PabLO: Improving Semi-Supervised Learning with Pseudolabeling Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modern semi-supervised learning (SSL) methods frequently rely on pseudolabeling and consistency regularization. The main technical challenge in pseudolabeling is identifying the points that can reliably be labeled. To address this challenge we propose a framework to learn confidence functions and thresholds explicitly aligned with the SSL task, obviating the need for manual designs. Our approach formulates an optimization problem over a flexible space of confidence functions and thresholds, allowing us to obtain optimal scoring functions—while remaining compatible with the most popular and performant SSL techniques today. Extensive empirical evaluation of our method shows up to 11% improvement in test accuracy over the standard baselines while requiring substantially fewer training iterations.

1 Introduction

Obtaining high-quality labeled data is a major bottleneck in machine learning. The semi-supervised learning (SSL) paradigm tackles this problem by training models on a small amount of labeled data and a large quantity of unlabeled data [7, 57, 47]. Modern SSL methods frequently rely on a pair of ideas: pseudolabeling [29, 2, 40, 27, 39] and consistency regularization [26, 4, 41, 11, 22]. SSL techniques marrying these ideas have delivered strong performance on a number of benchmark datasets. The main challenge with pseudolabeling is balancing accurate point selection with efficient model training. A promising solution is a framework that learns confidence functions and thresholds *explicitly aligned* with the SSL task, eliminating the need for manual experimentation. Inspired by threshold-based auto-labeling (TBAL) [50], a data development technique, we propose a framework that adapts TBAL principles to learn confidence functions and thresholds specifically for pseudolabeling-based SSL.

Our approach involves two aspects. First, we formulate an optimization problem over a flexible space of confidence functions and thresholds to optimize the quantity/quality tradeoff in pseudolabeling. The space we optimize over is broad enough to subsume many existing manually-designed approaches. That is, we *learn confidence functions and thresholds*. Second, we develop strategies to make the framework compatible with SSL approaches. Experimentally, we couple our framework to some of the most prominent SSL techniques in use today, including Fixmatch [45] and Freematch [52]. We observed accuracy lifts of up to 11%, 6%, and 3% on popular benchmarks like SVHN, CIFAR-10, and CIFAR-100 respectively, along with substantial improvements in convergence speed.

2 Background and Problem Setup

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

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

$\hat{y} := h(\mathbf{x})$. When the prediction $\hat{y}$ 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 score for class y.

2.1 Pseudolabeling-based Semi-Supervised Learning

Given 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 $\hat{h}_{\text{ssl}}$ 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, $\widehat{\mathcal{L}}_s(h|D_l^b) = \frac{1}{|D_l^b|}\sum_{(x,y)\in D_l^b} H(y,h,\mathbf{x})$. 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 $\tilde{y} = h(\mathbf{x})$ are computed for each $\mathbf{x} \in X_u^b$. Then, a pseudlabeling mask $S(\mathbf{x}, g, \mathbf{t} \mid h) = \mathbb{1}(g(\mathbf{x})[\tilde{y}] \geq \mathbf{t}[\tilde{y}])$, is 1 for points having confidence score bigger than predetermined threshold corresponding to the predicted class. Recent methods, couple this loss and consistency regularization together by doing pseudolabeling on weakly augmented data using weak transform ω and then defining the cross-entropy loss on the strongly augmented data using strong transformation Ω. The loss is

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

2.2 Problem Statement

The success of pseudolabeling-based SSL hinges heavily on maximizing the quality and quantity of the pseudolabels. These are defined as follows:

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

$$\widehat{\mathcal{P}}(g, \mathbf{t} \mid h, X) := \frac{1}{|X|}\sum_{(\mathbf{x})\in X} S(\mathbf{x}, g, \mathbf{t} \mid h)\,, \quad \mathcal{P}(g, \mathbf{t}|h) := \mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h)]. \tag{1}$$

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

$$\widehat{\mathcal{E}}(g, \mathbf{t} \mid h, D) := \frac{\sum_{(\mathbf{x},y,\tilde{y})\in D} S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)}{\sum_{(\mathbf{x},y,\tilde{y})\in D} S(\mathbf{x}, g, \mathbf{t} \mid h)}, \tag{2}$$

$$\mathcal{E}(g, \mathbf{t} \mid h) = \frac{\mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)]}{\mathcal{P}(g, \mathbf{t}|h)}. \tag{3}$$

Goal. We want to learn a classifier $\hat{h}_{\text{ssl}}$ that generalizes well on the unseen data.

3 Methodology

Our approach integrates learnable confidence functions and thresholds into existing pseudolabeling-based SSL pipelines. To do so, we build on a recently-developed technique [50] to improve the performance of threshold-based auto-labeling (TBAL) [43, 49, 38] systems. In order to make such an approach compatible with SSL, we apply a simple notion—*accumulating pseudolabels*—that may also be useful for other methods.

