
FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence

Kihyuk Sohn* David Berthelot* Chun-Liang Li Zizhao Zhang Nicholas Carlini
Ekin D. Cubuk Alex Kurakin Han Zhang Colin Raffel
Google Research
{kihyuks, dberth, chunliang, zizhaoz, ncarlini,
cubuk, kurakin, zhanghan, craffel}@google.com

Abstract

Semi-supervised learning (SSL) provides an effective means of leveraging unlabeled data to improve a model’s performance. This domain has seen fast progress recently, at the cost of requiring more complex methods. In this paper we propose FixMatch, an algorithm that is a significant simplification of existing SSL methods. FixMatch first generates pseudo-labels using the model’s predictions on weakly-augmented unlabeled images. For a given image, the pseudo-label is only retained if the model produces a high-confidence prediction. The model is then trained to predict the pseudo-label when fed a strongly-augmented version of the same image. Despite its simplicity, we show that FixMatch achieves state-of-the-art performance across a variety of standard semi-supervised learning benchmarks, including 94.93% accuracy on CIFAR-10 with 250 labels and 88.61% accuracy with 40 – just 4 labels per class. We carry out an extensive ablation study to tease apart the experimental factors that are most important to FixMatch’s success. The code is available at <https://github.com/google-research/fixmatch>.

1 Introduction

Deep neural networks have become the de facto model for computer vision applications. Their success is partially attributable to their *scalability*, i.e., the empirical observation that training them on larger datasets produces better performance [30, 20, 42, 55, 41, 21]. Deep networks often achieve their strong performance through supervised learning, which requires a labeled dataset. The performance benefit conferred by the use of a larger dataset can therefore come at a significant cost since labeling data often requires human labor. This cost can be particularly extreme when labeling must be done by an expert (for example, a doctor in medical applications).

A powerful approach for training models on a large amount of data without requiring a large amount of labels is *semi-supervised learning* (SSL). SSL mitigates the requirement for labeled data by providing a means of leveraging unlabeled data. Since unlabeled data can often be obtained with minimal human labor, any performance boost conferred by SSL often comes with low cost. This has led to a plethora of SSL methods that are designed for deep networks [33, 46, 24, 51, 4, 54, 3, 25, 45, 52].

A popular class of SSL methods can be viewed as producing an artificial label for unlabeled images and training the model to predict the artificial label when fed unlabeled images as input. For example, pseudo-labeling [25] (also called self-training [32, 55, 44, 47]) uses the model’s class prediction as a label to train against. Similarly, consistency regularization [2, 46, 24] obtains an artificial label using the model’s predicted distribution after randomly modifying the input or model function.

*Equal contribution.

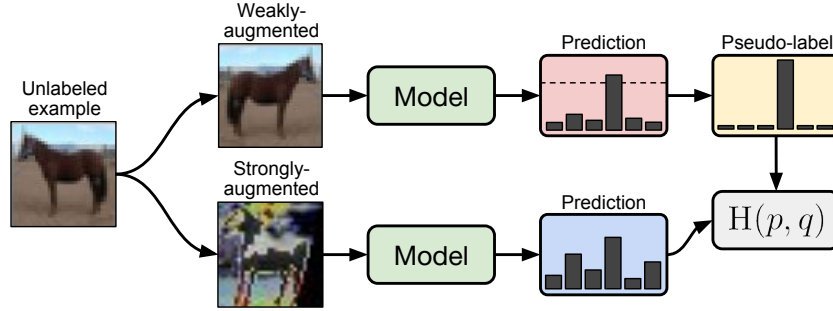


Figure 1: Diagram of FixMatch. A weakly-augmented image (top) is fed into the model to obtain predictions (red box). When the model assigns a probability to any class which is above a threshold (dotted line), the prediction is converted to a one-hot pseudo-label. Then, we compute the model’s prediction for a strong augmentation of the same image (bottom). The model is trained to make its prediction on the strongly-augmented version match the pseudo-label via a cross-entropy loss.

In this work, we break the trend of recent state-of-the-art methods that combine increasingly complex mechanisms [4, 54, 3] and produce a method that is simpler, but also more accurate. Our algorithm, FixMatch, produces artificial labels using both consistency regularization and pseudo-labeling. Crucially, the artificial label is produced based on a *weakly*-augmented unlabeled image (e.g., using only flip-and-shift data augmentation) which is used as a target when the model is fed a *strongly*-augmented version of the same image. Inspired by UDA [54] and ReMixMatch [3], we leverage Cutout [14], CTAugment [3], and RandAugment [11] for strong augmentation, which all produce heavily-distorted versions of a given image. Following the approach of pseudo-labeling [25], we only retain an artificial label if the model assigns a high probability to one of the possible classes. A diagram of FixMatch is shown in fig. 1.

Despite its simplicity, we show that *FixMatch obtains state-of-the-art performance on the most commonly-studied SSL benchmarks*. For example, FixMatch achieves 94.93% accuracy on CIFAR-10 with 250 labeled examples compared to the previous state-of-the-art of 93.73% [3] in the standard experimental setting from [36]. We also explore the limits of our approach by applying it in the extremely-scarce-labels regime, obtaining 88.61% accuracy on CIFAR-10 with only 4 labels per class. Since FixMatch is a simplification of existing approaches but achieves substantially better performance, we include an extensive ablation study to determine which factors contribute the most to its success. A key benefit of FixMatch being a simplification of existing methods is that it requires many fewer additional hyperparameters. As such, it allows us to perform an extensive ablation study of each of them. Our ablation study also includes basic fully-supervised learning experimental choices that are often ignored or not reported when new SSL methods are proposed (such as the optimizer or learning rate schedule).

2 FixMatch

FixMatch is a combination of two approaches to SSL: Consistency regularization and pseudo-labeling. Its main novelty comes from the combination of these two ingredients as well as the use of a separate weak and strong augmentation when performing consistency regularization. In this section, we first review consistency regularization and pseudo-labeling before describing FixMatch in detail. We also describe the other factors, such as regularization, which contribute to FixMatch’s empirical success.

For an L -class classification problem, let $\mathcal{X} = \{(x_b, p_b) : b \in (1, \dots, B)\}$ be a batch of B labeled examples, where x_b are the training examples and p_b are one-hot labels. Let $\mathcal{U} = \{u_b : b \in (1, \dots, \mu B)\}$ be a batch of μB unlabeled examples where μ is a hyperparameter that determines the relative sizes of \mathcal{X} and \mathcal{U} . Let $p_m(y | x)$ be the predicted class distribution produced by the model for input x . We denote the cross-entropy between two probability distributions p and q as $H(p, q)$. We perform two types of augmentations as part of FixMatch: strong and weak, denoted by $\mathcal{A}(\cdot)$ and $\alpha(\cdot)$ respectively. We describe the form of augmentation we use for \mathcal{A} and α in section 2.3.

2.1 Background

Consistency regularization is an important component of recent state-of-the-art SSL algorithms. Consistency regularization utilizes unlabeled data by relying on the assumption that the model should output similar predictions when fed perturbed versions of the same image. This idea was first proposed in [2] and popularized by [46, 24], where the model is trained both via a standard supervised classification loss and on unlabeled data via the loss function

$$\sum_{b=1}^{\mu B} \|p_m(y | \alpha(u_b)) - p_m(y | \alpha(u_b))\|_2^2 \quad (1)$$

Note that both α and p_m are stochastic functions, so the two terms in eq. (1) will indeed have different values. Extensions to this idea include using an adversarial transformation in place of α [33], using a running average or past model predictions for one invocation of p_m [51, 24], using a cross-entropy loss in place of the squared ℓ^2 loss [33, 54, 3], using stronger forms of augmentation [54, 3], and using consistency regularization as a component in a larger SSL pipeline [4, 3].

