Semi-Supervised learning with MixMatch

Martin Iglesias Goyanes
KTH Royal Institute of Technology
martinig@kth.se

Alex Loiko
KTH Royal Institute of Technology
loiko@kth.se

Abstract

MixMatch is a high-level algorithm for semi-supervised learning (SSL) that achieves state-of-the-art results on semi-supervised learning image classification tasks. Its key steps include mixing in unlabeled samples with labeled (MixUp) and penalizing inconsistent prediction of unlabeled samples. In this work, we reproduce parts of the key findings of [2], and also discuss the limitations of the method. In particular, we investigate the dependence of the results on the data augmentation used, the training length, and the unlabeled loss term of MixMatch. We are able to achieve 90.8% test accuracy on CIFAR10 when trained with 4000 labeled and 36000 unlabeled samples.

1 Introduction

SSL methods have proven to be effective for exploiting unlabeled data and reducing the need for huge, labeled datasets. It is a type of learning that falls between supervised and unsupervised. An SSL algorithm is given some supervision information in addition to unlabeled input — but not for all examples [1]. Big labeled datasets have contributed to recent progress in training massive, deep neural networks. However, collecting labeled data is costly for many learning tasks because it necessitates human expertise [2]. In practice, fully functional unsupervised learning or SSL would be highly desirable as the ground truth is not always present in real-world data sources. Recent advances in deep learning research have been delving into the idea of finding ways with training through unsupervised and semi-supervised data that could potentially decrease the need for having big labeled datasets [3, 4, 5]. One approach that has grown attention is SSL using the MixMatch algorithm [2]. It aims to guess low-entropy labels for data-augmented unlabeled samples and mixes labeled and unlabeled samples using MixUp [6], which itself is a simple learning principle to prevent the model from memorizing labels and make it less sensitive to adversarial examples. This approach has achieved state-of-the-art results such as an error rate of 6.24% with 4000 labels on CIFAR10 [2]. This project aimed to replicate the implementation of MixMatch and evaluate its performance as the label proportion changes. This was done by using a WideRes Net (WRN) 10-10, a certain proportion of the CIFAR 10 dataset remained labeled, and the rest of the unlabeled samples went through the steps of the MixMatch algorithm. A set of 4000 labels performed best with a 90.8% test accuracy.

2 Related Work

2.1 MixMatch Algorithm

The core of this paper's work is based on Berthelot et al. [2] who introduces MixMatch, which is their proposed SSL method. Given a batch X of labeled samples along with their one-hot targets

and a batch U with equal size but with unlabeled samples, the MixMatch algorithm augments and mixes the two batches and produces augmented sample batches X' and U'. The unlabeled samples in U' are assigned labels q through a multi-step algorithm which is described in list 2.1. In the end, MixMatch produces a batch of 'mostly labeled' samples X ' and a batch of 'mostly unlabeled' samples U '. Separate losses are computed separately from each batch and added together according to $L = L_X + \lambda_U L_U$.

$$L_X = \frac{1}{|X'|} \sum_{x,p \in X'} H\left(p, p_{\text{model}}\left(y \mid x; \theta\right)\right) \quad (1) \quad L_U = \frac{1}{L|U'|} \sum_{u,q \in U'} \|q - p_{\text{model}}\left(y \mid u; \theta\right)\|_2^2 \quad (2)$$

The cross-entropy between the distributions p and q (denoted as $H(p, q)$) is computed for the 'mostly labeled' examples X '. Each part of the algorithm in figure 1 is described below. Intuitively, the model should maximize the likelihood of the known true labels of X, hence the KL term. It should also give consistent predictions on unlabeled samples. This could hypothetically be enforced by another KL term, however, KL gives large penalties for predicting probabilities close to zero when the target is nonzero. It is less stable than using L2 distance, which is done in MixMatch.

1. **Data Augmentation**
   Following typical SSL methods, MixMatch uses augmentations on both labeled and unlabeled data. For each sample in X, it generates another transformed sample with the use of an augmentation function. And, for each sample in U, MixMatch generates K augmented samples.

2. **Label Guessing**
   MixMatch uses the model's predictions to make a "guess" for the label of each unlabeled example in U. The unsupervised loss term later uses this guess. To do so, it averages the predicted class distributions of the model across all K augmentations of x_b.