3.1 Pseudolabeling Optimization Framework

The fundamental problem in pseudolabeling is, given a classifier $\hat{h}_i$, to correctly identify the points in the pool of unlabeled data X_u where the predictions of $\hat{h}_i$ are correct. Since the classifier is frequently undertrained during the SSL process, it may not have high accuracy. That is, it might only be accurate in some small part of the feature space, which we hope to identify via the confidence scores and appropriate thresholds. As discussed earlier, existing solutions [27, 45, 52] use maximum softmax probability (MSP) from the model $\hat{h}_i$ in concert with heuristics for thresholds that are either fixed or vary dynamically based on the learning status of the model. Some recent works have observed that MSP scores tend to be miscalibrated and proposed solutions to obtain more calibrated scores [30, 28], which also led to performance gains.

Theoretical Framework. 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. (1)) while keeping the pseudolabeling error low (eq. (3)) i.e. have high quality. More specifically, one approach to formalizing this optimization problem is to seek to maximize the pseudolabeling coverage while ensuring pseudolabeling error is at most $\epsilon \in (0,1)$, for some hyperparameter ϵ. In other words, given the classifier $\hat{h}_i$ in any iteration i of SSL, then,

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

are the optimal confidence functions and thresholds for pseudolabeling using $\hat{h}_i$'s predictions. The *quality* of the pseudolabels can be controlled using ϵ. This follows the recipe for TBAL [50], with one additional complication: for SSL, it is not clear what value of ϵ is suitable, while in TBAL ϵ is a system-level constant provided as input.

The most attractive property of this framework is that, irrespective of the choice of ϵ, it provides the scores and threshold that yield maximum pseudolabeling coverage at that error level, freeing us from making arbitrary choices of confidence scores, calibration techniques, and thresholding heuristics. Instead, we solve the optimization problem over a flexible enough space will subsume specific strategies. We defer the discussion of making the framework practical into Appendix B.

3.2 Threshold Estimation

While we can obtain both the confidence scores and thresholds by solving (P1), we propose to adapt the threshold estimation procedure from [50] as it avoids potential generalization issues due to learning them simultaneously from the same data D_{cal} and ensures stricter control over the pseudolabeling errors. It is also decoupled from any particular choice of scoring function, hence it can replace the thresholding procedure in the existing SSL pipelines as well.

Our procedure is simple. It takes in a confidence function $\tilde{g}_i$ and another part of the held-out validation data referred to as D_{th}. It estimates thresholds for each class separately and estimates the pseudolabeling errors $\widehat{\mathcal{E}}(\tilde{g}_i, t \mid h, D_{\text{th}}, \tilde{y})$ on the super level sets of $\tilde{g}_i$. Here we slightly abuse notation: instead of $\mathbf{t} \in T^k$, we use $t \in T$, to indicate the estimate of pseudolabeling error at threshold t for class y. To obtain a threshold $\tilde{\mathbf{t}}[y]$ for class y, the procedure finds the smallest $t \in T$ such that $\widehat{\mathcal{E}}(\tilde{g}_i, t \mid h, D_{\text{th}}, \tilde{y}) + C_1\hat{\sigma}(\widehat{\mathcal{E}}) \leq \epsilon$. Here C_1 is a constant and $\hat{\sigma}(z) = \sqrt{z \cdot (1-z)}$ and $\widehat{\mathcal{E}}$ is used for brevity in place of $\widehat{\mathcal{E}}(\tilde{g}_i, t \mid h, D_{\text{th}}, \tilde{y})$. Using the thresholds found using this procedure ensures pseudolabeling error remains below (or close to) the a tolerance level ϵ. We refer to our method as `PabLO` . A more formal listing of the steps is detailed in Algorithm 1, deferred to Appendix B due to space constraints.

4 Experiments

We evaluate our method empirically to verify the following claims: **C1.** Our method produces models with improved test accuracy while taking fewer iterations. **C2.** In certain cases, we may wish to produce a high-quality dataset using pseudolabeling (rather than a single high-quality model). For such scenarios, `PabLO` achieves much higher dataset coverage and accuracy. Additionally, we conduct ablation studies, deferred to the Appendix C.

