Final Project: Semi-supervised Learning Methods

Adam Mann
Department of Computer Science
Columbia University
aim21272@columbia.edu

Bracha Blau
Department of Statistics
Columbia University
bsb2144@columbia.edu

Abstract

This project seeks to explore a body of research concerned with noisy semi-supervised approaches to training convolutional neural networks. This project is focused on experimenting with different settings on the classic student-teacher method described in Google's noisy student paper [3]. The three areas where we experiment with different settings are with the type of noise used, the number of student-teacher iterations, and the cutoff threshold for assigning pseudo-labels to unclassified data. All of our methods are tested on the The Street View House Numbers (SVHN) dataset. Our goal is to discover which set of the hyperparameters described above leads to the most accurate model. The repository containing our code and data can be found at the following link: https://github.com/bblau100/Deep-learning-final-project.

1 Introduction

Lately, state-of-the-art benchmarks on popular datasets have been achieved by focusing more on training methods than on model architectures. Many of these methods make use of preexisting architectures such as Google's EfficientNet or Facebook's ResNet and differ in how they go about feeding data into the model to train them. These methods often take a Semi-Supervised student-teacher approach. In this approach a Teacher model is trained on a labeled dataset and is then used to make predictions on an unlabeled dataset, creating pseudo-labels. These pseudo-labels are then used to augment the labeled data and train a new student model. This process can then be repeated iteratively multiple times by treating the former student as the new teacher.

Previous research on the use of these models tends to differ in many of the particularities of the approach, while maintaining the underlying student-teacher structure. This project is concerned with applying the methodologies in the literature to the SVNH dataset. This project hopes to add to the current literature by experimenting with a few different hyperparameters in the student-teacher setup. These hyperparameters are the type of noise introduced into the student model, the number of student-teacher iterations to run, and the threshold for assigning labels to the unlabeled test data. The types of noise we consider are RandAugment as described in [5], the more traditional random affine transformations, and no transformations. The thresholds we consider are the 97, 98, and 99th quantiles of the certainty of the predictions for each class of digits. We do not use a cutoff corresponding directly to loss of the model, because depending on the loss function the results may not be interpratable. Therefore, it is more stable to use thresholds based on the quantities of certainty. For the iterations, we run three complete cycles and track the performance after each one is complete. The metric that we use to calculate the performance of each model version is the model's accuracy. Throughout this project we use a ResNext model as our initial teacher model.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Data

SVNH: The SVNH dataset is a collection of images of house numbers collected from Google maps. There are two flavors of the available data. One with the individual digits separated and one without. For this project we are using the former. It consists of a training set of 73,257 images, a test set of 26,032 images and an extra set of 531,131 images. For this project we use one fourth of the training data as the initial "labeled" data, and the remaining three quarters as the "unlabeled data" after we throw away their labels. We use the test data for validation. We do no use the extra 531,131 images in this project, since they are not as difficult to classify as the original data.

3 Review of Related Work

Yalniz et al. [4] in conjunction with Facebook AI explore a simple student-teacher approach for semi-supervised learning. The key of their research is making use of an unlabeled dataset that grossly outnumbers the labeled data 10:1. With this data they take their off-the-shelf ResNet and ResNeXt architectures and train them using four steps. 1) Train the teacher model on the labeled data. 2) Create pseudo labels for the unlabeled data. 3) Train a student model on the pseudo-labeled data. 4) Fine-tune the student model on the labeled data. On ImageNet they scored a top-1 accuracy of 84.8

Xie et al. [3] take a different methodology and make three main changes to Yalniz et al.'s approach: 1) They repeat to teacher-student process multiple times iteratively gradually using larger and larger models. 2) They train the student on both the labeled and unlabeled data. 3) They add noise to the data (RandAugment) and to the model (dropout and stochastic depth). Xie et al. found that the last adjustment was the most significant in obtaining their top-1 accuracy score of 88.4% on ImageNet. They also experimented with two types of pseudo labeling (soft and hard) and found that when it came to high confidence labels both methods worked well, however, on lower confidence labels hard pseudo labels hurt performance.

