.03 ± 0.82	93.26 ± 2.36
Freematch + BaM	92.32 ± 0.25	62.13 ± 2.93	91.08 ± 3.72
Freematch + Ours	$\mathbf{93.10 \pm 0.28}$	$\mathbf{68.76 \pm 1.38}$	$\mathbf{96.65 \pm 0.26}$

Table 2. Top-1 Accuracy for CIFAR-10, CIFAR-100 and SVHN averaged across 3 random seeds. The best accuracy is **bolded.**

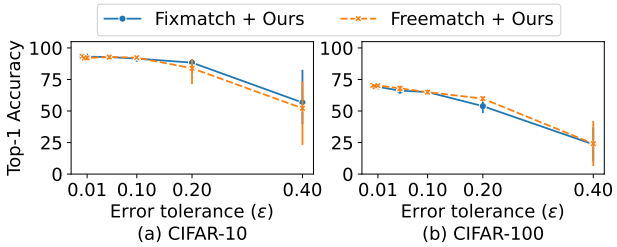

Figure 3. Top-1 accuracy of our method with different error tolerance ϵ on (a) CIFAR-10 and (b) CIFAR-100 dataset.

	CIFAR-10 Top-1 Accuracy	CIFAR-100 Top-1 Accuracy
Schedule 1 Fixmatch + Ours	91.11 ± 1.31	65.78 ± 1.36
Schedule 1 Freematch + Ours	91.09 ± 1.01	66.13 ± 0.48
Schedule 2 Fixmatch + Ours	35.38 ± 29.27	19.56 ± 3.07
Schedule 2 Freematch + Ours	30.48 ± 10.52	24.80 ± 3.49

Table 3. Top-1 accuracy for the two error tolerance (ϵ) scheduling. The table reports the ϵ we use between each iteration interval and the top-1 accuracy yielded by the corresponding schedule. For schedule 1, we set $\epsilon = 0.4, 0.2, 0.1, 0.05, 0.01$ when the number of iterations is in the following interval, respectively: $[0, 5\text{K}), [5\text{K}, 10\text{K}), [10\text{K}, 15\text{K}), [15\text{K}, 20\text{K}), [20\text{K}, 25\text{K})$. For schedule 2, we set $\epsilon = 0.01, 0.05, 0.1, 0.2, 0.4$, for the same intervals, respectively.

Freematch. SVHN is an easier setting; here, the improvements are marginal. With Fixmatch, our performance is similar to that of the baselines. But, using `PabLO` with Freematch improves the performance by 3%.

C2. Error tolerance affects performance. In our method, the error tolerance parameter ϵ is a knob to control the amount of noise in pseudolabels. A common wisdom in pseudolabeling is that higher noise will lead to worse performance, which is our expectation. To see this, we run our method with $\epsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ in CIFAR-10 and CIFAR-100 settings, each with three random seeds, and report the results in Figure 3. The results are as expected — higher values of ϵ lead to degraded test accuracy due to high noise in the pseudolabels and with decreasing ϵ leads to improved accuracy. These results also suggest that prioritizing the quality (accuracy) of pseudolabels over quantity is a better choice in pseudolabeling. The results are also summarized in Table 7 and Table 8 in the Appendix.

To investigate the error tolerance further, we designed error tolerance scheduling using different error tolerances during various stages of SSL training. Table 3 summarizes the error tolerance we set at different iterations of SSL training and the corresponding top-1 accuracy for the CIFAR-10 and CIFAR-100 datasets. As we see, starting SSL with a small error tolerance and ending with a large tolerance severely impacts the performance on both CIFAR-10 and CIFAR-100 datasets. While our findings suggest a lower error tolerance is preferable, this may not hold in general. Nevertheless, our framework provides the flexibility to control this explicitly and thus can be tuned by practitioners for the setting at hand.

C3. Is pseudolabel accumulation helpful? Accumulation allows the methods to use old pseudolabel for points that couldn't get pseudolabeled in the current iteration. Thus, we expect accumulation to help improve the utilization of unlabeled data and lead to better test accuracy in cases where the pseudolabel quality is assured to be high in all iterations. We run two variations of our method and baselines — with and without accumulation and report the results in Table 4. We observe that our method has similar test accuracy irrespective of accumulation. However, accumulation achieves better coverage in early iterations, as observed in Figure 6 in Appendix C. These results are unsurprising since our method ensures high quality of pseudolabels while maximizing coverage; it can eventually catch up with the version using accumulation, leading to similar final test ac-

Method	Acc—True	Acc—False
Fixmatch	67.62 ± 2.10	90.08 ± 0.78
Fixmatch + MR	64.78 ± 4.64	90.41 ± 0.83
Fixmatch + BaM	68.10 ± 2.02	90.67 ± 0.90
Freematch	85.40 ± 1.36	92.26 ± 0.18
Freematch + MR	83.59 ± 2.59	92.17 ± 0.36
Freematch + BaM	85.48 ± 3.02	92.32 ± 0.25
Fixmatch + Ours	$\mathbf{92.69 \pm 0.74}$	$\mathbf{92.80 \pm 0.56}$
Freematch + Ours	$\mathbf{93.10 \pm 0.28}$	$\mathbf{91.80 \pm 1.08}$

Table 4. Results on CIFAR-10 with and without pseudolabel accumulation (Acc) for all the methods.

curacies. On the other hand, having accumulation hurts the performance of baseline models. This might be because the pseudolabels generated by the baseline models are inaccurate, especially in the earlier iterations, thus degrading the overall performance. Overall, we believe accumulation will be helpful when we have pseudolabels with high accuracy. The plots for pseudolabeling coverage and accuracy over the entire run are in Figures 6, 7 in Appendix C.

