Effectiveness of Semi-Supervised learning - Mixmatch to FreeMatch, Towards FreeMatch++

Abbaas Alif Mohamed Nishar
Department of Computer Science
Georgia State University
Atlanta, GA 30303
amohamednishar1@student.gsu.edu

Abstract

The use of unlabeled data in semi-supervised learning has been shown to be effective in reducing the dependency on large-labeled datasets. We are conducting a survey of papers that were released in 2017 on the topic of semi-supervised learning up to 2023. One of the seminal papers that we chose was the MixMatch [1] that introduces the idea of Label consistency over augmentations to create pseudo labels for unlabeled data and then improve that over time. Then came FixMatch [2] in 2020 where they introduced the idea of optimizing the consistency of the label instead of using K different weak enhancements of unlabled data. Use a weak and strong augmented image and analyze consistency. This approach was useful, but the problem was that there was a manual thresholding factor which was used that limits the effectiveness of using all of the unlabled data then came along FreeMatch which was FixMatch added with Self-Adaptive thresholding and a fairness loss which ensures diverse and accurate predictions in unlabled dataset of SSL. In this work, we implement FreeMatch in pytorch and train it on a bench mark, which is CIFAR-10 with 4000 labels (400 labels per class). Then we introduced 3 new innovative techniques and incrementally modified the loss function of the FreeMatch paper. Although the modifications are not on par with the original technique, there are close enough without any explicit hyper parameter tuning which was done for the original paper. This shows potential promise in improving the paper and towards FreeMatch++. Code for this paper with results are available at https://github.com/abbaasalif/freematch-improved

1 Introduction

Recent advances in training large deep neural networks have been greatly influenced by the availability of extensive labeled datasets. However, the process of collecting labeled data can be costly for various learning tasks, as it requires the expertise of human annotators. This is particularly evident in medical tasks, where the acquisition of measurements involves expensive equipment and the labels are derived from time-consuming analyses carried out by multiple human experts. Additionally, data labels may contain sensitive information. In contrast, acquiring unlabeled data is often simpler or more cost-effective for many tasks.

Semi-supervised learning [3](SSL) aims to reduce the dependence on labeled data by allowing a model to utilize unlabeled data. To achieve this, many recent SSL approaches incorporate a loss term that is computed on unlabeled data. This loss term encourages the model to generalize better to unseen data. In current research, this loss term can be classified into three categories (further discussed in Section 2): entropy minimization [4, 5], consistency regularization, and generic regularization. Entropy minimization encourages the model to generate confident predictions on unlabeled data.

Preprint. Under review.

Consistency regularization encourages the model to produce the same output distribution when its inputs are perturbed. Generic regularization promotes good generalization and prevents overfitting to the training data.

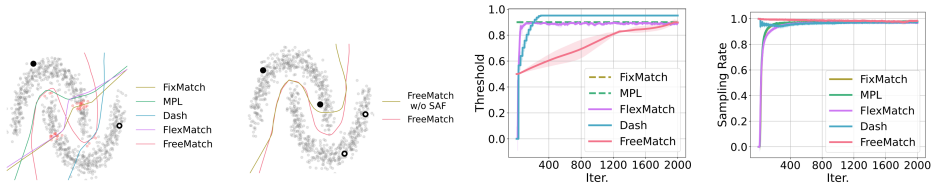
This paper presents MixMatch, an SSL algorithm that introduces a single loss term that effectively combines the dominant approaches to semi-supervised learning. Unlike previous methods, MixMatch targets all properties simultaneously, leading to the following advantages:

1. Experimental results demonstrate that MixMatch achieves state-of-the-art (SOTA) performance on all standard image benchmarks and reduces the error rate on CIFAR-10 by a factor of 4.
2. An ablation study reveals that MixMatch surpasses the combined performance of its individual components.
3. Furthermore, MixMatch has been shown to be beneficial for differentially private learning. It enables students in the PATE framework [6] to achieve new state-of-the-art results, enhancing both privacy guarantees and accuracy simultaneously.

Consequently, MixMatch introduces a unified loss term for unlabeled data that reduces entropy, maintains consistency, and is compatible with traditional regularization techniques.

But these techniques are older and require a lot of computations, especially intriduction of new augmentations can be an expensive operation and require numerous fourward and backward passes to get to a point where we have an acceptable performance. Also, we are not making any educated guesses as to where the model has performed well. *FreeMatch* introduces techniques that incorporate model learning status by analyzing the dark knowledge of the model that is learned by analyzing the logit layers of the model.

The key idea in paper like MixMatch and its predecessors is that the model should produce similar predictions or the same pseudo-labels for the same unlabeled data under different perturbations. The problem with these techniques is that we need a hyperparameter that is your threshold or ad hoc threshold adjusting scheme to compute the loss with only confident unlabeled samples. Most techniques like UDA [7] and FixMatch [2] maintain a fixed high threshold to ensure the quality of pseudo-labels. However, a fixed high threshold could lead to low data ultilization in the early training stages. The *FreeMatch* paper looks at the two-moon data set for a single labeled sample and shows that the model can obtain good decision limits for the data set.



