Improving Semi-Supervised Learning under Class Imbalance: A Comparative Study of FixMatch, AdaMatch, and a Hybrid FlexMatch-AdaMatch Approach

Zahra KANANI[1]

[1]Department of Computer Science, Université Grenoble Alpes, Grenoble, France

Abstract

Semi-supervised learning (SSL) leverages both labeled and unlabeled data to improve model performance when annotated data are limited. While methods such as FixMatch and AdaMatch have demonstrated strong results in balanced datasets, their robustness to class imbalance remains less explored. In this work, we evaluate the impact of labeled data imbalance using the CIFAR-10 benchmark. FixMatch demonstrates relative stability, maintaining evaluation accuracy near 93% under imbalance. AdaMatch, however, suffers a significant drop, with performance reduced to 73.6% due to its mechanisms of distribution alignment and relative confidence thresholding. A per-class analysis shows that minority classes are severely misclassified into majority ones. To address this, we propose integrating FlexMatch's class-wise thresholding strategy into AdaMatch, ensuring minority classes contribute effectively while retaining domain adaptation capability. The findings highlight the importance of threshold design in SSL under imbalance. This study provides insights into improving fairness and robustness of SSL methods, with implications for real-world domains where balanced datasets are rarely available.

Introduction

Semi-supervised learning

Today, labeling data is one of the challenges in machine learning. For some issues, such as privacy, we have limited access to labeled data. However, we can have large amounts of unlabeled data but labeling them can be expensive and time-consuming. Semi-supervised learning is a method that can benefit from both limited labeled data and abundant unlabeled data. In other words, semi-supervised learning is a paradigm between fully supervised and unsupervised learning. It combines the strengths of both approaches, using a small set of labeled examples to guide the learning process while extracting additional information from the unlabeled data to enhance performance.

SSL Methods

Consistency regularization, proxy-label methods, and graph-based models are different approaches to semi-supervised learning (SSL) that have been introduced over the years. In consistency regularization, the model is trained to produce consistent predictions on a given unlabeled example and its perturbed version. Proxy-label methods leverage a model trained on the labeled set to produce additional training examples by labeling instances from the unlabeled set. Graph-based methods represent data as nodes in a graph, link similar samples, and propagate label information through these connections [1].

Methods of SSL can be categorized into two types based on the learning pattern: transductive learning and inductive learning. Transductive learning learns to label only the unlabeled data it sees during training but can't handle new data it hasn't seen before. Inductive learning trains a model that can label new, unseen data later [1].

Other learning with few labels

Several learning paradigms are closely related to Semi-Supervised Learning (SSL), as they also aim to reduce dependence on large amounts of labeled data.

Active Learning (AL) chooses the most useful unlabeled examples to label, reducing labeling cost by selecting informative and representative samples. Thereby this approach can lead to maximized performance with minimal labeling effort. Active Learning and SSL both aim to learn with limited labeled data and are often combined to improve performance, reduce errors, and enhance learning from unlabeled data [1] [2].

Weakly-supervised learning (WSL) trains models using large amounts of low-quality or imprecise labels instead of expensive, manually labeled data. WSL can be combined with semi-supervised learning to boost performance using a small number of strong labels [1].

Learning with noisy labels deals with training data that contains unreliable labels, which can reduce performance. Thus, to solve the problem, for these types of data we can consider less sensitive in loss function, give them less weight during training, fix and relabel the incorrect labels or treating noisy data as unlabeled and applying semi-supervised learning (SSL) methods. These methods can help the model focus on cleaner data and improve its ability to generalize [1] [3].

Transfer Learning uses knowledge from one related but different (source) domain to help learning in another (target) domain, especially when labeled data is limited in the target. This method helps to achieve better accuracy on target data [1] [4] [5].

Domain Adaptation is one popular type of transfer learning. In these problems, like the other types of transfer learning, source domain and target domain are different but in this type of transfer learning, the task between two domains stays the same while it can be different or the same in other types. [1] [6] [7] [8].

Dissimilarity between domains arises frequently in real-world applications, where collecting and labeling large datasets for every possible scenario is impractical. For example, a model trained to recognize objects in high-quality images might perform poorly when applied to images from a surveillance camera, due to differences in lighting, resolution, and background. Thus, the goal is to build a model that can perform well across these different domains. For this purpose, the distributional discrepancy between domains must be reduced. Typically, labeled data is available in the source domain, while the target domain may have no labels (unsupervised domain adaptation), a few labels (semi-supervised domain adaptation), or full labels (supervised domain adaptation).