4.1 Experimental Setup

Methods. We use two simple base methods capturing the core ideas of pseudolabeling (PL) and consistency regularization (CR). The first is *Fixmatch* [45] which uses fixed thresholds on MSP scores

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

Dataset	CIFAR-10	CIFAR-100	SVHN
# Labels	250	2500	250
Fixmatch	88.15 ± 1.27	50.07 ± 1.12	96.54 ± 0.05
Fixmatch + MR	87.85 ± 1.10	44.75 ± 1.36	96.58 ± 0.04
Fixmatch + BaM	86.44 ± 1.47	44.58 ± 0.41	95.99 ± 0.06
Fixmatch + Ours	$\mathbf{93.03 \pm 0.44}$	$\mathbf{53.17 \pm 1.27}$	$\mathbf{96.61 \pm 0.16}$
Freematch	90.17 ± 0.13	57.21 ± 0.78	85.25 ± 1.70
Freematch + MR	90.17 ± 0.45	57.23 ± 1.18	84.65 ± 1.03
Freematch + BaM	88.34 ± 0.99	51.98 ± 1.74	86.28 ± 1.75
Freematch + Ours	$\mathbf{93.08 \pm 0.05}$	$\mathbf{60.96 \pm 0.53}$	$\mathbf{96.48 \pm 0.33}$

for PL along with CR. *Freematch* [52] 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)* [28] and *Margin Regularization (MR)*[1] [30] 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).

Datasets. We experiment with 3 datasets: *CIFAR-10* [21], *CIFAR-100* [21] and *SVHN* [32]. More details are summarized in Table 2 in Appendix C. We use a portion of the validation data (N_{val}) for our method, split into N_{cal}, used to calibrate the function g, and N_{th}, used to estimate the threshold.

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 [51]. We run till 25K iterations—in contrast to the extremely large number of iterations (2^{20}) in prior works—which may be unrealistic in practice due to resource constraints. For confidence functions class $\mathcal{G}$, we use a class of 2-layer neural nets and provide its last two layers representations from h as input, as in [49]. We use $\epsilon = 5\%$ across all settings. More experimental details are deferred to Appendix C.

4.2 Results and Discussion

C1. Test accuracy improvements. Our method maximizes pseudolabeling coverage and accuracy, producing more accurate pseudolabels. As Table 1 shows, integrating our method into Fixmatch and Freematch significantly improves test accuracy on CIFAR-10, CIFAR-100, and SVHN. Notably, we see a 6% improvement on CIFAR-10 with Fixmatch, a 3% improvement on the harder CIFAR-100 with Fixmatch, and an 11% improvement on SVHN with Freematch.

C2. Improved pseudolabeling coverage and accuracy. As our method is designed to maximize coverage and accuracy of pseudolabels, we expect high pseudolabeling accuracy and coverage from the beginning. To test this, we log the pseudolabeling coverage and accuracy in each iteration on the batch of unlabeled data used in that iteration. We refer to these as batch pseudolabeling coverage (batch-pl-cov) and batch pseudolabeling accuracy (batch-pl-acc). We show these for CIFAR-10 and CIFAR-100 settings in Figure 1 and 2 in the Appendix. As expected, the batch-pl-acc is high right from the beginning and it is close to the desired level of 95% (with $\epsilon = 5\%$) throughout for CIFAR-10. However, for CIFAR-100 possibly due to high class cardinality it drops to around 70%, This is similar to the baselines but yields much higher coverage. Similar results hold for SVHN (Figure 3).

5 Conclusion

We built a framework, inspired by ideas from autolabeling, that learns confidence functions and thresholds explicitly aligned with the SSL task. This approach eliminates the need for manual designs and hand-crafted notions of confidence, which can be limited in specialized data settings. By formulating an optimization problem over a flexible space of confidence functions and thresholds, we characterized optimal scoring functions. We derived our practical method to learn the scores and evaluated it empirically, where it achieved up to 11% improvement in test accuracy over standard baselines, while also reducing training iterations.

[1] We assign this name for convenience.

References

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

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

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

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

[5] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

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

[7] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.

[8] H. Chen, R. Tao, Y. Fan, Y. Wang, J. Wang, B. Schiele, X. Xie, B. Raj, and M. Savvides. Softmatch: Addressing the quantity-quality tradeoff in semi-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

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

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

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

[14] C. Gupta and A. Ramdas. Top-label calibration and multiclass-to-binary reductions. In *International Conference on Learning Representations*, 2022.

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

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

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

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

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

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

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

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

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

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

[25] A. Kumar, S. Sarawagi, and U. Jain. 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*, pages 2805–2814. PMLR, 10–15 Jul 2018.

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

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

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

[29] G. J. McLachlan. 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.

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

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

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

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

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

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

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

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

[38] H. Qiu, K. Chintalapudi, and R. Govindan. MCAL: Minimum cost human-machine active labeling. In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

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

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

[43] SGT. Aws sagemaker ground truth. https://aws.amazon.com/sagemaker/data-labeling/, 2022. Accessed: 2022-11-18.

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

[45] 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. *Advances in neural information processing systems*, 33:596–608, 2020.

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

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

[48] V. N. Vapnik, V. Vapnik, et al. Statistical learning theory. 1998.

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

[50] H. Vishwakarma, S. J. Tay, S. S. S. Namburi, F. Sala, R. K. Vinayak, et al. Pearls from pebbles: Improved confidence functions for auto-labeling. *arXiv preprint arXiv:2404.16188*, 2024.