Chen et al. proposed another framework using contrastive learning. Instead of using external unlabeled data they augment their current labeled data creating a normal and augmented set. These a neural network is then trained on the data using the following loss function

$$l_{i,j} = -log\frac{exp(sim(z,i,z_j)/\tau)}{\sum_{k-1}^{2N} \mathbb{1}_{[k\neq i]} exp(sim(z,i,z_j)/\tau)}.$$

The basic idea is that the network attempts to match the augmented (un-augmented) image with its un-augmented (augmented) counterpart and is penalized when it attempts to match it with any other image. The found that there were two factors that helped them achieve a top-1 accuracy of 85.8% on ImageNet: 1) A careful approach to choosing data augmentation (they found that RandAugment did not correctly match the task). 2) A large batch size as that maximized the number of contrasting images.

4 Methodology

Our approach to the model will closely follow that used by Xie et al. as their methodology is the forerunner on ImageNet accuracy. However, as the type, variety, and size of our data differs from theirs, we will take certain parts of departure with their approach to see if other another methodology might better fit out tasks. In particular, there are three main points of departure we will consider:

1 There is the question of how to pass pseudo labels into the models. Xie et al. [3] experimented with both soft and hard pseudo labels. It is unclear how exactly they went about this process, however, in their results they do mention that for in-sample unlabeled data they found hard pseudo labels to outperform soft pseudo labels. Therefore, we suggest the following approaches: Experimenting with hard pseudo labels following Chen and Yang et al. [2] this means limit labels to those within a certain threshold of certainty. The thresholds we consider are the following quantities of certainty (for each digit):

   i. 97
   ii. 98

iii. 99

These thresholds also ensure that the ratio of our pseudo-labeled and labeled data is similar those that used by Xie et al.

2 In conjunction with Chen et al. [1] who found that the RandAugment mused by Xie et al.[3] was not always the most effective, we will attempt to formulate a pipeline of augmentations to maximize the effectiveness of noise within our model. We therefore experiment with different types of noise:

   i. RandAugment
   ii. RandAffine
   iii. No transformations

3 How many iterations of the teacher-student process is required before overfitting becomes a point of concern. We experiment with up to three complete cycles.

Additionally, it is important to mention that our base model will have two unique facets when compared that that use by Xie et al.

1 While Xie et al. [3] used Google's EfficientNet architecture we are curious how Facebook's ResNext (50_32x4d) would perform on our task given the models' similar base level performances.

2 We consider the task using a much smaller dataset than Google's noisy student model. In fact, we only use a quarter of the training set to train our base model (about 18,400 samples) and reserve the rest for pseudo-labeling purposes for the student-teacher iterations. This follows as we found that the total training data available was not sufficient to train ResNext from scratch yet too much to use a pre-trained ResNext and achieve meaningful results. Thus we cut down on the amount of 'labeled' data to allow provide our student-teacher model with a more feasible learning curve.

5 Architecture

The architecture of our code is broken down into three main parts. The first function, train_base_model(), trains the initial teacher model on approximately 18,400 samples using a ResNext pre-trained on Imagenet. The second function, student_training(), adds the noise, sets the new labels, and trains the student model. Separately, we loop through each configuration of the noise type and threshold we want to use and train three full cycles for each pair. The first two methods are in a file called SVHN Model, and the second part is in another notebook entitled Run Models. It is important to note that only pseudo-labeled data is noised with the idea that that should minimize the effects of incorrect labels.

6 Results

In this section we report the results of the different model configurations. We analyze which combinations perform the best on the SVHN data, and we compare the results to our baseline model.In Tables 1, 2, and 3, we show the accuracy for each combination of noise type, threshold, and cycle number. It is important to note that the baseline model achieved a validation accuracy of **.9012**, hence all of the below models outperform the initial baseline.