(a) Decision boundary (b) Self-adaptive fairness (c) Confi. threshold (d) Sampling rate

Figure 1: Demonstration of how FreeMatch works on the "two-moon" dataset. (a) Decision boundary of FreeMatch and other SSL methods. (b) Decision boundary improvement of self-adaptive fairness (SAF) on two labeled samples per class. (c) Class-average confidence threshold. (d) Class average sampling rate of FreeMatch during training. Figures are from the FreeMatch paper

2 Related Work

To introduce the context for MixMatch, we first discuss the existing methods for SSL. Our main focus is on the state-of-the-art (SOTA) techniques and the ones that MixMatch is based on. There is a vast literature on semi-supervised learning (SSL) techniques that we do not cover in this discussion. (e.g., "transductive" models [8, 9, 10], graph-based methods [11, 3, 12], generative modeling [13, 14, 15, 16, 17], etc.).Additional information and comprehensive explanations are given in [11, 3]. In the subsequent discussion, we will make use of a general model $p_{model}(yx;\theta)$ that generates a probability distribution for class labels y given an input x and parameters θ.

2.1 Consistency Regularization

Data augmentation is a commonly used regularization technique in supervised learning. It involves applying transformations to the input data that are assumed to preserve the class semantics. In the context of image classification, typical data augmentation techniques include elastic deformation and adding noise to the input images. These transformations can significantly alter the pixel content of an image while keeping its label unchanged. [18, 19, 20]. In simple terms, this technique can increase the size of a training set by generating a large amount of new modified data. Consistency regularization is used in semi-supervised learning to apply data augmentation. The main idea behind this approach is that a classifier should produce the same class distribution for an unlabeled example, even after it has been augmented. To be more precise, consistency regularization ensures that an unlabeled example x should be classified similarly to Augment(x), which is an augmentation of the original example.

In the simplest case, for unlabeled points x, prior work [21, 22] adds the loss term

$$\|p_{model}(y|\text{Augment}(x);\theta) - p_{model}(y|\text{Augment}(x);\theta)\|_2^2 \tag{1}$$

Note that Augment(x) is a stochastic transformation, so the two terms in Eq. 1 are not identical. MixMatch utilizes a form of consistency regularization through the use of standard data augmentation for images(random horizontal flips and crops).

2.2 Entropy Minimization

Numerous semi-supervised learning techniques are based on the premise that the classifier's decision boundary should not intersect regions of high data density in the marginal distribution. One strategy to enforce this is by imposing the constraint that the classifier generates predictions with low entropy for unlabeled data. In [4], this is achieved through a loss term that minimizes the entropy of $P_{model}(yx;\theta)$ for unlabeled data x. In order to improve the results, the entropy minimization technique was merged with VAT in the study by Miyato et al. [23]. Another method, called "Pseudo-Label" [24], indirectly achieves entropy minimization by producing hard (1-hot) labels based on confident predictions made on unlabeled data. These labels are subsequently utilized as training targets in a conventional cross-entropy loss. Furthermore, MixMath implicitly performs entropy minimization employing a "sharpening" function in the target distribution for the unlabeled data.

2.3 Traditional Regularization

Regularization refers to the general approach of imposing a constraint on a model to make it harder to memorize training data and, therefore, to hopefully make it generalize better on unseen data [25]. MixMatch uses weight decay that penalizes the $\mathbb{L}_2$ norm of the model parameters [26]. They also use MixUp in MixMatch to encourage convex behavior "between" the examples. Mixup has a dual role of regularization and SSL method.

3 MixMatch

MixMatch was a technique that unifies several existing semi-supervised learning approaches. MixMatch guesses low-entropy and high-consistency labels for the unlabeled samples and also mixes unlabeled data with labeled examples using MixUp [27].

The core concepts that build MixMatch are **Entropy Minimization** - The classifier must be confident [24] [28] [29]. **Label Consistency** - should produce same class for weak augmentations of one image [30] [31].

The general architecture of MixMatch is shown in Figure 2. The models first have a batch of both labeled and unlabeled data. When we have labeled data, we use cross-entropy to train the model. When we encounter unlabeled data, we first perform weak K augmentations. Then we take the average output of the model for all the augmentations and then we apply a sharpening technique to get the pseudo-label - Sharpening is basically taking power $\frac{1}{T}$ and then applying softmax to get sharpened outputs. The unlabeled data we compute the Brier score loss. The labeled loss is given by

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} H(p, p_{\text{model}}(y|x;\theta))$$

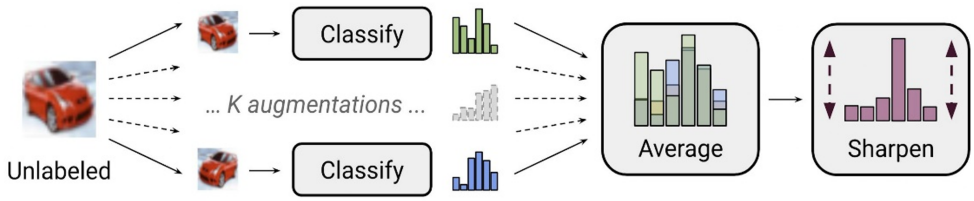


Figure 2: This Figure illustrates the technique of MixMatch.