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

[52] Y. Wang, H. Chen, Q. Heng, W. Hou, Y. Fan, Z. Wu, J. Wang, M. Savvides, T. Shinozaki, B. Raj, B. Schiele, and X. Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[54] 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 *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.

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

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

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

Supplementary Material

We discuss related works in Appendix A formal algorithm in Appendix B. Additional experimental results and details are in Appendix C.

A Related Work

Semi-supervised learning (SSL). There is a rich literature on SSL spanning multiple decades [57, 7, 44, 36]. This literature comprises of a wide variety of approaches. Among these significant focus has been placed on self-training (also called pseudolabeling) [42, 6, 40, 27, 37, 2], generative models [34, 1, 20], graph-based strategies [5, 35, 46], and transductive approaches [48, 17]. Due to their simplicity, pseudolabeling-based approaches have gained prominence and are widely used in application areas such as NLP [19], speech recognition [18], and protein prediction [10]. 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 [3] i.e., when a model starts to reinforce its own mistakes. To overcome this and to maintain high quality of pseudolabels, confidence-based thresholding is applied. Here only the unlabeled data where confidence is higher than a particular threshold is used [45]. Due to the limitations of fixed thresholds, adaptive thresholds based on the classifier's learning status have been introduced to improve performance [54, 56, 52]. Nearly all of these methods also use some form of consistency regularization [26, 4, 41, 11, 22] 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 [53, 52, 45, 56, 8, 54].

Confidence functions and calibration. Miscalibration (overconfidence) in neural networks plagues various applications [33, 15, 13], including SSL. To mitigate this in general, a range of solutions have been proposed, including training-time methods [31, 25, 16, 9, 12] and post-hoc methods [13, 24, 14, 23, 55]. In pseudolabeling based SSL, recent works [39, 28, 30] noted the issue of miscalibration. To promote calibration, **(author?)** [28] use Bayesian neural nets by replacing the model's final layer with a Bayesian layer. **(author?)** [39] improve pseudolabeling with negative labels and an uncertainty-aware pseudolabel selection technique. **(author?)** [30] incorporate a regularizer in pseudolabeling to encourage calibration.

While calibration is generally desirable, it may not be enough to solve the overconfidence issue in SSL and other applications. Pseudolabeling requires scores that effectively distinguish correct from incorrect predictions, aligning with the ordinal ranking criterion [15, 31, 12, 9]. Instead of trial-and-error with various options, we propose a flexible framework that learns confidence functions directly optimized for pseudolabeling objectives. This builds upon principles used in threshold-based auto-labeling (TBAL) [50], a technique for creating labeled datasets.

B Appendix to the Method Section

Practical Version. The optimization problem discussed earlier involves population-level quantities which are usually not accessible in practice. Thus we have to fall back to using their finite sample estimates and smooth variations to make the optimization problem tractable. We adapt the steps from [50] to obtain such a practical version of the optimization problem. There, the authors first estimate the coverage and error using a small amount held-out labeled data (called calibration data D_{cal}) curated from the validation data. They then introduce differentiable surrogates for the 0-1 variables. Let $\sigma(\alpha, z) := 1/(1 + \exp(-\alpha z))$ denote the sigmoid function on $\mathbb{R}$ with scale parameter $\alpha \in \mathbb{R}$. The surrogates are as follows,

$$\widetilde{\mathcal{P}}(g, \mathbf{t}|h, D_{\text{cal}}) := \frac{1}{|D_{\text{cal}}|} \sum_{(\mathbf{x},y,\tilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\tilde{y}] - \mathbf{t}[\tilde{y}]\big), \tag{4}$$

$$\widetilde{\mathcal{E}}(g, \mathbf{t} \mid h, D_{\text{cal}}) := \frac{\sum_{(\mathbf{x},y,\tilde{y}) \in D_{\text{cal}}} \mathbb{1}(y \neq \tilde{y})\, \sigma\big(\alpha, g(\mathbf{x})[\tilde{y}] - \mathbf{t}[\tilde{y}]\big)}{\sum_{(\mathbf{x},y,\tilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\tilde{y}] - \mathbf{t}[\tilde{y}]\big)}. \tag{5}$$

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

Algorithm 1 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.

