NNTI Project: Semi-supervised Image Classification

Vikram Singh
Department of Computer Science
Saarland University
Saarbrücken, Saarland, 66123
visi00003@stud.uni-saarland.de

Zeba Karin Ahmad
Department of Computer Science
Saarland University
Saarbrücken, Saarland, 66123
zeah00001@stud.uni-saarland.de

Abstract

In this project, we address the computer vision task of image classification using semi-supervised learning (SSL) techniques. In the first and second tasks, we train a WideResNet using the two popular SSL algorithms, namely Pseudo-Labeling and Virtual Adversarial Training (VAT) respectively on CIFAR-10 and CIFAR-100 dataset. For the third task, we try to improve upon the existing SSL technique Pseudo-Labeling by adding a Siamese Network based on a triplet loss on top of it for both CIFAR-10 and CIFAR-100 dataset. The code for our project is available at our GitHub repository: https://github.com/vsingh1998/NNTI_CV_project

1 Introduction

Deep learning has been used in various domains such as computer vision Krizhevsky et al. [2017], natural language processing Kwiatkowski et al. [2019], data mining Zhang et al. [2018], etc. It has achieved great success in these fields, especially in supervised learning tasks where a huge amount of labeled data is available. Such large labeled datasets, however, are difficult to acquire and are not always available. For example, labeled medical datasets require expert supervision to generate accurate labels. This task of generating labels is quite tiresome and time-consuming. In the case of fewer labeled samples, our current supervised learning algorithms fail to generalize and produce incorrect results. By contrast, there is an abundance of unlabeled datasets, which is comparatively easy to acquire than labeled datasets. But the problem is to devise an algorithm that can use these unlabeled samples to improve the model training. This is where semi-supervised learning techniques come into play. The idea is to leverage the availability of unlabeled data in addition to the fewer labeled data to generate accurate predictions. That is why semi-supervised learning has become an active research field nowadays.

Semi-supervised learning was first conceptualized in the 1970s Agrawala [1970], Fralick [1967]. Since then, many traditional SSL methods have been proposed Zhu and Goldberg [2009]. However, with the advent of deep neural networks, SSL has also been adopted and developed for deep learning. A plethora of SSL methods exists, each with its own advantages and characteristics. The most recent survey Yang et al. [2021] characterizes the current SSL methods, based on model design and loss function, into generative methods (such as semi-supervised GANs Sajun and Zualkernan [2022] and VAEs Kingma and Welling [2014]), consistency regularization methods (such as Mean Teacher Tarvainen and Valpola [2017] and VAT Miyato et al. [2019]), graph-based methods (such as AutoEncoder Abbasnejad et al. [2017] and GNN based models Scarselli et al. [2009]), pseudo-labeling methods (such as deep co-training Blum and Mitchell [1998] and pseudo-label Lee [2013]) and hybrid methods (such as MixMatch Berthelot et al. [2019] and FixMatch Sohn et al. [2020]).

For this project, we delve deeper into Pseudo-Labeling Lee [2013], Virtual Adversarial Training (VAT) Miyato et al. [2019], and Siamsese Networks Koch [2015] (for SSL). Pseudo-Labeling is

Submitted to Neural Networks: Theory and Implementation 2021-22.

an intuitive SSL algorithm in which we train the model initially on labeled data and then make predictions on the unlabeled data. If the confidence of a class or the maximum predicted probability on the unlabeled data is greater than a specified threshold value (τ), we assign this (pseudo) label to the unlabeled data and add the unlabeled samples to the pseudo dataset. We then train the model on this combined dataset including pseudo data and labeled data. On the other hand, VAT is a regularization method utilizing a virtual adversarial loss. In contrast to adversarial training, VAT generates an adversarial direction without using labels. Initially, we generate predictions for the unlabeled image and try to find perturbation such that it maximizes the divergence between that distribution and the distribution of the perturbed image. The divergence we used in the VAT loss algorithm is Kullback–Leibler divergence Kullback and Leibler [1951], often referred to as KL divergence. During training, the model tries to generate a perturbation such that the prediction of the image will change. After generating this perturbation, we try to minimize the VAT loss. This loss function is added to our regular loss function to make it work as a regularization factor. For the third task, we utilized the Siamese Network Bromley et al. [1993] based upon the triplet loss function to improve upon the Pseudo-Labeling algorithm in Task I. Siamese Networks are neural networks that are mainly used in case where there are higher number of classes with few labeled samples per class. These networks can be seen as multiple networks sharing same weights with the same model architecture and are used to compute embeddings of the image. The triplet loss function has been widely used in face recognition Schroff et al. [2015] to differentiate between multiple faces. That is why we used Siamese Network with triplet loss to fine-tune our trained model so as to differentiate among classes better by forming clusters of the different classes.

