# Computer Vision Project Report: Semi-supervised Image Classification

Daniel Porawski 2580636
MS Teams: s8dapora@uni-saarland.de

Michel Scherer 2578585
MS Teams: s8mlscer@teams.uni-saarland.de

March 11, 2022

**Abstract**

In this report we want to show the challenges and solutions we found, whilst implementing pseudo-labeling (Task1) and virtual adversarial training (Task2) as well as their effect on training image classification models for the cifar10/100 datasets. Lastly we will present our own idea on how to improve upon FixMatch (Task3).

## 1 Task 1 - Pseudo-labeling

### 1.1 Implementation

The core idea of pseudo-labeling is taken from Dong-Hyun Lee (2013) [1], who describes important factors to take into account for pseudo-labeling. These include a rising influence of pseudo label loss and a normalization of the labeled and pseudo-labeled loss by their respective set sizes. As loss-function we choose cross entropy from $torch.nn.CrossEntropyLoss()$ The optimizier is from $torch.optim.SGD()$ First, we split up a part of the unlabeled set to create a validation set, which we used to find hyper parameters and determine early stopping to save computational time. The rest of the unlabeled set became the new unlabeled set. Then, our implementation trains the model on only the labeled set for half the epochs and then increases the influence, alpha, linear from 0 to 3 until 80% of the epochs, after that it will be constant 3. As mentioned after 50% of the epochs the model will created a new pseudo label set each epoch by making predictions on the whole unlabeled set. Since the labeled and pseudo-labeled set are of different sizes now, we iterate once over the larger of the two sets, while the other will be repeated. We observed that when pseudo-labels are used, the number of batches in pseudo-label set $n_p$ is almost always larger than the number of batches in the label set $n_l$. For each iteration $\hat{y}_l$ the logits from our model for a labeled image batch, $y_l$ the true labels, $\hat{y}_p$ logits of prediction for the

Table 1: Task 1: error rates on test set

| | CIFAR-10 | | CIFAR-100 | |
|---|---|---|---|---|
| threshold | 250 labels | 4000 labels | 2500 labels | 10000 labels |
| baseline | 63.74 | 23.74 | 80.21 | 49.38 |
| 0.95 | 66.06 | 21.18 | 82.13 | 56.34 |
| 0.75 | 68.50 | 20.88 | 78.83 | 53.05 |
| 0.6 | 67.02 | 20.82 | 77.10 | 52.68 |

pseudo-labels and $y_p$ the pseudo-labels are combined to the loss with reference to the combined loss by Dong-Hyun Lee (2013) [1]:

$$L = L(\hat{y}_l, y_l)/n_l + \alpha(epoch) \cdot L(\hat{y}_p, y_p)/n_p$$

For each epoch we determine accuracy on test set, test and evaluation error and make a deepcopy respectively in case the current model has a better accuracy or validation error than in all previous epochs. We used a learning rate scheduler that exponentially shrinks the learning rate each epoch by the factor $\gamma$, we came across that a good value seemed to be $\gamma = .99$

### 1.2 Results

Error rate for a baseline model, that does not use pseudo-labels, and different thresholds are reported in table 1. We observed the following:

- Test error will still decrease while test loss increases.
- Pseudo-labels usage decreases validation and test error
- Pseudo-labels usage increases accuracy slightly, when the model already has a good accuracy
- Pseudo-labels usage decreases accuracy significantly, when the model is not pre-trained well enough
- Training process with pseudo-labels is more robust

## 2 Task 2 - Virtual Adversarial Training

### 2.1 Implementation

Virtual Adversarial Training is a method for semi-supervised learning first proposed by T. Miyato et al. (2018) [3]. The core idea is adding random noise to unlabeled data to find the noise vector which maximizes the KL-divergence between the network's predictions on unlabeled data with and without the noise. This aims to further stabilize the networks predictions with regards to small

irregularities in the images. In our implementation we chose to combine the batches of labeled and unlabeled images and use this batch to compute the VAT Loss. Our reasoning is that this should only further aid the stabilizing effect of the loss. After computing this VA loss, we use Cross Entropy Loss on the labeled images' predictions and their true labels. The VA loss is then scaled by the hyperparameter $\lambda$, which determines the impact of the VA loss on training, and eventually added to the Cross Entropy loss. Therefore the final Loss is:

$$L = CE(\hat{y_l}, y_l) + \lambda KL(\hat{y}, \hat{z})$$

where $\hat{y}$ is the prediction on the labeled and unlabeled images and $\hat{z}$ is the prediction on the labeled and unlabeled images combined with the noise.