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

$\widetilde{Y} \leftarrow [0] \times n_u$, $S \leftarrow [0] \times n_u$, $i \leftarrow 1$.
D_{cal}, $D_{\text{th}} \leftarrow$ `draw_randomly`$(D_{\text{val}}, N_{\text{cal}}, N_{\text{th}})$
while $i \leq$ num_iters **do**
  D_l^b, X_u^b, $I_u^b \leftarrow$ `draw_random_batch`$(\mu D_l, \mu X_u, B)$
  $X_{u,w}^b$, $X_{u,s}^b \leftarrow \omega(X_u^b)$, $\Omega(X_u^b)$
  if use-`PabLO` **then**
    if $i \% F = 0$ **then**
      $\hat{g}_i \leftarrow$ `solve_opt_problem_P1`$(\hat{h}_i, D_{\text{cal}})$
      $\hat{\mathbf{t}}_i \leftarrow$ `estimate_thresholds`$(\hat{h}_i, \hat{g}_i, D_{\text{th}})$
      $\widetilde{Y}^f \leftarrow \hat{h}_i(\omega(X_u)), \quad S^f \leftarrow \mathbb{1}(\hat{g}_i(\omega(X_u)) \geq \hat{\mathbf{t}})$
      if use-accumulation **then**
        $\widetilde{Y}$, $S \leftarrow S^f \widetilde{Y}^f + (1 - S^f)\widetilde{Y}; \quad S \leftarrow S \vee S^f$
      else
        $\widetilde{Y}$, $S \leftarrow \widetilde{Y}^f$, S^f
      end if
    end if
    $\widetilde{Y}^b$, $S^b \leftarrow \widetilde{Y}[I_u^b], \;\; S[I_u^b]$
  else
    $\widetilde{Y}^b$, $S^b \leftarrow$ `baseline_pseudo_labeling`$(\hat{h}_i, X_{u,w}^b)$
    if use-accumulation **then**
      for $j \in I_u^b$ **do**
        $\widetilde{Y}[j] \leftarrow S^b[j]\widetilde{Y}^b[j] + (1 - S^b[j])\widetilde{Y}[j]$
        $S[j] \leftarrow S[j] \vee S^b[j]$
      end for
    end if
  end if
  $\widehat{\mathcal{L}}_s(\hat{h}_i) \leftarrow$ `supervised_loss`(h, D_l^b)
  $\widehat{\mathcal{L}}_u(\hat{h}_i) \leftarrow$ `unsupervised_loss`$(h, X_{u,w}^b X_{u,s}^b, \widetilde{Y}^b, S^b)$
  $\widehat{\mathcal{L}}_r(\hat{h}_i) \leftarrow$ `baseline_regularizers()`
  $\widehat{\mathcal{L}}(\hat{h}_i) \leftarrow \widehat{\mathcal{L}}_s(\hat{h}_i) + \lambda_u \widehat{\mathcal{L}}_u(\hat{h}_i) + \lambda_r \widehat{\mathcal{L}}_r(\hat{h}_i)$
  $\hat{h}_{i+1} \leftarrow$ `SGD_update`$(\widehat{\mathcal{L}}(\hat{h}_i)); \quad i \leftarrow i + 1$
  if $i \%$eval_freq $= 0$ **then**
    eval_acc $\leftarrow$ `evaluate_model`$(\hat{h}_i, D_{\text{val}})$
    If eval_acc is best so far then $\hat{h}_{\text{ssl}} = \hat{h}_i$.
  end if
end while

importance of the pseudolabeling error and coverage.

$$\hat{g}_i, \hat{\mathbf{t}}_i \in \underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\min} \quad -\widetilde{\mathcal{P}}(g, \mathbf{t} \mid \hat{h}_i, D_{\text{cal}}) + \lambda \, \widetilde{\mathcal{E}}(g, \mathbf{t} \mid \hat{h}_i, D_{\text{cal}}) \tag{P1}$$

We use 2-layer neural nets as a choice of $\mathcal{G}$. The optimization problem (P1) is nonconvex, 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.

The full algorithm we use is:

Table 2: Details of the dataset we use in experiments. k is the no. of classes. N_l is the no. of labeled data points used for training the backbone model h. N_u is the no. of unlabelled data points used for consistency regularization and pseudolabeling for all the methods. N_{val} is the no. of points used for model selection in all methods. N_{test} is the no. of test data points. N_{cal} is the number of points used for learning the g function. N_{th} is the no. of points used for threshold estimation.

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

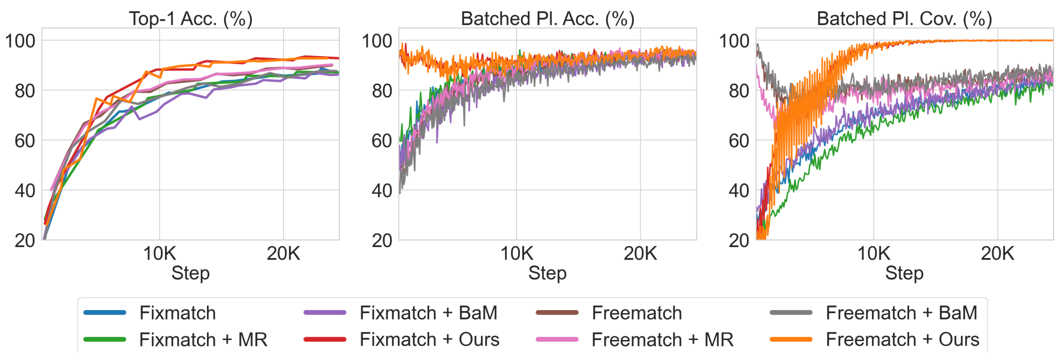

Figure 1: Left to Right: Top-1 accuracy, Batched pseudolabeling accuracy and Batched pseudolabeling coverage of our method and baselines on CIFAR-10. We plot the values for every 200 steps.