5. Related Work

Semi-supervised learning (SSL). There is a rich literature on SSL (Zhu, 2005; Chapelle et al., 2006; Singh et al., 2008; Oliver et al., 2018). This literature comprises of a wide variety of approaches. Among these, significant focus has been placed on self-training (also called pseudolabeling) (Scudder, 1965; Blum & Mitchell, 1998; Rosenberg et al., 2005; Lee, 2013; Oymak & Gulcu, 2020; Amini et al., 2023), generative models (Nigam et al., 2000; Adams & Ghahramani, 2009; Kingma et al., 2014), graph-based strategies (Blum & Chawla, 2001; Niyogi, 2013; Subramanya & Talukdar, 2022), and transductive approaches (Vapnik et al., 1998; Joachims, 1999). Due to their simplicity, pseudolabeling-based approaches have gained prominence and are widely used in application areas such as NLP (Karamanolakis et al., 2021), speech recognition (Kahn et al., 2020), and protein prediction (El-Manzalawy et al., 2016). Our paper focuses on recent variants of this, discussed next.

Pseudolabeling based SSL. These methods generate artificial labels for unlabeled data and use them for training the model. A crucial challenge here is the issue of confirmation bias (Arazo et al., 2020), i.e., when a model starts to reinforce its own mistakes. To overcome this and to maintain a high quality of pseudolabels, confidence-based thresholding is applied. Here, only the unlabeled data with confidence higher than a particular threshold is used (Sohn et al., 2020). Due to the limitations of fixed thresholds, adaptive thresholds based on the classifier's learning status have been introduced to improve performance (Xu et al., 2021; Zhang et al., 2021; Wang et al., 2023). Nearly all of these methods also use some form of consistency regularization (Laine & Aila, 2017; Bachman et al., 2014; Sajjadi et al., 2016; Fan et al., 2021; Kukačka et al., 2017) where the core idea is that the model should produce similar prediction when presented with different versions (perturbations) of inputs and all the present SSL methods (Xie et al., 2020; Wang et al., 2023; Sohn et al., 2020; Zhang et al., 2021; Chen et al., 2023; Xu et al., 2021).

Confidence functions and calibration. Miscalibration (overconfidence) in neural networks plagues various applications (Nguyen et al., 2015; Hendrycks & Gimpel, 2017; Guo et al., 2017), including SSL. To mitigate this in general, a range of solutions have been proposed, including training-time methods (Moon et al., 2020; Kumar et al., 2018; Hui et al., 2023; Corbière et al., 2019; Foret et al., 2021) and post-hoc methods (Guo et al., 2017; Kumar et al., 2019; Gupta & Ramdas, 2022; Kull et al., 2019; Zadrozny & Elkan, 2002). In pseudolabeling based SSL, recent works (Rizve et al., 2021; Loh et al., 2023; Mishra et al., 2024) noted the issue of miscalibration. To promote calibration, Loh et al. (2023) use Bayesian neural nets by replacing the model's final layer with a Bayesian layer. Rizve et al. (2021) utilize negative labels and an uncertainty-aware pseudolabel selection technique. Mishra et al. (2024) incorporate a regularizer to encourage calibration.

While calibration is a reasonable goal in general, it may not be sufficient to address the overconfidence problem in SSL and other applications. In pseudolabeling, we seek the use of scores that can easily segregate the model's correct and incorrect predictions, which is closely related to the ordinal ranking criterion (Hendrycks & Gimpel, 2017; Moon et al., 2020; Foret et al., 2021; Corbière et al., 2019). Rather than experimenting with several such choices, ideally, we would have a flexible framework that can learn confidence functions explicitly optimizing pseudolabeling objectives.

6. Conclusion

Common semi-supervised learning (SSL) methods rely on pseudolabeling, but their effectiveness is limited by unreliable confidence scores and heuristic thresholding strategies. We address these issues by introducing a principled framework for learning confidence scores and thresholds with explicit control over pseudolabeling error. We adapt existing SSL methods with this framework and empirically show that the adapted methods achieve a higher test accuracy compared to their standard versions. Additionally, we introduce pseudolabel accumulation and analyze its impact, showing that it benefits methods with reliable pseudolabels, such as those using our framework. In sum, by providing a principled, data-driven approach to obtaining scores and thresholds for pseudolabeling, our work enhances SSL methods and opens the door to more reliable and efficient pseudolabeling-based SSL.

7. Impact Statement

This research improves semi-supervised learning, enabling more accurate and efficient machine learning in settings where labeled data is hard to obtain by following first principles in designing thresholds and confidence functions. Our work has various potential societal implications, with no specific concerns that require special attention in this context.

References

Adams, R. P. and Ghahramani, Z. Archipelago: nonparametric bayesian semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1–8, 2009.

Amini, M.-R., Feofanov, V., Pauletto, L., Hadjadj, L., Devijver, E., and Maximov, Y. Self-training: A survey, 2023.

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., and McGuinness, K. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2020.