Pseudo-labeling leverages the idea of using the model itself to obtain artificial labels for unlabeled data [32, 47]. Specifically, this refers to the use of “hard” labels (i.e., the $\arg \max$ of the model’s output) and only retaining artificial labels whose largest class probability fall above a predefined threshold [25]. Letting $q_b = p_m(y | u_b)$, pseudo-labeling uses the following loss function:

$$\frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) H(\hat{q}_b, q_b) \quad (2)$$

where $\hat{q}_b = \arg \max(q_b)$ and τ is the threshold. For simplicity, we assume that $\arg \max$ applied to a probability distribution produces a valid “one-hot” probability distribution. The use of a hard label makes pseudo-labeling closely related to entropy minimization [17, 45], where the model’s predictions are encouraged to be low-entropy (i.e., high-confidence) on unlabeled data.

2.2 Our Algorithm: FixMatch

The loss function for FixMatch consists of two cross-entropy loss terms: a supervised loss ℓ_s applied to labeled data and an unsupervised loss ℓ_u . Specifically, ℓ_s is just the standard cross-entropy loss on weakly augmented labeled examples:

$$\ell_s = \frac{1}{B} \sum_{b=1}^B H(p_b, p_m(y | \alpha(x_b))) \quad (3)$$

FixMatch computes an artificial label for each unlabeled example² which is then used in a standard cross-entropy loss. To obtain an artificial label, we first compute the model’s predicted class distribution given a *weakly*-augmented version of a given unlabeled image: $q_b = p_m(y | \alpha(u_b))$. Then, we use $\hat{q}_b = \arg \max(q_b)$ as a pseudo-label, except we enforce the cross-entropy loss against the model’s output for a *strongly*-augmented version of u_b :

$$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) H(\hat{q}_b, p_m(y | \mathcal{A}(u_b))) \quad (4)$$

where τ is a scalar hyperparameter denoting the threshold above which we retain a pseudo-label. The loss minimized by FixMatch is simply $\ell_s + \lambda_u \ell_u$ where λ_u is a fixed scalar hyperparameter denoting the relative weight of the unlabeled loss. We present a complete algorithm for FixMatch in algorithm 1 of the supplementary material.

While eq. (4) is similar to the pseudo-labeling loss in eq. (2), it is crucially different in that the artificial label is computed based on a weakly-augmented image and the loss is enforced against the model’s output for a strongly-augmented image. This introduces a form of consistency regularization which, as we will show in section 5, is crucial to FixMatch’s success. We also note that it is typical in modern SSL algorithms to increase the weight of the unlabeled loss term (λ_u) during training [51, 24, 4, 3, 36]. We found that this was unnecessary for FixMatch, which may be due to the fact

²In practice, we include all labeled data as part of unlabeled data without their labels when constructing \mathcal{U} .

that $\max(q_b)$ is typically less than τ early in training. As training progresses, the model’s predictions become more confident and it is more frequently the case that $\max(q_b) > \tau$. This suggests that pseudo-labeling may produce a natural curriculum “for free”. Similar justifications have been used in the past for ignoring low-confidence predictions in visual domain adaptation [15].

2.3 Augmentation in FixMatch

FixMatch leverages two kinds of augmentations: “weak” and “strong”. In all of our experiments, weak augmentation is a standard flip-and-shift augmentation strategy. Specifically, we randomly flip images horizontally with a probability of 50% on all datasets except SVHN and we randomly translate images by up to 12.5% vertically and horizontally.

For “strong” augmentation, we experiment with two methods based on AutoAugment [10], which are then followed by the Cutout [14]. AutoAugment uses reinforcement learning to find an augmentation strategy comprising transformations from the Python Imaging Library.³ This requires labeled data to learn the augmentation strategy, making it problematic to use in SSL settings where limited labeled data is available. As a result, variants of AutoAugment which do not require the augmentation strategy to be learned ahead of time with labeled data, such as RandAugment [11] and CTAugment [3], have been proposed. Instead of using a learned strategy, both RandAugment and CTAugment randomly select transformations for each sample. For RandAugment, the magnitude that controls the severity of all distortions is randomly sampled from a pre-defined range (RandAugment with random magnitude was also used for UDA by [54]), whereas the magnitudes of individual transformations are learned on-the-fly for CTAugment. Refer to appendix E for more details.

2.4 Additional important factors

Semi-supervised performance can be substantially impacted by factors other than the SSL algorithm used because considerations like the amount of regularization can be particularly important in the low-label regime. This is compounded by the fact that the performance of deep networks trained for image classification can heavily depend on the architecture, optimizer, training schedule, etc. These factors are typically not emphasized when new SSL algorithms are introduced. Instead, we endeavor to quantify their importance and highlight which ones have a significant impact on performance. Most analysis is performed in section 5. In this section we identify a few key considerations.

First, as mentioned above, we find that regularization is particularly important. In all of our models and experiments, we use simple weight decay regularization. We also found that using the Adam optimizer [22] resulted in worse performance and instead use standard SGD with momentum [50, 40, 34]. We did not find a substantial difference between standard and Nesterov momentum. For a learning rate schedule, we use a cosine learning rate decay [28] which sets the learning rate to $\eta \cos\left(\frac{7\pi k}{16K}\right)$ where η is the initial learning rate, k is the current training step, and K is the total number of training steps. Finally, we report final performance using an exponential moving average of model parameters.

2.5 Extensions of FixMatch

Due to its simplicity, FixMatch can be readily extended with techniques in SSL literature. For example, both Augmentation Anchoring (where M strong augmentations are used for consistency regularization for each unlabeled example) and Distribution Alignment (which encourages the model predictions to have the same class distribution as the labeled set) from ReMixMatch [3] can be straightforwardly applied to FixMatch. Moreover, one may replace strong augmentations in FixMatch with modality-agnostic augmentation strategies, such as MixUp [59] or adversarial perturbations [33]. We present some exploration and experiments with these extensions in appendix D.

3 Related work

Semi-supervised learning is a mature field with a huge diversity of approaches. In this review, we focus on methods closely related to FixMatch. Broader introductions are provided in [60, 61, 6].

The idea behind self-training has been around for decades [47, 32]. The generality of self-training (i.e., using a model’s predictions to obtain artificial labels for unlabeled data) has led it to be applied in many domains including NLP [31], object detection [44], image classification [25, 55], domain

³<https://www.pythonware.com/products/pil/>

Algorithm	Artificial label augmentation	Prediction augmentation	Artificial label post-processing	Notes
TS / II-Model	Weak	Weak	None	
Temporal Ensembling	Weak	Weak	None	Uses model from earlier in training
Mean Teacher	Weak	Weak	None	Uses an EMA of parameters
Virtual Adversarial Training	None	Adversarial	None	
UDA	Weak	Strong	Sharpening	Ignores low-confidence artificial labels
MixMatch	Weak	Weak	Sharpening	Averages multiple artificial labels
ReMixMatch	Weak	Strong	Sharpening	Sums losses for multiple predictions
FixMatch	Weak	Strong	Pseudo-labeling	

Table 1: Comparison of SSL algorithms which include a form of consistency regularization and which (optionally) apply some form of post-processing to the artificial labels. We only mention those components of the SSL algorithm relevant to producing the artificial labels (for example, Virtual Adversarial Training additionally uses entropy minimization [17], MixMatch and ReMixMatch also use MixUp [59], UDA includes additional techniques like training signal annealing, etc.).

adaptation [62], to name a few. Pseudo-labeling refers to a specific variant where model predictions are converted to hard labels [25], which is often used along with a confidence-based thresholding that retains unlabeled examples only when the classifier is sufficiently confident (e.g., [44]). While some studies have suggested that pseudo-labeling is not competitive against other modern SSL algorithms on its own [36], recent SSL algorithms have used pseudo-labeling as a part of their pipeline to produce better results [1, 39]. As mentioned above, pseudo-labeling results in a form of entropy minimization [17] which has been used as a component for many SSL techniques [33].