Semi-supervised domain adaptation

SSDA addresses the challenge of adapting a model trained on a source domain to perform well on a target domain, where the target domain has limited labeled data and abundant unlabeled data. In traditional supervised learning, models assume that training and testing data are drawn from the same distribution. However, in real-world problems, this assumption does not always hold and can lead to poor performance.

SSDA aims to overcome domain shift by utilizing both the labeled data from the source domain and the small amount of labeled data along with unlabeled data from the target domain. And to achieve this, SSDA typically exploits consistency regularization and pseudo-labeling strategies. That can encourage the model to produce stable predictions under input perturbations and assign pseudo labels to unlabeled target data based on model confidence. Some popular SSDA approaches include FixMatch, UDA, and AdaMatch. These strategies, making SSDA an effective solution for addressing distributional discrepancies between source and target domains.

Main Assumptions in SSL

Before we can use semi-supervised learning (SSL) algorithms, we need to understand when they are applicable. These algorithms only work well if certain assumptions are true. Without these assumptions, it would be impossible to make reliable predictions on new, unseen data. The key assumptions that SSL relies on are illustrated in Figure 1 [1]:

- **Cluster Assumption:** Data tends to form clusters, and points in the same cluster are likely to belong to the same class. And decision boundaries should lie in low-density regions (between clusters), not through high-density ones.
- **Continuity / Smoothness Assumption:** Points that are close in the input space are likely to have the same label. And the model should produce similar outputs for similar inputs.
- **Manifold Assumption:** Although the data belongs to the high-dimensional input space, it follows a low-dimensional embedded manifold. So, the learning task can thus be performed effectively on this manifold, not the full space.

Besides these assumptions, most of the studies in the field of SSL have not considered many issues that lead to these algorithms failing in real-world applications. There are some practical recommendations to close SSL research towards real-world applicability [9]:

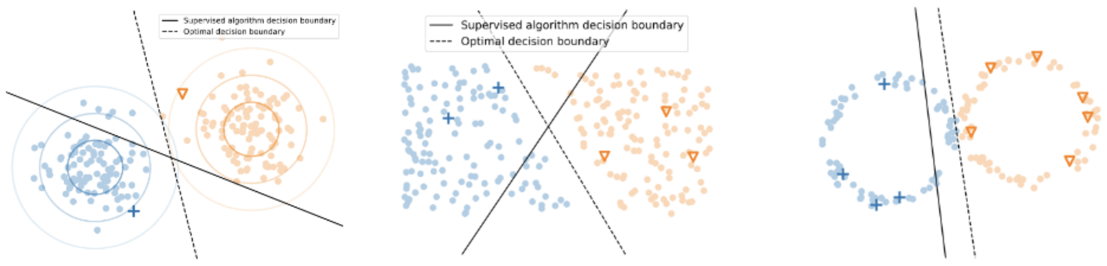


Figure 1: Visual representation of the key assumptions in semi-supervised learning — cluster, continuity/smoothness, and manifold assumptions.

- **A Shared Implementation:** To fairly compare different SSL methods, it is recommended to use a common codebase and identical model architecture.
- **High-Quality Supervised Baseline:** Try to include a carefully tuned supervised baseline trained only on the labeled data to see whether SSL truly outperforms strong baselines.
- **Comparison to Transfer Learning:** Compare SSL methods against transfer learning from a related labeled dataset, as this is a strong and often more practical alternative in many scenarios.
- **Considering Class Distribution Mismatch:** To evaluate how well SSL methods work in real-world situations, test them in the face of unlabeled data that comes from different classes than the labeled data.
- **Varying the Amount of Labeled and Unlabeled Data:** Systematically test SSL performance across different quantities of labeled and unlabeled data to understand robustness in limited- and abundant-data regimes.
- **Realistically Small Validation Sets:** Assess how small validation sets affect hyperparameter tuning and model selection, since real applications rarely have large validation sets available.

