Small scale replication and extension of *MixMatch: A Holistic Approach to Semi-Supervised Learning*

Sonja Horn **Povel Forsare Källman** **Carl Mannerstråle** **Filip Christiansen**

Abstract

Semi-supervised learning (SLL) combines supervised and unsupervised learning methods to mitigate the reliance on large labeled datasets. This study aims to reproduce parts of the paper *MixMatch: A Holistic Approach to Semi-Supervised Learning* which introduced a high performing SSL algorithm, especially effective on CIFAR-10. Additionally, we investigate different methods of manipulating the amount of regularization and what effects they have on performance. Our best results were achieved with 4000 labeled examples (like in the original paper), reaching a test accuracy of 83.77% after only 50 epochs of training. Furthermore, our results show that decreasing the influence of the unlabeled loss term early in the training (essentially decreasing the amount of regularization) does not have significant effects on the test accuracy. However, using additional data augmentation in the MixMatch algorithm by applying random brightness and contrast to images (increasing regularization) had a slight negative effect on performance, as expected. Interesting future work includes a more exhaustive search for the optimal ramp-up length for λ_u for shorter training time, and perhaps investigating non-linear ramp-up of λ_u and thereby the unlabeled loss term.

1 Introduction

Almost all supervised machine learning models need labeled data to train. Even though there exists tons of data including texts, images, time-series and more, only a small fraction is labeled. Labeling instances of data can often be difficult and time consuming as it requires the efforts of experienced human annotators. Concurrently, unlabeled data is relatively easy to obtain but fewer algorithms use them. As a consequence, semi-supervised learning (SSL) algorithms have been developed to address this issue. SSL combines supervised and unsupervised learning methods to mitigate the reliance on large labeled datasets. By leveraging small fractions of labeled data, a semi-supervised model labels the unlabeled data during training by predicting sample outputs using its current model [13].

This study aims to reproduce parts of the paper *MixMatch: A Holistic Approach to Semi-Supervised Learning* by David Berthelot et al, first published in 2019 in *Advances in Neural Information Processing Systems 32* [4]. The paper introduces the semi-supervised learning algorithm MixMatch, which is set apart by its loss function that combines the advantages of the three most common loss terms used for unlabeled data in SSL models. The results show that MixMatch is reaching state-of-the-art performance on several benchmark datasets and especially good results for CIFAR-10, for which the error rate was reduced by a factor 4.

To confirm and further investigate the performance of MixMatch, this study will implement the MixMatch algorithm and investigate how performance is affected by altering the amount of regularisation in different ways. First we manipulate the the linearly ramped up influence of the unlabeled loss on the total loss to decrease the amount of regularisation. Then, we increase the amount of regularization by adding additional types of augmentation to the labeled and unlabeled data. When replicating the experiments of the original paper, we were able to reach a test accuracy of 83.77% in only 50 epochs of training. The results of the regularization experiments suggest that a lower amount if regularization

Project report for DD2424

is needed for such short training, but this could not be confirmed due to the time and computational limitations of this project.

2 Related work

Using 4000 labeled samples for training, MixMatch reached an error rate of 6.24% on CIFAR-10, while the next best model at the time (Mean Teacher) had an error rate of 10.36% [4]. Since *MixMatch: A Holistic Approach to Semi-Supervised Learning* was first published, David Berthelot with colleagues have published improved versions of MixMatch. The first follow-up, ReMixMatch, was published later in 2019 and featured distribution alignment and augmentation anchoring, which increased the test accuracy to 93.73% [3]. A later version called FixMatch (published in 2020) used consistency regularization and pseudo-labeling which further increased the test accuracy to 95.69% on CIFAR-10 and 4000 labels [8]. This model is the second best performing SSL image classifier on CIFAR-10 and 4000 labels at the moment, the best model only 0.13% better [7].