Consistency regularization was first proposed by [2] and later referred to as “Transformation/Stability” (or TS for short) [46] or the “II-Model” [43]. Early extensions included using an exponential moving average of model parameters [51] or using previous model checkpoints [24] when producing artificial labels. Several methods have been used to produce random perturbations including data augmentation [15], stochastic regularization (e.g. Dropout [49]) [46, 24], and adversarial perturbations [33]. More recently, it has been shown that using strong data augmentation can produce better results [54, 3]. These heavily-augmented examples are almost certainly outside of the data distribution, which has in fact been shown to be beneficial for SSL [12]. Noisy Student [55] has integrated these techniques into a self-training framework and demonstrated impressive performance on ImageNet with additional massive amount of unlabeled data.

Of the aforementioned work, FixMatch bears the closest resemblance to two recent methods: Unsupervised Data Augmentation (UDA) [54] and ReMixMatch [3]. They both use a weakly-augmented example to generate an artificial label and enforce consistency against strongly-augmented examples. Neither of them uses pseudo-labeling, but both approaches “sharpen” the artificial label to encourage the model to produce high-confidence predictions. UDA in particular also only enforces consistency when the highest probability in the predicted class distribution for the artificial label is above a threshold. The thresholded pseudo-labeling of FixMatch has a similar effect to sharpening. In addition, ReMixMatch anneals the weight of the unlabeled data loss, which we omit from FixMatch because we posit that the thresholding used in pseudo-labeling has a similar effect (as mentioned in section 2.2). These similarities suggest that FixMatch can be viewed as a substantially simplified version of UDA and ReMixMatch, where we have combined two common techniques (pseudo-labeling and consistency regularization) while removing many components (sharpening, training signal annealing from UDA, distribution alignment and the rotation loss from ReMixMatch, etc.).

Since the core of FixMatch is a simple combination of two existing techniques, it also bears substantial similarities to many previously-proposed SSL algorithms. We provide a concise comparison of each of these techniques in table 1 where we list the augmentation used for the artificial label, the model’s prediction, and any post-processing applied to the artificial label. A more thorough comparison of these different algorithms and their constituent approaches is provided in the following section.

4 Experiments

We evaluate the efficacy of FixMatch on several SSL image classification benchmarks. Specifically, we perform experiments with varying amounts of labeled data and augmentation strategies on CIFAR-10/100 [23], SVHN [35], STL-10 [9], and ImageNet [13], following standard SSL evaluation protocols [36, 4, 3]. In many cases, we perform experiments with fewer labels than previously considered since FixMatch shows promise in extremely label-scarce settings. Note that we use an identical set of

Method	CIFAR-10			CIFAR-100			SVHN			STL-10
	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	40 labels	250 labels	1000 labels	1000 labels
II-Model	-	54.26 \pm 3.97	14.01 \pm 0.38	-	57.25 \pm 0.48	37.88 \pm 0.11	-	18.96 \pm 1.92	7.54 \pm 0.36	26.23 \pm 0.82
Pseudo-Labeling	-	49.78 \pm 0.43	16.09 \pm 0.28	-	57.38 \pm 0.46	36.21 \pm 0.19	-	20.21 \pm 1.09	9.94 \pm 0.61	27.99 \pm 0.83
Mean Teacher	-	32.32 \pm 2.30	9.19 \pm 0.19	-	53.91 \pm 0.57	35.83 \pm 0.24	-	3.57 \pm 0.11	3.42 \pm 0.07	21.43 \pm 2.39
MixMatch	47.54 \pm 11.50	11.05 \pm 0.86	6.42 \pm 0.10	67.61 \pm 1.32	39.94 \pm 0.37	28.31 \pm 0.33	42.55 \pm 14.53	3.98 \pm 0.23	3.50 \pm 0.28	10.41 \pm 0.61
UDA	29.05 \pm 5.93	8.82 \pm 1.08	4.88 \pm 0.18	59.28 \pm 0.88	33.13 \pm 0.22	24.50 \pm 0.25	52.63 \pm 20.51	5.69 \pm 2.76	2.46 \pm 0.24	7.66 \pm 0.56
ReMixMatch	19.10 \pm 9.64	5.44 \pm 0.05	4.72 \pm 0.13	44.28 \pm 2.06	27.43 \pm 0.31	23.03 \pm 0.56	3.34 \pm 0.20	2.92 \pm 0.48	2.65 \pm 0.08	5.23 \pm 0.45
FixMatch (RA)	13.81 \pm 3.37	5.07 \pm 0.65	4.26 \pm 0.05	48.85 \pm 1.75	28.29 \pm 0.11	22.60 \pm 0.12	3.96 \pm 2.17	2.48 \pm 0.38	2.28 \pm 0.11	7.98 \pm 1.50
FixMatch (CTA)	11.39 \pm 3.35	5.07 \pm 0.33	4.31 \pm 0.15	49.95 \pm 3.01	28.64 \pm 0.24	23.18 \pm 0.11	7.65 \pm 7.65	2.64 \pm 0.64	2.36 \pm 0.19	5.17 \pm 0.63

Table 2: Error rates for CIFAR-10, CIFAR-100, SVHN and STL-10 on 5 different folds. FixMatch (RA) uses RandAugment [11] and FixMatch (CTA) uses CTAugment [3] for strong-augmentation. All baseline models (II-Model [43], Pseudo-Labeling [25], Mean Teacher [51], MixMatch [4], UDA [54], and ReMixMatch [3]) are tested using the same codebase.

hyperparameters ($\lambda_u = 1$, $\eta = 0.03$, $\beta = 0.9$, $\tau = 0.95$, $\mu = 7$, $B = 64$, $K = 2^{20}$)⁴ across all amounts of labeled examples and datasets other than ImageNet. A complete list of hyperparameters is reported in appendix B.1. We include an extensive ablation study in section 5 to tease apart the importance of the different components and hyperparameters of FixMatch, including factors that are not explicitly part of the SSL algorithm such as the optimizer and learning rate.

4.1 CIFAR-10, CIFAR-100, and SVHN

We compare FixMatch to various existing methods on the standard CIFAR-10, CIFAR-100, and SVHN benchmarks. As suggested by [36], we reimplemented all existing baselines and performed all experiments using the same codebase. In particular, we use the same network architecture and training protocol, including the optimizer, learning rate schedule, data preprocessing, etc. across all SSL methods. Following [4], we used a Wide ResNet-28-2 [56] with 1.5M parameters for CIFAR-10 and SVHN, WRN-28-8 for CIFAR-100, and WRN-37-2 for STL-10. For baselines, we consider methods that are similar to FixMatch and/or are state-of-the-art: II-Model [43], Mean Teacher [51], Pseudo-Label [25], MixMatch [4], UDA [54], and ReMixMatch [3]. Besides [3], previous work has not considered fewer than 25 labels per class on these benchmarks. Performing better with less supervision is the central goal of SSL in practice since it alleviates the need for labeled data. We also consider the setting where only 4 labeled images are given for each class on each dataset. As far as we are aware, we are the first to run *any* experiments at 4 labels per class on CIFAR-100.

We report the performance of all baselines along with FixMatch in table 2. We compute the mean and variance of accuracy when training on 5 different “folds” of labeled data. We omit results with 4 labels per class for II-Model, Mean Teacher, and Pseudo-Labeling since the performance was poor at 250 labels. MixMatch, ReMixMatch, and UDA all perform reasonably well with 40 and 250 labels, but we find that FixMatch substantially outperforms each of these methods while nevertheless being simpler. For example, FixMatch achieves an average error rate of 11.39% on CIFAR-10 with 4 labels per class. As a point of reference, among the methods studied in [36] (where the same network architecture was used), the lowest error rate achieved on CIFAR-10 with 400 labels per class was 13.13%. Our results also compare favorably to recent state-of-the-art results achieved by ReMixMatch [3], despite the fact that we omit various components such as the self-supervised loss.

Our results are state-of-the-art on all datasets except for CIFAR-100 where ReMixMatch performs a bit better. To understand why ReMixMatch performs better than FixMatch, we experimented with a few variants of FixMatch which copy various components of ReMixMatch into FixMatch. We find that the most important term is Distribution Alignment (DA), which encourages the model predictions to have the same class distribution as the labeled set. Combining FixMatch with DA reaches a **40.14%** error rate with 400 labeled examples, which is substantially better than the 44.28% achieved by ReMixMatch.