Bachman, P., Alsharif, O., and Precup, D. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Blum, A. and Chawla, S. Learning from labeled and unlabeled data using graph mincuts. 2001.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proc. of the eleventh annual conference on Computational learning theory*, pp. 92–100. ACM, 1998.

Chapelle, O., Schölkopf, B., and Zien, A. (eds.). *Semi-Supervised Learning*. The MIT Press, 2006. ISBN 9780262033589.

Chen, H., Tao, R., Fan, Y., Wang, Y., Wang, J., Schiele, B., Xie, X., Raj, B., and Savvides, M. Softmatch: Addressing the quantity-quality tradeoff in semi-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ymt1zQXBDiF.

Corbière, C., THOME, N., Bar-Hen, A., Cord, M., and Pérez, P. Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems 32*, pp. 2902–2913. 2019.

Djurisic, A., Bozanic, N., Ashok, A., and Liu, R. Extremely simple activation shaping for out-of-distribution detection. In *The Eleventh International Conference on Learning Representations*, 2023.

El-Manzalawy, Y., Munoz, E. E., Lindner, S. E., and Honavar, V. Plasmosep: Predicting surface-exposed proteins on the malaria parasite using semisupervised self-training and expert-annotated data. *Proteomics*, 16(23): 2967–2976, 2016.

Fan, Y., Kukleva, A., and Schiele, B. Revisiting consistency regularization for semi-supervised learning, 2021.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Gupta, C. and Ramdas, A. Top-label calibration and multiclass-to-binary reductions. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WqoBaaPHS-.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Hui, L., Belkin, M., and Wright, S. Cut your losses with squentropy. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 14114–14131, 2023.

Joachims, T. Transductive inference for text classification using support vector machines. In Bratko, I. and Dzeroski, S. (eds.), *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pp. 200–209, 1999.

Kahn, J., Lee, A., and Hannun, A. Self-training for end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7084–7088. IEEE, 2020.

Karamanolakis, G., Mukherjee, S., Zheng, G., and Awadallah, A. H. Self-training with weak supervision. *arXiv preprint arXiv:2104.05514*, 2021.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kukačka, J., Golkov, V., and Cremers, D. Regularization for deep learning: A taxonomy, 2017.

Kull, M., Perello Nieto, M., Kängsepp, M., Silva Filho, T., Song, H., and Flach, P. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Kumar, A., Sarawagi, S., and Jain, U. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2805–2814. PMLR, 10–15 Jul 2018.

Kumar, A., Liang, P. S., and Ma, T. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *Fifth International Conference on Learning Representations*, 2017.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

Li, M., Wu, R., Liu, H., Yu, J., Yang, X., Han, B., and Liu, T. Instant: Semi-supervised learning with instance-dependent thresholds. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Loh, C., Dangovski, R., Sudalairaj, S., Han, S., Han, L., Karlinsky, L., Soljacic, M., and Srivastava, A. On the importance of calibration in semi-supervised learning, 2022.

Loh, C., Dangovski, R., Sudalairaj, S., Han, S., Han, L., Karlinsky, L., Soljacic, M., and Srivastava, A. Mitigating confirmation bias in semi-supervised learning via efficient bayesian model averaging. *Transactions on Machine Learning Research*, 2023.

McLachlan, G. J. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

Mishra, S., Murugesan, B., Ayed, I. B., Pedersoli, M., and Dolz, J. Do not trust what you trust: Miscalibration in semi-supervised learning, 2024.

Moon, J., Kim, J., Shin, Y., and Hwang, S. Confidence-aware learning for deep neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 7034–7044, 2020.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 7. Granada, Spain, 2011.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39:103–134, 2000.

Niyogi, P. Manifold regularization and semi-supervised learning: Some theoretical analyses. *Journal of Machine Learning Research*, 14(5), 2013.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Oymak, S. and Gulcu, T. C. Statistical and algorithmic insights for semi-supervised learning with self-training. *arXiv preprint arXiv:2006.11006*, 2020.

Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021.

Rosenberg, C., Hebert, M., and Schneiderman, H. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, volume 1, pp. 29–36, 2005. doi: 10.1109/ACVMOT.2005.107.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 1171–1179, 2016.

Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

Singh, A., Nowak, R., and Zhu, J. Unlabeled data: Now it helps, now it doesn't. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Subramanya, A. and Talukdar, P. P. *Graph-based semi-supervised learning*. Springer Nature, 2022.

van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, 109:373 – 440, 2019.

Vapnik, V. N., Vapnik, V., et al. Statistical learning theory. 1998.

Vishwakarma, H., Lin, H., Sala, F., and Vinayak, R. K. Promises and pitfalls of threshold-based auto-labeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Wang, Y., Chen, H., Fan, Y., Sun, W., Tao, R., Hou, W., Wang, R., Yang, L., Zhou, Z., Guo, L.-Z., Qi, H., Wu, Z., Li, Y.-F., Nakamura, S., Ye, W., Savvides, M., Raj, B., Shinozaki, T., Schiele, B., Wang, J., Xie, X., and Zhang, Y. Usb: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022.

Wang, Y., Chen, H., Heng, Q., Hou, W., Fan, Y., Wu, Z., Wang, J., Savvides, M., Shinozaki, T., Raj, B., Schiele, B., and Xie, X. Freematch: Self-adaptive thresholding for semi-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PDrUPTXJI_A.

Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Xu, Y., Shang, L., Ye, J., Qian, Q., Li, Y.-F., Sun, B., Li, H., and Jin, R. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pp. 11525–11536. PMLR, 2021.

Zadrozny, B. and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 694–699, 2002.

Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., and Shinozaki, T. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34: 18408–18419, 2021.

Zhu, X. Semi-supervised learning literature survey. In *University of Wisconsin-Madison, Department of Computer Sciences*, 2005.

Supplementary Material

The supplementary material is organized as follows. First, we summarize the notations in Table 5 in Appendix A, then we provide formal algorithms in Appendix B and additional experimental results and details are provided in Appendix C.

A. Glossary

The notations are summarized in Table 5 below.

B. Detailed Algorithms

Algorithm 1 Estimate Pseudolabeling Thresholds Classwise

Input: Confidence function $\widehat{g}_i$, classifier $\widehat{h}_i$, Part of validation data $D_{\text{th}}^{(i)}$ for threshold estimation, pseudolabeling error tolerance ϵ, space of thresholds T, label space $\mathcal{Y}$.
Output: Pseudolabeling thresholds $\widehat{\mathbf{t}}_i$, where $\widehat{\mathbf{t}}_i[y]$ is the threshold for class y.
for $y \in \mathcal{Y}$ **do**
  # Extract the set of points $D_{\text{th}}^{(y)}$ for which the groundtruth class is y.
  $D_{\text{th}}^{(y)} \leftarrow \{(\mathbf{x}', y') \in D_{\text{th}} : y' = y\}$
  $T'_y \leftarrow T \cup \{\infty\}$.
  # Estimate pseudolabeling error at each threshold on class specific data $D_{\text{th}}^{(y)}$. Pick the smallest threshold with the sum of the estimated error and C_1 times the std. deviation is below ϵ. Here C_1 is set to 0.25 and $\widehat{\sigma}(z) = \sqrt{z(1-z)}$.
  $\widehat{\mathbf{t}}_i[y] \leftarrow \min\{t \in T'_y : \widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}^{(y)}) + C_1 \widehat{\sigma}(\widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}^{(y)})) \leq \epsilon\}$,
end for
return $\widehat{\mathbf{t}}_i$

Algorithm 2 Estimate Pseudolabeling Threshold Jointly for All Classes

Input: Confidence function $\widehat{g}_i$, classifier $\widehat{h}_i$, Part of validation data $D_{\text{th}}^{(i)}$ for threshold estimation, pseudolabeling error tolerance ϵ, space of thresholds T, label space $\mathcal{Y}$.
Output: Pseudolabeling thresholds $\widehat{\mathbf{t}}_i$, where $\widehat{\mathbf{t}}_i[y]$ is the threshold for class y.
$T' \leftarrow T \cup \{\infty\}$
Estimate pseudolabeling error at each threshold on the entire set D_{th}. Pick the smallest threshold with the sum of the estimated error and C_1 times $\widehat{\sigma}$ is below ϵ. Here C_1 is set to 0.25 and $\widehat{\sigma}(z) = \sqrt{z(1-z)}$.
$t \leftarrow \min\{t \in T' : \widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}) + C_1 \widehat{\sigma}(\widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}})) \leq \epsilon\}$.
for $y \in \mathcal{Y}$ **do**
  $\widehat{\mathbf{t}}_i[y] \leftarrow t$
end for
return $\widehat{\mathbf{t}}_i$

Algorithm 3 Pseudolabeling Based SSL with `PabLO`

Input: Labeled data for training D_l, validation data D_{val}, unlabeled pool X_u, error tolerance ϵ, use-accumulation flag, num_iters, batch size B, replication factor μ, weak ω and strong Ω augmentations, number of calibration points N_{cal}, num. of threshold estimation points N_{th}, frequency of invoking `PabLO` F, space of thresholds T, label space $\mathcal{Y}$.