where, $\mathcal{L}_{\mathcal{X}}$: This is the loss function calculated on the set of processed labeled examples $\mathcal{X}$. In a machine learning context, the loss function quantifies how well the model's predictions match the actual data. $\frac{1}{|\mathcal{X}'|}$ represents the average for all the examples in $\mathcal{X}'$. The vertical bars $|\cdot|$ denote the cardinality of a set, so $|\mathcal{X}'|$ is the number of elements in $\mathcal{X}'$. $H(p, p_{\text{model}}(y|x;\theta))$: H is the cross-entropy function, which measures the difference between two probability distributions. Here, p represents the true distribution (or true label) of the data, and $p_{\text{model}}(y|x;\theta)$ is the predicted probability distribution from the model, parameterized by θ, given the input x. The un-labled data loss is given by:

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - p_{\text{model}}(y|u;\theta)|_2 \, 2$$

where - $\mathcal{L}_{\mathcal{U}}$: This is the loss function calculated on the set of processed unlabeled examples $\mathcal{U}'$. Measures the quality of model predictions on unlabeled data. $\|q - p_{\text{model}}(y|u;\theta)\|_2^2$: This term represents the squared Euclidean norm (also known as the L2 norm) of the difference between q and $p_{\text{model}}(y|u;\theta)$. The q is the guessed label distribution for the unlabeled data obtained from the model's predictions, and $p_{\text{model}}(y|u;\theta)$ is the model's predicted distribution for the unlabeled data u, with θ being the model parameters. Squaring the L2 norm penalizes larger discrepancies between the predicted and guessed labels. Here, q is the label guessed after sharpening.

Algorithm 1: MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ of processed labeled examples (resp. $\mathcal{U}'$ of processed unlabeled examples with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \{(\mathbf{x}_b, \mathbf{p}_b); b \in (1, \ldots, B)\}$, batch of unlabeled examples $\mathcal{U} = \{(\mathbf{u}_b; b \in (1, \ldots, B)\})$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.

Output: A collection $\mathcal{X}'$ of processed labeled examples, a collection $\mathcal{U}'$ of processed unlabeled examples with guessed labels.

for $b = 1$ **to** B **do**
  $\tilde{\mathbf{x}}_b = \text{Augment}(\mathbf{x}_b)$ `// Apply data augmentation to` $\mathbf{x}_b$
  for $k = 1$ **to** K **do**
    $\tilde{\mathbf{u}}_{b,k} = \text{Augment}(\mathbf{u}_b)$ `// Apply` k^{th} `round of data augmentation to` $\mathbf{u}_b$
  end
  $\bar{\mathbf{q}}_b = \frac{1}{K} \sum_k \mathbf{p}_{\text{model}}(y|\tilde{\mathbf{u}}_{b,k};\theta)$ `// Compute average predictions across all augmentations of` $\mathbf{u}_b$
  $\mathbf{q}_b = \text{Sharpen}(\bar{\mathbf{q}}_b, T)$ `// Apply temperature sharpening to the average prediction`
end
$\mathcal{X}' = \{(\tilde{\mathbf{x}}_b, \mathbf{p}_b); b \in (1, \ldots, B)\}$ `// Augmented labeled examples and their labels`
$\mathcal{U}' = \{(\tilde{\mathbf{u}}_{b,k}, \mathbf{q}_b); b \in (1, \ldots, B), k \in (1, \ldots, K)\}$ `// Augmented unlabeled examples, guessed labels`
$W = \text{Shuffle}(\text{Concat}(\mathcal{X}', \mathcal{U}'))$ `// Combine and shuffle labeled and unlabeled data`
$\mathcal{X}' = \text{MixUp}(\tilde{\mathbf{x}}_i, \mathbf{W}_i); i \in (1, \ldots, |\mathcal{X}'|)$ `// Apply MixUp to labeled data and entries from` W
$\mathcal{U}' = \text{MixUp}(\tilde{\mathbf{u}}_i, \mathbf{W}_{i+|\mathcal{X}'|}); i \in (1, \ldots, |\mathcal{U}'|)$ `// Apply MixUp to unlabeled data and the rest of` W
return $\mathcal{X}', \mathcal{U}'$

But the problem with this algorithm is that we need to do multiple augmentations which are expensive, and then we have to run the algorithm multiple times which makes it computationally expensive.

4 FixMatch

FixMatch [2] is a semi-supervised learning algorithm that seeks to utilize the abundance of unlabeled data in conjunction with a smaller set of labeled data to train machine learning models. The central idea of FixMatch is to predict pseudo-labels for unlabeled data using the predictions made by the model on weakly augmented versions of that data.

Algorithm 2: FixMatch algorithm.

Input: Labeled batch $\mathcal{X} = \{(\mathbf{x}_b, \mathbf{p}_b) : b \in (1, \dots, B)\}$, unlabeled batch $\mathcal{U} = \{\mathbf{u}_b : b \in (1, \dots, \mu B)\}$, confidence threshold τ, unlabeled data ratio μ, unlabeled loss weight λ_u.

Output: Loss $\mathcal{L} = \mathcal{L}_s + \lambda_u \mathcal{L}_u$.