Using this new loss, we backpropagate through the network and adapt the weights accordingly. We chose Adam as our optimizer, as it outperformed SGD and also used the same learning rate scheduler as in Task 1. We also implemented early stopping after 50 epochs, with the stopping criterion being the accuracy of our model on the test set. We chose the accuracy rather than the loss as it continued to improve while the loss very slowly started to increase. However, we assume that the large gain in accuracy outweighs the small increase in loss.

## 2.2 Results

We first computed the model's base performance without VAT. Surprisingly we found that without VAT, SGD outperformed Adam as an optimizer. After finding the best hyperparameters for our model on Cifar-10 with 4,000 labels, we used the same on Cifar-10 with 250 labels, as well as on Cifar-100 with 10,000 and 2,500 labels. These hyperparameters are:

- Learning Rate: 0.005
- Weight decay: 0.01
- Momentum: 0.5
- Learning Rate decay (Scheduler): 0.98

To achieve comparable results we also searched for the best hyperparameters when using Adam, the results are:

- Learning Rate: 0.003
- Weight decay: 0.0005
- Learning Rate decay (Scheduler): 0.98

Using these hyperparameters we did an additional search for the best value of $\epsilon$ in VAT, while keeping $\alpha = 1$ and $\xi = 10$ fixed, as recommended in the original VAT paper. Doing this we found the best achieving $\epsilon$ are:

- Cifar-10 / 4,000 labels: 8.0

Table 2: Task 2: error rates on test set

| | CIFAR-10 | | CIFAR-100 | |
|---|---|---|---|---|
| | 250 labels | 4000 labels | 2500 labels | 10000 labels |
| Base | 74.96 | 27.68 | 79.36 | 53.78 |
| VAT | 61.16 | 23.55 | 76.66 | 49.69 |

- Cifar-10 / 250 labels: 8.0
- Cifar-100 / 10,000 labels: 8.0 / 30.0
- Cifar-100 / 2,500 labels: 8.0

Finally the resulting error rates can be found in Table 2. We see that in all cases VAT slightly improved the error rates. Additionally to the improvement in accuracy, we also noticed that the losses when compared to the base model also always improved. The losses on the training set decreased faster than those of the base model, but to the same value. The losses on the test data on the other hand not only converged faster, but were also smaller than those of the base model. We give some examples of the images produced by the noise in the appendix. The noise in each image is not recognizable by humans, although $\epsilon$ was set to 8. Given that in the original paper the noise was clearly visible with such high values for $\epsilon$ and that our method did influence the accuracy, we assume our method for visualization did not work as intended. Judging by these observations we find that implementing VAT training when having just a small amount of labeled data with a large amount of unlabeled data seems to generally improve test accuracy and test loss.

# 3 Task 3 - Challenge Task

## 3.1 Implementation

Using the FixMatch algorithm as proposed in [2] as foundation of this SSL task, we added an additional loss that penalizes the change in softmax logits for an unlabeled image that is weakly augmented and then strongly augmented. The change in logits is measured with MSE:

$$\text{our\_loss} = \frac{1}{N}\sum_{c}(\hat{y}_{\text{weak, c}} - \hat{y}_{\text{strong, c}})^2$$

With $N$ the number of classes, $\hat{y}_{\text{weak, c}}$ the softmax logit the model predicts for the weakly augmented image for class $c$ and $\hat{y}_{\text{strong, c}}$ for the strongly augmented image. Weakly augmentation is only doing a horizontal flip with $p = .5$ while strongly augmentation additionally uses vertical flip with $p = .5$, color jittering and distortion from torchvision.transforms. Our main idea was to boost the