In addition to simplifying assumptions that limit the effectiveness of learning algorithms in real-world scenarios, many semi-supervised learning (SSL) prototypes also assume a balanced class distribution in the labeled data, as well as an equal ratio of labeled to unlabeled samples for each class [10] [11] [12]. However, in real-world applications, such balanced conditions are rarely achievable, and for unseen data, we typically have no control over the number of samples per class. This study aims to:

(1) examine the impact of imbalanced class distributions in the labeled data and unequal labeled-to-unlabeled ratios across classes on model performance;

(2) explore methods to improve model performance under these imbalanced conditions.

Materials and Methods

Notation

In this study, we use X, Y, and Z to denote examples, labels, and logits, respectively. Specifically, X_S and Y_S refer to the data and their labels in the source domain, while X_U denotes the unlabeled target data. Here, J is the number of classes, and N_S and N_U are the numbers of labeled source samples and unlabeled target samples, respectively. Importantly, the source and target domains share the same classification task, so the number of classes J and the image dimension d are the same for both. Subscripts s and w indicate strongly or weakly augmented examples. The notation $\hat{Y}$ represents a pseudo-label predicted by the model for unlabeled data before any adjustment, and $\tilde{Y}$ denotes the distribution-aligned pseudo-label after rescaling. The symbol θ denotes all model parameters—the weights and biases learned during training. $E[\hat{Y}]$ indicates the average predicted class probabilities across all examples in a batch. Finally, the notation $H(p, q)$ denotes the cross-entropy loss between a target distribution p and a prediction q.

Dataset Preparation and Class Imbalance Setup

In this study, the CIFAR-10 dataset, which contains 10 different categories, was used as the test case. As specified by the Unified Semi-Supervised Learning Benchmark (USB), the balanced configuration includes 250 labeled examples per class, with the remaining data used as unlabeled. In the imbalanced setup, the number of labeled examples has to be adjusted so that each class has a different amount. To ensure a fair comparison between the balanced and imbalanced setups, the total number of labeled and unlabeled samples should remain the same in both cases.

$$10 \times 250 = 2500 \text{ labeled samples.}$$

Let r_i, for $i = 1, 2, \ldots, 10$, represent the ratio applied to the number of labeled samples in class i under the imbalanced setup. To preserve the total number of labeled samples, the following condition must hold:

$$\sum_{i=1}^{10} r_i \times 250 = 2500 \quad \Rightarrow \quad \sum_{i=1}^{10} r_i = 10.$$

An example set of class-wise ratios r_i that satisfies this constraint is:

$$\begin{aligned} r_1 &= 0.1, & r_6 &= 1.9, \\ r_2 &= 0.2, & r_7 &= 1.8, \\ r_3 &= 0.3, & r_8 &= 1.7, \\ r_4 &= 0.4, & r_9 &= 1.6, \\ r_5 &= 0.5, & r_{10} &= 1.5. \end{aligned}$$

This setup introduces an imbalanced distribution of labeled data across classes while keeping the overall quantity unchanged, enabling a controlled and fair evaluation of the effect of class imbalance. The imbalance function iterates over all classes, selects a subset of samples from each class based on its corresponding ratio r_c, and concatenates them to form the imbalanced dataset. If N_c is the number of labeled samples in class c in the balanced dataset, then the number of samples retained for that class in the imbalanced set is calculated as $r_c \cdot N_c$.

This approach ensures that the total number of labeled samples across all classes remains equal to the original 2500 samples, allowing for a fair comparison with the balanced configuration. The method enables a controlled evaluation of semi-supervised learning algorithms under class imbalance conditions.

In the following, we used FixMatch to examine its sensitivity to class imbalance in the dataset and compared its performance with that of AdaMatch during training. AdaMatch extends the FixMatch method by introducing several additional components designed to address domain shifts and improve flexibility. In the next section, we briefly introduce the main components of the AdaMatch algorithm and describe the methods it employs for leveraging unlabeled data and addressing distribution discrepancies.

AdaMatch: A Unified Approach to Semi-Supervised Learning and Domain Adaptation

AdaMatch is a semi-supervised learning (SSL) and domain adaptation algorithm designed to improve performance when there is a distribution shift between labeled and unlabeled data. It extends the FixMatch method by introducing several components that make it especially robust to domain shifts, enabling it to work effectively in SSL, unsupervised domain adaptation (UDA), and semi-supervised domain adaptation (SSDA) settings [13].