We find that in most cases the performance of FixMatch using CTAugment and RandAugment is similar, except in the settings where we have 4 labels per class. This may be explained by the fact that these results are particularly high-variance. For example, the variance over 5 different folds for CIFAR-10 with 4 labels per class is 3.35%, which is significantly higher than that with 25 labels per class (0.33%). The error rates are also affected significantly by the random seeds when the number of labeled examples per class is extremely small, as shown in table 8 of supplementary material.

⁴ β refers to a momentum in SGD optimizer. The definition of other hyperparameters are found in section 2.



Figure 2: FixMatch reaches 78% CIFAR-10 accuracy using only above 10 labeled images.

4.2 STL-10

The STL-10 dataset contains 5,000 labeled images of size 96×96 from 10 classes and 100,000 unlabeled images. There exist out-of-distribution images in the unlabeled set, making it a more realistic and challenging test of SSL performance. We test SSL algorithms on five of the predefined folds of 1,000 labeled images each. Following [4], we use a WRN-37-2 network (comprising 5.9M parameters).⁵ As in table 2, FixMatch achieves the state-of-the-art performance of ReMixMatch [3] despite being significantly simpler.

4.3 ImageNet

We evaluate FixMatch on ImageNet to verify that it performs well on a larger and more complex dataset. Following [54], we use 10% of the training data as labeled and treat the rest as unlabeled examples. We use a ResNet-50 network architecture and RandAugment [11] as strong augmentation for this experiment. We include additional implementation details in appendix C. FixMatch achieves a top-1 error rate of $28.54 \pm 0.52\%$, which is 2.68% better than UDA [54]. Our top-5 error rate is $10.87 \pm 0.28\%$. While S⁴L [57] holds state-of-the-art on semi-supervised ImageNet with a 26.79% error rate, it leverages 2 additional training phases (pseudo-label re-training and supervised fine-tuning) to significantly lower the error rate from 30.27% after the first phase. FixMatch outperforms S⁴L after its first phase, and it is possible that a similar performance gain could be achieved by incorporating these techniques into FixMatch.

4.4 Barely Supervised Learning

To test the limits of our proposed approach, we applied FixMatch to CIFAR-10 with **only one example per class**.⁶ We conduct two sets of experiments.

First, we create four datasets by randomly selecting one example per class. We train on each dataset four times and reach between 48.58% and 85.32% test accuracy with a median of 64.28%. The inter-dataset variance is much lower, however; for example, the four models trained on the first dataset all reach between 61% and 67% accuracy, and the second dataset reaches between 68% and 75%.

We hypothesize that this variability is caused by the quality of the 10 labeled examples comprising each dataset and that sampling low-quality examples might make it more difficult for the model to learn some particular class effectively. To test this, we construct eight new training datasets with examples ranging in “prototypicality” (i.e., representative of the underlying class). Specifically, we take the ordering of the CIFAR-10 training set from [5] that sorts examples from those that are most representative to those that are least. This example ordering was determined after training many CIFAR-10 models with all labeled data. We thus do not envision this as a practical method for choosing examples for use in SSL, but rather to experimentally verify that examples that are more representative are better suited for low-label training. We divide this ordering evenly into eight buckets (so all of the most representative examples are in the first bucket, and all of the outliers in the last). We then create eight labeled training sets by randomly selecting one labeled example of each class from the same bucket.

Using the same hyperparameters, the model trained only on the most prototypical examples reaches a median of 78% accuracy (with a maximum of 84% accuracy); training on the middle of the distribution reaches 65% accuracy; and training on only the outliers fails to converge completely, with 10% accuracy. Figure 2 shows the full labeled training dataset for the split where FixMatch achieved a median accuracy of 78%. Further analysis is presented in Appendix B.7.

⁵We clarify that both FixMatch and ReMixMatch [3], which has reported an incorrect number of network parameters (23.8M), are tested with the same network architecture containing 5.9M parameters.

⁶The experimental protocol of barely supervised learning (BSL) shares similarities to those of few-shot learning (FSL) [37] as they both assume a limited availability (e.g., 1 or 5) of labeled examples from categories of interest. However, two protocols have a critical difference, where for FSL one is provided with extra labeled training examples from regular classes, whereas for BSL one is given additional unlabeled training examples.

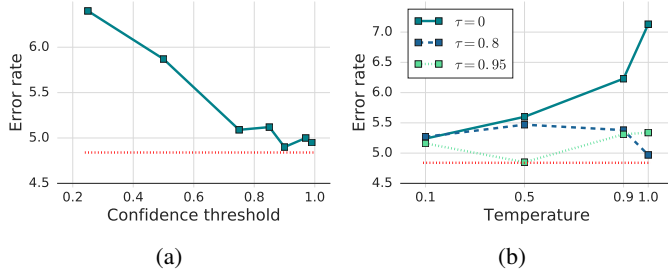


Figure 3: Plots of ablation studies on FixMatch. (a) Varying the confidence threshold for pseudo-labels. (b) Measuring the effect of “sharpening” the predicted label distribution while varying the confidence threshold (τ). Error rate of FixMatch with default hyperparameters is in red dotted line.

Ablation	Error
FixMatch	4.84
Only Cutout	6.15
No Cutout	6.15

Table 3: Ablation study with different strong data augmentation of FixMatch. Error rates are reported on a single 250-label split from CIFAR-10.

5 Ablation Study

Since FixMatch comprises a simple combination of two existing techniques, we perform an extensive ablation study to better understand why it is able to obtain state-of-the-art results. Due to the number of experiments in our ablation study, we focus on studying with a single 250 label split from CIFAR-10 and only report results using CTAugment. Note that FixMatch with default parameters achieves 4.84% error rate on this particular split. We present complete ablation results, including optimizer (appendix B.3), learning rate decay schedule (appendix B.4), weight decay (appendix B.6), labeled to unlabeled data ratio μ (appendix B.5), in the supplementary material.

5.1 Sharpening and Thresholding

A “soft” version of pseudo-labeling can be designed by sharpening the predicted distribution. This formulation appears in UDA and is of general interest since other methods such as MixMatch and ReMixMatch also make use of sharpening (albeit without thresholding). Using sharpening instead of an arg max introduces a hyper-parameter: the temperature T [4, 54, 3].

We study the interactions between the temperature T and the confidence threshold τ . Note that pseudo-labeling in FixMatch is recovered as $T \rightarrow 0$. The results are presented in fig. 3a and fig. 3b. The threshold value of 0.95 shows the lowest error rate, though increasing it to 0.97 or 0.99 did not hurt much. In contrast, accuracy drops by more than 1.5% when using a small threshold value. Note that the threshold value controls the trade-off between the quality and the quantity of pseudo-labels. As discussed in appendix B.2, the accuracy of pseudo-labels for unlabeled data increases with higher threshold values, while the amount of unlabeled data contributing to ℓ_u in eq. (4) decreases. This suggests that the quality of pseudo-labels is more important than the quantity for reaching a high accuracy. Sharpening, on the other hand, did not show a significant difference in performance when a confidence threshold is used. In summary, we observe that swapping pseudo-labeling for sharpening and thresholding would introduce a new hyperparameter while achieving no better performance.

5.2 Augmentation Strategy

We conduct an ablation study on different strong data augmentation policies as it plays a key role in FixMatch. Specifically, we chose RandAugment [11] and CTAugment [3], which have been used for state-of-the-art SSL algorithms such as UDA [54] and ReMixMatch [4] respectively. On CIFAR-10, CIFAR-100, and SVHN we observed highly comparable results between the two policies, whereas in STL-10 (table 2), we observe a significant gain by using CTAugment.

We measure the effect of Cutout in table 3, which is used by default after strong augmentation in both RandAugment and CTAugment. We find that both Cutout and CTAugment are required to obtain the best performance; removing either results in a significant increase in error rate.