C Additional Experiments and Details

Compute. For all our experiments, we used an NVIDIA RTX A6000 which has 48GB of VRAM and an NVIDIA RTX 4090 with 24GB of VRAM. The runtime depends on several factors including CPU I/O and GPU load, but on average, the baselines took around 8 hours, while our method took around 15 hours for 25K iterations.

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 3.

C.1 Ablation Studies

We perform ablations that give insights into the role of various parts of it. We run all the ablation experiments on the CIFAR-10 data setting.

Table 3: Hyperparameters used for our method.

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

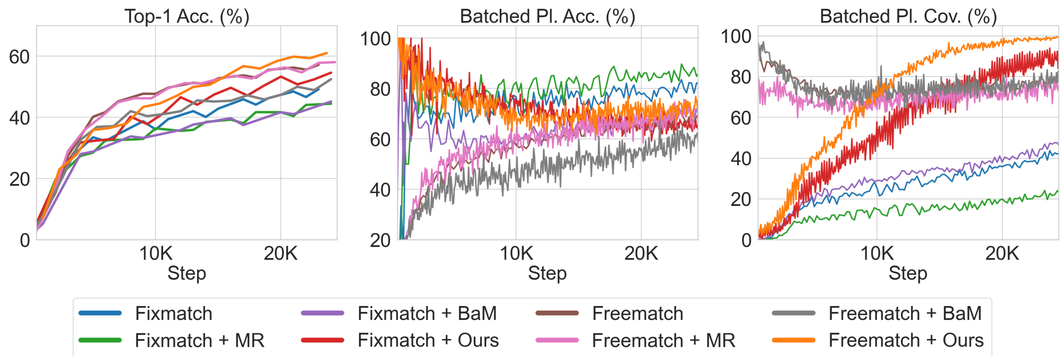

Figure 2: Left to Right: Top-1 accuracy, Batched pseudolabeling accuracy and Batched pseudolabeling coverage of our method and baselines on CIFAR-100. We plot the values for every 200 steps.

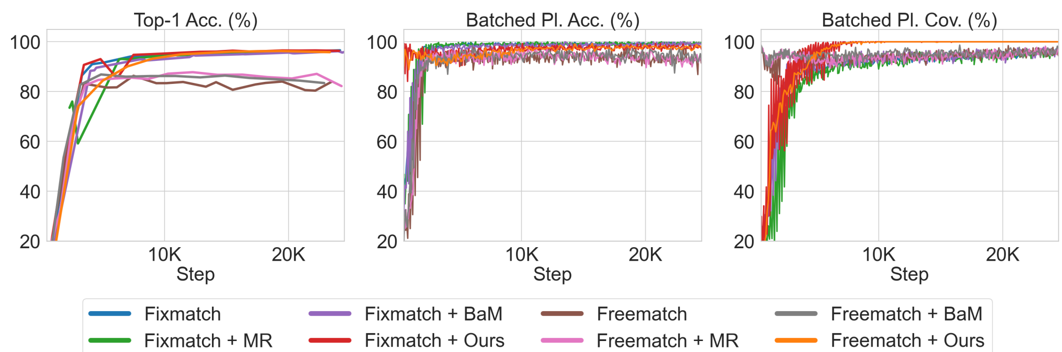

Figure 3: Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and batched pseudolabeling coverage of our method and various baselines on SVHN. We plots the values for every 200 steps.

A1. 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 could help in improving the utilization of unlabeled data and could 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 accumulation and without it and report the results in Table 4. We observe that our method has similar test accuracy irrespective of accumulation. However, with accumulation it achieves better coverage in early iterations as observed in Figure 6. These results are not surprising, since our method ensures high quality of pseudolabels while maximizing coverage, it is able to eventually catch up with the version using accumulation, leading to similar final test accuracies. On the other hand, having accumulation hurts the performance of baseline models. This might be because the pseudo labels generated by the baseline models are not accurate especially in the earlier iterations, thus degrading the overall performance. Overall, we believe accumulation is going to be helpful when we have pseudolabels with high accuracy. The plots for coverage and accuracy over the entire run are in Figures 7, 8 in the Appendix C.