The main idea of AdaMatch is to combine a small amount of labeled data (often from a source domain) with large quantities of unlabeled data (from either the same or a different target domain). The key components of AdaMatch are:

Random Logit Interpolation

For each labeled example, the model computes logits in two ways: once using only the labeled (source) batch normalization and once using a mixed batch that combines labeled and unlabeled data. These logits are then randomly interpolated. This technique encourages the model to produce consistent predictions regardless of the domain.

Distribution Alignment

To prevent the model from over-predicting certain classes, AdaMatch adjusts the pseudo-label predictions of the unlabeled data so that their average class distribution matches the mean distribution observed on the labeled data.

Relative Confidence Thresholding

Traditional pseudo-labeling relies on a fixed confidence threshold (e.g., 0.95) to decide which predictions are trustworthy. AdaMatch instead defines a relative threshold by scaling the fixed threshold by the average of top confidence on labeled examples.

Augmentation Strategy

Similar to FixMatch, AdaMatch uses both weak and strong augmentations for each example. A pseudo-label is predicted from the weakly augmented version, and the strongly augmented version is trained to match this pseudo-label only if the prediction exceeds the relative confidence threshold.

During training, AdaMatch computes two losses:

- A supervised loss on the labeled examples.
- An unsupervised loss on unlabeled examples.

In the first step, the model learns to predict correctly on labeled data under weak and strong augmentation using the following supervised loss function:

$$L_{\text{source}}(\theta) = \frac{1}{N_S}\sum_{i=1}^{N_S} H\big(Y_S^{(i)},\, Z_{S,w}^{(i)}\big) + \frac{1}{N_S}\sum_{i=1}^{N_S} H\big(Y_S^{(i)},\, Z_{S,s}^{(i)}\big)$$

That uses the cross-entropy between the true label and the model's logits under augmentations for each labeled sample i.

In the second step, for each unlabeled sample i, the model generates a pseudo-label $\widehat{Y}_{U,w}^{(i)}$ by applying a weak augmentation. It then adjusts this pseudo-label to match the overall class distribution seen in the labeled data by scaling it proportionally to the average class frequencies of both labeled and unlabeled batches:

$$\widetilde{Y}_{U,w}^{(i)} = \text{normalize}\left(\widehat{Y}_{U,w}^{(i)} \times \frac{E\big[\widehat{Y}_{S,w}\big]}{E\big[\widehat{Y}_{U,w}\big]}\right).$$

If the prediction is confident enough, then the pseudo-label is used in the unsupervised loss:

$$L_{\text{target}}(\theta) = \frac{1}{N_U}\sum_{i=1}^{N_U} H\left(\text{stop\_gradient}\big(\widetilde{Y}_{U,w}^{(i)}\big),\, Z_{U,s}^{(i)}\right) \cdot \text{mask}(i).$$

Here, stop_gradient ensures the pseudo-label remains fixed during training, and mask(i) is set to 1 if the prediction passes the threshold and 0 otherwise.

Finally, the total loss combines both terms as follows:

$$L(\theta) = L_{\text{source}}(\theta) + \mu(t) \cdot L_{\text{target}}(\theta).$$

In this setup, the weighting schedule $\mu(t)$ gradually increases the contribution of the unsupervised loss over time, enabling the model to learn reliably from labeled data at the start and progressively incorporate confident pseudo-labels from the unlabeled data to enhance performance.

Compared to prior methods, AdaMatch has been shown to perform strongly across different tasks and datasets, often surpassing state-of-the-art results without requiring dataset-specific hyperparameter tuning. It is particularly designed to be effective in settings where there is a distribution shift between labeled and unlabeled data, making it a flexible method for real-world semi-supervised learning and domain adaptation scenarios. However, as shown in our experiments, its relative performance may vary depending on dataset characteristics and label distribution.

Results

This section presents the results of applying two semi-supervised learning algorithms—FixMatch and AdaMatch—to the CIFAR-10 dataset under two different labeled data conditions: balanced and imbalanced class distributions. Figures 2 through 5 illustrate the trends in training loss, evaluation loss, and evaluation accuracy across approximately one million training iterations. Our findings reveal notable differences in performance, particularly under class imbalance.