We also study different combinations of weak and strong augmentations for pseudo-label generation and prediction (i.e., the upper and lower paths in fig. 1). When we replaced the weak augmentation for label guessing with strong augmentation, we found that the model diverged early in training. Conversely, when replacing weak augmentation with *no* augmentation, the model overfits the guessed unlabeled labels. Using weak augmentation in place of strong augmentation to generate the model’s prediction for training peaked at 45% accuracy but was not stable and progressively collapsed to 12%,

suggesting the importance of strong data augmentation. This observation is well-aligned with those from supervised learning [10].

6 Conclusion

There has been rapid recent progress in SSL. Unfortunately, much of this progress comes at the cost of increasingly complicated learning algorithms with sophisticated loss terms and numerous difficult-to-tune hyper-parameters. We introduce FixMatch, a simpler SSL algorithm that achieves state-of-the-art results across many datasets. We show how FixMatch can begin to bridge the gap between low-label semi-supervised learning and few-shot learning or clustering: we obtain surprisingly-high accuracy with just one label per class. Using only standard cross-entropy losses on both labeled and unlabeled data, FixMatch’s training objective can be written in just a few lines of code.

Because of this simplicity, we are able to thoroughly investigate how FixMatch works. We find that certain design choices are important (and often underemphasized) – most importantly, weight decay and the choice of optimizer. The importance of these factors means that even when controlling for model architecture as is recommended in [36], the same technique can not always be directly compared across different implementations.

On the whole, we believe that the existence of such simple but performant semi-supervised machine learning algorithms will help to allow machine learning to be deployed in increasingly many practical domains where labels are expensive or difficult to obtain.

Broader Impact

FixMatch helps democratize machine learning in two ways: first, its simplicity makes it available to a wider audience, and second, its accuracy with only a few labels means that it can be applied to domains where previously machine learning was not feasible. The flip side of democratization of machine learning research is that it becomes easy for both good and bad actors to apply. We hope that this ability will be used for good—for example, obtaining medical scans is often far cheaper than paying an expert doctor to label every image. However, it is possible that more advanced techniques for semi-supervised learning will allow for more advanced surveillance: for example, the efficacy of our one-shot classification might allow for more accurate person identification from a few images. Broadly speaking, any progress on semi-supervised learning will have these same consequences.

Funding Disclosure

Google is the sole source of funding for this work.

Acknowledgment

We thank Qizhe Xie, Avital Oliver, Quoc V. Le, and Sercan Arik for their feedback on this paper.

References

- [1] E. Arazo, D. Ortego, P. Albert, N. E. O’Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. *arXiv preprint arXiv:1908.02983*, 2019. 5, 15
- [2] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *Advances in neural information processing systems*, pages 3365–3373, 2014. 1, 3, 5
- [3] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. Remix-match: Semi-supervised learning with distribution matching and augmentation anchoring. In *Eighth International Conference on Learning Representations*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 18, 19, 20, 21
- [4] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2019. 1, 2, 3, 5, 6, 7, 8, 14, 19

- [5] N. Carlini, Ú. Erlingsson, and N. Papernot. Distribution density, tails, and outliers in machine learning: Metrics and applications. *arXiv preprint arXiv:1910.13427*, 2019. 7, 16
- [6] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006. 4
- [7] J. Chen and Q. Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018. 15
- [8] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019. 15
- [9] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. 5
- [10] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 4, 9, 20
- [11] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2019. 2, 4, 6, 7, 8, 18, 20
- [12] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. R. Salakhutdinov. Good semi-supervised learning that requires a bad GAN. In *Advances in Neural Information Processing Systems*, 2017. 5
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 5
- [14] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2, 4
- [15] G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. In *Sixth International Conference on Learning Representations*, 2018. 4, 5
- [16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 16
- [17] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, 2005. 3, 5
- [18] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 15
- [19] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 15
- [20] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017. 1
- [21] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 1
- [22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Third International Conference on Learning Representations*, 2015. 4
- [23] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

- [24] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. In *Fifth International Conference on Learning Representations*, 2017. 1, 3, 5
- [25] 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. 1, 2, 3, 4, 5, 6
- [26] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018. 15
- [27] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017. 15
- [28] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Fifth International Conference on Learning Representations*, 2017. 4, 15
- [29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Sixth International Conference on Learning Representations*, 2018. 15
- [30] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1
- [31] D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, 2006. 4
- [32] 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. 1, 3, 4
- [33] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 3, 4, 5, 19
- [34] Y. E. Nesterov. A method of solving a convex programming problem with convergence rate $O(k^{-2})$. *Doklady Akademii Nauk*, 269(3), 1983. 4
- [35] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 5
- [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*, pages 3235–3246, 2018. 2, 3, 5, 6, 9
- [37] B. Oreshkin, P. R. López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018. 7
- [38] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019. 19
- [39] H. Pham and Q. V. Le. Semi-supervised learning by coaching. *Submitted to the 8th International Conference on Learning Representations*, 2019. <https://openreview.net/forum?id=rJe04p4YDB>. 5
- [40] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5), 1964. 4
- [41] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019. 1

- [42] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019. 1
- [43] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015. 5, 6
- [44] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 2005. 1, 4, 5
- [45] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *IEEE International Conference on Image Processing*, 2016. 1, 3
- [46] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2016. 1, 3, 5
- [47] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3), 1965. 1, 3, 4
- [48] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015. 19
- [49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 2014. 5
- [50] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, 2013. 4
- [51] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, 2017. 1, 3, 5, 6
- [52] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019. 1
- [53] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017. 15
- [54] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 16, 20
- [55] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020. 1, 4, 5
- [56] S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 6
- [57] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. S4l: Self-supervised semi-supervised learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 7, 19
- [58] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018. 15
- [59] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. MixUp: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 4, 5, 19
- [60] X. Zhu. Semi-supervised learning literature survey. Technical Report TR 1530, Computer Sciences, University of Wisconsin – Madison, 2008. 4

- [61] X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1), 2009. [4](#)
- [62] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 289–305, 2018. [5](#)

A Algorithm

We present the complete algorithm for FixMatch in algorithm 1.

Algorithm 1 FixMatch algorithm.

- 1: **Input:** Labeled batch $\mathcal{X} = \{(x_b, p_b) : b \in (1, \dots, B)\}$, unlabeled batch $\mathcal{U} = \{u_b : b \in (1, \dots, \mu B)\}$, confidence threshold τ , unlabeled data ratio μ , unlabeled loss weight λ_u .
 - 2: $\ell_s = \frac{1}{B} \sum_{b=1}^B H(p_b, \alpha(x_b))$ {Cross-entropy loss for labeled data}
 - 3: **for** $b = 1$ **to** μB **do**
 - 4: $q_b = p_m(y \mid \alpha(u_b); \theta)$ {Compute prediction after applying weak data augmentation of u_b }
 - 5: **end for**
 - 6: $\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}\{\max(q_b) > \tau\} H(\arg \max(q_b), p_m(y \mid \mathcal{A}(u_b)))$ {Cross-entropy loss with pseudo-label and confidence for unlabeled data}
 - 7: **return** $\ell_s + \lambda_u \ell_u$
-

B Comprehensive Experimental Results

B.1 Hyperparameters

As mentioned in section 4, we used almost identical hyperparameters of FixMatch on CIFAR-10, CIFAR-100, SVHN and STL-10. Note that we used similar network architectures for these datasets, except that more convolution filters were used for CIFAR-100 (WRN-28-8) to handle larger label space and more convolutions were used for STL-10 (WRN-37-2) to deal with larger input image size. Following the suggestion in [4], we doubled the weight decay parameter for WRN-28-8 to avoid overfitting. Here, we provide a complete list of hyperparameters in table 4. Note that we did ablation study for most of these hyperparameters in section 5 (τ in section 5.1, μ in appendix B.5, lr and β (momentum) in appendix B.3, and weight decay in appendix B.6).