3. **Sharpening**
   Given the average prediction over augmentations q_b, we use a sharpening function to lower the entropy of the label distribution when creating a label estimate. In practice, we modify the "temperature" of this categorical distribution [7], which is characterized as the operation of sharpening. $Sharpen(p)_i = p_i^{1/T} / (\sum p_i^{1/T})$ where p is the input categorical distribution (in MixMatch it is the average class predicted over augmentations). T is a hyperparameter that controls the temperature, the lower, the more lower-entropy predictions the model will make. Intuitively, a well-performing model should produce low-entropy predictions for all samples, and different augmentations of the same samples should yield similar label distributions.

4. **MixUp Algorithm**
   Unlike other SSL works, the authors of MixMatch use MixUp with a combination of labeled and unlabeled samples with guesses. For a pair of samples and labels $(x1, p1)$ and $(x2, p2)$, MixUp produces the interpolated $(x', p') = \lambda(x1, p1) + (1 - \lambda)(x2, p2)$ where λ is a random number from a distribution biased towards 1. Intuitively, MixUp mixes a small random amount of $(x2, p2)$ in to $(x1, p1)$, which lets information about ground truth labels more easily be incorporated into the consistency loss term. The distribution of λ is $\lambda = max(\lambda', 1 - \lambda')$ where λ' is $Beta(\alpha, \alpha)$, and $\alpha < 1$ is a hyperparameter. In our experiments and in [2], $\alpha = 0.75$.

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).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \mathrm{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \mathrm{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \mathrm{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \mathrm{Shuffle}\big(\mathrm{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\mathrm{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\mathrm{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

Figure 1: MixMatch pseudocode implementation [2].

The algorithm above is based on the ideas of consistency regularization, entropy minimization and traditional regularization which are prevalent throughout the SSL field. Therefore, we will focus on explaining these concepts and leave out other methods such as transductive models, graph-based methods, and generative modeling.

2.2 Consistency Regularization

Data augmentation is a frequent regularization strategy in supervised learning for classification. Input is altered with the assumption that class semantics are unaffected. In image classification, for example, it is typical to elastically deform or add noise to an input picture, which can drastically affect the pixel content of an image without changing its label [8]. This may be used to artificially increase the size of a training set by creating an almost limitless stream of fresh, changed data. Consistency regularization adds data augmentation to semi-supervised learning by exploiting the principle that a classifier should produce the same class distribution for an unlabeled sample even after it has been augmented. Consistency regularization requires that an unlabeled sample x be classified as the augment of X, a self-augmentation. Different augmentations of the same sample should be classified similarly. Mathematically, this means that for an unlabeled sample x, we add to the loss the following term:

$$\|p_{\text{model}}\left(y \mid \text{ Augment }(x); \theta\right) - p_{\text{model}}\left(y \mid \text{ Augment }(x); \theta\right)\|_2^2$$

Note: The terms in the equation 5 are not identical since the augmentation is a stochastic operation. "Mean Teacher" [9] substitutes the output of the model using an exponential moving average of model parameter values for one of the components in equation 4. This creates a more steady target, which has been shown to enhance performance empirically. The employment of domain-specific data augmentation procedures is a disadvantage of these approaches. Instead, "Virtual Adversarial Training" [10] solves the problem by determining an additive perturbation to apply to the input that affects the output class distribution most. MixMatch employs conventional data augmentation for photos as a type of consistency regularization (random horizontal flips and crops).

2.3 Entropy Minimization

The decision boundary of the classifier should not cross into high-density portions of the marginal data distribution, according to a common underlying principle in many semi-supervised learning approaches. One technique to enforce this is to require low-entropy predictions from the classifier on unlabeled data. In [11] This is implemented explicitly by adding a term to the loss that minimizes the entropy of unlabeled data x on $p_{\text{model}}\left(y \mid x; \theta\right)$. In [10] this approach for minimizing the entropy is

combined with VAT (Virtual Adversarial Training), which obtained better results. Other techniques are "Pseudo-label" [12] - an extension to the training procedure which implicitly minimizes entropy. "Pseudo-label" constructs labels from model predictions that are of high confidence to use them inside the typical cross-entropy loss. Similar to "Pseudo-label", MixMatch's approach to minimizing entropy is also implicit. The unlabeled data's target distribution is "sharpened" through a procedure described in the MixMatch algorithm section.

2.4 Traditional Regularization

Regularization is the process of placing a restriction on a model to make it more difficult to memorize the training data and, as a result, make it more generalizable to unknown data. We employ weight decay, which penalizes the model parameters' L2 norm. In MixMatch, we also employ MixUp [6] to induce convex behavior "between" samples. MixUp is used as a regularizer (on labeled data points) as well as a semi-supervised learning tool (with unlabeled data points).