In the **balanced setting**, FixMatch achieved strong performance, with the training loss decreasing to approximately 0.12 and the evaluation loss reaching 0.25. The evaluation accuracy in this setting was around 95%, indicating that the model was able to generalize well when trained on a uniformly labeled dataset. Under the **imbalanced setting**, FixMatch experienced a moderate decline in performance. While the training loss remained low (below 0.13), the evaluation loss increased to approximately 0.58, and the evaluation accuracy dropped slightly to around 93%. These results suggest that although FixMatch is somewhat sensitive to class imbalance, it retains relatively strong generalization compared to more complex approaches.

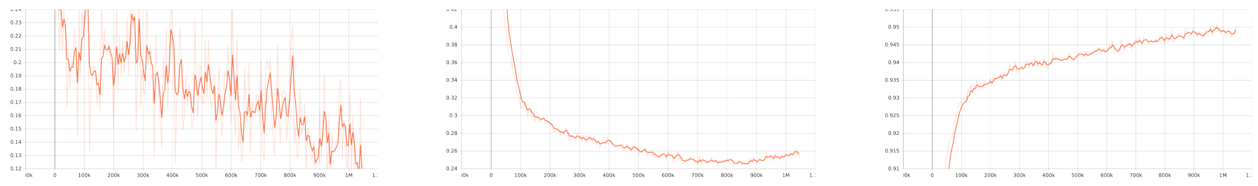

(a) (b) (c)

Figure 2: Training and evaluation performance of FixMatch on the CIFAR-10 dataset under a balanced class distribution, showing: a) training loss, b) evaluation loss, and c) evaluation accuracy.

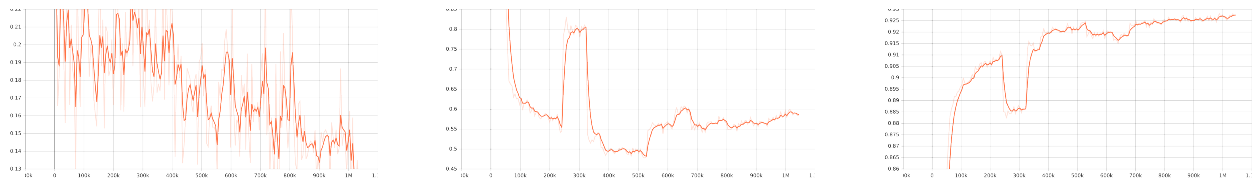
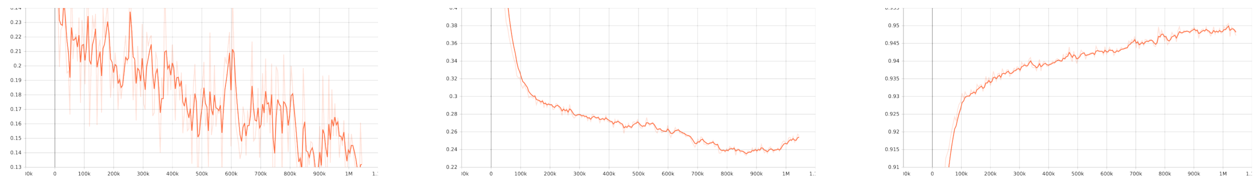

(a) (b) (c)

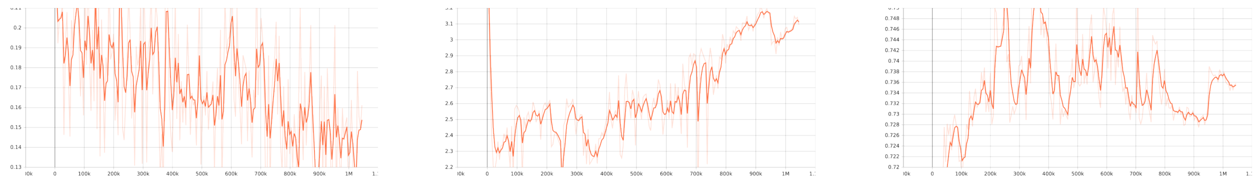
Figure 3: Results of FixMatch on the CIFAR-10 dataset with an imbalanced class distribution, depicting: (a) training loss, (b) evaluation loss, and (c) evaluation accuracy.