$\mathcal{L}_s = \frac{1}{B} \sum_{b=1}^{B} H(\mathbf{p}_b, \alpha(\mathbf{x}_b))$ `// Cross-entropy loss for labeled data`

for $b = 1$ **to** μB **do**

  $\mathbf{q}_b \leftarrow p_m(y|\alpha(\mathbf{u}_b))$ `// Compute prediction after applying weak data augmentation of` $\mathbf{u}_b$

$\mathcal{L}_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}\{\max(\mathbf{q}_b) > \tau\} H(\text{argmax}(\mathbf{q}_b), p_m(y|A(\mathbf{u}_b)))$ `// Cross-entropy loss with pseudo-label and confidence for unlabeled data`

return $\mathcal{L}_s + \lambda_u \mathcal{L}_u$

So, now you are not using multiple augumentations anymore and use only one augmentation per image. Figure 3 shows how the technique works. we first train the model on the batch data with labeled data, then we take a weakly augmented unlabeled data and then get the corresponding pseudo-label then we take the same image with strong augmentation then get the prediction and apply cross-entropy over it.

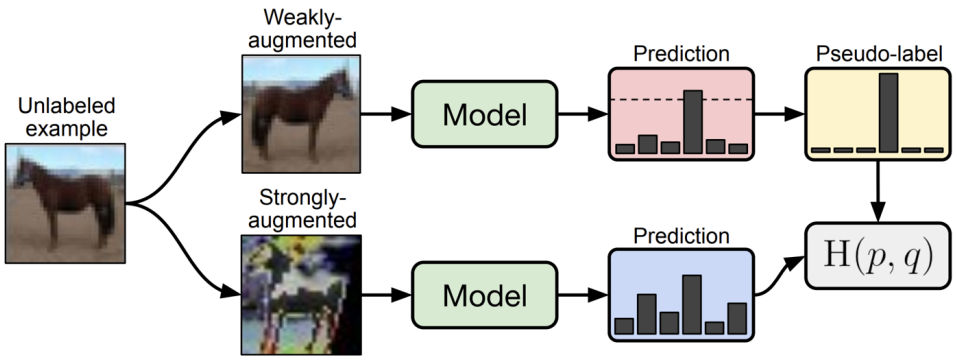


Figure 3: This Figure illustrates the technique of FixMatch.

The problem with this technique is that the there is a fixed threshold value in the unlabled loss $\mathcal{L}_u$ which limits the feedback that we can use for the model and that only improves performance on images that look similar to each other and not have diversity over classes.

5 FreeMatch

FreeMatch [32] extends the work done in FixMatch but introduces a self-adaptive thresholding.

5.1 Self-Adaptive Thresholding Loss

We assert that the crucial factor in determining SSL thresholds is their reflection of the learning status. The learning impact can be gauged by the prediction confidence of a properly calibrated model (Guo et al., 2017). Consequently, we introduce self-adaptive thresholding (SAT) that autonomously sets and adaptively modifies the confidence threshold for each class, capitalizing on the model's predictions during the training process. Initially, SAT estimates a global threshold based on the EMA of the model's confidence. Subsequently, SAT adjusts the global threshold using the local class-specific thresholds, which are estimated based on the EMA of the probability for each class from the model. At the onset of training, the threshold is set low to accommodate more potentially correct samples. As the confidence of the model increases, the threshold is adaptively raised to exclude potentially incorrect samples, thereby minimizing the confirmation bias. Hence, as shown in Figure 2, we characterize SAT as $tau_t(c)$, representing the threshold for class c at the iteration t.

Self Adaptive Global Threshold The global threshold is decided based on two principles - the global threshold in SAT should be related to the model's confidence on unlabeled data, reflecting the overall learning status. Furthermore, it is imperative that the threshold incrementally escalates throughout the training process to effectively eliminate erroneous pseudo-labels. The global threshold, denoted as τ_t, is established as the mean confidence derived from the interaction of the model with unlabeled data, with t symbolizing the t-th time step. Nevertheless, the computation of confidence for the entirety of unlabeled data at each temporal increment or even each training epoch would be an excessively time-intensive process due to the substantial volume of data. As an alternative, we approximate the global confidence using the exponential moving average (EMA) of the confidence at each individual training step. The initial value of τ_t is set as $frac1C$, where C denotes the total number of classes. The global threshold τ_t is subsequently defined and adjusted as follows:

$$\tau_t = \begin{cases} \frac{1}{C_t}, & \text{if } t = 0, \\ \lambda \tau_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} \max(q_b), & \text{otherwise}, \end{cases} \tag{2}$$

where, $\lambda \in (0, 1)$ is the momentum decay of EMA.

Self-adaptive Local Threshold The local threshold aims to adapt the global threshold in a class-specific manner to account for the intra-class diversity and the possible class adjacency. We compute the expectation of the model's predictions on each class c to estimate the class-specific learning status:

$$\tilde{p}_t(c) = \begin{cases} \frac{1}{C_t}, & \text{if } t = 0, \\ \lambda \tilde{p}_{t-1}(c) + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} q_b(c) & \text{otherwise}, \end{cases} \tag{3}$$

where $\tilde{p}_t = [\tilde{p}_t(1), \tilde{p}_t(2), \ldots, \tilde{p}_t(C)]$ is the list containing all $\tilde{p}_t(c)$.