Table 3: Task 3: error rates on test set

| | CIFAR-10 | | CIFAR-100 | |
|---|---|---|---|---|
| model | 250 labels | 4000 labels | 2500 labels | 10000 labels |
| FM only | 65.01 | 31.55 | 77.91 | 58.94 |
| $\mu = 1$, $\omega = 0.1$ | 63.56 | 27.54 | 80.28 | 59.86 |
| $\mu = 1$, $\omega = 0.6$ | 66.6 | 27.97 | 81.19 | 63.04 |
| $\mu = 1$, $\omega = 1$ | 64.91 | 31.55 | 81.28 | 63.27 |

models confidence for images that, at least from a logit perspective, look similar to the model. This way the unlabeled data could help to increase the amount of information the training can obtain from the labeled data. We used the wide-res-net from the previous tasks in order to transfer the gained knowledge about hyper-parameters and be able to compare the results. Since FixMatch makes use of pseudo labels, like in Task 1, on epoch equal to iterating over the larger set, pseudo label or labeled data, while the smaller set will be reused once all samples were used. The learning rate scheduler and early stopping from previous tasks were used again. The final loss is described by:

$$\text{loss} = (\omega \cdot \text{our\_loss} + \mu \cdot \text{FM\_loss} + \text{Class\_loss})/(1 + \mu + \omega)$$

with FM_loss the FixMatch loss and Class_loss being the supervised classification error on a labeled batch. $\mu$ and $\omega$ are hyper-parameter, that respectively determine the influence of the FixMatch loss and our_loss. With regards to the original paper we kept $\mu = 1$ fixed in order to find the influence of our loss to the model's accuracy.

## 3.2 Results

Our initial idea was more sophisticated: For each unlabeled image that our model could predict a pseudo label with confidence above .95 a so called partner-image, with the most similar logits, had to be found in the labeled images. Then the unlabeled image and its partner would both be strongly augmented and the resulting difference in the changing logits for these augmented versions would be the loss. Unfortunately this resulted in a huge computational effort and unfeasible training times. Therefore we backed down to the simpler variant we presented. As shown in Table 3 and the appending training graphs we could achieve some improvement for Cifar10 4k labels scenario but limited to that only. We assume that our method needs a solid accuracy in order to work and under-performs when the model lacks training data to make good confident predictions. One thing to mention is that our method seems to indeed stabilize the test loss (Figure 11) in contrast to what we observered in task 1. While the test loss for the baseline increases after some time in training, the test loss of our method keeps declining.

# 4 Conclusion

Frankly, we can not be satisfied with the resulting accuracy of any task compared to state of the art methods. Concerning task 2, the original paper achieves a test error rate of 11.36 on Cifar-10 with 4,000 labels, which is halve the error rate our implementation achieved. We cannot explain this difference solely on implementational details, which brings us to the conclusion that further optimization would be needed to reproduce the paper's results. Similarly the FixMatch paper claims error rates of below 6 on Cifar-10 with 250 and 4,000 labels, while our implementation performs far worse. The difference is similar on Cifar-100. Therefore we again assume that not only our implementation differs vastly from the original one, but also further optimization is needed to replicate these results. The differences between our implementations and the original ones becomes even clearer when using the hyper-parameters found by the respective authors, we could not achieve an accuracy better than educated guessing with those. Though finding an tuning hyper-parameter on our own was time consuming, we gained a lot of practical insight on how they influence the training process. Surprisingly, the Adam optimizer performed worse than SGD with momentum in all our experiments of Task 1. It also performed worse when ignoring VAT training in Task 2, however with VAT Adam was superior to SGD. To us it seemed that when using weight decay the effect of dropout gets diminished. There was no difference to observe between dropout 0.2 and 0.5. Additionally we can conclude that this project helped us to learn much about working as a team, planing tasks, understand neural network experimenting and time management.

# References


[1] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on challenges in representation learning*, 3(2):896, 2013.

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

[3] Takeru Miyato et al. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41.8:1979–1993, 2018.