In comparison, AdaMatch performed similarly to FixMatch in the **balanced scenario**, with the training loss decreasing to around 0.13 and the evaluation loss reaching 0.25. The evaluation accuracy also reached 95%, indicating good generalization when the labeled data was evenly distributed. However, under the **imbalanced condition**, AdaMatch experienced a much larger drop in performance than FixMatch. The training loss increased slightly to above 0.15, but the evaluation loss rose to 3.1, and the evaluation accuracy fell to 73.6%. These results indicate that AdaMatch was more sensitive to class imbalance than FixMatch in this experimental setup.



(a) (b) (c)

Figure 4: Training and evaluation performance of AdaMatch on the CIFAR-10 dataset with a balanced class distribution, illustrating: (a) training loss, (b) evaluation loss, and (c) evaluation accuracy.



(a) (b) (c)

Figure 5: Performance of AdaMatch on the CIFAR-10 dataset with an imbalanced class distribution, showing: (a) training loss, (b) evaluation loss, and (c) evaluation accuracy.

To better understand these differences, the model was modified to calculate per-class accuracies. The comparative results are presented in tables 1 and 2 to illustrate whether the decrease in overall accuracy is uniformly distributed across all classes or if certain classes contribute more significantly to the decline.

Class number	0	1	2	3	4	5	6	7	8	9
Adamatch	0.980	0.984	0.918	0.850	0.954	0.904	0.988	0.982	0.984	0.980
FixMatch	0.978	0.986	0.920	0.876	0.960	0.876	0.998	0.976	0.974	0.974

Table 1: Per-class accuracy of AdaMatch and FixMatch on the CIFAR-10 dataset under a balanced class distribution.

Class number (ratio)	0 (0.1)	1 (0.2)	2 (0.3)	3 (0.4)	4 (0.5)	5 (1.9)	6 (1.8)	7 (1.7)	8 (1.6)	9 (1.5)
Adamatch	0.092	0.468	0.664	0.642	0.654	0.916	0.996	0.970	0.990	0.988
FixMatch	0.972	0.980	0.914	0.820	0.966	0.916	0.992	0.980	0.982	0.978

Table 2: Per-class accuracy of AdaMatch and FixMatch on the CIFAR-10 dataset under an imbalanced class distribution.

In the balanced configuration, both AdaMatch and FixMatch achieved consistently high accuracy across all classes, with only minor variations between them. These differences were relatively small and did not affect the general conclusion that both methods generalize well when the labeled data is evenly distributed.

The contrast between the two methods becomes much more pronounced in the imbalanced configuration. The per-class results reveal that AdaMatch suffers a dramatic decline in performance on minority classes. For instance, the accuracy for class 0 dropped to only 9%, and class 1 fell to below 50%, while FixMatch maintained substantially higher accuracies for the same classes. In contrast, for the majority classes with higher ratios, both methods continued to perform strongly, often exceeding 90% accuracy. These effects are also reflected in Table 3, the confusion matrix of AdaMatch on the imbalanced dataset: minority classes are frequently misclassified into majority ones, such as class 0 being predicted predominantly as class 8 and class 1 as class 9. These systematic misclassifications clearly illustrate the model's bias toward over-represented categories.

The difference between the results of AdaMatch and FixMatch mainly comes from how each method handles pseudo-labeling and class distribution. FixMatch uses a simple fixed confidence threshold to decide which unlabeled samples should contribute to training, which makes it relatively stable even when the labeled dataset is imbalanced. AdaMatch, on the other hand, introduces two additional mechanisms: distribution alignment and relative confidence thresholding. Distribution alignment forces the pseudo-label distribution to follow the labeled data distribution. When the labeled data is imbalanced, this alignment makes the imbalance worse, causing the model to rely more on the majority classes. Relative confidence thresholding further depends on the confidence statistics of labeled samples, which are unreliable for low-sample classes, leading to very few accepted pseudo-labels for minorities. As a result, AdaMatch performs similarly to FixMatch under balanced conditions, but its additional mechanisms backfire under imbalance, making it more biased toward majority classes, while FixMatch maintains robustness.

To improve the performance of AdaMatch on imbalanced datasets, we explored an alternative approach by modifying the thresholding mechanism, adopting a strategy that does not suppress minority classes while still benefiting from AdaMatch's adaptive mechanisms. One solution is to replace AdaMatch's relative confidence thresholding with the class-wise thresholding strategy used in FlexMatch.

