Revisiting FixMatch for Our Semi-Supervised Learning Problem

Donggyu Kim
20180065
Korea Advanced Institute of Science and Technology
eaststar@kaist.ac.kr

Abstract

Semi-supervised learning (SSL) is a fine approach to solve image classification problems, only with small supervision for a dataset. Google Research recently showed SSL algorithm called FixMatch, which achieves state-of-the-art performance in various SSL benchmarks. My goal was to implement FixMatch as base algorithm and deal with some additional problems. However, what I found is that FixMatch does not work so well; the given MixMatch baseline code appears the best in test. My code is available at `https://github.com/EaststarKim/CS492I_CV-project`.

1. Introduction

Nowadays, deep neural networks hold the foremost position in computer vision area. Many researches shows the success of deep neural networks in various problems. Their high performances are often acheived through supervised learning, which needs a labeld dataset. However, labeling a whole dataset is entirely assigned to humans, which is costly.

Semi-supervised learning (SSL) appeared in this background. The goal of SSL is to train DNN model with a dataset where only small portion of data is labeld. Since there are too many unlabeled data in a dataset, SSL requires special algorithm that differs from supervised learning to deal with both labeled and unlabeled data, *e.g.* MixMatch [1].

In this paper, FixMatch [2] is used with the fashion dataset provided by NAVER. The purpose was to develop this existing algorithm to fit our objective dataset more properly. Though, what I found in real is that FixMatch does not work so well.

2. FixMatch

The idea of FixMatch is simple. It just combined *consistency regularization* and *pseudo-labeling*, which are commonly used approaches in SSL. For unlabled data, FixMatch model first predicts a label for weakly-augmented version of an unlabled image, which becomes pseudo label of that image. Then, model predicts a label for strongly-augmented version of the image this time, and calculated cross-entropy loss with the prediction and pseduo label.

2.1. Loss Function

FixMatch optimizes the following loss function:

$$\mathcal{L} = \mathcal{L}_s + \lambda_u \mathcal{L}_u \tag{1}$$

$\mathcal{L}_s$ and $\mathcal{L}_u$ denotes supervised loss and unsupervised loss each. λ_u, a fixed scalar hyperparameter, is the weight of the unsupervised loss.

$\mathcal{L}_s$ and $\mathcal{L}_u$ are based on cross-entropy loss as follows:

$$\mathcal{L}_s = \frac{1}{B}\sum_{b=1}^{B} H(p_b, p_m(y \mid \alpha(x_b))) \tag{2}$$

$$\mathcal{L}_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1} max(q_b \geq \tau) H(\hat{q_b}, p_m(y \mid \mathcal{A}(u_b))) \tag{3}$$

$q_b = p_m(y \mid \alpha(u_b))$ is the prediction of FixMatch model for weakly-augmented version of unlabeld data, and $\hat{q_b} = argmax(q_b)$ is pseudo label. If the model's prediciton is not reliable, pseudo label should not be used. Therefore, a fixed scalar hyperparameter τ is used as a threshold.

2.2. Augmentation

FixMatch uses two versions of augmentation: *weak* and *strong*. I used baseline code transformations for weak augmentation, while using RandAugment[3] for strong augmentation.

I adopted this code as a baseline: `https://github.com/kekmodel/FixMatch-pytorch/blob/10db592088432a0d18f95d74a2e3f6a2dbc25518/dataset/randaugment.py`. For strong augmentation, 2 out of 14 transformations are randomly chosen, and after they are performed cutout is done as a default. Example of strong augmentation is shown in Figure 1.

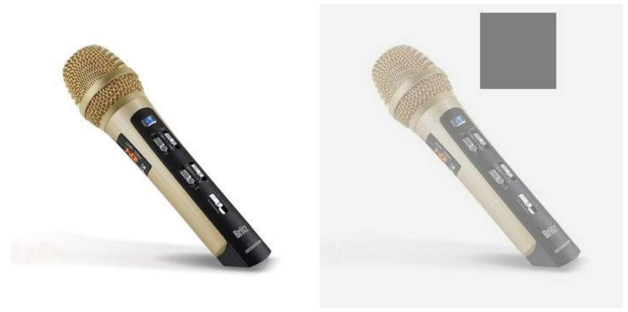

Figure 1. Original unlabeld image (Left), strongly-augmented version of the image (Right)

3. Experiments

3.1. Fashion Dataset

For training a model, fashion dataset is provided by NAVER, which has 265 classes of fashion item images. Among 60K data, only 1325 (about 2 percent of total) images are labeled, so SSL is required for image classification. After training with this dataset, test is done with 200K dataset from NAVER shopping website.

3.2. MixMatch vs. FixMatch

After implementing a rough FixMatch model, I compared the results of MixMatch and FixMatch. I expected FixMatch to give at least slightly better score than baseline MixMatch, but it did not.

Method	Top 1 Val. Acc.	Test score
MixMatch	8.85	0.177
FixMatch	13.2	0.151

Table 1. FixMatch showed better accuracy in validation, but test score was lower.

I simply used initial settings: hyperparameter values $\lambda = 10, B = 200, \mu = 1, \tau = 0.7, lr = 1e-4$, using Adam optimizer, and no EMA.

