MixMatch: A Holistic Approach to Semi-Supervised Learning

David Berthelot  
Google Research  
dberth@google.com

Nicholas Carlini  
Google Research  
ncarlini@google.com

Ian Goodfellow  
Work done at Google  
ian-academic@mailfence.com

Avital Oliver  
Google Research  
avitalo@google.com

Nicolas Papernot  
Google Research  
papernot@google.com

Colin Raffel  
Google Research  
craffel@google.com

Abstract

Semi-supervised learning has proven to be a powerful paradigm for leveraging unlabeled data to mitigate the reliance on large labeled datasets. In this work, we unify the current dominant approaches for semi-supervised learning to produce a new algorithm, MixMatch, that guesses low-entropy labels for data-augmented unlabeled examples and mixes labeled and unlabeled data using MixUp. MixMatch obtains state-of-the-art results by a large margin across many datasets and labeled data amounts. For example, on CIFAR-10 with 250 labels, we reduce error rate by a factor of 4 (from 38% to 11%) and by a factor of 2 on STL-10. We also demonstrate how MixMatch can help achieve a dramatically better accuracy-privacy trade-off for differential privacy. Finally, we perform an ablation study to tease apart which components of MixMatch are most important for its success. We release all code used in our experiments.

1 Introduction

Much of the recent success in training large, deep neural networks is thanks in part to the existence of large labeled datasets. Yet, collecting labeled data is expensive for many learning tasks because it necessarily involves expert knowledge. This is perhaps best illustrated by medical tasks where measurements call for expensive machinery and labels are the fruit of a time-consuming analysis that draws from multiple human experts. Furthermore, data labels may contain private information. In comparison, in many tasks it is much easier or cheaper to obtain unlabeled data.

Semi-supervised learning [6] (SSL) seeks to largely alleviate the need for labeled data by allowing a model to leverage unlabeled data. Many recent approaches for semi-supervised learning add a loss term which is computed on unlabeled data and encourages the model to generalize better to unseen data. In much recent work, this loss term falls into one of three classes (discussed further in Section 2): entropy minimization [18, 28]—which encourages the model to output confident predictions on unlabeled data; consistency regularization—which encourages the model to produce the same output distribution when its inputs are perturbed; and generic regularization—which encourages the model to generalize well and avoid overfitting the training data.

In this paper, we introduce MixMatch, an SSL algorithm which introduces a single loss that gracefully unifies these dominant approaches to semi-supervised learning. Unlike previous methods, MixMatch targets all the properties at once which we find leads to the following benefits:

---

https://github.com/google-research/mixmatch

Figure 1: Diagram of the label guessing process used in MixMatch. Stochastic data augmentation is applied to an unlabeled image $K$ times, and each augmented image is fed through the classifier. Then, the average of these $K$ predictions is “sharpened” by adjusting the distribution’s temperature. See algorithm 1 for a full description.

- Experimentally, we show that MixMatch obtains state-of-the-art results on all standard image benchmarks (section 4.2), and reducing the error rate on CIFAR-10 by a factor of 4;
- We further show in an ablation study that MixMatch is greater than the sum of its parts;
- We demonstrate in section 4.3 that MixMatch is useful for differentially private learning, enabling students in the PATE framework [36] to obtain new state-of-the-art results that simultaneously strengthen both privacy guarantees and accuracy.

In short, MixMatch introduces a unified loss term for unlabeled data that seamlessly reduces entropy while maintaining consistency and remaining compatible with traditional regularization techniques.

2 Related Work

To set the stage for MixMatch, we first introduce existing methods for SSL. We focus mainly on those which are currently state-of-the-art and that MixMatch builds on; there is a wide literature on SSL techniques that we do not discuss here (e.g., “transductive” models [14, 22, 21], graph-based methods [49, 4, 29], generative modeling [3, 27, 41, 9, 17, 23, 38, 34, 42], etc.). More comprehensive overviews are provided in [49, 6]. In the following, we will refer to a generic model $p_{\text{model}}(y | x; \theta)$ which produces a distribution over class labels $y$ for an input $x$ with parameters $\theta$.

2.1 Consistency Regularization

A common regularization technique in supervised learning is data augmentation, which applies input transformations assumed to leave class semantics unaffected. For example, in image classification, it is common to elastically deform or add noise to an input image, which can dramatically change the pixel content of an image without altering its label [7, 43, 10]. Roughly speaking, this can artificially expand the size of a training set by generating a near-infinite stream of new, modified data. Consistency regularization applies data augmentation to semi-supervised learning by leveraging the idea that a classifier should output the same class distribution for an unlabeled example even after it has been augmented. More formally, consistency regularization enforces that an unlabeled example $x$ should be classified the same as $\text{Augment}(x)$, an augmentation of itself.