	CIFAR-10	CIFAR-100	SVHN	STL-10
τ		0.95		
λ_u		1		
μ		7		
B		64		
lr		0.03		
β		0.9		
Nesterov		True		
weight decay	0.0005	0.001	0.0005	0.0005

Table 4: Complete list of FixMatch hyperparameters for CIFAR-10, CIFAR-100, SVHN and STL-10.

B.2 Trade-off between the Quality and the Quantity of Pseudo-Labels with Confidence

To better understand the role of thresholding in FixMatch, we present in table 5 two additional measurements along with the test set accuracy: the impurity (the error rate of unlabeled data that falls above the threshold) and the mask rate (the number of examples which are masked out) which are computed as follows:

$$\text{impurity} = \frac{\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) \mathbb{1}(y_b \neq \hat{q}_b)}{\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau)} \quad (5)$$

$$\text{mask rate} = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) \quad (6)$$

As shown in table 5, when using small threshold values, most unlabeled examples' confidence is above the threshold. Consequently, they all contribute to the unlabeled loss in eq. (4). Unfortunately, pseudo-labels of these examples are not always correct and the learning process is significantly

τ	mask rate	impurity	error rate
0.25	100.00	6.39	6.40
0.5	100.00	5.40	5.87
0.75	99.82	5.35	5.09
0.85	99.31	4.32	5.12
0.9	99.21	3.85	4.90
0.95	98.13	3.47	4.84
0.97	96.35	2.30	5.00
0.99	92.14	2.06	5.05

Table 5: The mask rate and impurity at the end of the training along with the test set error rate of FixMatch using different threshold values on a single 250-label split from CIFAR-10.

Decay Schedule	Error
Cosine (FixMatch)	4.84
Linear Decay (end 0.01)	4.95
Linear Decay (end 0.02)	5.55
No Decay	5.70

Table 6: Ablation study on learning rate decay schedules. Error rates are reported on a single 250-label split from CIFAR-10.

impeded by noisy pseudo-labeled examples. This behavior is known as confirmation bias [1]. On the other hand, using high threshold values allows a smaller fraction of ostensibly higher-quality unlabeled examples to contribute to the unlabeled loss, effectively reducing the confirmation bias with strong data augmentation, resulting in lower error rates on the test set. Given our observation on the trade-off between the quality and the quantity of pseudo-labels, combining improved techniques for confidence calibration and uncertainty estimation [18, 27, 26, 19] into FixMatch would be a promising future direction.

B.3 Ablation Study on Optimizer

While the study of different optimizers and their hyperparameters is seldom done in previous SSL works, we found that they can have a strong effect on performance. We present ablation results on optimizers in table 7. First, we studied the effect of momentum (β) for the SGD optimizer. We found that the performance is somewhat sensitive to β and the model did not converge when β is set too large. On the other hand, small values of β still worked fine. When β is small, increasing the learning rate improved the performance, though they are not as good as the best performance obtained with $\beta = 0.9$. Nesterov momentum resulted in a slightly lower error rate than that of standard momentum SGD, but the difference was not significant.

As studied in [53, 29], we did not find Adam performing better than momentum SGD. While the best error rate of the model trained with Adam is only 0.53% larger than that of momentum SGD, we found that the performance was much more sensitive to the change of learning rate (e.g., increase in error rate by more than 8% when increasing the learning rate to 0.002) than momentum SGD. Additional exploration along this direction to make Adam more competitive includes the use of weight decay [29, 58] instead of L2 weight regularization and a better exploration of hyperparameters [7, 8].

B.4 Ablation Study on Learning Rate Schedule

It is a popular choice in recent works [28] to use a cosine learning rate decay. As shown in table 6, a linear learning rate decay performed nearly as well. Note that, as for the cosine learning rate decay, picking a proper decaying rate is important. Finally, using no decay results in worse accuracy (a 0.86% degradation).

Optimizer	Hyperparameters			Error
SGD	$\eta = 0.03$	$\beta = 0.90$	Nesterov	4.84
SGD	$\eta = 0.03$	$\beta = 0.999$	Nesterov	84.33
SGD	$\eta = 0.03$	$\beta = 0.99$	Nesterov	21.97
SGD	$\eta = 0.03$	$\beta = 0.50$	Nesterov	5.79
SGD	$\eta = 0.03$	$\beta = 0.25$	Nesterov	6.42
SGD	$\eta = 0.03$	$\beta = 0$	Nesterov	6.76
SGD	$\eta = 0.05$	$\beta = 0$	Nesterov	6.06
SGD	$\eta = 0.10$	$\beta = 0$	Nesterov	5.27
SGD	$\eta = 0.20$	$\beta = 0$	Nesterov	5.19
SGD	$\eta = 0.50$	$\beta = 0$	Nesterov	5.74
SGD	$\eta = 0.03$	$\beta = 0.90$		4.86
Adam	$\eta = 0.002$	$\beta_1 = 0.9$	$\beta_2 = 0.00$	29.42
Adam	$\eta = 0.002$	$\beta_1 = 0.9$	$\beta_2 = 0.90$	14.42
Adam	$\eta = 0.002$	$\beta_1 = 0.9$	$\beta_2 = 0.99$	15.44
Adam	$\eta = 0.002$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	13.93
Adam	$\eta = 0.0008$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	7.35
Adam	$\eta = 0.0006$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	6.12
Adam	$\eta = 0.0005$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	5.95
Adam	$\eta = 0.0004$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	5.44
Adam	$\eta = 0.0003$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	5.37
Adam	$\eta = 0.0002$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	5.57
Adam	$\eta = 0.0001$	$\beta_1 = 0.9$	$\beta_2 = 0.999$	7.90

Table 7: Ablation study on optimizers. Error rates are reported on a single 250-label split from CIFAR-10.

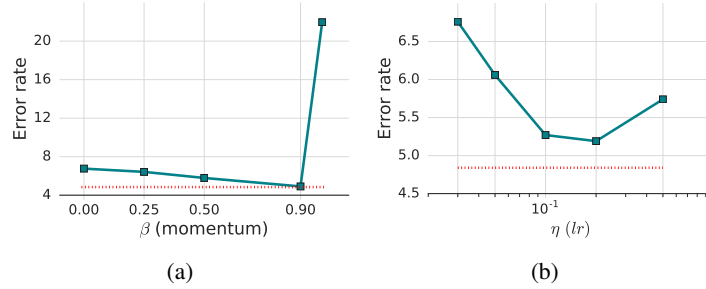


Figure 4: Plots of ablation studies on optimizers. (a) Varying β . (b) Varying η with $\beta = 0$.

B.5 Ratio of Labeled to Unlabeled Data in Minibatch

In fig. 5a we plot the error rates of FixMatch with different ratios of unlabeled data (μ) in each minibatch. We observe a significant decrease in error rates by using a large amount of unlabeled data, which is consistent with the finding in UDA [54]. In addition, scaling the learning rate η linearly with the batch size (a technique for large-batch supervised training [16]) was effective for FixMatch, especially when μ is small.

B.6 Weight Decay

While the value 0.0005 appeared as a good default choice for WRN-28-2 across datasets, we find that the weight decay could have a huge impact on performance when tuned incorrectly for low label regimes: choosing a value that is just one order of magnitude larger or smaller than optimal can cost ten percentage points or more, as shown in fig. 5b.

B.7 Labeled Data for Barely Supervised Learning

In addition to fig. 2, we visualize the full labeled training images obtained by ordering mechanism [5] used for barely supervised learning in fig. 6. Each row contains 10 images from 10 different classes

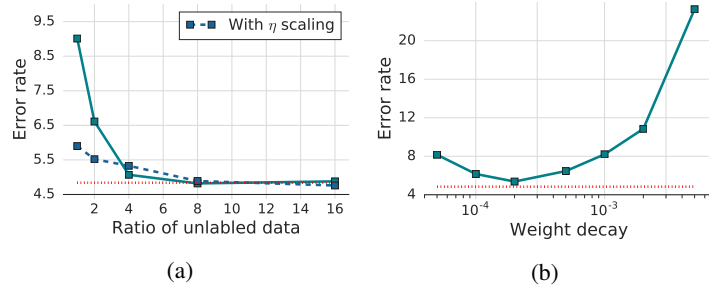


Figure 5: Plots of ablation studies on FixMatch. (a) Varying the ratio of unlabeled data (μ) with different learning rate (η) scaling strategies. (b) Varying the loss coefficient for weight decay. Error rate of FixMatch with default hyperparameters is in red dotted line.