3.3. Optimizer & Scheduler

This time, I re-implemeted the model to strictly follow what FixMatch paper did. SGD optimizer is used instead, since the paper mentioned that Adam was worse than SGD. Also, I used EMA, multistep scheduler, and linear learning rate warm up. I fixed other hyperparameters and varied only threshold value.

The models showed significantly low train accuracy (about 50% top 1 accuracy) and very noisy validation accuracy graph during their training. Therefore, it is not surprising that the results were worse than before.

τ	Top 1 Val. Acc.	Test score
0.95	10.6	0.092
0.85	10.2	-
0.70	10.4	-

Table 2. $\lambda = 1, B = 64, \mu = 3, lr = 0.4$ fixed.

I also tried cosine annealing scheduler with several hyperparameter options, and some of the results are shown below.

B	μ	lr	Top 1 Val. Acc.	Test score
64	3	1e-4	1.37	-
64	3	0.03	28.5	0.140
100	1	0.03	10.4	-

Table 3. $\lambda = 1, \tau = 0.95$ fixed.

Although the second model showed remarkable validation result compare to previous implementations, the test score was worse.

3.4. Noise Elimination

Almost every FixMatch models' accuracy graphs appeared quite noisy as shown in Figure 2. This noise should occur because of learning unlabeled data.

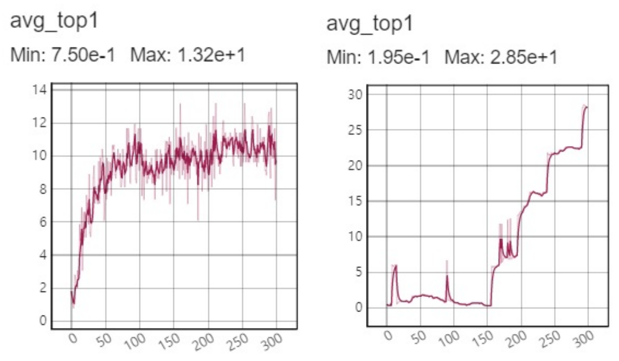

Figure 2. Graphs appear quite noisy.

Since the model is trained with labeled data at the beginning because of threshold τ, my idea was to testing labeled data in unlabeled way, too. I thought this might be helpful to balance the weight of supervised learning and consistnecy regularization&pseudo-labeling in FixMatch.

I adapted this idea to the models in Section 3.3 for comparison. Noise is eliminated evidently when $\lambda = 1, \mu = 3, \tau = 0.95, lr = 0.4$. See Figure 3. Though, there was no significant improvements in performances.

I left this implementation and tried for further models, but there still exist some noises.

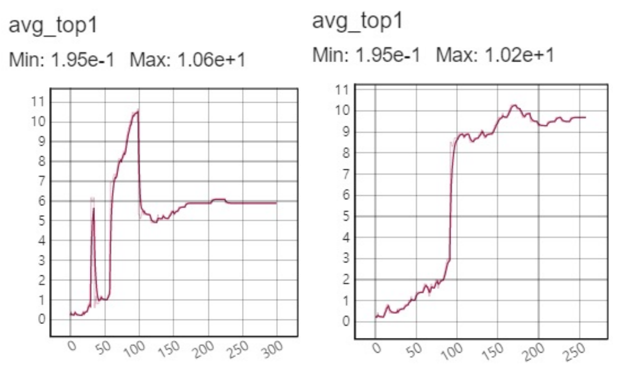

Figure 3. Noise is eliminated, but performance is similar.

3.5. Additional Approaches

After the final presentation, I could get two ideas from other teams. One is about strong augmentation. If strong augmentation is *too strong*, the model may learn nothing from unlabeled dataset or get overfitted. Especially, some transformations that changes the color of unlabeled image might be *too strong*. Thus, I tried some variants of the augmentation code to make strong augmentation weaker or stronger, but the effect was not recognizable in model's performance.

Another idea is about threshold. A model learns and becomes reliable as epoch increases, and this means that pseudo label might be also getting credible. Therefore, it is acceptable to decay the threshold and reflect unlabeled loss more. Thus, I implemented threshold scheduling so that the threshold exponentially decays with the rate of 0.99.

I couldn't use other neural net structure than Res18 because the NSML server was always saturated. However, after the final presentation, the NSML server had enough space so I tried Res34 for additional approaches.

Method	Top 1 Val. Acc.	Test score
Original	17.2	0.141
Weaker augmentation	33.2	0.129
Threshold scheduling	35.4	0.154

Table 4. $\lambda = 1, \tau = 0.95$ fixed.

It is disappointing that even the models showed the highest accuracies among overall experiments I performed, their test score failed to beat MixMatch.

4. Conclusion

In this project, I revisited FixMatch to solve SSL problem. Although I tried several implementations and experiments to find the best state for the FixMatch model, it never outperformed the baseline MixMatch model.

Since FixMatch showed very doubtful performances for all the teams that used it, there should be further studies to check whether FIxMatch is practical or not.

References

[1] Berthelot, David, et al. "Mixmatch: A holistic approach to semi-supervised learning." Advances in Neural Information Processing Systems. 2019.

[2] Sohn, Kihyuk, et al. "Fixmatch: Simplifying semi-supervised learning with consistency and confidence." arXiv preprint arXiv:2001.07685 (2020).

[3] Cubuk, Ekin D., et al. "Randaugment: Practical automated data augmentation with a reduced search space." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.