In the simplest case, for unlabeled points $x$, prior work [25, 40] adds the loss term
\[
\| p_{\text{model}}(y | \text{Augment}(x); \theta) - p_{\text{model}}(y | \text{Augment}(x); \theta) \|^2.
\]

(1)

Note that $\text{Augment}(x)$ is a stochastic transformation, so the two terms in eq. (1) are not identical. “Mean Teacher” [44] replaces one of the terms in eq. (1) with the output of the model using an exponential moving average of model parameter values. This provides a more stable target and was found empirically to significantly improve results. A drawback to these approaches is that they use domain-specific data augmentation strategies. “Virtual Adversarial Training” [31] (VAT) addresses this by instead computing an additive perturbation to apply to the input which maximally changes the output class distribution. 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

A common underlying assumption in many semi-supervised learning methods is that the classifier’s decision boundary should not pass through high-density regions of the marginal data distribution.
One way to enforce this is to require that the classifier output low-entropy predictions on unlabeled data. This is done explicitly in [18] with a loss term which minimizes the entropy of \( p_{\text{model}}(y \mid x; \theta) \) for unlabeled data \( x \). This form of entropy minimization was combined with VAT in [31] to obtain stronger results. “Pseudo-Label” [28] does entropy minimization implicitly by constructing hard (1-hot) labels from high-confidence predictions on unlabeled data and using these as training targets in a standard cross-entropy loss. MixMatch also implicitly achieves entropy minimization through the use of a “sharpening” function on the target distribution for unlabeled data, described in section 3.2.

2.3 Traditional Regularization

Regularization refers to the general approach of imposing a constraint on a model to make it harder to memorize the training data and therefore hopefully make it generalize better to unseen data [19]. We use weight decay which penalizes the \( L_2 \) norm of the model parameters [30, 46]. We also use MixUp [47] in MixMatch to encourage convex behavior “between” examples. We utilize MixUp as both a regularizer (applied to labeled datapoints) and a semi-supervised learning method (applied to unlabeled datapoints). MixUp has been previously applied to semi-supervised learning; in particular, the concurrent work of [45] uses a subset of the methodology used in MixMatch. We clarify the differences in our ablation study (section 4.2.3).

3 MixMatch

In this section, we introduce MixMatch, our proposed semi-supervised learning method. MixMatch is a “holistic” approach which incorporates ideas and components from the dominant paradigms for SSL discussed in section 2. Given a batch \( X \) of labeled examples with one-hot targets (representing one of \( L \) possible labels) and an equally-sized batch \( U \) of unlabeled examples, MixMatch produces a processed batch of augmented labeled examples \( X' \) and a batch of augmented unlabeled examples with “guessed” labels \( U' \). \( U' \) and \( X' \) are then used in computing separate labeled and unlabeled loss terms. More formally, the combined loss \( \mathcal{L} \) for semi-supervised learning is defined as

\[
\mathcal{L} = \mathcal{L}_X + \lambda_{U} \mathcal{L}_U
\]

where \( \mathcal{L}_X \) is the cross-entropy between distributions \( p \) and \( q \), and \( T, K, \alpha, \) and \( \lambda_{U} \) are hyperparameters described below. The full MixMatch algorithm is provided in algorithm 1 and a diagram of the label guessing process is shown in fig. 1. Next, we describe each part of MixMatch.

3.1 Data Augmentation

As is typical in many SSL methods, we use data augmentation both on labeled and unlabeled data. For each \( x_b \) in the batch of labeled data \( X \), we generate a transformed version \( \hat{x}_b = \text{Augment}(x_b) \) (algorithm 1 line 3). For each \( u_b \) in the batch of unlabeled data \( U \), we generate \( K \) augmentations \( \hat{u}_{b,k} = \text{Augment}(u_b), k \in (1, \ldots, K) \) (algorithm 1 line 5). We use these individual augmentations to generate a “guessed label" \( q_b \) for each \( u_b \), through a process we describe in the following subsection.

3.2 Label Guessing

For each unlabeled example in \( U \), MixMatch produces a “guess” for the example’s label using the model’s predictions. This guess is later used in the unsupervised loss term. To do so, we compute the average of the model’s predicted class distributions across all the \( K \) augmentations of \( u_b \) by

\[
\bar{q}_b = \frac{1}{K} \sum_{k=1}^{K} p_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)
\]

in algorithm 1 line 7. Using data augmentation to obtain an artificial target for an unlabeled example is common in consistency regularization methods [25, 30, 44].
Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