In this project, we have made the following contributions:

- Task I: Implemented Pseudo-Labeling algorithm using 16-8 WideResNet for both CIFAR-10 (4000 and 250 labeled samples) and CIFAR-100 (10000 and 2500 labeled samples) with different thresholds for generating a pseudo dataset from the unlabeled dataset.
- Task II: Implemented Virtual Adversarial Training (VAT) algorithm using 16-8 WideResNet for both CIFAR-10 (4000 and 250 labeled samples) and CIFAR-100 (10000 and 2500 labeled samples) utilizing VAT loss for the unlabeled dataset only.
- Task III: Implemented a Siamese network over the Pseudo-Labeling algorithm used in Task I for both CIFAR-10 (4000 and 250 labeled samples) and CIFAR-100 (10000 and 2500 labeled samples) with an objective of minimizing triplet loss.

2 Model Architecture

In this section, the model architecture used for the project is discussed.

Wide Residual Networks or WideResNet was proposed by Zagoruyko and Komodakis [2016] to tackle the problem of diminishing feature reuse in deep residual networks by decreasing the depth and increasing the width of the residual networks. WRNs help in reducing the training time of the model but results in an increase in the number of parameters due to the widening of the network. The paper showed that WRNs perform better than their thin and very deep counterparts. In WRNs, different combinations of the following parameters are tested:

1. Design of the ResNet block
2. Deepening factor l
3. Widening factor k

Table 2 provides the general structure for the WRNs. It comprises of an initial convolutional layer represented by conv1 followed by residual blocks conv2, conv3, and conv4 having size N. The structure is followed by an average pooling layer and a classification layer in the end. The residual blocks in the groups conv2-conv4 are scaled by altering the value of the widening factor k while the size of conv1 remains fixed. When the value of k is set to 1, the WRN's width becomes similar to ResNet. In our project, we have used WideResNet with a depth of 16 and a width of 8 which has a total of 11.0 million parameters.

3 Our Implementation

In this section, we discuss the implementation details of the different tasks of the project.

Task I We built upon Algorithm 2 in the project handout for Pseudo-Labeling. We made a few additions to the algorithm: We first train the model on only labeled data for the first T1 epochs (supervised learning). This was done so that the model could learn some representations beforehand. After T1 epochs, we consider the unlabeled samples and train on them as well with their pseudo labels. Along with the loss on our supervised samples, we consider the pseudo loss on the unlabeled samples and give it appropriate weightage depending on the epoch. This is the same as discussed in the original paper Lee [2013], termed as alpha weight. It makes sense because as we start to include unlabeled samples after T1 epochs, we give low weightage to their corresponding loss. But as the model learns on more unlabeled data and gets confident, we increase the alpha weight (proportional to epoch) to penalize the pseudo loss. After T2 epochs, this weight is set constant ($= 3$, from Lee [2013]).

For training, we use cross-entropy as our loss function and a batch size of 64 for training and testing both. The Stochastic Gradient Descent (SGD) is our choice of optimizer with momentum $= 0.9$, learning rate $= 0.03$, and weight decay $= 0.00005$. We also employ a MultiStep learning rate scheduler to decay the learning rate after 30 and 80 epochs to converge the model faster with higher accuracy. Moreover, we used a dropout of 0.3 during training so as to reduce overfitting of our model.

Task II We built upon Algorithm 3 and Algorithm 4 in the project handout for Virtual Adversarial Training. To the pseudo algorithm we added an additional cost, conditional entropy, given by equation 1. This cost was first introduced by Grandvalet and Bengio [2004] and was also used in VAT Miyato et al. [2019]. The conditional entropy minimization exaggerates the prediction of the model on every data point. The final objective function comes out to be the addition of the negative log-likelihood of the labeled dataset, regularizer R_{adv} (VAT loss), and the conditional entropy term. As we minimize this loss function, we are basically training our model to classify the perturbed and original image as the same. This loss makes the model robust to external noise and thus reduces overfitting.

$$\begin{aligned} \mathcal{R}_{\text{ent}} &= \mathcal{H}(Y \mid X) \\ &= -\frac{1}{N_l + N_{ul}} \sum_{x \in \mathcal{D}_l, \mathcal{D}_{ul}} \sum_{y} p(y \mid x, \theta) \log p(y \mid x, \theta) \end{aligned} \tag{1}$$

For training, we use cross-entropy as our loss function for labeled data and VAT loss for unlabeled data. Our combined loss function to minimise is then the sum of cross-entropy loss, α weighted VAT loss and conditional entropy loss. The Stochastic Gradient Descent (SGD) with momentum $= 0.9$, learning rate $= 0.03$, and weight decay $= 0.00005$ has been used. We used the following parameters in VAT: $\xi = 1 \times 10^{-6}$, $\epsilon = 6.0$, and $\alpha = 1.0$. We choose $\epsilon = 6.0$ because Miyato et al. [2019] reports the validation error to be reducing with increase in ϵ as shown in Figure 1. Batch size is set to 64 for both training and testing. Additionally, batch normalization is turned off while calling VAT loss.