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	0.092	0.000	0.004	0.000	0.000	0.008	0.006	0.000	0.884	0.006
C1	0.000	0.468	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.530
C2	0.000	0.000	0.664	0.006	0.216	0.024	0.026	0.014	0.050	0.000
C3	0.000	0.000	0.004	0.642	0.008	0.150	0.174	0.008	0.012	0.002
C4	0.000	0.000	0.002	0.002	0.654	0.302	0.020	0.014	0.006	0.000
C5	0.000	0.000	0.006	0.014	0.006	0.916	0.042	0.014	0.002	0.000
C6	0.000	0.000	0.000	0.000	0.000	0.002	0.996	0.000	0.002	0.000
C7	0.000	0.000	0.000	0.000	0.010	0.018	0.002	0.970	0.000	0.000
C8	0.000	0.002	0.000	0.000	0.000	0.000	0.000	0.002	0.990	0.006
C9	0.000	0.006	0.000	0.000	0.000	0.002	0.002	0.000	0.002	0.988

Table 3: Confusion matrix of AdaMatch on the CIFAR-10 dataset under an imbalanced class distribution.

FlexMatch is a semi-supervised learning algorithm that does not apply a single global threshold across all classes. Instead, it maintains a separate threshold for each class. This design allows easier pseudo-label acceptance for classes where the model is still weak and has stricter thresholds for classes where the model already performs well [14]. In an imbalanced scenario, such class-aware thresholding ensures that minority classes are not suppressed by majority ones, giving them a fairer chance to contribute during training.

By integrating this class-wise thresholding into AdaMatch, the proposed method combines the strengths of both approaches. Distribution alignment can still guide global prediction distribution, while FlexMatch's thresholding ensures that pseudo-label selection remains balanced across classes. This hybrid strategy is expected to mitigate AdaMatch's bias toward majority classes and yield more consistent performance across both balanced and imbalanced datasets.

The results of the proposed approach are presented in Table 4, which shows that integrating FlexMatch's class-wise thresholding into AdaMatch substantially improves minority-class performance. For example, accuracy for class 0 increased from 9% to 32% and for class 1 from 47% to 74%, while majority classes retained high accuracy above 90%. This demonstrates that the hybrid method mitigates AdaMatch's bias toward majority classes and achieves more balanced performance across all categories.

	Class number									
	0	1	2	3	4	5	6	7	8	9
Accuracy	0.32	0.738	0.678	0.752	0.942	0.948	0.992	0.970	0.926	0.976

Table 4: Per-class accuracy of the proposed hybrid AdaMatch–FlexMatch approach on the CIFAR-10 dataset.

Discussion

The experimental results highlight the different performance of FixMatch and AdaMatch to class imbalance in semi-supervised learning settings. While both methods performed comparably well in the balanced scenario and achieved evaluation accuracy around 95%, the robustness of AdaMatch decreased significantly under the imbalanced setup. This method experienced a severe performance drop, reaching an evaluation accuracy of just 73.6%, while FixMatch retained stable performance.

The main reason for this difference lies in how each method handles pseudo-labeling and class distribution. By using a fixed threshold, FixMatch mitigates the influence of class imbalance and preserves more uniform pseudo-label contributions. In contrast, AdaMatch introduces distribution alignment and relative confidence thresholding, both of which amplify imbalance effects. If the labeled data are imbalanced, distribution alignment forces pseudo-labels to mimic the imbalanced distribution, while relative confidence thresholds prevent minority classes from contributing effectively. This results in misclassification of minority classes into majority ones, as illustrated in the confusion matrices.

The per-class accuracy analysis provides further evidence of this imbalance sensitivity. Minority classes (e.g., class 0 and class 1) showed significant drops in accuracy with AdaMatch, while majority classes retained high accuracy, indicating the model's tendency toward majority categories. FixMatch, however, demonstrated more consistent performance across both minority and majority classes.

To enhance AdaMatch's performance on imbalanced datasets, we integrated class-wise thresholding from FlexMatch into AdaMatch. This adjustment leverages AdaMatch's adaptive distribution mechanisms while allowing minority classes to contribute more effectively through lower thresholds. As a result, the hybrid strategy mitigates AdaMatch's bias toward dominant classes. Whereas AdaMatch alone showed very low accuracy on underrepresented categories, the hybrid version achieved a more balanced performance without reducing the accuracy of majority classes. This demonstrates that class-wise thresholding is an effective solution to the imbalance sensitivity observed in the earlier results. Although further validation is needed to confirm its impact across different settings, the approach shows strong potential for improving the practicality of AdaMatch in real-world scenarios with imbalanced data.