Intergrating the global and local thresholds, we get the final self adaptive threshold $\tau_t(c)$ as:

$$\tau_t(c) = \text{MaxNorm}(\tilde{p}_t(c)) \cdot \tau_t = \frac{\tilde{p}_t(c)}{\max\{\tilde{p}_t(c) : c \in [C]\}} \cdot \tau_t, \tag{4}$$

where MaxNorm is the Maximum Normalization operation (i.e., $x' = \frac{x}{\max(x)}$). Finally the unsupervised training objective $\mathcal{L}_u$ at the t-th iteration is:

$$\mathcal{L}_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) > \tau_t(\arg\max(q_b))) \cdot \mathcal{H}(\tilde{q}_b, Q_b). \tag{5}$$

5.2 Self-Adaptive Fairness Loss

The FreeMatch paper also introduces a class fariness objective to encourage the model or make diverse predictions for each class, and thus produce a meaningful self-adaptive threshold, especially under the settings where the labeled data are rare. They use the EMA of the model predictions $\tilde{p}_t$ from 3 as an estimate of the expectation of prediction distribution over unlabeled data. We optimize the cross-entropy of $\tilde{p}_t$ and $\bar{p} = \mathbb{E}_{\mu B}[p_m(y|\Omega(u_b)))]$ over the mini-batch as an estimate of $\mathcal{H}(\mathbb{E}_u[p_m(y|u)])$. The underlying pseduo label distribution may not be uniform, they propose to modulate the fairness objective in a self-adaptive technique, i.e., normalizing the expectation

of probability by the histogram distribution of the pseudo labels to counter the negative effect of imbalance as:

$$\begin{aligned} \bar{p} &= \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau_t(\arg\max(q_b)))Q_b, \\ \bar{h} &= \text{Hist}_{\mu B}\left(\mathbb{1}(\max(q_b) \geq \tau_t(\arg\max(q_b)))\hat{Q}_b\right). \end{aligned} \tag{6}$$

Similarly like $\tilde{p}_t$ we also compute $\tilde{h}_t$ as:

$$\tilde{h}_t = \lambda \tilde{h}_{t-1} + (1-\lambda)\text{Hist}_{\mu B}(\hat{q}_b). \tag{7}$$

Then Self-Adaptive Fairness Loss is formulated as:

$$\mathcal{L}_f = \mathcal{H}\left(\text{SumNorm}\left(\frac{\tilde{p}_t}{\tilde{h}_t}\right), \text{SumNorm}\left(\frac{\bar{p}}{\bar{h}}\right)\right) \tag{8}$$

where SumNorm $= (\cdot) \,/\, \sum(\cdot)$. In essence, SAF encourages the expectation of model output for each batch to be close to the marginal class distribution of the model.

So, the overall loss of the freematch algorithm can be formulated as

$$\mathcal{L} = \mathcal{L}_s + w_u \mathcal{L}_u + w_f \mathcal{L}_f, \tag{9}$$

where w_u and w_f are represents the loss weights for SAT and SAF respectively.

The algorithm of freematch can be written as follows:

Algorithm 3: FreeMatch algorithm at the t-th iteration.

Input: Number of classes C, labeled batch $\mathcal{X} = \{(\mathbf{x}_b, y_b) : b \in (1, 2, ..., B)\}$, unlabeled batch $\mathcal{U} = \{\mathbf{u}_b : b \in (1, 2, ..., \mu B)\}$, unsupervised loss weight w_u, fairness loss weight w_f, and EMA decay λ.

Compute L_s for labeled data

$L_s = \frac{1}{B}\sum_{b=1}^{B} H(y_b, p_m(y|\mathbf{x}_b))$; `// Cross-entropy loss for labeled data`

Update the global threshold

$\tau_t = \lambda\tau_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B}\max(q_b)$; `//` q_b `is an abbreviation of` $p_m(y|\mathbf{u}_b)$`, shape of` τ_t`:` $[1]$

Update the local threshold

$\tilde{p}_t = \lambda\tilde{p}_{t-1} + (1-\lambda)\frac{1}{\mu B}\sum_{b=1}^{\mu B} q_b$; `// Shape of` $\tilde{p}_t$`:` $[C]$

Update histogram for $\tilde{p}_t$

$\tilde{h}_t = \lambda\tilde{h}_{t-1} + (1-\lambda)\text{Hist}_{\mu B}(q_b)$; `// Shape of` $\tilde{h}_t$`:` $[C]$

for $c = 1$ **to** C **do**

  $\tau_t(c) = \text{MaxNorm}(\tilde{p}_t(c))$; `// Calculate SAT`

Compute L_u on unlabeled data

$L_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbb{1}(\max(q_b) \geq \tau_t(\arg\max(q_b)))H(\tilde{q}_b, Q_b)$; `//`

Compute the expectation of probability on unlabeled data.

$\tilde{p} = \frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbb{1}(\max(q_b) \geq \tau_t(\arg\max(q_b)))Q_b$; `//`

Compute the histogram for $\tilde{p}$

$\tilde{h} = \text{Hist}_{\mu B}(\mathbb{1}(\max(q_b) \geq \tau_t(\arg\max(q_b)))\hat{Q}_b)$; `//`

Compute L_f on unlabeled data.