Table 1: Accuracy with RandAugment

Cycle Number	**Threshold: 0.97**	**Threshold: 0.98**	**Threshold: 0.99**
0	0.9223	0.9198	0.9120
1	0.9211	0.9212	0.9213
2	0.9147	0.9192	0.9201

Table 2: Accuracy with RandAffine

Cycle Number	Threshold: 0.97	Threshold: 0.98	Threshold: 0.99
0	0.9166	0.9226	0.9181
1	0.9177	0.9226	0.9212
2	0.9214	0.9223	0.9196

Table 3: Accuracy with No Transforms

Cycle Number	Threshold: 0.97	Threshold: 0.98	Threshold: 0.99
0	0.9216	0.9247	0.9216
1	0.9199	0.9234	0.9142
2	0.9151	0.9231	0.9224

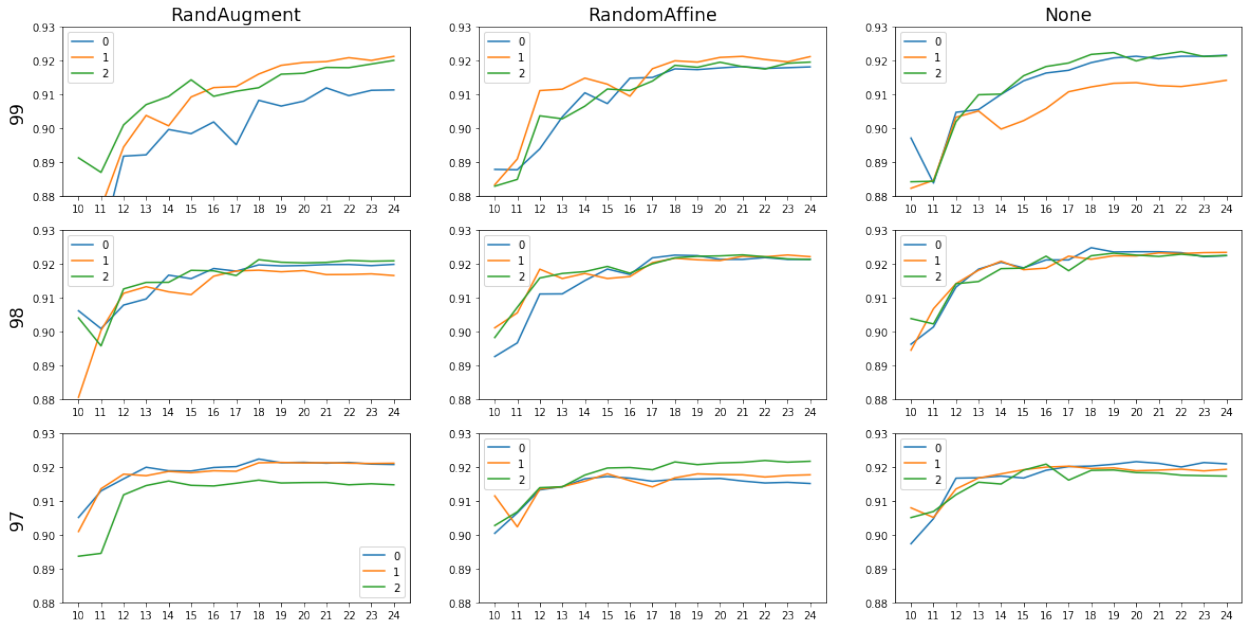

Figure 1: The performance of the different model configurations after at three different iterations.

7 Discussion

7.1 Thresholds

A quick examination of the results shows that the 88th percentile threshold overall outperforms the others while holding all else constant. It is important to note that this means that the model was using almost an equivalent amount of label and pseud-labeled data for training. It is interesting to note that under these conditions the number of iterations plays little role. Additionally, in reference to the Xie et al. while they found little benefit from using more data than an equal ration we find (as in the case of using the 97th percent) that performance begins to regress. Hence there appears to be a sweet spot for the ratio of pseudo-labeled samples.