# Appendix

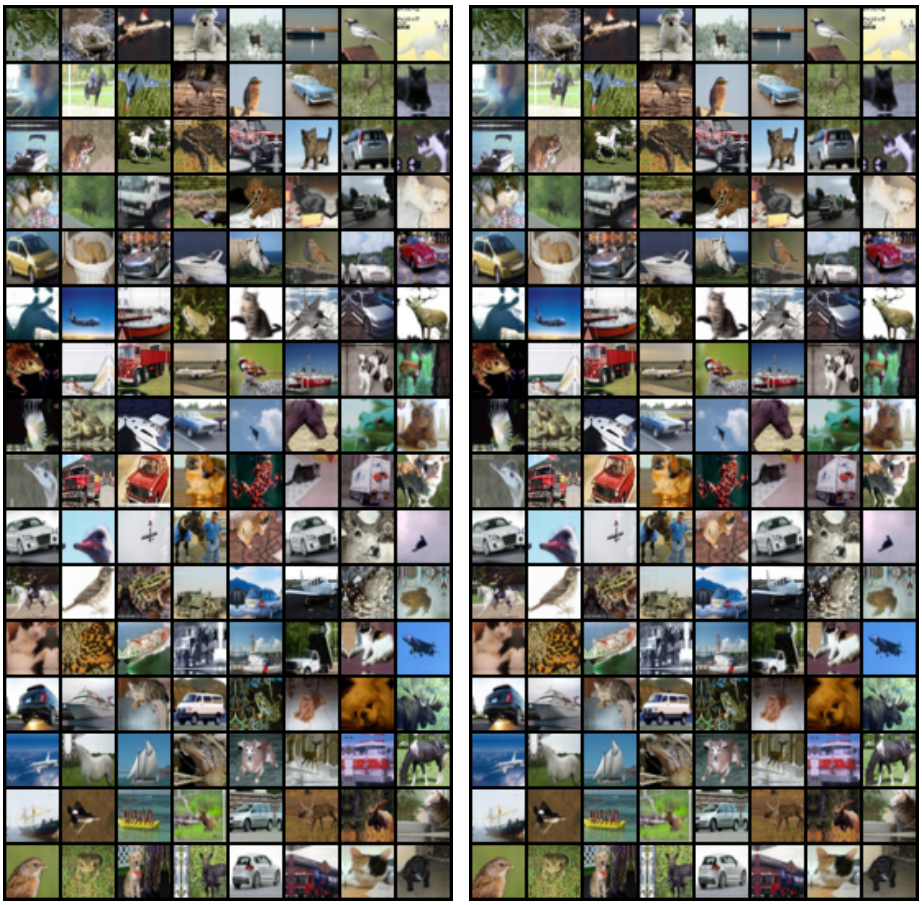

Figure 1: Image comparison on Cifar-10 with $\epsilon = 30$. Left: Images without noise. Right: Images with noise