Output: $\widehat{h}_{\text{ssl}}$, model with the best validation accuracy.

```
# Set initial pseudolabels and masks to 0.
Ỹ_u^(0) ← [0, 0, ..., 0], S_u^(0) ← [0, 0, ..., 0], i ← 1
# Draw calibration and threshold estimation sets from D_val.
D_cal, D_th ← DrawRandomly(D_val, N_cal, N_th).
# Training loop with pseudolabeling.
while i ≤ num_iters do
  # Draw batches D_l^b, X_u^b of labeled and unlabeled points, I_u^b denotes the indices corresponding to points in X_u^b.
  D_l^b, X_u^b, I_u^b ← DrawRandomBatch(μD_l, μX_u, B)
  # Create weak and strong augmentations of X_u^b.
  X_{u,w}^b, X_{u,s}^b ← ω(X_u^b), Ω(X_u^b)
  /** Begin Pseudolabeling Block **/
  # Perform pseudolabeling using PabLO.
  if i%F = 0 then
    # Get ĝ_i by solving optimization (P1).
    ĝ_i, t̂_i' ← SolveOptProblemP1(ĥ_i, D_cal)
    # Estimate pseudolabeling thresholds.
    if estimate threshold classwise then
      # Use Algorithm 1.
      t̂_i ← ClasswiseThreshold(ĝ_i, ĥ_i, D_th, ε, T, 𝒴)
    else
      # Use Algorithm 2.
      t̂_i ← JointThreshold(ĝ_i, ĥ_i, D_th, ε, T, 𝒴)
    end if
    # Compute fresh psuedolabels Ỹ_u^(i) and pseudolabeling masks S_u^(i) for all points in X_u.
    Ỹ_u^(i) ← ĥ_i(ω(X_u)),   S_u^(i) ← 𝟙(ĝ_i(ω(X_u)) ≥ t̂)
    if use-accumulation then
      # Apply pseudolabel accumulation if enabled.
      Ỹ_u^(i) ← S_u^(i) Ỹ_u^(i) + (1 − S_u^(i)) Ỹ_u^(i−1)
      S_u^(i) ← S_u^(i) ∨ S_u^(i−1)
    end if
  else
    ĝ_i, t̂_i = ĝ_{i−1}, t̂_{i−1}
  end if
  /** End Pseudolabeling Block **/
  # Extract pseudolabels and masks for the current unlabeled batch. Then compute supervised and unsupervised losses.
  Ỹ_u^b, S_u^b ← Ỹ_u[I_u^b],  S_u[I_u^b]
  ℒ̂_s(ĥ_i) ← supervised_loss(h, D_l^b)
  ℒ̂_u(ĥ_i) ← unsupervised_loss(h, X_{u,w}^b X_{u,s}^b, Ỹ_u^b, S_u^b)
  ℒ̂(ĥ_i) ← ℒ̂_s(ĥ_i) + λ_u ℒ̂_u(ĥ_i)
  # Perform a gradient descent step to get new model ĥ_{i+1}.
  ĥ_{i+1} ← SGD_update(ℒ̂(ĥ_i));   i ← i + 1
  # Evaluate model on D_val to keep track of the best model.
  if i%eval_freq = 0 then
    eval_acc ← evaluate_model(ĥ_i, D_val)
    If eval_acc is best so far then ĥ_ssl = ĥ_i
  end if
end while
```

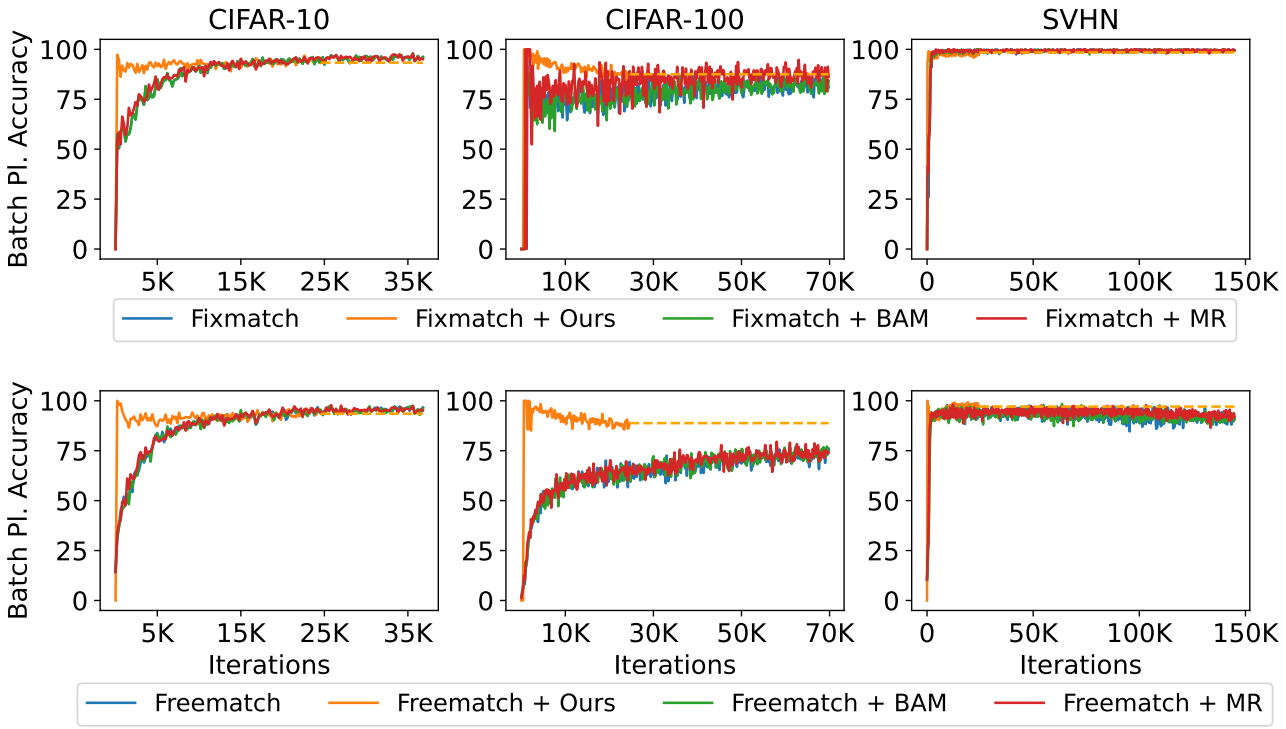

Figure 4. Batch pseudolabel accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

C. Additional Experiments and Details

Compute. We ran all of our experiments on a high-throughput system with various GPUs. Therefore, each individual experiment task may be scheduled among NVIDIA A100 SXM4-40GB, NVIDIA A100 SXM4-80GB, NVIDIA L40, and NVIDIA H100 80GB HBM3. We measured the runtime of our algorithm on a desktop with a single NVIDIA RTX 4090. On CIFAR-10, it took about 0.203 seconds for each iteration for our method and around 0.140 seconds for the baselines. On CIFAR-100, it took about 0.396 seconds for each iteration for our method and around 0.143 seconds for the baselines. On SVHN, it took about 1.275 seconds for each iteration for our method and around 0.225 seconds for the baselines.