1: **Input:** Batch of labeled examples and their one-hot labels $\mathcal{X} = ((x_b, p_b); b \in (1, \ldots, B))$, batch of unlabeled examples $\mathcal{U} = ((u_b; b \in (1, \ldots, B))$, sharpening temperature $T$, number of augmentations $K$, Beta distribution parameter $\alpha$ for MixUp.
2: for $b = 1$ to $B$ do
3: \hspace{1em} $\hat{x}_b =$ Augment($x_b$) // Apply data augmentation to $x_b$
4: for $k = 1$ to $K$ do
5: \hspace{2em} $\hat{u}_{b,k} =$ Augment($u_b$) // Apply $k^{th}$ round of data augmentation to $u_b$
6: end for
7: $\bar{q}_b = \frac{1}{K} \sum_k \Pr_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ // Compute average predictions across all augmentations of $u_b$
8: $q_b =$ Sharpen($\bar{q}_b, T$) // Apply temperature sharpening to the average prediction (see eq. (7))
9: end for
10: $\mathcal{X}' =$ MixUp($\mathcal{X}$, $\mathcal{W}$); $i \in (1, \ldots, [\mathcal{X}])$ // Apply MixUp to labeled data and entries from $\mathcal{W}$
11: $\mathcal{U}' =$ MixUp($\mathcal{U}$, $\mathcal{W} + [\mathcal{X}]$); $i \in (1, \ldots, [\mathcal{U}])$ // Apply MixUp to unlabeled data and the rest of $\mathcal{W}$
12: return $\mathcal{X}', \mathcal{U}'$

**Sharpening.** In generating a label guess, we perform one additional step inspired by the success of entropy minimization in semi-supervised learning (discussed in section 2.2). Given the average prediction over augmentations $\bar{q}_b$, we apply a sharpening function to reduce the entropy of the label distribution. In practice, for the sharpening function, we use the common approach of adjusting the “temperature” of this categorical distribution $\Pr_i$, which is defined as the operation

$$\text{Sharpen}(p, T)_i := p_i^p \left/ \sum_{j=1}^{B} p_j^p \right.$$ (7)

where $p$ is some input categorical distribution (specifically in MixMatch, $p$ is the average class prediction over augmentations $\bar{q}_b$, as shown in algorithm 1 line 8) and $T$ is a hyperparameter. As $T \to 0$, the output of $\text{Sharpen}(p, T)$ will approach a Dirac (“one-hot”) distribution. Since we will later use $q_b = \text{Sharpen}(\bar{q}_b, T)$ as a target for the model’s prediction for an augmentation of $u_b$, lowering the temperature encourages the model to produce lower-entropy predictions.

### 3.3 MixUp

We use MixUp for semi-supervised learning, and unlike past work for SSL we mix both labeled examples and unlabeled examples with label guesses (generated as described in section 3.2). To be compatible with our separate loss terms, we define a slightly modified version of MixUp. For a pair of two examples with their corresponding labels probabilities $(x_1, p_1), (x_2, p_2)$ we compute $(x', p')$ by

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$ (8)

$$\lambda' = \max(\lambda, 1 - \lambda)$$ (9)

$$x' = \lambda x_1 + (1 - \lambda') x_2$$ (10)

$$p' = \lambda p_1 + (1 - \lambda') p_2$$ (11)

where $\alpha$ is a hyperparameter. Vanilla MixUp omits eq. (9) (i.e. it sets $\lambda' = \lambda$). Given that labeled and unlabeled examples are concatenated in the same batch, we need to preserve the order of the batch to compute individual loss components appropriately. This is achieved by eq. (9), which ensures that $x'$ is closer to $x_1$ than to $x_2$. To apply MixUp, we first collect all augmented labeled examples with their labels and all unlabeled examples with their guessed labels into

$$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, \ldots, B))$$ (12)

$$\hat{\mathcal{U}} = ((\hat{u}_{b,k}, q_b); b \in (1, \ldots, B), k \in (1, \ldots, K))$$ (13)
(algorithm 1, lines 10-11). Then, we combine these collections and shuffle the result to form \( W \) which will serve as a data source for MixUp (algorithm 1, line 12). For each the \( i \)th example-label pair in \( \hat{X} \), we compute \( \text{MixUp}(\hat{X}_i, W_i) \) and add the result to the collection \( X' \) (algorithm 1, line 13). We compute \( U'_i = \text{MixUp}(\hat{U}_i, W_i + |\hat{X}|) \) for \( i \in (1, \ldots, |U|) \), intentionally using the remainder of \( W \) that was not used in the construction of \( X' \) (algorithm 1, line 14). To summarize, MixMatch transforms \( X \) into \( X' \), a collection of labeled examples which have had data augmentation and MixUp (potentially mixed with an unlabeled example) applied. Similarly, \( U \) is transformed into \( U' \), a collection of multiple augmentations of each unlabeled example with corresponding label guesses.