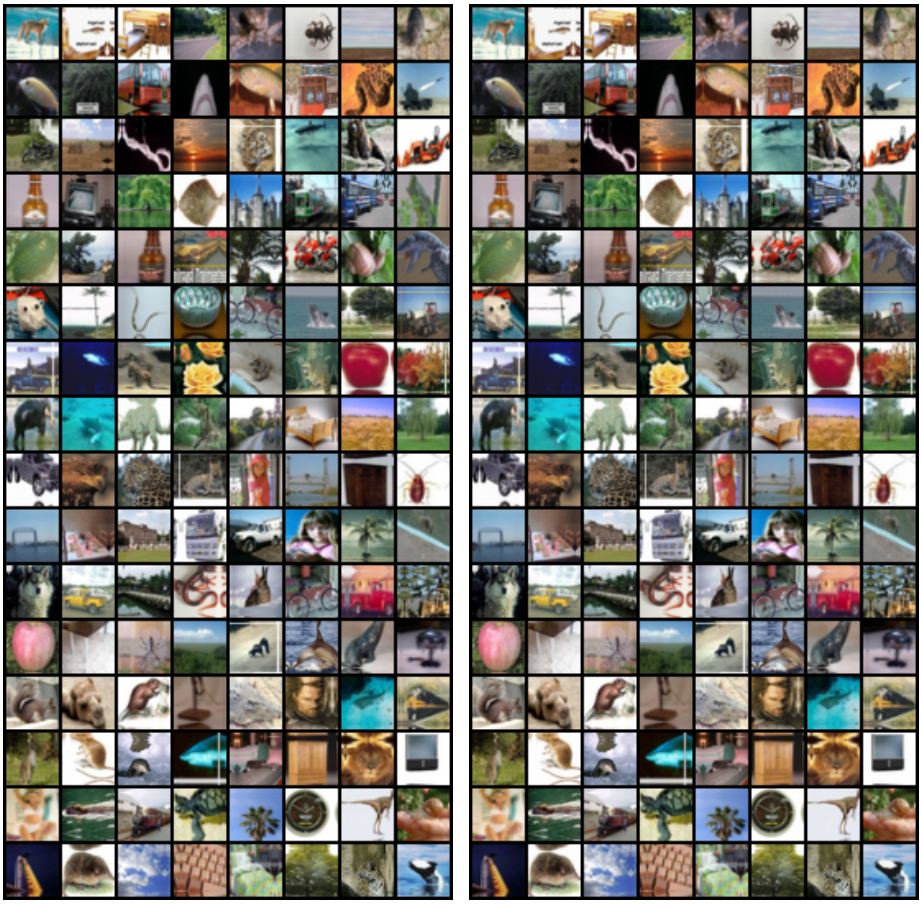

Figure 2: Image comparison on Cifar-100 with $\epsilon = 30$. Left: Images without noise. Right: Images with noise

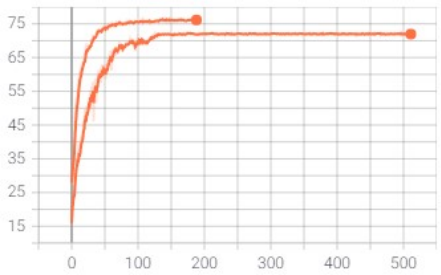


Figure 3: Task 2: Test Accuracy for Cifar-10 with 4,000 labels. Upper Orange: With VAT. Lower Orange: Base Model

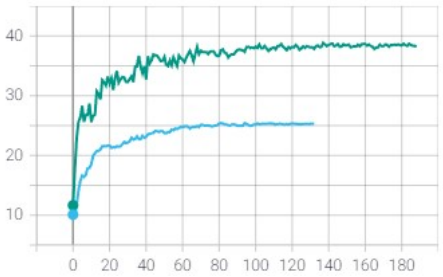


Figure 4: Task 2: Test Accuracy for Cifar-10 with 250 labels. Green: With VAT. Cyan: Base Model

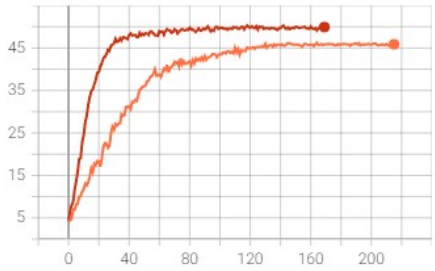


Figure 5: Task 2: Test Accuracy for Cifar-100 with 10,000 labels. Red: With VAT. Orange: Base Model

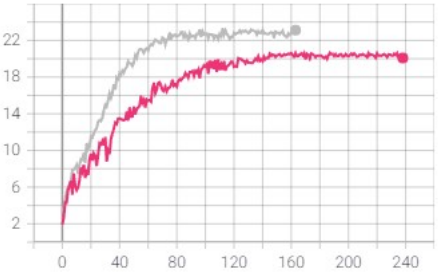


Figure 6: Task 2: Test Accuracy for Cifar-100 with 2,500 labels. Grey: With VAT. Pink: Base Model

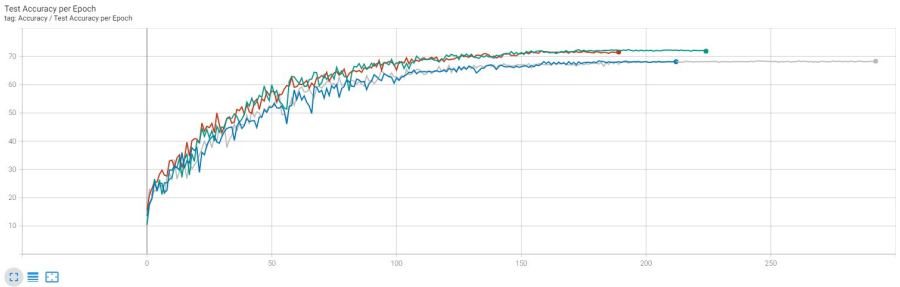


Figure 7: Task 3: Test Accuracy for Cifar-10 with 4,000 labels. Blue: Fixmatch Only. Green: Omega=0.1, Red: Omega=0.6, Grey: Omega=1