MixMatch has been surpassed by the paper *EnAET: Self-Trained Ensemble AutoEncoding Transformations for Semi-Supervised Learning* [10], which holds the current state-of-the-art results on CIFAR-10 for semi-supervised learning, with a error rate of 5.35% with 4000 labeled samples. EnAET builds on MixMatch, by incorporating Auto-Encoding Transformations (AET) [12], a representational method with previous success in self-supervised learning. In the first step of AET, an ensemble of both spatial and non-spatial transformations is applied to a given image. Then, the representations from an encoder, of both the original and the transformed image, are used as inputs to a decoder. The task of the decoder is then to decode the applied transformations. The idea behind this is to force the encoder to learn a good visual representation with sufficient information for the decoder to be able to successfully recover the parameterization of the applied transformations. The AET-loss is computed as the mean-squared error between the predicted and applied transformations. Furthermore, to enforce consistent and transformation-invariant predictions, the KL-divergence[1] between the predicted conditional class probabilities, of the original and the transformed image is added to the overall loss. EnAET trains a classifier by minimizing a linear combination of the MixMatch-loss, the AET-loss, and the KL-divergence-loss.

Looking at the current leaderboard for CIFAR-10, the best performing model as of now is called BiT-L and presented in the paper Big Transfer (BiT): General Visual Representation Learning. Like Mixmatch, BiT-L was also developed by Google Research and is a model applying transfer learning in which the model is trained in sequences. Initially, the model is trained once on a large dataset. Consequently, the weights are then used to initialize subsequent tasks. These tasks can then be performed and solved using fewer data points, thus requiring less computational power. BiT-L achieves an impressive accuracy of 99.4% on CIFAR-10. Additionally, BiT-L records an accuracy of 97.0% with only 10 examples per class, adding up to 100 samples in total.[2]

Excluding the BiT-L model, the above mentioned models all use residual network, so called ResNets, which are convolutional neural nets with skip connections, using ReLU and batch normalization [11]. There are a number of variations of ResNets depending on the depth of the network and width of the layers, but MixMatch, ReMixMatch and FixMatch all use ResNet-28-2 for CIFAR-10. [4, 3, 8]. The name of this network architecture reveals that it is a residual net with depth 28 and width 2.

3 Data

The data used in this project is the benchmark dataset CIFAR-10. The CIFAR-10 dataset was retrieved from the Toronto university page of Alex Krizhevsky [1]. CIFAR-10 consists of 32x32 colour images (3 color channels) and has 10 classes, all classes being mutually exclusive and the data equally distributed among the classes. There are 50000 images for training and an additional 10000 images in the test set [1]. Except for normalizing the dataset and encoding the labels into one-hot, there was no additional preprocessing of CIFAR-10.

Generally when training SSL models, it is common practice to use a small portion of the training data as labeled, while the majority of the data is used as unlabeled [4]. This simply means that for a

[1]The KL-divergence is a measure of difference between two probability distributions.

part of the training set, the labels remain unused. For this study, splits of (250, 49750), (1000, 49000) and (4000, 46000) labeled respectively unlabeled were constructed.

4 Method

4.1 MixMatch algorithm

This section describes the fundamental parts of the MixMatch algorithm, while we refer to the original paper for the step-by-step functionality of the algorithm (*Algorithm 1*) [4]. In this section we will denote the labeled data as $X = \{x_n, p_n\}_{n=1}^{N}$ and the unlabeled data as $U = \{u_n\}_{n=1}^{N}$. The MixMatch algorithm uses augmentation on both the labeled data and the unlabeled data, where $\hat{X} = \{\hat{x}_n, p_n\}_{n=1}^{N}$ is the result of augmenting each $x_n \in X$ once, keeping the same labels. The unlabeled data samples are augmented K times, where $k \in \{1, K\}$ is a hyperparameter of the model. For each augmented unlabeled sample $\hat{u}_{n,k}$ a label is guessed, and the final guessed label for all k augmentations is set to the average label of all K augmentations, determined by Eq. 1 and sharpened according to Eq. 2. The hyperparameter T determines how much the model prediction should be sharpened, where $T \to 0$ results in a one-hot encoding. The result of the augmentation of the unlabeled data is $\hat{U} = \{\hat{u}_{n,k}, q_n\}_{n=k=1}^{N,K}$.