3.4 Loss Function

Given our processed batches \( X' \) and \( U' \), we use the standard semi-supervised loss shown in eqs. (3) to (5). Equation (5) combines the typical cross-entropy loss between labels and model predictions from \( X' \) with the squared \( L_2 \) loss on predictions and guessed labels from \( U' \). We use this \( L_2 \) loss in eq. (4) (the multiclass Brier score [5]) because, unlike the cross-entropy, it is bounded and less sensitive to incorrect predictions. For this reason, it is often used as the unlabeled data loss in SSL [25, 44] as well as a measure of predictive uncertainty [26]. We do not propagate gradients through computing the guessed labels, as is standard [25, 44, 31, 35].

3.5 Hyperparameters

Since MixMatch combines multiple mechanisms for leveraging unlabeled data, it introduces various hyperparameters – specifically, the sharpening temperature \( T \), number of unlabeled augmentations \( K \), \( \alpha \) parameter for Beta in MixUp, and the unsupervised loss weight \( \lambda_U \). In practice, semi-supervised learning methods with many hyperparameters can be problematic because cross-validation is difficult with small validation sets [35, 39, 35]. However, we find in practice that most of MixMatch's hyperparameters can be fixed and do not need to be tuned on a per-experiment or per-dataset basis. Specifically, for all experiments we set \( T = 0.5 \) and \( K = 2 \). Further, we only change \( \alpha \) and \( \lambda_U \) on a per-dataset basis; we found that \( \alpha = 0.75 \) and \( \lambda_U = 100 \) are good starting points for tuning. In all experiments, we linearly ramp up \( \lambda_U \) to its maximum value over the first 16,000 steps of training as is common practice [44].

4 Experiments

We test the effectiveness of MixMatch on standard SSL benchmarks (section 4.2). Our ablation study teases apart the contribution of each of MixMatch's components (section 4.2.3). As an additional application, we consider privacy-preserving learning in section 4.3.

4.1 Implementation details

Unless otherwise noted, in all experiments we use the “Wide ResNet-28” model from [35]. Our implementation of the model and training procedure closely matches that of [35] (including using 5000 examples to select the hyperparameters), except for the following differences: First, instead of decaying the learning rate, we evaluate models using an exponential moving average of their parameters with a decay rate of 0.999. Second, we apply a weight decay of 0.0004 at each update for the Wide ResNet-28 model. Finally, we checkpoint every \( 2^{16} \) training samples and report the median error rate of the last 20 checkpoints. This simplifies the analysis at a potential cost to accuracy by, for example, averaging checkpoints [2] or choosing the checkpoint with the lowest validation error.

4.2 Semi-Supervised Learning

First, we evaluate the effectiveness of MixMatch on four standard benchmark datasets: CIFAR-10 and CIFAR-100 [24], SVHN [32], and STL-10 [8]. Standard practice for evaluating semi-supervised learning on the first three datasets is to treat most of the dataset as unlabeled and use a small portion as labeled data. STL-10 is a dataset specifically designed for SSL, with 5,000 labeled images and 100,000 unlabeled images which are drawn from a slightly different distribution than the labeled data.
250 500 1000 2000 4000
Number of Labeled Datapoints
0%
10%
20%
30%
40%
Test Error
-Model
Mean Teacher
VAT
Pseudo-Label
MixUp
MixMatch
Supervised

Figure 2: Error rate comparison of MixMatch to baseline methods on CIFAR-10 for a varying number of labels. Exact numbers are provided in table 5 (appendix). “Supervised” refers to training with all 50000 training examples and no unlabeled data. With 250 labels MixMatch reaches an error rate comparable to next-best method’s performance with 4000 labels.

250 500 1000 2000 4000
Number of Labeled Datapoints
0%
10%
20%
30%
40%
Test Error
-Model
Mean Teacher
VAT
Pseudo-Label
MixUp
MixMatch
Supervised

Figure 3: Error rate comparison of MixMatch to baseline methods on SVHN for a varying number of labels. Exact numbers are provided in table 6 (appendix). “Supervised” refers to training with all 73257 training examples and no unlabeled data. With 250 examples MixMatch nearly reaches the accuracy of supervised training for this model.

4.2.1 Baseline Methods

As baselines, we consider the four methods considered in [35] (II-Model [25 40], Mean Teacher [44], Virtual Adversarial Training [31], and Pseudo-Label [28]) which are described in section 2. We also use MixUp [47] on its own as a baseline. MixUp is designed as a regularizer for supervised learning, so we modify it for SSL by applying it both to augmented labeled examples and augmented unlabeled examples with their corresponding predictions. In accordance with standard usage of MixUp, we use a cross-entropy loss between the MixUp-generated guess label and the model’s prediction. As advocated by [35], we reimplemented each of these methods in the same codebase and applied them to the same model (described in section 4.1) to ensure a fair comparison. We re-tuned the hyperparameters for each baseline method, which generally resulted in a marginal accuracy improvement compared to those in [35], thereby providing a more competitive experimental setting for testing out MixMatch.