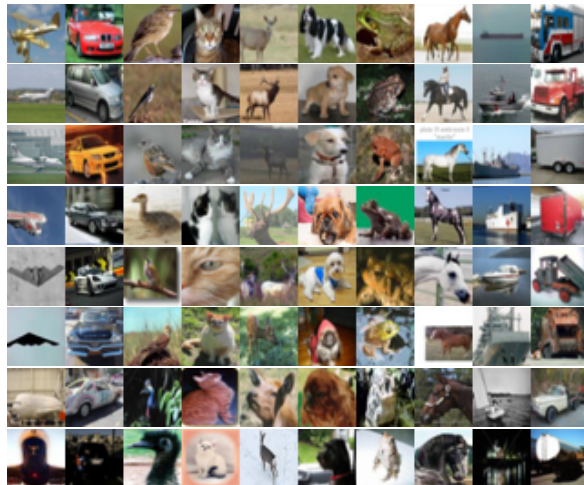


Figure 6: Labeled training data for the 1-label-per-class experiment. Each row corresponds to the complete labeled training set for one run of our algorithm, sorted from the most prototypical dataset (first row) to least prototypical dataset (last row).

of CIFAR-10 and is used as the complete labeled training dataset for one run of FixMatch. The first row contains the most prototypical images of each class, while the bottom row contains the least prototypical images. We train two models for each dataset and compute the mean accuracy between the two and plot this in fig. 7. Observe that we obtain over 80% accuracy when training on the best examples.

B.8 Comparison to Supervised Baselines

In table 9 and table 10, we present the performance of models trained only with the labeled data using strong data augmentations to highlight the effectiveness of using unlabeled data in FixMatch.

C Implementation Details for Section 4.3

For our ImageNet experiments we use standard ResNet50 pre-activation model trained in a distributed way on a TPU device with 32 cores.⁷ We report results over five random folds of labeled data. We use the following set of hyperparameters for our ImageNet model:

- **Batch size.** On each step our batch contains 1024 labeled examples and 5120 unlabeled examples.

⁷<https://github.com/tensorflow/tpu/tree/master/models/official/resnet>

Dataset	1	2	3	4	5
CIFAR-10	5.46	6.17	9.37	10.85	13.32
SVHN	2.40	2.47	6.24	6.32	6.38

Table 8: Error rates of FixMatch (CTA) on a single 40-label split of CIFAR-10 and SVHN with different random seeds. Runs are ordered by accuracy.

Method	CIFAR-10			CIFAR-100			SVHN		
	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	40 labels	250 labels	1000 labels
Supervised (RA)	64.01 \pm 0.76	39.12 \pm 0.77	12.74 \pm 0.29	79.47 \pm 0.18	52.88 \pm 0.51	32.55 \pm 0.21	52.68 \pm 2.29	22.48 \pm 0.55	10.89 \pm 0.12
Supervised (CTA)	64.53 \pm 0.83	41.92 \pm 1.17	13.64 \pm 0.12	79.79 \pm 0.59	54.23 \pm 0.48	35.30 \pm 0.19	43.05 \pm 2.34	15.06 \pm 1.02	7.69 \pm 0.27
FixMatch (RA)	13.81 \pm 3.37	5.07 \pm 0.65	4.26 \pm 0.05	48.85 \pm 1.75	28.29 \pm 0.11	22.60 \pm 0.12	3.96 \pm 2.17	2.48 \pm 0.38	2.28 \pm 0.11
FixMatch (CTA)	11.39 \pm 3.35	5.07 \pm 0.33	4.31 \pm 0.15	49.95 \pm 3.01	28.64 \pm 0.24	23.18 \pm 0.11	7.65 \pm 7.65	2.64 \pm 0.64	2.36 \pm 0.19

Table 9: Error rates for CIFAR-10, CIFAR-100 and SVHN on 5 different folds. Models with (RA) uses RandAugment [11] and the ones with (CTA) uses CTAugment [3] for strong-augmentation. All models are tested using the same codebase.

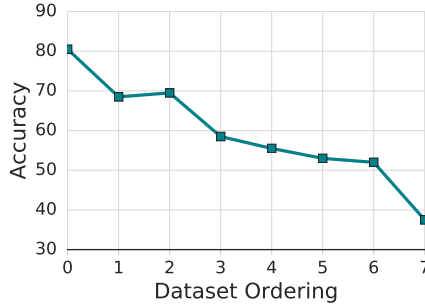


Figure 7: Accuracy of the model when trained on the 1-label-per-class datasets from Figure 6, ordered from most prototypical (top row) to least (bottom row).

Method	Error rate	Method	Error rate
Supervised (RA)	20.66 \pm 0.83	FixMatch (RA)	7.98 \pm 1.50
Supervised (CTA)	19.86 \pm 0.66	FixMatch (CTA)	5.17 \pm 0.63

Table 10: Error rates for STL-10 on 1000-label splits. All models are tested using the same codebase.

- **Training time.** We train our model for 300 epochs of unlabeled examples⁸.
- **Learning rate schedule.** We utilize linear learning rate warmup for the first 5 epochs until it reaches an initial value of 0.4. Then we decay learning rate at epochs 60, 120, 160, and 200 epoch by multiplying it by 0.1.
- **Optimizer.** We use Nesterov Momentum optimizer with momentum 0.9.
- **Exponential moving average (EMA).** We utilize EMA technique with decay 0.999.
- **FixMatch loss.** We use unlabeled loss weight $\lambda_u = 10$ and confidence threshold $\tau = 0.7$ in FixMatch loss.
- **Weight decay.** Our weight decay coefficient is 0.0003. Similarly to other datasets we perform weight decay by adding L2 penalty of all weights to model loss.
- **Augmentation of unlabeled images.** For strong augmentation we use RandAugment with random magnitude [11]. For weak augmentation we use a random horizontal flip.
- **ImageNet preprocessing.** We randomly crop and rescale to 224×224 size all labeled and unlabeled training images prior to performing augmentation. This is considered a standard ImageNet preprocessing technique.

⁸Note that one epoch of unlabelled examples contains all 1.2 million examples from Imagenet training set and it correspond to 10 passes through labelled set for 10% Imagenet task.

D Extensions of FixMatch

D.1 Augmentation Anchoring and Distribution Alignment

Augmentation Anchoring, where M strong augmentations are applied to each unlabeled example for consistency regularization, and Distribution Alignment, which encourages the model predictions to have same class distribution as the labeled set, are two important techniques to the success of ReMixMatch [3]. Thanks to its simplicity and clean formulation, incorporating these techniques into FixMatch becomes straightforward. Firstly, we incorporate Augmentation Anchoring into FixMatch as follows:

$$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau) \times \frac{1}{M} \sum_{i=1}^M H(\hat{q}_b, p_m(y | \mathcal{A}(u_b))) \quad (7)$$

Note that the strong augmentation $\mathcal{A}(u_b)$ is a stochastic process and it produces M different strongly-augmented examples of an unlabeled example u_b . Using $M = 4$ and $\mu = 4$, FixMatch (CTA) with Augmentation Anchoring reduces the error rate averaged over 5 different folds on CIFAR-10 with 250 labeled examples from 5.07% to 4.81%.