Hyperparameters. For the baselines, we have used their default settings. To maintain consistency and experiment the efficiency of method, we used WRN-28-2 which is 1.4M parameter model for all the datasets. We summarize the main hyperparameters we have used in our method in Table 9.

Note that the number of epochs we used to train the function g and to estimate $\mathbf{t}$ is dynamic. That is, its actual value depends on and is proportional to the current number of iterations of the SSL training. More concretely, at iteration i of SSL training, we use $\min(\lfloor i/25 \rfloor, \text{max epoch})$ number of epochs to find g and $\mathbf{t}$.

We additionally conduct the following ablation study to study our technique's dependence on the amount of data used in learning g and thresholds.

A2. How much data is needed to learn the g and $\mathbf{t}$? We take N_{cal} and N_{th} from the validation data to learn the confidence function g and estimate the thresholds $\mathbf{t}$ respectively. Intuitively larger values of these should lead to good g and $\mathbf{t}$ that can extract the expected level of pseudolabeling coverage and accuracy from the classifier at hand. However, the task of learning good g and estimating thresholds is not super hard and we expect it will take fewer samples to be successful. To understand this better we run our method with N_{cal} and N_{th} in $\{250, 500, 750, 1000\}$ on CIFAR-10 setting for 3 random seeds and report the result in Fig 8. We observe that our method can achieve desired performance with just 500 labeled points (i.e 50 labels per class). This is interesting because we can achieve 90% accuracy by just using 250 points (N_l) for training h and a total of 1K for learning g. Refer to Table 6 for more details.

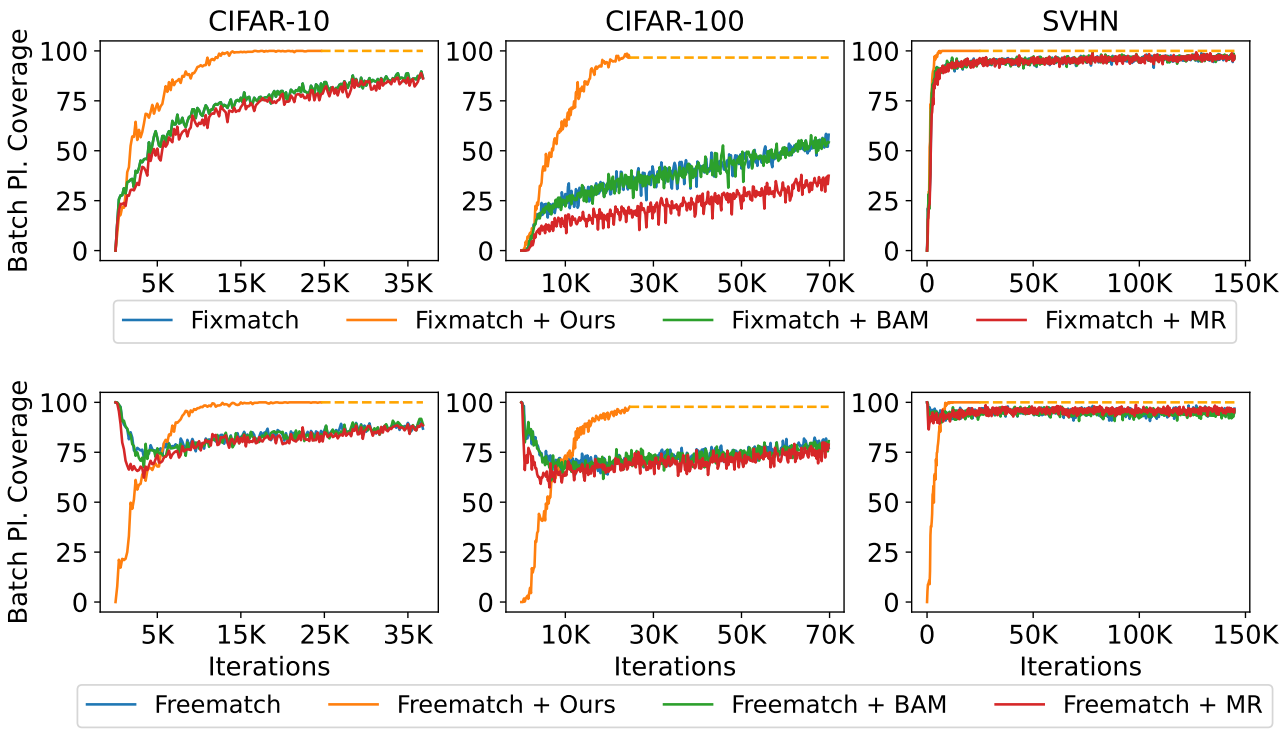

Figure 5. Batch pseudolabel coverage of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