$L_f = -H(\text{SumNorm}(\tilde{p}), \text{SumNorm}(\tilde{h}))$; `//`

return $L_s + w_u \cdot L_u + w_f \cdot L_f$

6 Experiments

We tried to implement the experiments of the original paper and implemented it using pytorch. The original freematch paper has several benchmarks and was run up to 2^20 epochs, but we are implementing the CIFAR-10 dataset, which has 50,000 images for 10 classes in the training set and 10,000 images in its test set. For the labeled set we randomly take 400 labeled images from each class, which is a total of 4000 labeled data points and then the rest is unlabled and learned in the semi-supervised fashion. We also train the models only up to 45000 epochs.

6.1 Modification 1

We are trying to improve the paper incrementally using some modifications to the original algorithm. So modification one had added to the current losses extra terms as an improvement. We modified **the supervised loss** $\mathcal{L}_s$ to include the smoothing of the label [33]. It is a technique where instead using the direct one-hot vector (which has 1 for the class index corresponding to the true class of the image and zero in the rest of the indexes) we use say $1 - \epsilon + \frac{\epsilon}{C}$ for the true class and $\frac{\epsilon}{C}$ for the rest of the classes in one vector. This can be mathematically formulated as follows.

$$\mathcal{L}_s(p, y') = -\sum_{i=1}^{C} y'_i \log(p_i) \tag{10}$$

where

$$y'_i = \begin{cases} 1 - \epsilon + \frac{\epsilon}{C} & \text{if } i = \text{true class} \\ \frac{\epsilon}{C} & \text{otherwise} \end{cases} \tag{11}$$

We also added improvements to **consistency loss** that is used in the FixMatch paper and has been adopted into Freematch. The consistency loss now becomes follwing.

Given logits l, true targets y, an optional mask m, consistency weight λ_c, and regularization weight λ_r, the consistency loss with softmax weights is defined as:

$$\mathcal{L}_{\text{cons}} = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} m_i \cdot \max(\text{softmax}(l_i)) \cdot \mathcal{H}(y_i, \text{softmax}(l_i)) \\ \quad + \lambda_c \cdot \frac{1}{N-1} \sum_{i=1}^{N-1} \|\text{softmax}(l_i) - \text{softmax}(l_{i+1})\|_2^2 \\ \quad + \lambda_r \cdot \frac{1}{N} \sum_{i=1}^{N} \|l_i\|_2^2, & \text{if mask is provided} \\ \frac{1}{N} \sum_{i=1}^{N} H(y_i, \text{softmax}(l_i)) + \lambda_r \cdot \frac{1}{N} \sum_{i=1}^{N} \|l_i\|_2^2, & \text{otherwise} \end{cases} \tag{12}$$

In this formula:

- $\max(\text{softmax}(l_i))$ represents the maximum softmax probability (confidence) for the i-th sample.
- $\mathcal{H}(y_i, \text{softmax}(l_i))$ is the cross-entropy loss between the true label y_i and the softmax probabilities of the logits l_i.
- m_i is an element of the mask which, when provided, determines whether a sample loss is included in the final calculation. If $m_i = 0$, the sample loss does not contribute to the total loss.

In this context, consistency loss is employed as an integral component of self-adaptive thresholding, wherein the primary function of the SAT mechanism is to establish thresholds for the generation of pseudo-labels, which subsequently serve as targets within consistency loss. A notable limitation of this loss function lies in the segment in which we incorporate the loss term of consistency, denoted by $\lambda_c \cdot \frac{1}{N-1} \sum_{i=1}^{N-1} \|\text{softmax}(l_i) - \text{softmax}(l_{i+1})\|_2^2$. This term would have been beneficial if the batch had temporal relationships; however, it did not, rendering the inclusion of this term a regrettable decision in retrospective analysis. Also we introduced a bunch of hyperparmeters that need to be carefully tuned, which is left out as future work.

Improvements in SAT loss we added temperature to the logits for regularization and then added a diversity loss within the threshold loss for more diverse predictions.

so, we can mathematically formulate it as folliws.

Given logits $\text{logits}_{\text{ulb\_w}}$ (logits of weakly augmented unlabeled sample)and $\text{logits}_{\text{ulb\_s}}$(logits of strongly augmented unlabeled sample), temperature temperature, exponential moving average of the self-adaptive threshold sat_ema, and diversity weight diversity_weight, the loss of the self-adaptive threshold is calculated as follows:

Update temperature-scaled softmax probabilities:

$$\text{probs}_{\text{ulb\_w}} = \text{softmax}\left(\frac{\text{logits}_{\text{ulb\_w}}}{\text{temperature}}, \text{dim} = -1\right) \tag{13}$$

Update τ_t and p_t using a weighted moving average:

$$\tau_t = \tau_t \cdot \text{sat\_ema} + (1 - \text{sat\_ema}) \cdot \text{mean}(\max(\text{probs}_{\text{ulb\_w}})) \tag{14}$$

$$p_t = p_t \cdot \text{sat\_ema} + (1 - \text{sat\_ema}) \cdot \text{mean}(\text{probs}_{\text{ulb\_w}}, \text{dim} = 0) \tag{15}$$