As already reported in section 4.1, combining Distribution Alignment into FixMatch improves the SSL performance significantly, especially when the number of labeled training data is very limited. Specifically, we align the predictive distribution of a weakly-augmented example $q_b = p_m(y | \alpha(u_b))$ using the dataset’s marginal class distribution estimated using the labeled data and the running average of the model’s prediction estimated by the unlabeled data as follows:

$$\tilde{q}_b = \text{Normalize} \left(q_b \times \frac{p(y | \mathcal{X})}{p_m(y | \mathcal{U})} \right) \quad (8)$$

where $\text{Normalize}(x)_i = x_i / \sum_j x_j$. Now, eq. (4) can be modified as follows accordingly:

$$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}(\max(\tilde{q}_b) \geq \tau) H(\hat{\tilde{q}}_b, p_m(y | \mathcal{A}(u_b))) \quad (9)$$

FixMatch (CTA) with distribution alignment reduces the error rate averaged over 5 different folds on CIFAR-10 with 40 labeled examples from 11.38% to 9.47%. On CIFAR-100 with 400 labeled examples, it reduces the error rate from 49.95% to 40.14%, which is also lower than 44.28% of ReMixMatch. In addition to Section 4.3, we conduct SSL experiments on ImageNet using 1% of its data as labeled examples. We find that in this regime the role of distribution alignment becomes more critical – FixMatch model does not train well without distribution alignment. On the other hand, after a proper tuning of hyperparameters (weight of unlabeled loss $\lambda_u = 3$ and confidence threshold $\tau = 0.9$), FixMatch (RA) model with distribution alignment achieves 67.1% top-1 and 47.7% top-5 error rate (this correspond to 32.9% top-1 and 52.3% top-5 accuracy and similar to [57] results).

D.2 Datatype-Agnostic Data Augmentation

Strong augmentation plays a key role in FixMatch. Applying FixMatch to different problem domains than vision thus requires one to come up with a novel augmentation strategy. While there are domain-specific data augmentation strategies for different application domains, such as back-translation [48] for text classification or SpecAugment [38] for speech recognition, it is desirable if FixMatch can also be combined with datatype-agnostic data augmentation methods.

In this section, we consider two such augmentation schemes, namely MixUp [59] and Virtual Adversarial Training (VAT) [33], as a replacement of RandAugment or CTAugment in FixMatch for image classification. For MixUp, instead of mixing both input and label, we mixed random pairs of inputs only using $\alpha = 9$ to be more consistent with the data augmentation in FixMatch. For VAT, we used $\tau = 0.5$. We evaluated on CIFAR-10 with 250 labeled data protocol and report the mean and standard deviation over 5 different folds in table 11. We make comparisons of our FixMatch variants to MixMatch [4] and VAT [33]. The FixMatch variant with (input) MixUp obtained comparable error rates to MixMatch, while the variant with VAT achieved significantly lower error rates than VAT. This suggests the generality of the FixMatch’s framework against different data augmentation strategies.

FixMatch + Input MixUp	MixMatch	FixMatch + VAT	VAT
10.99±0.50	11.05±0.86	32.26±2.24	36.03±2.82

Table 11: Error rates for CIFAR-10 with 250 labeled examples on 5 different folds. All models are tested using the same codebase.

E List of Data Transformations

Given a collection of transformations (e.g., color inversion, translation, contrast adjustment, etc.), RandAugment randomly selects transformations for each sample in a mini-batch. As originally proposed, RandAugment uses a single fixed global magnitude that controls the severity of all distortions [11]. The magnitude is a hyperparameter that must be optimized on a validation set e.g., using grid search. We found that sampling a random magnitude from a pre-defined range at each training step (instead of using a fixed global value) works better for semi-supervised training, similar to what is used in UDA [54].

Instead of setting the transformation magnitudes randomly, CTAugment [3] learns them online over the course of training. To do so, a wide range of transformation magnitude values is divided into bins (as in AutoAugment [10]) and a weight (initially set to 1) is assigned to each bin. All examples are augmented with a pipeline consisting of two transformations which are sampled uniformly at random. For a given transformation, a magnitude bin is sampled randomly with a probability according to the (normalized) bin weights. To update the weights of the magnitude bins, a labeled example is augmented with two transformations whose magnitude bins are sampled uniformly at random. The magnitude bin weights are then updated according to how close the model’s prediction is to the true label. Further details on CTAugment can be found in [3].

We used the same sets of image transformations used in RandAugment [11] and CTAugment [3]. For completeness, we listed all transformation operations for these augmentation strategies in table 12 and table 13.

Transformation	Description	Parameter	Range
Autocontrast	Maximizes the image contrast by setting the darkest (lightest) pixel to black (white).		
Brightness	Adjusts the brightness of the image. $B = 0$ returns a black image, $B = 1$ returns the original image.	B	[0.05, 0.95]
Color	Adjusts the color balance of the image like in a TV. $C = 0$ returns a black & white image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Contrast	Controls the contrast of the image. A $C = 0$ returns a gray image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Equalize	Equalizes the image histogram.		
Identity	Returns the original image.		
Posterize	Reduces each pixel to B bits.	B	[4, 8]
Rotate	Rotates the image by θ degrees.	θ	[-30, 30]
Sharpness	Adjusts the sharpness of the image, where $S = 0$ returns a blurred image, and $S = 1$ returns the original image.	S	[0.05, 0.95]
Shear_x	Shears the image along the horizontal axis with rate R .	R	[-0.3, 0.3]
Shear_y	Shears the image along the vertical axis with rate R .	R	[-0.3, 0.3]
Solarize	Inverts all pixels above a threshold value of T .	T	[0, 1]
Translate_x	Translates the image horizontally by $(\lambda \times \text{image width})$ pixels.	λ	[-0.3, 0.3]
Translate_y	Translates the image vertically by $(\lambda \times \text{image height})$ pixels.	λ	[-0.3, 0.3]

Table 12: List of transformations used in RandAugment [11].

Transformation	Description	Parameter	Range
Autocontrast	Maximizes the image contrast by setting the darkest (lightest) pixel to black (white), and then blends with the original image with blending ratio λ .	λ	[0, 1]
Brightness	Adjusts the brightness of the image. $B = 0$ returns a black image, $B = 1$ returns the original image.	B	[0, 1]
Color	Adjusts the color balance of the image like in a TV. $C = 0$ returns a black & white image, $C = 1$ returns the original image.	C	[0, 1]
Contrast	Controls the contrast of the image. A $C = 0$ returns a gray image, $C = 1$ returns the original image.	C	[0, 1]
Cutout	Sets a random square patch of side-length ($L \times$ image width) pixels to gray.	L	[0, 0.5]
Equalize	Equalizes the image histogram, and then blends with the original image with blending ratio λ .	λ	[0, 1]
Invert	Inverts the pixels of the image, and then blends with the original image with blending ratio λ .	λ	[0, 1]
Identity	Returns the original image.		
Posterize	Reduces each pixel to B bits.	B	[1, 8]
Rescale	Takes a center crop that is of side-length ($L \times$ image width), and rescales to the original image size using method M .	L	[0.5, 1.0]
		M	see caption
Rotate	Rotates the image by θ degrees.	θ	[-45, 45]
Sharpness	Adjusts the sharpness of the image, where $S = 0$ returns a blurred image, and $S = 1$ returns the original image.	S	[0, 1]
Shear_x	Shears the image along the horizontal axis with rate R .	R	[-0.3, 0.3]
Shear_y	Shears the image along the vertical axis with rate R .	R	[-0.3, 0.3]
Smooth	Adjusts the smoothness of the image, where $S = 0$ returns a maximally smooth image, and $S = 1$ returns the original image.	S	[0, 1]
Solarize	Inverts all pixels above a threshold value of T .	T	[0, 1]
Translate_x	Translates the image horizontally by ($\lambda \times$ image width) pixels.	λ	[-0.3, 0.3]
Translate_y	Translates the image vertically by ($\lambda \times$ image height) pixels.	λ	[-0.3, 0.3]

Table 13: List of transformations used in CTAugment [3]. The ranges for the listed parameters are discretized into 17 equal bins, except for the M parameter of the Rescale transformation, which takes one of the following six options: anti-alias, bicubic, bilinear, box, hamming, and nearest.