$$q_n = \frac{1}{K} \sum_{k=1}^{K} p_{model}(|\hat{u}_{n,k}; \theta) \tag{1}$$

$$\text{sharpen}(p, T) = \frac{p^{1/T}}{\sum_{j=1}^{L} p_j^{1/T}} \tag{2}$$

After augmenting the labeled and unlabeled data and combining and shuffling them to create a new set, W, an adapted version of MixUp is applied to further augment the data. Let (x_1, p_1) and (x_2, p_2) be a pair of samples and their corresponding labels or guessed labels, and α the beta distribution hyperparameter. Eq. 3 then results in samples and labels that are a mix of the pair. MixUp is applied to W and $\hat{X}$ and W and $\hat{U}$ separately, resulting in the final sets X' and U' which are used for calculating the loss and updating the model parameters.

$$\begin{aligned} \lambda &\sim Beta(\alpha, \alpha) \\ \lambda' &= max(\lambda, 1 - \lambda) \\ x' &= \lambda' x_1 + (1 - \lambda') x_2 \\ p' &= \lambda' p_1 + (1 - \lambda') p_2 \end{aligned} \tag{3}$$

The MixMatch loss function leverages the advantages of the three most common loss terms for unlabeled data in SSL when calculating the overall loss. The MixMatch loss is defined according to Eq. 4 where L is the number of classes, and uses X' and U' obtained by the main MixMatch algorithm. The fourth and final hyperparameter λ_u is introduced here.

$$\begin{aligned} L_x &= \frac{1}{|X'|} \sum_{x,p \in X'} H(p, p_{model}(y|x; \theta)) \\ L_u &= \frac{1}{L|U'|} \sum_{u,q \in U'} \|q - p_{model}(y|u; \theta)\|_2^2 \\ L &= L_x + \lambda_u L_u \end{aligned} \tag{4}$$

The hyperparameters of the model used in the main MixMatch algorithm (*Algorithm 1* [4]) are hence K, T, and λ_u. In the paper, these parameters were fixed in all CIFAR-10 experiments. Each part of MixMatch was implemented from scratch but strictly following the equations in the paper (presented above). When translating the equations to python code, we were inspired by the MixMatch GitHub repository in how TensorFlow was used to interpret the equations [4, 6].

4.2 Data augmentation

A central part of MixMatch is data augmentation of both labeled and unlabeled data as described in the previous subsection. The data augmentation both extends the data set and provides consistency regularization by requiring an augmented image to have the same label as the original image [4].

As basic augmentation method we have used horizontal flips and random crops, following the original paper. The horizontal flips were achieved by a TensorFlow built-in function. The random crops are accomplished by adding a reflective padding around the original image making it temporarily larger, and then returning a randomly placed crop the size of the original image. When an image was augmented, it was first augmented by a random horizontal flip and then by making the random crop.

In our experiments when we use additional augmentation methods to vary the amount of regularisation, the images are also manipulated with random brightness and contrast. These augmentations were also achieved by TensorFlow built-in functions.

4.3 Wide ResNet & high-level implementation

To be able to reproduce the results of [4] for CIFAR-10, Wide ResNet-28-2 was used. The implementation of the actual Wide ResNet-28 model was borrowed from [9], and is the only part of our code that was not implemented by us. We used the TensorFlow library throughout the implementation instead of PyTorch or other deep learning libraries, since one of our main learning goals was to gain experience using this library. When interpreting the MixMatch equations in python and TensorFlow code, we were inspired by the original GitHub repository [6].

Wide ResNet-28-2 has 1.47 million parameters, that we optimize using the Adam optimizer. Adam uses adaptive learning rates that adapt to each individual network parameter during training by utilizing an exponential moving average [5]. We strictly follow the learning rate schedule by Berthelot et al., meaning *"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"* [4]. Also following the original paper, we linearly ramp up λ_u to its maximum value by specifying a ramp-up length that decides how fast (in how many update steps) λ_u should reach its maximum value.