Update label histogram:

$$\text{label\_hist} = \text{label\_hist} \cdot \text{sat\_ema} + (1 - \text{sat\_ema}) \cdot \left(\frac{\text{histogram}}{\text{sum}(\text{histogram})}\right) \tag{16}$$

The consistency loss $\mathcal{L}_{\text{cons}}$ is calculated with the masked logits, and diversity loss The "SelfAdaptiveThresholdLoss" incorporates several components in its calculation:

1. **Temperature-scaled softmax** - This applies a temperature parameter to the logits before computing the softmax probabilities. The temperature affects the distribution's entropy, making it more uniform (higher temperature) or more peaked (lower temperature).
2. **Weighted moving average updates** for τ_t and p_t - The algorithm maintains a moving average of the threshold and class probabilities, which are updated with each batch using the exponential moving average method.
3. **Label histogram update** - A histogram of the predicted classes is maintained and updated similarly, providing a historical view of the predictions of the classes.
4. **Masked consistency loss** - A mask is created based on the threshold τ_t and the updated probabilities. This mask is used to compute the consistency loss, likely by ignoring certain samples that do not meet a confidence threshold.
5. **Diversity loss** - A diversity loss term encourages the model to spread its predictions across the classes, potentially preventing it from becoming too confident in a small subset of classes.
6. **Total SAT loss** - Total SAT loss is the sum of consistency loss and scaled diversity loss.

$$\mathcal{L}_{\text{total SAT}} = \mathcal{L}_{\text{cons}}(\text{logits}_{\text{ulb\_s}}, \text{max\_idx\_w}, \text{mask}) + \text{diversity\_weight} \cdot \mathcal{L}_{\text{div}}(\text{logits}_{\text{ulb\_s}}) \tag{17}$$

where

- $\mathcal{L}_{\text{cons}}$ is the masked consistency loss based on the strongly augmented logits and the mask determined by the self-adaptive threshold.
- $\mathcal{L}_{\text{div}}$ is the diversity loss, which is the negative sum of the element-wise product of softmax probabilities and log softmax probabilities of strongly augmented logits.
- diversity_weight is a hyperparameter to balance the two loss components.

The mask for consistency loss is computed as:

$$\text{mask} = \max(\text{softmax}(\frac{\text{logits}_{\text{ulb\_w}}}{\text{temperature}})) \geq \tau_t \cdot \frac{p_t}{\max(p_t)} \tag{18}$$

Improvements in Self-Adaptive Fairness Loss We also added two new terms to the SAF loss. Self-Entropy for the logits of the unlabled weak augmentations which would calculate predicted probabilities for the diversity and also a KL divergence term for additional regularization.

We can mathematically model that as follows.

$$\mathcal{L}_{\text{entropy}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} p_{i,c}\log(p_{i,c}+\epsilon) \tag{19}$$

$$\mathcal{L}_{\text{CE}} = \sum_{c=1}^{C} p_{t,c}\log(\text{label\_hist}_c+\epsilon) \tag{20}$$

$$\mathcal{L}_{\text{KL}} = \sum_{c=1}^{C} \text{label\_hist}_c(\log(\text{label\_hist}_c+\epsilon) - \log(p_{t,c}+\epsilon)) \tag{21}$$

$$\mathcal{L}_{\text{total SAF}} = -\mathcal{L}_{\text{CE}} + \lambda_{\text{entropy}}\cdot\mathcal{L}_{\text{entropy}} + \lambda_{\text{KL}}\cdot\mathcal{L}_{\text{KL}} \tag{22}$$

where label_hist is the $\bar{h}$ as described in the paper.

6.2 Modification 2

It is very similar to modification 1 but we removed some components that we thought were not necessary.

1. **Supervised Loss** Use as it is we did not use Label Smoothing
2. **Consistency Loss** we removed other parameters and only kept L2 regualrization.
3. **SAT loss** removed temperature for logits and kept the diversity loss
4. **SAF loss** same as modification 1

6.3 Modification 3

In modification 3, we retained all the losses from the original implementation, but then we added an attention layer in the logits to weight the importance of each batch sample as we train over the batch and these weight get updated as we train the model further.

The modification of the new $\mathcal{L}_{SAT}$ is:

$$\tau_t = \tau_t \times \text{sat\_ema} + (1-\text{sat\_ema}) \times \text{mean}(\max(\text{softmax}(\text{logits\_ulb\_w}))) \tag{23}$$

$$p_t = p_t \times \text{sat\_ema} + (1-\text{sat\_ema}) \times \text{mean}(\text{softmax}(\text{logits\_ulb\_w}), \text{dim}=0) \tag{24}$$

$$\text{attention\_weights} = \sigma(\text{logits\_ulb\_w}\cdot W^T + b) \tag{25}$$

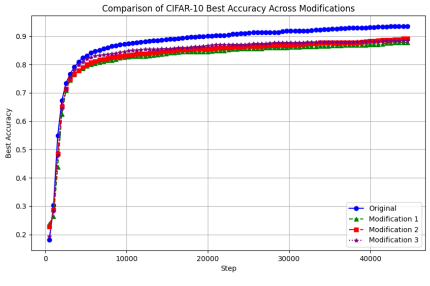
$$\text{weighted\_logits\_ulb\_w} = \text{logits\_ulb\_w}\cdot\text{attention\_weights} \tag{26}$$

$$\text{mask} = \text{max\_probs\_w} \geq \tau_t \times \frac{p_t}{\max(p_t)} \tag{27}$$