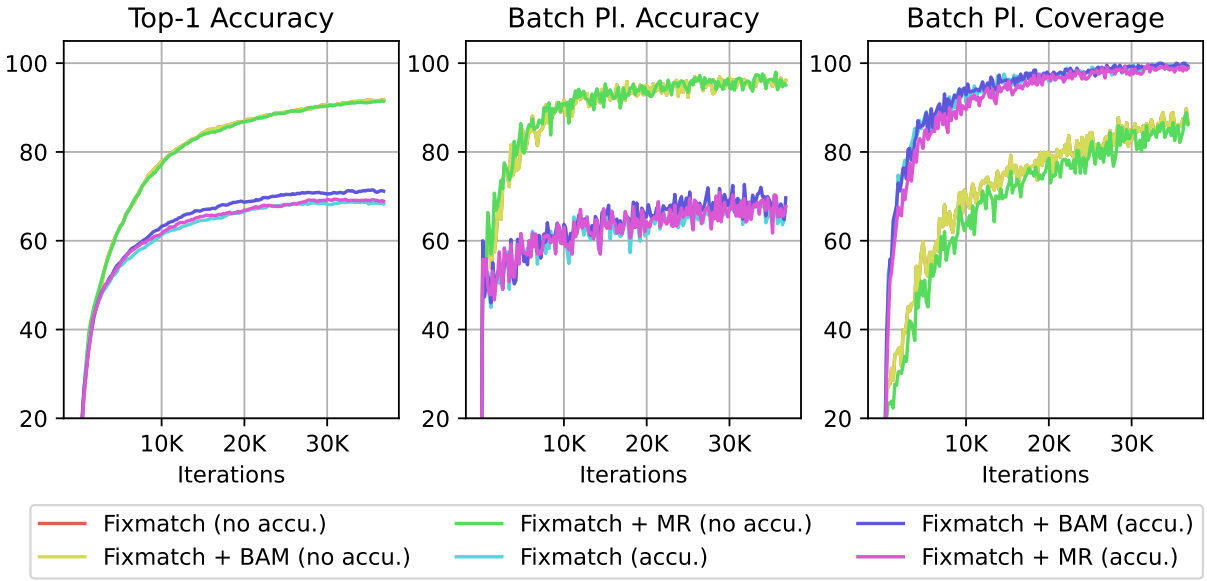

Figure 6. **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Fixmatch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsen the performance of baseline methods in terms of accuracy and coverage.

Symbol	Definition
$\mathcal{X}$	feature space.
$\mathcal{Y}$	label space i.e. $1, 2, \dots k$.
$\mathcal{H}$	hypothesis space (model class for the classifiers).
$\mathcal{G}$	space of confidence functions.
k	number of classes.
$\mathbf{x}, y$	$\mathbf{x}$ is a datapoint in $\mathcal{X}$ and y is its true label (if available).
h	a model $h : \mathcal{X} \to \mathcal{Y}$.
g	confidence function $g : \mathcal{X} \to T^k \subseteq \mathbb{R}^k$
$\widehat{y}$	hard label prediction.
$\widetilde{y}$	$\widehat{y}$ is used as pseudolabel.
$\widehat{h}_{\text{ssl}}$	a best learned model using SSL.
ϵ	pseudolabeling error tolerance.
$g_i^\star$	optimal confidence function at i iteration.
$\mathbf{t}_i^*$	optimal threshold at i iteration.
X_u	available unlabeled data drawn from the distribution P_x over $\mathcal{X}$.
X_u^b	batch of unlabeled data.
D_l	set of labeled data points drawn from the distribution P_{xy}.
D_l^b	batch of labeled data.
D_{val}	validation data.
D_{cal}	calibration data; part of validation data used to optimize surrogate functions.
D_{th}	part of validation data to estimate threshold $\mathbf{t}$.
$\mathbf{t}$	k dimensional vector of thresholds representing for k classes.
$\mathbf{t}[y]$	yth entry of $\mathbf{t}$ i.e. the threshold for class y.
n_u	number of unlabeled points, i.e. size of X_u used for consistency regularization and pseudolabeling.
N_l	number of labeled points, i.e. size of D_l. Usual SSL setting has, $N_l \ll n_u$.
N_{val}	number of points used for model selection.
N_{test}	number of test data points.
N_{cal}	number of points used for learning the g function.
N_{th}	number of data points used for threshold estimation.
$\widehat{\mathcal{L}}_s$	supervised loss.
$\widehat{\mathcal{L}}_u$	unsupervised loss with weighted importance λ_u.
$\widehat{\mathcal{L}}_r$	sum of regularization terms for supervised and unsupervised loss with weighted importance λ_r.
$H(y, h, \mathbf{x})$	standard cross-entropy loss.
$S(\mathbf{x}, g, \mathbf{t} \mid h)$	pseudolabeleing mask.
ω	weak transformation, $\omega : \mathcal{X} \mapsto \mathcal{X}$.
Ω	strong transformation, $\Omega : \mathcal{X} \mapsto \mathcal{X}$.
α_o, α_b	average time taken by our method and baseline methods. These are used for adjusted iterations for baselines.
$\widehat{\mathcal{P}}(g, \mathbf{t} \mid h, X)$	estimated pseudolabeling coverage, see eq. (1).
$\mathcal{P}(g, \mathbf{t} \mid h)$	population level pseudolabeling coverage, see eq. (2).
$\widehat{\mathcal{E}}(g, \mathbf{t} \mid h, D)$	estimated pseudolabeling error, see eq. (3).
$\mathcal{E}(g, \mathbf{t} \mid h)$	population level pseudolabeling error, see eq. (4).
$\widetilde{\mathcal{P}}(g, \mathbf{t} \mid h, D)$	surrogate estimated pseudolabeling coverage, see eq. (6).
$\widetilde{\mathcal{E}}(g, \mathbf{t} \mid h, D)$	surrogate estimated pseudolabeling error, see eq. (7).
λ	hyperparamter controlling the importance of pseudolabeleing coverage and error in (P1).