4.2.2 Results

CIFAR-10 For CIFAR-10, we evaluate the accuracy of each method with a varying number of labeled examples from 250 to 4000 (as is standard practice). The results can be seen in fig. 2. We used \( \lambda_U = 75 \) for CIFAR-10. We created 5 splits for each number of labeled points, each with a different random seed. Each model was trained on each split and the error rates were reported by the mean and variance across splits. We find that MixMatch outperforms all other methods by a significant margin, for example reaching an error rate of 6.24% with 4000 labels. For reference, on the same model, fully supervised training on all 50000 samples achieves an error rate of 4.17%. Furthermore, MixMatch obtains an error rate of 11.08% with only 250 labels. For comparison, at 250 labels the next-best-performing method (VAT [31]) achieves an error rate of 36.03, over 4.5 times higher than MixMatch considering that 4.17% is the error limit obtained on our model with fully supervised learning. In addition, at 4000 labels the next-best-performing method (Mean Teacher [44]) obtains an error rate of 10.36%, which suggests that MixMatch can achieve similar performance with only 1/16 as many labels. We believe that the most interesting comparisons are with very few labeled data points since it reveals the method’s sample efficiency which is central to SSL.

CIFAR-10 and CIFAR-100 with a larger model Some prior work [44, 12] has also considered the use of a larger, 26 million-parameter model. Our base model, as used in [35], has only 1.5 million parameters which confounds comparison with these results. For a more reasonable comparison to these results, we measure the effect of increasing the width of our base ResNet model and evaluate MixMatch’s performance on a 28-layer Wide Resnet model which has 135 filters per layer, resulting in 26 million parameters. We also evaluate MixMatch on this larger model on CIFAR-100 with 10000 labels, to compare to the corresponding result from [2]. The results are shown in table 1. In general, MixMatch matches or outperforms the best results from [2], though we note that the comparison still remains problematic due to the fact that the model from [44, 2] also uses more
We thus compare in both experimental settings. With extra α we used for CIFAR-10 and CIFAR-100 error rate (with 4,000 and 10,000 labels respectively) with larger models (26 million parameters).

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Teacher</td>
<td>6.28</td>
<td>-</td>
</tr>
<tr>
<td>SWA</td>
<td>5.00</td>
<td>28.80</td>
</tr>
<tr>
<td>MixMatch</td>
<td>4.95 ± 0.08</td>
<td>25.88 ± 0.30</td>
</tr>
</tbody>
</table>

Table 1: CIFAR-10 and CIFAR-100 error rate.

<table>
<thead>
<tr>
<th>Method</th>
<th>1000 labels</th>
<th>5000 labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>CutOut</td>
<td>-</td>
<td>12.74</td>
</tr>
<tr>
<td>IIC</td>
<td>-</td>
<td>11.20</td>
</tr>
<tr>
<td>SWWAPE</td>
<td>25.70</td>
<td>-</td>
</tr>
<tr>
<td>CC-GAN²</td>
<td>22.20</td>
<td>-</td>
</tr>
<tr>
<td>MixMatch</td>
<td>10.18 ± 1.46</td>
<td>5.59</td>
</tr>
</tbody>
</table>

Table 2: STL-10 error rate using 1000-label splits or the entire 5000-label training set.

Table 3: Comparison of error rates for SVHN and SVHN+Extra for MixMatch. The last column ("All") contains the fully-supervised performance with all labels in the corresponding training set.

MixMatch 4.2.3 Ablation Study

As with CIFAR-10, we evaluate the performance of each SSL method on SVHN with a varying number of labels from 250 to 4000. As is standard practice, we first consider the setting where the 73257-example training set is split into labeled and unlabeled data. The results are shown in fig. 3. We used λ_U = 250. Here again the models were evaluated on 5 splits for each number of labeled points, each with a different random seed. We found MixMatch’s performance to be relatively constant (and better than all other methods) across all amounts of labeled data. Surprisingly, after additional tuning we were able to obtain extremely good performance from Mean Teacher [44], though its error rate was consistently slightly higher than MixMatch’s.

Note that SVHN has two training sets: train and extra. In fully-supervised learning, both sets are concatenated to form the full training set (604388 samples). In SSL, for historical reasons the extra set was left aside and only train was used (73257 samples). We argue that leveraging both train and extra for the unlabeled data is more interesting since it exhibits a higher ratio of unlabeled samples over labeled ones. We report error rates for both SVHN and SVHN+Extra in table 3. For SVHN+Extra we used α = 0.25, λ_U = 250 and a lower weight decay of 0.000002 due to the larger amount of available data. We found that on both training sets, MixMatch nearly matches the fully-supervised performance on the same training set almost immediately – for example, MixMatch achieves an error rate of 2.22% with only 250 labels on SVHN+Extra compared to the fully-supervised performance of 1.71%. Interestingly, on SVHN+Extra MixMatch outperformed fully supervised training on SVHN without extra (2.59% error) for every labeled data amount considered. To emphasize the importance of this, consider the following scenario: You have 73257 examples from SVHN with 250 examples labeled and are given a choice: You can either obtain 8× more unlabeled data and use MixMatch or obtain 293× more labeled data and use fully-supervised learning. Our results suggest that obtaining additional unlabeled data and using MixMatch is more effective, which conveniently is likely much cheaper than obtaining 293× more labels.