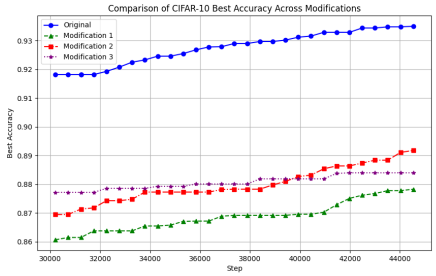
Here, σ represents the sigmoid function, W represents the weights, and b represents the biases in the dynamic linear attention mechanism. The weights and biases are dynamically adjusted based on the input feature size, and the sigmoid function is used to ensure the attention weights are between 0 and 1.

7 Results

We ran the original algorithm and the modifications for 45K epochs the original technique hyperparameters were taken from the paper and while the other we just used best-practice values from other papers and techniques as we commonly see them and based on intuition. From Figure 4, we can see that the original technique performs the best followed by modification 2, modification 3 and modification 1 respectively. From Table 1 we can see the best accuracy values in the last step of training. We also have logged a lot of other metrics, which can be found in the tensorboard logs, which is attached with the code. The code is also available at `https://github.com/abbaasalif/freematch-improved`



(a) Best Validation Accuracy values plotted over the train steps



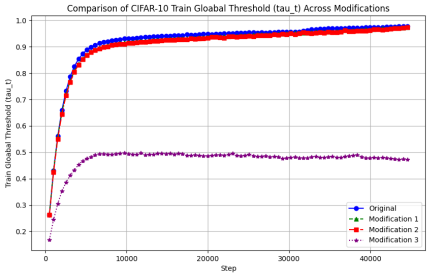
(b) Zoomed version over the last steps

Figure 4: These Figures show the Best Validation Accuracy over all the techniques proposed and then the zoomed version over the last 20k epochs.

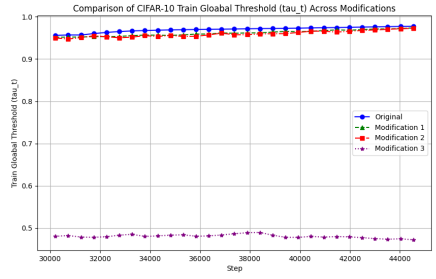
Configuration	Best Accuracy
Original	0.935
Modification 1	0.878
Modification 2	0.891
Modification 3	0.884

Table 1: Best Accuracy Values at the Last Step Across Different Configurations

Another notable result was the global threshold τ_t (Refer Figure 5), which as per the paper should first be small and then at the end of the training should be closer to 1. In the case of modification 3 this did not happen which we see as an issue and want to investigate further.



(a) Train τ_t values plotted over the train steps



(b) Train τ_t over the last steps

Figure 5: These Figures show the Train global threshold τ_t over all the techniques proposed and then the zoomed version over the last 20k epochs.

8 Conclusion

The results show good promise that these techniques can reach on par or even surpass the state-of-the-art technique FreeMatch. We might need to do some more hyperparameter tuning and then see where

these results take us. Also modification 3 global threshold inspection gave some information about the problems it has, we might need to look into it further. The goal of this project was to implement the original algorithm and propose some incremental improvements to it, which is the main purpose of the project.

9 Future Work

The furture work is to finish the hyper parameter tuning and look into modification 3 why there is no increase in the global threshold. Also, we would like to extend the same technique in the domain of image segmentation where we are doing pixel-wise classification and also need to consider spatial relationships, but we would like to use the principles of free match.

References

[1] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[3] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. Adaptive Computation and Machine Learning series. MIT Press, London, England, September 2006.

[4] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[5] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, 07 2013.

[6] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. October 2016.

[7] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[8] Alex Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. *CoRR*, abs/1301.7375, 2013.

[9] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 290–297. AAAI Press, 2003.

[10] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[11] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 912–919. AAAI Press, 2003.

[12] Bin Liu, Zhirong Wu, Han Hu, and Stephen Lin. Deep metric transfer for label propagation with limited annotated data. *CoRR*, abs/1812.08781, 2018.

[13] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[14] Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, page 87–94, USA, 2006. IEEE Computer Society.

[15] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[16] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 921–928, Madison, WI, USA, 2011. Omnipress.

[17] Ian J. Goodfellow, Aaron Courville, and Yoshua Bengio. Spike-and-slab sparse coding for unsupervised feature discovery, 2012.

[18] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, December 2010.

[19] Patrice Simard, David Steinkraus, and John Platt. Best practices for convolutional neural networks applied to visual document analysis. pages 958–962, 01 2003.

[20] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation policies from data. May 2018.

[21] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning, 2017.

[22] Vikas Verma, Kenji Kawaguchi, Alex Lamb, Juho Kannala, Arno Solin, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. March 2019.

[23] Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2019.

[24] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin 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.

[25] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, page 5–13, New York, NY, USA, 1993. Association for Computing Machinery.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[27] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[28] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning, 2018.

[29] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[30] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning, 2017.

[31] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[32] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning, 2023.

[33] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help?, 2020.