Table 5. Glossary of variables and symbols used in this paper.

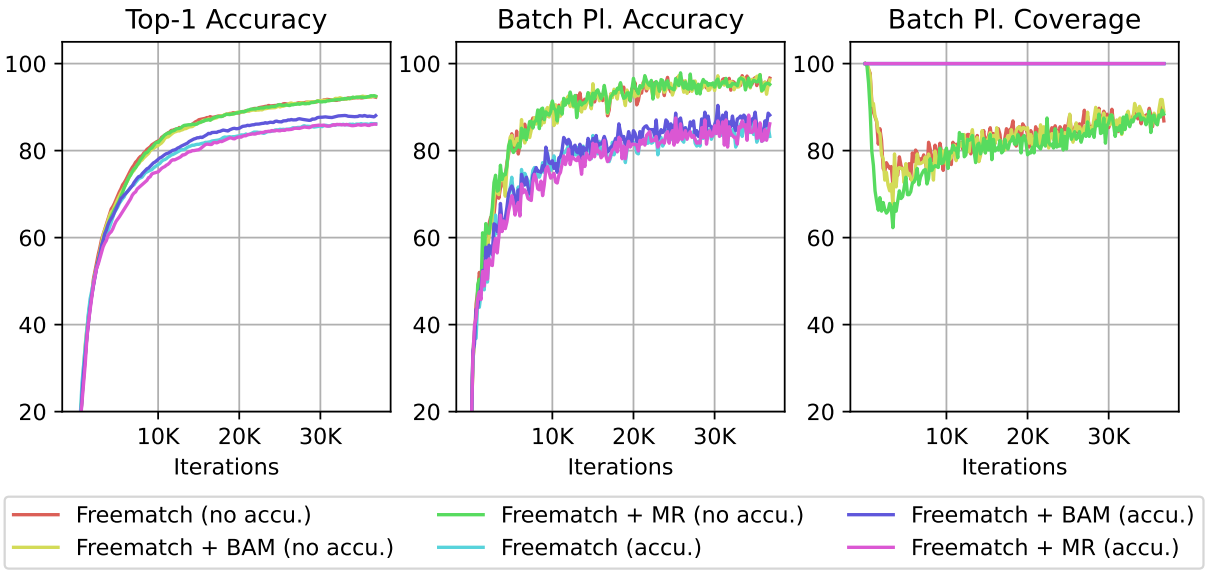

Figure 7. **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Freematch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsen the performance of baseline methods in terms of accuracy and coverage.

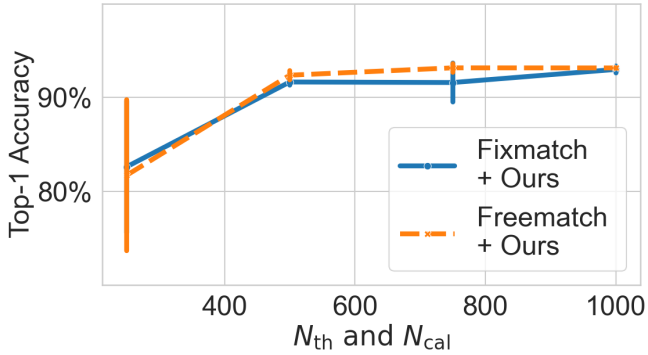

Figure 8. Top-1 accuracy of our method with different N_{th} and N_{cal}.

Method	$N_{\text{cal}} = N_{\text{th}} = 250$	$N_{\text{cal}} = N_{\text{th}} = 500$	$N_{\text{cal}} = N_{\text{th}} = 750$
Fixmatch + Ours	82.67 ± 7.08	91.74 ± 0.41	91.66 ± 2.11
Freematch + Ours	82.13 ± 7.93	92.33 ± 0.49	93.20 ± 0.53

Table 6. Results on CIFAR-10 with varying N_{cal} and N_{th}.

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Fixmatch + Ours	93.05 ± 0.54	91.54 ± 0.95	88.35 ± 2.90	56.72 ± 22.25
Freematch + Ours	92.11 ± 1.18	92.31 ± 0.16	83.89 ± 10.36	52.17 ± 25.36

Table 7. Results on CIFAR-10 with varying ϵ.

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Fixmatch + Ours	69.19 ± 1.13	65.01 ± 0.34	53.88 ± 8.15	23.58 ± 18.21
Freematch + Ours	70.13 ± 0.67	64.95 ± 1.41	59.83 ± 1.32	24.09 ± 17.22

Table 8. Results on CIFAR-100 with varying ϵ.

Method	**Hyperparameter**	**Values**
Learning g function	optimizer	SGD
	learning rate	0.01
	batch size	64
	max epoch	500
	weight decay	0.01
	momentum	0.9
Estimating $\mathbf{t}$	optimizer	SGD
	learning rate	0.01
	batch size	64
	max epoch	500
	weight decay	0.01
	momentum	0.9

Table 9. Hyperparameters used for our method.