3 Method

3.1 Data

This paper uses the CIFAR-10 data set. Papers with code [13] present a list of top-performing methods on CIFAR-10, of which the top three performing methods are presented in this paper (see table 1).

Name	Year	Accuracy (%)
CaiT-M-36 U 224 [14]	2021	99.4
CvT-W24 [15]	2021	99.39
BiT-L [16]	2019	99.37

Table 1: Some of the state-of-the-art performing methods on CIFAR 10

As an alternative to the convolutional neural network, Touvron et al. [14] built and optimized a deeper transformer network for image classification, in doing so, they introduce Class-Attention in Image Transformers (CaiT). This model is built upon a vision transformer but they introduce LayerScale in each residual block and they also introduce their own built class-attention layers [14]. Wu et al. [15] present in their paper a new architecture, named Convolutional vision Transformer (CvT), that improves Vision Transformer (ViT) in performance and efficiency by introducing convolutions into ViT to yield the best of both designs. Kolesnikov et al. [16] revisit the paradigm of pre-training on large supervised datasets and fine-tunes the model by applying it to a target dataset. Big Transfer (BiT) combines a few carefully selected components and transfers using a simple heuristic [16], using a vanilla ResNet-v2 architecture for all their pre-trainings and they introduce BiT-HyperRule which is their heuristics that selects the most important hyperparameter for fine-tuning [16].

3.2 Model

3.2.1 Architecture

In our experiments, we used a variant of the residual network WideResNet-10-10 and WideResNet-28-2 [17]. It is chosen because of its performance and ease of implementation.

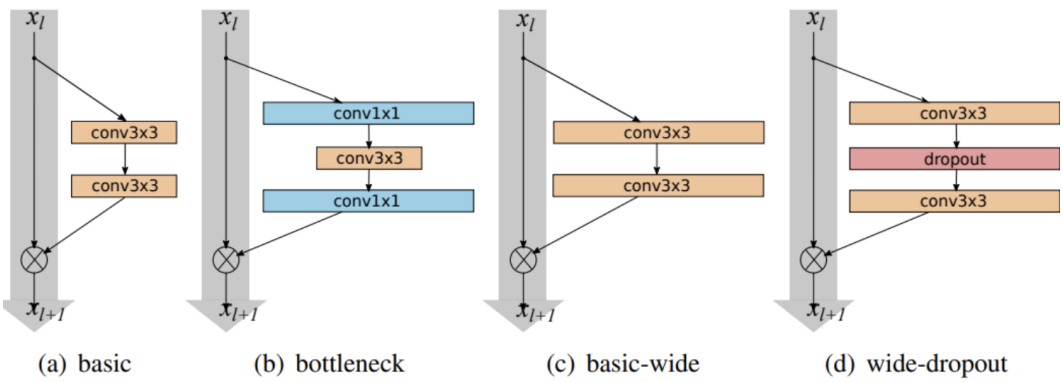

Figure 2: Res-net types [17].

An alternative approach would be using a regular ResNet. Figure 2 shows the difference between a wide residual block and a regular residual block. The difference here is that the wide residual block has more feature maps, hence making it "wider". This difference results in notably fewer layers than a regular ResNet has which makes it two times faster [17], a 16-layer of WRN has the same accuracy as a 1000-layer thin neural network [17]. Because of that, the WRN is faster to train [17] which is why WRN was chosen.