A2. Does error tolerance affect 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 higher noise will lead to worse performance, which is our expectation too. To see this, we run our method with $\epsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ in the CIFAR-10 setting. We run each setting with 3 random seeds and report the results in Figure 5. The results are as expected — higher values of ϵ lead to

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

Method	Acc—True	Acc—False
Fixmatch	66.30 ± 1.68	88.15 ± 1.27
Fixmatch + MR	64.24 ± 1.93	87.85 ± 1.10
Fixmatch + BaM	84.50 ± 2.60	86.44 ± 1.47
Freematch	85.17 ± 4.74	90.17 ± 0.13
Freematch + MR	80.67 ± 2.39	90.17 ± 0.45
Freematch + BaM	88.92 ± 0.49	88.34 ± 0.99
Fixmatch + Ours	$\mathbf{93.03 \pm 0.44}$	$\mathbf{93.34 \pm 0.50}$
Freematch + Ours	$\mathbf{93.08 \pm 0.05}$	$\mathbf{93.01 \pm 0.24}$

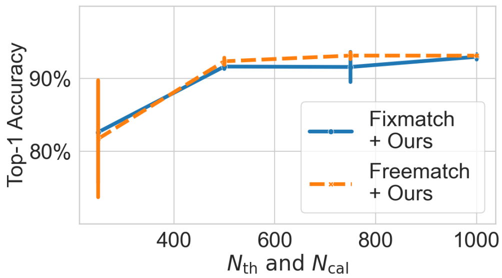

Figure 4: Top-1 accuracy of our method with different N_{th} and N_{cal}.

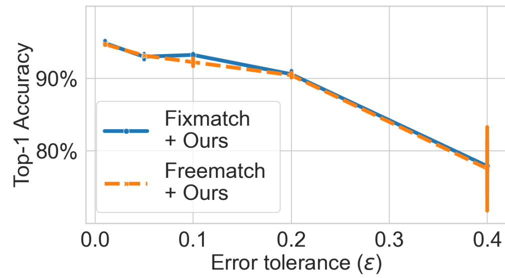

Figure 5: Top-1 accuracy of our method with different error tolerance ϵ.

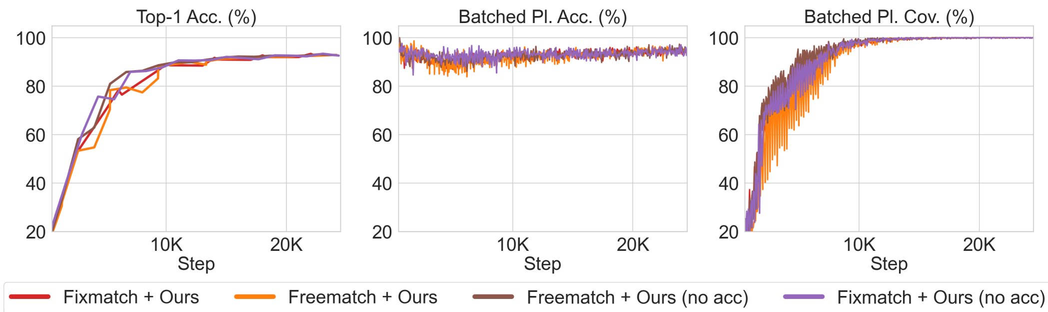

Figure 6: Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and Batched pseudolabeling coverage of our method with and without pseudolabeling accumulation enabled.

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 6 and Figure 10.

A3. 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 4. 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 Table 5 and Figure 9 for more details.

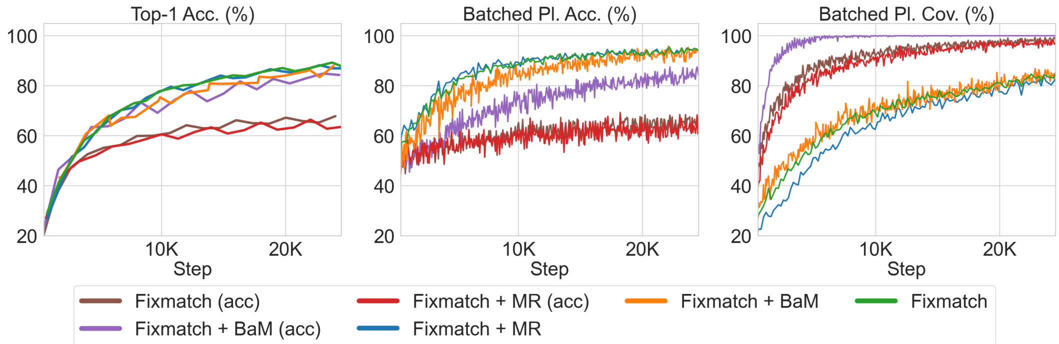

Figure 7: **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and batched pseudolabeling coverage of Fixmatch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsen the performance of baseline methods in terms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