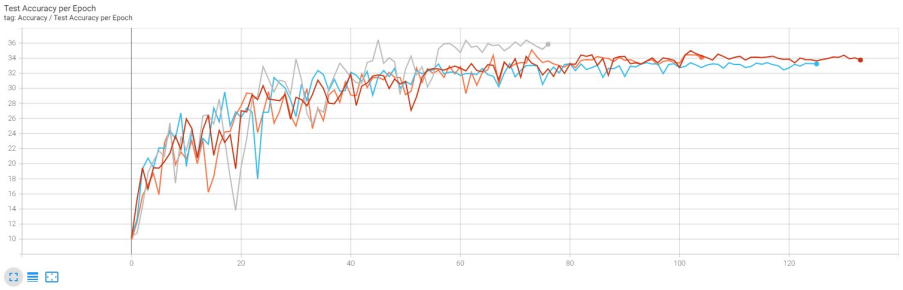


Figure 8: Task 3: Test Accuracy for Cifar-10 with 250 labels. Red: Fixmatch Only. Gray: Omega=0.1, Cyan: Omega=0.6, Orange: Omega=1

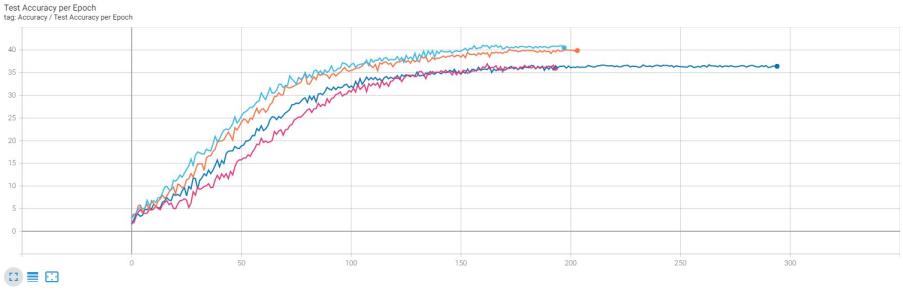


Figure 9: Task 3: Test Accuracy for Cifar-100 with 10,000 labels. Cyan: Fixmatch Only. Orange: Omega=0.1, Pink: Omega=0.6, Blue: Omega=1

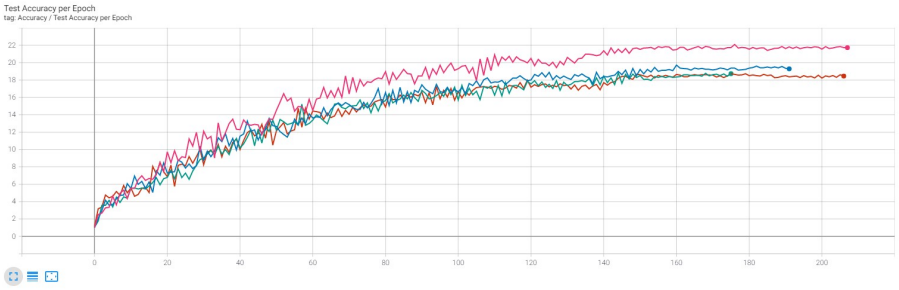


Figure 10: Task 3: Test Accuracy for Cifar-100 with 2,500 labels. Pink: Fixmatch Only. Blue: Omega=0.1, Green: Omega=0.6, Red: Omega=1

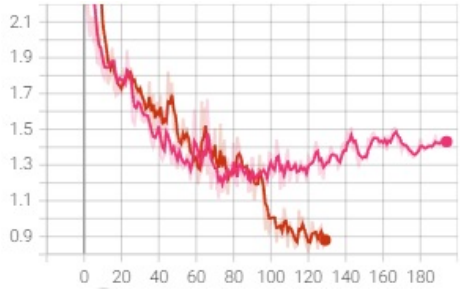


Figure 11: Task 3: Test Loss for Cifar-100 with 2,500 labels. Pink: Basemodel, Red: Omega=1, Mu=1