3.2.2 Implementation details

For our main experiments, we trained WRN-10-10 models for 50 epochs, where an epoch is a pass-through of all of the unlabeled data. MixMatch parameters introduce a couple of parameters, namely the number of augmentations K, sharpening temperature T, MixUp parameter α, and the unsupervised loss weight λ_U. In this paper, we chose the same values as the ones that Berthelot et al. [2] used, namely, $K = 2, T = 0.5, \alpha = 0.75$, and $\lambda_U = 75$. We linearly scaled λ_U from 0 to its final value (75) throughout the training. Each X and U batch were 64 samples. We used Adam with momentums 0.9 and 0.999, and an initial learning rate of $\mathbf{lr}$ 0.002, which decayed by 0.9 per epoch. We used a default momentum of $\mathbf{0.1}$ for batch normalization and no dropout in our WRN models. This paper used the PyTorch library to conduct the replication of MixMatch as opposed to TensorFlow, which was used by the original paper [2]. The decision of using PyTorch was based on the authors' level of familiarity with the framework. We first created a fully functional supervised-learning network architecture using the Wide ResNet-10-10 model [17]. We then evaluated the effectiveness of MixMatch on CIFAR-10. The evaluation was made by conducting a standard practice evaluation for SSL, which according to Berthelot et al. [2] is by treating most of the dataset as unlabeled and letting a small proportion of the dataset be labeled samples. We split up the training data into 50 000 training and 10 000 validation samples, according to the way Berthelot et al. [2] made the split. The training samples were split into labeled and unlabeled in five ways: 250, 500, 1000, 2000, and 4000 labeled samples with the remainder unlabeled, the reason why these splits were chosen was that Berthelot et al. [2] use the same type of splitting, which would make it easier to compare our results to theirs. We did not use exponential model averaging for making predictions of the unlabeled samples, the reason for that is according to the results of [2], it did not make any significant results that would lead to better performance. We first conducted a run with more complex augmentations than what [2] performed to understand what impact the augmentations have on MixMatch's result. Then we conducted a run with simple augmentations and compared the performance between the first and last run. Lastly, a comparison was made between our experiment results and [2] to understand what impact a shallower architecture has on the results of MixMatch.

4 Experiments and Results

We found that the kind of augmentation used greatly affects the results. Our default augmentation technique of [18] slowed down the training considerably compared to using a much simpler pad, crop, and flip technique described in [2]. We believe that at the start of training, it is most beneficial for the model to learn simple cues like pixel color, which enables it to quickly give high-confidence

Labeled samples	250	500	1000	2000	4000
Validation accuracy	51.49%	60.3%	71.05%	79.03%	84.03%

Table 2: Classifcation accuracy as labeled samples increase

predictions on some unlabeled images. Using an advanced augmentation technique forces the model to care less about simple features. We believe the model has too few labeled samples to learn advanced at the start of training, and therefore cannot make confident predictions for most of the unlabeled data.

The final validation accuracies can be seen in table 2. Due to resource limitations, we only performed one run. Thinking that our results were significantly worse than [2], we identified differences between our implementation and that of [2]. In addition to different augmentation, we saw that [2] used a constant learning rate, ca. 10 times longer training, a deeper WRN-28-2 model instead of our WRN-10-10, lower batch normalization momentum, weight decay, and exponential moving average parameters for the model that was evaluated. Changing all but the augmentation to the hyperparameter settings of [2], we obtained 90.8% test accuracy for the 4k labeled case.

5 Discussion and Conclusions

What we have learned from this experiment is that MixMatch is dependent on a deeper neural network than that of a Wide Residual Network with a depth of 10 to gain state-of-the-art results, and MixMatch also requires a minimal amount of augmentations, anymore augmentations than necessary could lead to worse performance. Due to the time limit, we could not apply MixMatch to other domains than computer vision as [2] suggested which we would highly suggest that future research should look into. Future research should also look at how variations of WRN perform on MixMatch as we can say from our findings, a shallower neural network does not seem to yield the ideal results that [2] managed to find with their WRN 28-2 model. We have, unfortunately, been unable to fully replicate the exact results from [2] due to the time and computationally constraints we had. The level of hyper-parameter tuning performed by the researchers at Google is something we are far from been capable with the hardware we had available. The same applies for techniques such as exponential moving average for unlabeled guesses. Nevertheless, these are not key in the research problem, and we were able of replicating the most interesting and powerful result, which is been able to achieve almost state-of-the-art supervised accuracy with a semi-supervised technique. Moreover, we were able to demonstrate how the proportion of labeled samples affects the results, and how impressive these are.

References

[1] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. 09 2006.

[2] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," 05 2019.

[3] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," 01 2020.

[4] A. Tarvainen and H. Valpola, "Weight-averaged consistency targets improve semi-supervised deep learning results," *CoRR*, vol. abs/1703.01780, 2017.

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

[6] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 10 2017.

[7] I. Goodfellow, *Deep learning*. Adaptive computation and machine learning, 2016.

[8] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[9] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," 2017.

[10] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1979–1993, 2019.

[11] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," vol. 17, 01 2004.

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

[13] "Papers with code - cifar-10 benchmark (image classification)."

[14] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 32–42, October 2021.

[15] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31, October 2021.

[16] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, *Big Transfer (BiT): General Visual Representation Learning*, pp. 491–507. 10 2020.

[17] S. Zagoruyko and N. Komodakis, "Wide residual networks," 05 2016.

[18] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," 2018.