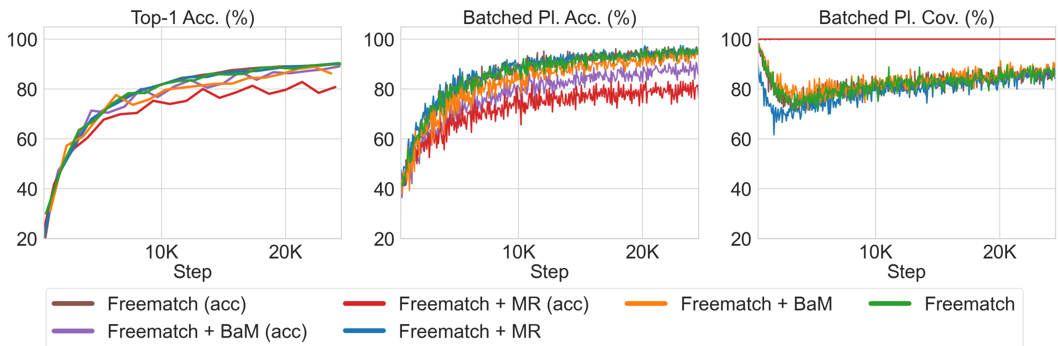

Figure 8: **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and batched pseudolabeling coverage of Freematch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsen the performance of baseline methods in terms of accuracy and coverage.

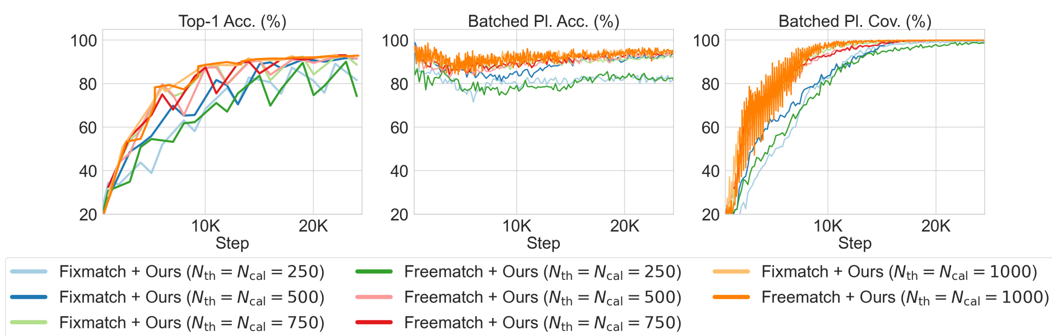

Figure 9: **(A3.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and batched pseudolabeling coverage of our method with $N_{\text{th}} = N_{\text{cal}} \in \{250, 500, 750, 1000\}$ on CIFAR-10. We observe that having more calibration and threshold estimation points benefits the performance of our method.

Table 5: Results on CIFAR-10 with varying N_{cal} and N_{th}.

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

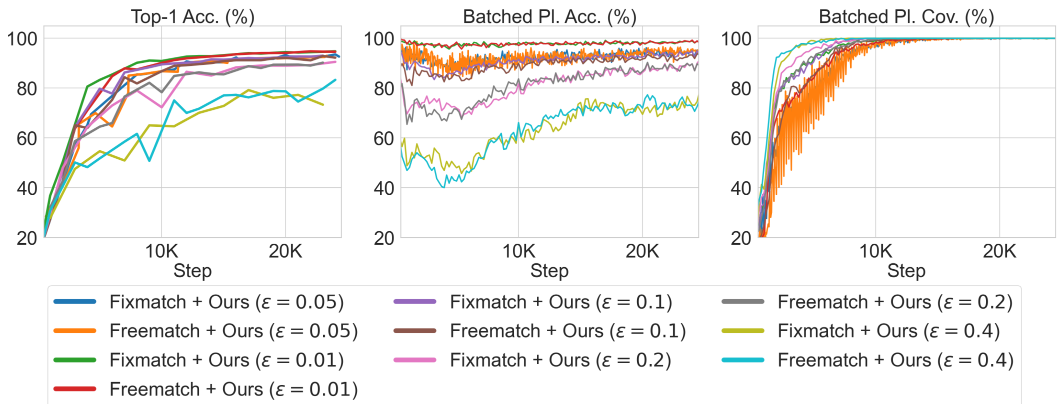

Figure 10: **(A2.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy and batched pseudolabeling coverage of our method with $\epsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ on CIFAR-10. Although having a looser constraint on the error encourages more coverage, the pseudolabeling drops as a trade-off.

Table 6: Results on CIFAR-10 with varying ϵ.

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Fixmatch + Ours	**94.85** $\pm$ **0.28**	93.24 $\pm$ 0.18	90.52 $\pm$ 0.43	80.62 $\pm$ 1.22
Freematch + Ours	**94.67** $\pm$ **0.09**	92.11 $\pm$ 0.84	90.20 $\pm$ 0.65	82.23 $\pm$ 1.31