Conclusion

This study investigated the effects of class imbalance on semi-supervised learning algorithms, focusing on FixMatch and AdaMatch using the CIFAR-10 dataset. The results demonstrate that while both methods perform well under balanced conditions, they respond very differently to class imbalance. FixMatch maintains stable performance even when labeled data are unequally distributed, whereas AdaMatch's adaptive mechanisms, designed for domain adaptation, backfire under imbalance, severely reducing accuracy on minority classes.

To overcome this limitation, we proposed modifying AdaMatch's relative confidence thresholding by incorporating FlexMatch's class-wise thresholding strategy. This hybrid design has the potential to preserve AdaMatch's advantages in handling domain shifts while improving fairness and consistency in imbalanced data scenarios. The evidence from Table 4 supports this claim, demonstrating that the hybrid approach can substantially rebalance per-class accuracy, closing the gap between minority and majority classes and making AdaMatch more reliable in imbalanced real-world conditions.

Overall, this study shows that improving the handling of imbalance is key to unlocking the full potential of semi-supervised learning in practical applications. Future work could involve:

1. **Extending to larger datasets:** Apply the hybrid AdaMatch–FlexMatch approach to larger and more complex benchmarks (e.g., CIFAR-100, ImageNet-LT) to validate its scalability and generalization ability.

2. **Exploring other imbalance scenarios:** Evaluate the method under a wider range of imbalance conditions, including long-tailed distributions and extreme minority-class cases, to better assess robustness.

3. **Adapting for domain adaptation tasks:** Investigate the effectiveness of the hybrid AdaMatch–FlexMatch approach in domain adaptation problems, examining whether the modification improves transfer performance or risks reducing AdaMatch's strengths in handling distribution shifts.

References

[1] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[2] Jiujun He, Bin Liu, and Guosheng Yin. Enhancing semi-supervised domain adaptation via effective target labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12385–12393, 2024.

[3] Yu-Chu Yu and Hsuan-Tien Lin. Semi-supervised domain adaptation with source label adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24100–24109, 2023.

[4] Mengmeng Zhan, Zongqian Wu, Rongyao Hu, Ping Hu, Heng Tao Shen, and Xiaofeng Zhu. Towards dynamic-prompting collaboration for source-free domain adaptation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 1643–1651, 2024.

[5] Song Tang, Wenxin Su, Mao Ye, and Xiatian Zhu. Source-free domain adaptation with frozen multimodal foundation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23711–23720, 2024.

[6] Xinyang Huang, Chuang Zhu, Ruiying Ren, Shengjie Liu, and Tiejun Huang. Source-free semantic regularization learning for semi-supervised domain adaptation. *IEEE Transactions on Multimedia*, 2025.

[7] Jichang Li, Guanbin Li, and Yizhou Yu. Inter-domain mixup for semi-supervised domain adaptation. *Pattern Recognition*, 146:110023, 2024.

[8] Wenyu Zhang, Qingmu Liu, Felix Ong Wei Cong, Mohamed Ragab, and Chuan-Sheng Foo. Universal semi-supervised domain adaptation by mitigating common-class bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23912–23921, 2024.

[9] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[10] Lingfei Deng, Changming Zhao, Zhenbang Du, Kun Xia, and Dongrui Wu. Semisupervised transfer boosting (ss-trboosting). *IEEE Transactions on Artificial Intelligence*, 5(7):3431–3444, 2024.

[11] Lihua Zhou, Nianxin Li, Mao Ye, Xiatian Zhu, and Song Tang. Source-free domain adaptation with class prototype discovery. *Pattern recognition*, 145:109974, 2024.

[12] Ba Hung Ngo, Ba Thinh Lam, Thanh Huy Nguyen, Quang Vinh Dinh, and Tae Jong Choi. Dual dynamic consistency regularization for semi-supervised domain adaptation. *IEEE Access*, 2024.

[13] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv:2106.04732*, 2021.

[14] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in neural information processing systems*, 34:18408–18419, 2021.