Generally, our implementation looks quite different from the original one. One reason for this is that the original MixMatch implementation is extremely dynamic and flexible which also makes it difficult to understand, consisting of many unnecessary components with respect to our task. Our goal was to have a concise and intuitive implementation that was restricted in functionality to the task at hand, considering the limited time frame of this project. Our approach of implementing from scratch also allowed us to understand all parts of MixMatch which would not have been the case if we had just altered some existing implementation.

4.4 Limitations

Berthelot et al. varied the number of labeled examples between 250 and 4000 during training while using the rest as unlabeled data. More specifically, they trained with 250, 500, 1000, 2000 and 4000 labeled exampled. We will adapt the same method in order to compare our results, but due to computational and time constraints we have restricted this study to only 250, 1000 and 4000 labeled examples. Furthermore, the original paper only states the test error achieved after 1024 epochs of training. Again, due to restricted time and computational resources we are not able to train for long enough, and thereby not able to fully reproduce the results of the original paper. However, we will make some assumptions about our ability to replicate the methods of Berthelot et al. based on trends observed in the results.

5 Experiments

Our experiments are split in three parts according to the problem we are working on. Firstly, we want to reproduce the results for CIFAR-10 to confirm the results of Berthelot et al. and ensure that our implementation is correct. Secondly, we experiment with the ramp-up length to try and adapt the model to shorter training by decreasing the amount of regularisation. Lastly, we want to

investigate how additional augmentation methods affect the performance of the model through the added regularisation.

5.1 Reproducing results

For the CIFAR-10 reproducing experiments, the hyperparameters K, T and λ_u were fixed to $K = 2$, $T = 0.5$ and $\lambda_u = 75$, following the experiments of the original paper. We trained our model with 250, 1000 and 4000 labeled examples, where every training run was 50 epochs. A batch size of 64 was used as well as a ramp-up length of 16 just like in the original paper. To follow the evolution of the model performance, we evaluate the model on the test set after every 10th epoch. No parameters were updated during the evaluations and no other learning was made during this evaluation. The test accuracy after 10, 20, 30, 40 and 50 epochs of training are presented in Table 1.

Table 1: Test accuracy achieved after 10 to 50 epochs of training using 250, 1000 and 4000 labeled examples on CIFAR-10

Labels	250	1000	4000
10 epochs	56.05%	68.00%	76.67%
20 epochs	59.79%	70.98%	79.74%
30 epochs	60.80%	72.65%	81.67%
40 epochs	61.24%	73.13%	83.89%
50 epochs	62.72%	73.61%	83.49%

Table 1 shows that our model reaches very high test accuracies after only 50 epochs of training, especially for 4000 labeled examples. It also shows that there is a steady increase in test accuracy as the training proceeds. We thereby find it reasonable to assume that the test accuracy would increase further if training was run for a longer period of time, and that the test accuracies would eventually reach similar values as in the original paper.

5.2 Linear ramp-up

In these experiments we are letting λ_u reach its maximum value slower. λ_u controls the influence of the unlabeled loss (see Eq. 4) on the total loss, which is a form of regularisation. A larger value for λ_u means that the unlabeled data with guessed labels has more influence over the loss, and thereby that the regularization is stronger. When training for a long time like in the original paper, more regularisation is required in order to prevent the model from overfitting to the training data. This can be enabled by using a higher λ_u at an earlier stage in training. Since our experiments run training for a shorted time, it makes sense to decrease the amount of regularisation by letting λ_u reach its maximum value slower. Table 2 shows the same experiments as in Table 1, but with ramp-up length 50 ($\lambda_u = 75$ at epoch 50).