7.2 Noises

In terms of noise we find that random affine and no noise both tend to produce better results than RandAugment. This is partly unsurprising as we saw in Chen et al. that RandAugment is not necessarily suited to every task. In fact we chose random affine as we believe it would be better suited here. What is surprising is the overall strong results of using no noise. In fact, this is perhaps the most interesting finding of this paper. The idea that using no noise could outperform models that use noise for in-sample data could suggest that there is a fundamental misunderstanding about what is happening under the hood in semi-supervised learning. It was previously believed that noise would help by minimizing the impact of incorrectly labeled data. But in truth it is possible that by limiting the positive impact of correctly labeled data, it is detrimental to overall model accuracy.

7.3 Iterations

In our frame work it is clear that while holding all else constant, multiple iterations provides little advantage to overall performance. We believe that this is chiefly due to the fact that we were not upgrading the complexity of the model between each iteration as Xie et al. Hence in our case we do little more than retraining the model on the same data with only small variation in the pseudo-labeled data.

8 Implementation

As described in section 5, the two methods for training the data are train base model and student training. In this section, we will discuss both functions in detail.

8.1 Train base model

This is the first stage where we perform the initial training on the labeled data set. For this model, we do not add any noise. In the first cycle, we start with downloading the SVHN data from the library of torchvision datasets. We only need to do this once, and afterwards the data is stored locally. Then we perform standard transformations on the data to normalize it. We initialize our model as a pretrained ResNext with 50 layers from the torchvision library, and add an extra layer for transfer learning. Next we define the training criteria to use stochastic gradient descent on the cross entropy loss function. From experimentation, we found that a starting learning rate of .05, a step size of 6, and a learning decay rate of 0.5 worked well with `torch.optim.lr_scheduler.StepLR`. We then train the model for 20 epochs recording the train loss, train accuracy, validation loss, validation accuracy, and learning rate. Once the the training is complete, we save the model, and return the model's validation accuracy as a metric of its performance.

8.2 Student training

The student training method takes in several parameters to determine how to set up the model. The two most important ones are the kind of noise to use, and the threshold for assigning hard pseudo-labels. First, the function loads the teacher model that was trained in stage one. Then we use that model to predict the classifications for all of the unlabeled data. Once we have all of these predictions, we perform calculations to determine which output score from the model is high enough to pass our given percentile threshold. Next, we apply the noise transformations to that data. We are now ready to train the student model. As in the previous model, we use stochastic gradient descent with the cross entropy loss function. We run this model for 25 epochs, with the same learning rate parameters. After each epoch is complete, we output the same metrics as we did for the teacher model after every iteration. After the training is finished, we return the validation accuracy.

8.3 Running the models

In the third and final stage of the code, we iterate through all of the model configurations that we want to test out. We have one array for the thresholds and one array for the noise types. We use a double for each loop to get all of the configurations. Nested inside of them we have another for loop to perform 4 complete iterations of using the teacher and student models.

9 Conclusion

After a series of ablation studies involving noisy student-teacher semi-supervised training on ResNext (50_32x4d) we have seen a number of interesting outcomes summarized here. Primarily, without expanding the complexity of the model in each iteration we did not see any improvement through iterations. We have found that using a certainty of 99% on unlabeled samples was ideal for accuracy boosting. Finally, we have seen that not only does RandAugment underperform in comparison to other transformations, we have seen that there is reason to believe that applying no noise whatsoever to pseudo-labeled data might be the best way to go, which as an unusual finding considering patterns in the field.

10 References

[1] Chen, T., Kornblith, S., Norouzi, M., Hinton, G. (2020). A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709.

[2] Chen, Y., Yang, Y., Zhang, M., Kuo, C. C. J. (2019, September). Semi-supervised learning via feedforward-designed convolutional neural networks. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 365-369). IEEE.

[3] Xie, Q., Hovy, E., Luong, M. T., Le, Q. V. (2019). Self-training with Noisy Student improves ImageNet classification. arXiv preprint arXiv:1911.04252.

[4] Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. arXiv preprint arXiv:1905.00546.

[5] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le (2019). RandAugment: Practical automated data augmentation with a reduced search space. arXiv preprint arXiv:1909.13719.