STL-10 STL-10 contains 5000 training examples aimed at being used with 10 predefined folds (we use the first 5 only) with 1000 examples each. However, some prior work trains on all 5000 examples. We thus compare in both experimental settings. With 1000 examples MixMatch surpasses both the state-of-the-art for 1000 examples as well as the state-of-the-art using all 5000 labeled examples. Note that none of the baselines in table 3 use the same experimental setup (i.e. model), so it is difficult to directly compare the results; however, because MixMatch obtains the lowest error by a factor of two, we take this to be a vote in confidence of our method. We used λ_U = 50.

4.2.3 Ablation Study

Since MixMatch combines various semi-supervised learning mechanisms, it has a good deal in common with existing methods in the literature. As a result, we study the effect of removing or
Ablation study results. All values are error rates on CIFAR-10 with 250 or 4000 labels.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>250 labels</th>
<th>4000 labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>MixMatch</td>
<td>11.80</td>
<td>6.00</td>
</tr>
<tr>
<td>MixMatch without distribution averaging ($K = 1$)</td>
<td>17.09</td>
<td>8.06</td>
</tr>
<tr>
<td>MixMatch with $K = 3$</td>
<td>11.55</td>
<td>6.23</td>
</tr>
<tr>
<td>MixMatch with $K = 4$</td>
<td>12.45</td>
<td>5.88</td>
</tr>
<tr>
<td>MixMatch without temperature sharpening ($T = 1$)</td>
<td>27.83</td>
<td>10.59</td>
</tr>
<tr>
<td>MixMatch with parameter EMA</td>
<td>11.86</td>
<td>6.47</td>
</tr>
<tr>
<td>MixMatch without MixUp</td>
<td>39.11</td>
<td>10.97</td>
</tr>
<tr>
<td>MixMatch with MixUp on labeled only</td>
<td>32.16</td>
<td>9.22</td>
</tr>
<tr>
<td>MixMatch with MixUp on unlabeled only</td>
<td>12.35</td>
<td>6.83</td>
</tr>
<tr>
<td>MixMatch with MixUp on separate labeled and unlabeled</td>
<td>12.26</td>
<td>6.50</td>
</tr>
<tr>
<td>Interpolation Consistency Training [45]</td>
<td>38.60</td>
<td>6.81</td>
</tr>
</tbody>
</table>

We carried out the ablation on CIFAR-10 with 250 and 4000 labels; the results are shown in table 4. We find that each component contributes to MixMatch’s performance, with the most dramatic differences in the 250-label setting. Despite Mean Teacher’s effectiveness on SVHN (fig. 3), we found that using a similar EMA of parameter values hurt MixMatch’s performance slightly.

4.3 Privacy-Preserving Learning and Generalization

Learning with privacy allows us to measure our approach’s ability to generalize. Indeed, protecting the privacy of training data amounts to proving that the model does not overfit: a learning algorithm is said to be differentially private (the most widely accepted technical definition of privacy) if adding, modifying, or removing any of its training samples is guaranteed not to result in a statistically significant difference in the model parameters learned [13]. For this reason, learning with differential privacy is, in practice, a form of regularization [33]. Each training data access constitutes a potential privacy leakage, encoded as the pair of the input and its label. Hence, approaches for deep learning from private training data, such as DP-SGD [11] and PATE [36], benefit from accessing as few labeled private training points as possible when computing updates to the model parameters. Semi-supervised learning is a natural fit for this setting.

We use the PATE framework for learning with privacy. A student is trained in a semi-supervised way from public unlabeled data, part of which is labeled by an ensemble of teachers with access to private labeled training data. The fewer labels a student requires to reach a fixed accuracy, the stronger is the privacy guarantee it provides. Teachers use a noisy voting mechanism to respond to label queries from the student, and they may choose not to provide a label when they cannot reach a sufficiently strong consensus. For this reason, if MixMatch improves the performance of PATE, it would also illustrate MixMatch’s improved generalization from few canonical exemplars of each class.

We compare the accuracy-privacy trade-off achieved by MixMatch to a VAT [31] baseline on SVHN. VAT achieved the previous state-of-the-art of 91.6% test accuracy for a privacy loss of $\varepsilon = 4.96$ [37]. Because MixMatch performs well with few labeled points, it is able to achieve $95.21 \pm 0.17\%$ test
accuracy for a much smaller privacy loss of $\varepsilon = 0.97$. Because $e^\varepsilon$ is used to measure the degree of privacy, the improvement is approximately $e^4 \approx 55\times$, a significant improvement. A privacy loss $\varepsilon$ below 1 corresponds to a much stronger privacy guarantee. Note that in the private training setting the student model only uses 10,000 total examples.