Table 2: Test accuracy for 250, 1000 and 4000 labeled examples using ramp-up length 50, after 50 epochs of training

Labels	250	1000	4000
10 epochs	52.05%	68.39%	74.66%
20 epochs	53.68%	71.49%	78.67%
30 epochs	52.98%	72.93%	81.01%
40 epochs	53.24%	73.82%	82.95%
50 epochs	53.97%	75.07%	83.77%

The results show that using a higher ramp-up length does not improve the test accuracy for a shorter training time except slightly for 1000 labeled examples. This might be a result of the ramp-up length being increased too much or not being increased enough. We also observe a remarkable decrease for

250 labeled examples. It is probably that the model suffered from overfitting when trained with only 250 labeled examples and the steeply decreased regularization.

5.3 Additional data augmentation

For the following experiment more regularization was added in terms of additional data augmentation in the MixMatch algorithm. This additional data augmentation (on top of random horizontal flips and crops) consisted of a random increase or decrease in brightness and contrast of the images. Table 3 shows the test accuracy at different epochs in the training when training with additional augmentation. Compared to the original data augmentation schedule, we can generally see a slight decrease in performance when using additional augmentation.

Table 3: Test accuracy for 250, 1000 and 4000 labeled examples using additional data augmentation, after 50 epochs of training

Labels	250	1000	4000
10 epochs	44.97%	62.77%	69.06%
20 epochs	57.31%	69.15%	76.78%
30 epochs	59.22%	71.16%	79.72%
40 epochs	58.79%	71.13%	79.78%
50 epochs	60.31%	72.16%	81.58%

5.4 Discussion

For the second and third experiments the aim was to decrease and increase the regularization, respectively. Even though Wide ResNet-28 is a relatively complex model, our hypothesis was that the model would benefit from less regularization due to the low amount of labeled data (compared to unlabeled data) along with such short training. However, the results in Table 2 did not show a significant difference in test accuracy compared to the first experiment. We speculate that a ramp-up length between 16 and 50 could be a good setting for shorter training schedules of this model, and that the increase from 16 to 50 might have been excessive. As previously mentioned, these speculations could not be explored further due to the time and computational limitations of this project.

Regarding experiment three where we instead increased the regularization with additional data augmentation techniques, we expected a drop in test accuracy. This expectation was also motivated by the fact that shorter training should require less regularization. In the results we observed a slight decrease compared to the results in Table 1, vaguely confirming our hypothesis. However, the decrease is not significant enough to draw any definite conclusions, but it does improve our confidence in this statement.

6 Conclusion

The main objective of the study was to replicate parts of the paper *MixMatch: A Holistic Approach to Semi-Supervised Learning* by Berthelot et al. and investigate how increasing and decreasing the amount of regularization using different methods would affect the performance. The MixMatch algorithm was successfully implemented with Wide ResNet-28 and we found that test accuracies close to 85% could be reached in just 50 epochs. We also found that for shorter training time, increasing the regularization by adding additional data augmentation had somewhat negative effects on the test accuracy as expected. Decreasing the regularization however did not result in higher test accuracies, perhaps because it was decreased too much leading to overfitting. As future work or extension of this study, it would be interesting to do a more exhaustive search for the optimal value of the ramp-up length of λ_u when training for fewer epochs (requiring less regularization). Another interesting idea would be to investigate if the performance would benefit from a non-linear ramp-up of λ_u.

References

[1] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. The cifar-10 dataset, 2009.

[2] Xiaohua Zhai Joan Puigcerver Jessica Yung Sylvain Gelly Alexander Kolesnikov, Lucas Beyer and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2020.

[3] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5049–5059. Curran Associates, Inc., 2019.

[5] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning, 2017. `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/`.

[6] Google Research. Mixmatch. `https://github.com/google-research/mixmatch`, 2019.

[7] Papers with code. Semi-supervised image classification on cifar-10, 4000 labels, 2020. `https://paperswithcode.com/paper/fixmatch-simplifying-semi-supervised-learning`.

[8] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[9] Nathan Tozer. mixmatch-tensorflow2.0. `https://github.com/ntozer/mixmatch-tensorflow2.0`, 2019.

[10] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning, 2019.

[11] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[12] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[13] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.