5 Conclusion

We introduced MixMatch, a semi-supervised learning method which combines ideas and components from the current dominant paradigms for SSL. Through extensive experiments on semi-supervised and privacy-preserving learning, we found that MixMatch exhibited significantly improved performance compared to other methods in all settings we studied, often by a factor of two or more reduction in error rate. In future work, we are interested in incorporating additional ideas from the semi-supervised learning literature into hybrid methods and continuing to explore which components result in effective algorithms. Separately, most modern work on semi-supervised learning algorithms is evaluated on image benchmarks; we are interested in exploring the effectiveness of MixMatch in other domains.

Acknowledgement

We would like to thank Balaji Lakshminarayanan for his helpful theoretical insights.

References


### A Notation and definitions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(p, q)$</td>
<td>Cross-entropy between “target” distribution $p$ and “predicted” distribution $q$</td>
</tr>
<tr>
<td>$x$</td>
<td>A labeled example, used as input to a model</td>
</tr>
<tr>
<td>$p$</td>
<td>A (one-hot) label</td>
</tr>
<tr>
<td>$L$</td>
<td>The number of possible label classes (the dimensionality of $p$)</td>
</tr>
<tr>
<td>$\mathcal{X}$</td>
<td>A batch of labeled examples and their labels</td>
</tr>
<tr>
<td>$\mathcal{X}'$</td>
<td>A batch of processed labeled examples produced by MixMatch</td>
</tr>
<tr>
<td>$u$</td>
<td>An unlabeled example, used as input to a model</td>
</tr>
<tr>
<td>$q$</td>
<td>A guessed label distribution for an unlabeled example</td>
</tr>
<tr>
<td>$\mathcal{U}$</td>
<td>A batch of unlabeled examples</td>
</tr>
<tr>
<td>$\mathcal{U}'$</td>
<td>A batch of processed unlabeled examples with their label guesses produced by MixMatch</td>
</tr>
<tr>
<td>$\theta$</td>
<td>The model’s parameters</td>
</tr>
<tr>
<td>$p_{\text{model}}(y \mid x; \theta)$</td>
<td>The model’s predicted distribution over classes</td>
</tr>
<tr>
<td>$\text{Augment}(x)$</td>
<td>A stochastic data augmentation function that returns a modified version of $x$. For example, $\text{Augment}(\cdot)$ could implement randomly shifting an input image, or implement adding a perturbation sampled from a Gaussian distribution to $x$.</td>
</tr>
<tr>
<td>$\lambda_\mathcal{U}$</td>
<td>A hyper-parameter weighting the contribution of the unlabeled examples to the training loss</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Hyperparameter for the Beta distribution used in MixUp</td>
</tr>
<tr>
<td>$T$</td>
<td>Temperature parameter for sharpening used in MixMatch</td>
</tr>
<tr>
<td>$K$</td>
<td>Number of augmentations used when guessing labels in MixMatch</td>
</tr>
</tbody>
</table>
## B Tabular results

### B.1 CIFAR-10

Training the same model with supervised learning on the entire 50000-example training set achieved an error rate of 4.13%.

<table>
<thead>
<tr>
<th>Methods/Labels</th>
<th>250</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PiModel</td>
<td>53.02 ± 2.05</td>
<td>41.82 ± 1.52</td>
<td>31.53 ± 0.98</td>
<td>23.07 ± 0.66</td>
<td>17.41 ± 0.37</td>
</tr>
<tr>
<td>PseudoLabel</td>
<td>49.98 ± 1.17</td>
<td>40.55 ± 1.70</td>
<td>30.91 ± 1.73</td>
<td>21.96 ± 0.42</td>
<td>16.21 ± 0.11</td>
</tr>
<tr>
<td>Mixup</td>
<td>47.43 ± 0.92</td>
<td>36.17 ± 1.36</td>
<td>25.72 ± 0.66</td>
<td>18.14 ± 1.06</td>
<td>13.15 ± 0.20</td>
</tr>
<tr>
<td>VAT</td>
<td>36.03 ± 2.82</td>
<td>26.11 ± 1.52</td>
<td>18.68 ± 0.40</td>
<td>14.40 ± 0.15</td>
<td>11.05 ± 0.31</td>
</tr>
<tr>
<td>MeanTeacher</td>
<td>47.32 ± 4.71</td>
<td>42.01 ± 5.86</td>
<td>17.32 ± 4.00</td>
<td>12.17 ± 0.22</td>
<td>10.36 ± 0.25</td>
</tr>
<tr>
<td>MixMatch</td>
<td>11.08 ± 0.87</td>
<td>9.65 ± 0.94</td>
<td>7.75 ± 0.32</td>
<td>7.03 ± 0.15</td>
<td>6.24 ± 0.06</td>
</tr>
</tbody>
</table>

Table 5: Error rate (%) for CIFAR10.

### B.2 SVHN

Training the same model with supervised learning on the entire 73257-example training set achieved an error rate of 2.59%.

<table>
<thead>
<tr>
<th>Methods/Labels</th>
<th>250</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PiModel</td>
<td>17.65 ± 0.27</td>
<td>11.44 ± 0.39</td>
<td>8.60 ± 0.18</td>
<td>6.94 ± 0.27</td>
<td>5.57 ± 0.14</td>
</tr>
<tr>
<td>PseudoLabel</td>
<td>21.16 ± 0.88</td>
<td>14.35 ± 0.37</td>
<td>10.19 ± 0.41</td>
<td>7.54 ± 0.27</td>
<td>5.71 ± 0.07</td>
</tr>
<tr>
<td>Mixup</td>
<td>39.97 ± 1.89</td>
<td>29.62 ± 1.54</td>
<td>16.79 ± 0.63</td>
<td>10.47 ± 0.48</td>
<td>7.96 ± 0.14</td>
</tr>
<tr>
<td>VAT</td>
<td>8.41 ± 1.01</td>
<td>7.44 ± 0.79</td>
<td>5.98 ± 0.21</td>
<td>4.85 ± 0.23</td>
<td>4.20 ± 0.15</td>
</tr>
<tr>
<td>MeanTeacher</td>
<td>6.45 ± 2.43</td>
<td>3.82 ± 0.17</td>
<td>3.75 ± 0.10</td>
<td>3.51 ± 0.09</td>
<td>3.39 ± 0.11</td>
</tr>
<tr>
<td>MixMatch</td>
<td>3.78 ± 0.26</td>
<td>3.64 ± 0.46</td>
<td>3.27 ± 0.31</td>
<td>3.04 ± 0.13</td>
<td>2.89 ± 0.06</td>
</tr>
</tbody>
</table>

Table 6: Error rate (%) for SVHN.
### B.3 SVHN+Extra

Training the same model with supervised learning on the entire 604388-example training set achieved an error rate of 1.71%.

<table>
<thead>
<tr>
<th>Methods/Labels</th>
<th>250</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PiModel</td>
<td>13.71 ± 0.32</td>
<td>10.78 ± 0.59</td>
<td>8.81 ± 0.33</td>
<td>7.07 ± 0.19</td>
<td>5.70 ± 0.13</td>
</tr>
<tr>
<td>PseudoLabel</td>
<td>17.71 ± 0.78</td>
<td>12.58 ± 0.59</td>
<td>9.28 ± 0.38</td>
<td>7.20 ± 0.18</td>
<td>5.56 ± 0.27</td>
</tr>
<tr>
<td>Mixup</td>
<td>33.03 ± 1.29</td>
<td>24.52 ± 0.59</td>
<td>14.05 ± 0.79</td>
<td>9.06 ± 0.55</td>
<td>7.27 ± 0.12</td>
</tr>
<tr>
<td>VAT</td>
<td>7.44 ± 1.38</td>
<td>7.37 ± 0.82</td>
<td>6.15 ± 0.53</td>
<td>4.99 ± 0.30</td>
<td>4.27 ± 0.30</td>
</tr>
<tr>
<td>MeanTeacher</td>
<td>2.77 ± 0.10</td>
<td>2.75 ± 0.07</td>
<td>2.69 ± 0.08</td>
<td>2.60 ± 0.04</td>
<td>2.54 ± 0.03</td>
</tr>
<tr>
<td>MixMatch</td>
<td>2.22 ± 0.08</td>
<td>2.17 ± 0.07</td>
<td>2.18 ± 0.06</td>
<td>2.12 ± 0.03</td>
<td>2.07 ± 0.05</td>
</tr>
</tbody>
</table>

Table 7: Error rate (%) for SVHN+Extra.

Figure 4: Error rate comparison of MixMatch to baseline methods on SVHN+Extra for a varying number of labels. With 250 examples we reach nearly the state of the art compared to supervised training for this model.

### C 13-layer ConvNet results

Early work on semi-supervised learning used a 13-layer convolutional network architecture [31, 44, 25]. In table 8, we present results on a similar architecture. We caution against comparing these numbers directly to previous work as we use a different implementation and training process [35].

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>SVHN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250</td>
<td>4000</td>
</tr>
<tr>
<td>Mean Teacher</td>
<td>46.34</td>
<td>88.57</td>
</tr>
<tr>
<td>MixMatch</td>
<td>85.69</td>
<td>93.16</td>
</tr>
</tbody>
</table>

Table 8: Results on a 13-layer convolutional network architecture.