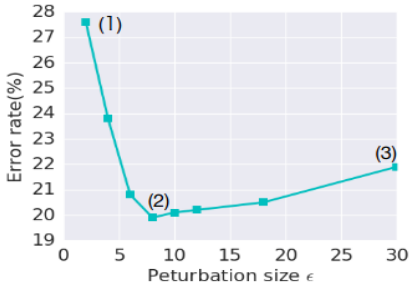


Figure 1: Validation error vs ϵ on CIFAR-10. From Miyato et al. [2019]

Task III In this task, we start with our best-performing trained models from Task I for different labeled data for CIFAR-10 and CIFAR-100. We use this model as a Siamese network after removing the last fully connected layer and adding three additional linear layers to it. By using this network as a Siamese network, images can be mapped to a lower dimensional Euclidean space with the distance corresponding to measure of similarity Schroff et al. [2015]. This is done by minimizing the Triplet loss Chechik et al. [2010] given by Equation 2 which takes in L2 normalised embeddings of anchor (A), positive (P), and negative (N). A randomly selected reference image is called the anchor, another image with the same class as the anchor is called positive, and an image with a different class than anchor is called negative. The triplet loss tries to minimize the distance between anchor and positive image while maximizing the distance between anchor and negative. So, as our model learns by minimizing the loss, it forms clusters of the different classes in feature space. We trained our network on the combination of labeled and pseudo-labeled data. After training the Siamese Network, we added a classification layer on top for classification task and then trained the network again. The layers before the new classification layer were frozen as we were only fine-tuning our model for classification.

$$\mathcal{L}(A, P, N) = \max\left(\|\mathrm{f}(A) - \mathrm{f}(P)\|^2 - \|\mathrm{f}(A) - \mathrm{f}(N)\|^2 + \alpha, 0\right) \tag{2}$$

For training, we use Triplet loss as our loss function for the triplets of images and a batch size of 4 for training and 64 for testing. The weights of the last classification layer are learned by minimizing cross entropy loss. Stochastic Gradient Descent (SGD) with momentum $= 0.9$, learning rate $= 0.03$, and weight decay $= 0.00005$ has been used as optimizer. We also used a MultiStep learning rate scheduler to decay the learning rate after 100 and 150 epochs for faster model convergence.

4 Experiments and Results

This section is divided into 4 parts. The first section discuss the datasets that were used for training and inference. The second, third, and fourth section demonstrates the experiments and results corresponding to Task I, Task II, and Task III of the project respectively. The training and testing code is written in PyTorch and Python. Moreover, the graphical plots are generated using Matplotlib.

4.1 Dataset

In this project, we have used two datasets namely, CIFAR-10 and CIFAR-100. The brief description of the datasets are as follows:

1. **CIFAR-10:** CIFAR-10 dataset is one of the most widely used datasets for image classification. It contains 60000 coloured images of size 32×32 in 10 classes, having 6000 images per class. The images are divided into training images and testing images with 50000 and 10000 images respectively.
2. **CIFAR-100:** CIFAR-100 is another famous dataset used for image classification and is similar to CIFAR-10. It also contains 60000 coloured images of size 32×32 but in 100 classes, having 600 images per class. The images are divided into training images and testing images with 500 and 100 images per class respectively. The 100 classes of this dataset is further grouped into 20 superclasses. Each image in the dataset is assigned a "fine" label which represents the class and a "coarse" label which corresponds to the superclass.

4.2 Semi-supervised learning using Pseudo Labeling

In this section, we discuss the different set of experiments performed during the implementation of Task I. For our model's architecture, we used WideResNet (WRN) with a depth of 16 and a width of 8 as it performed better than other model configurations. While going deeper with depth 28 and the same width of 8, we did note an accuracy of 85.70% on CIFAR-10 with 4000 labeled samples, which came at the cost of longer training time and a huge model with more than 30M parameters. The 16-8 WRN gave 80.78% accuracy for the same task while taking way less training time and the model being lighter (11M parameters).

We also experimented with different hyperparameters including iterations per epoch, epochs,

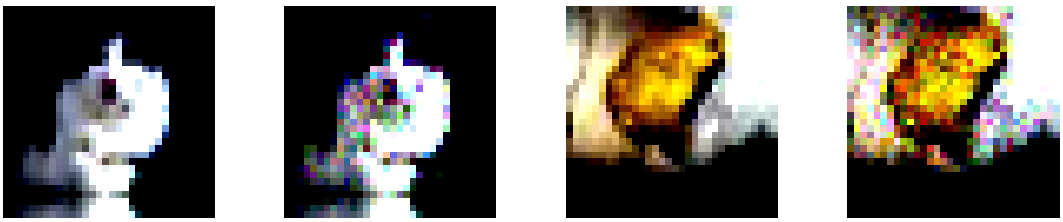
learning rate, T1 value, T2 value, and dropout for both CIFAR-10 and CIFAR-100 with different labeled samples. We report the hyperparameter values used for CIFAR-10 and CIFAR-100 for the different numbers of labeled samples in Table 3. Smaller iterations per epoch are chosen for 250 and 2500 labeled samples in CIFAR-10 and CIFAR-100 to reduce overfitting on the labeled data. We experimented with 256, 512, and 1024 iterations per epoch for the similar number of epochs but none of them performed better. We run for lesser epochs on CIFAR-10 with 250 labeled samples and CIFAR-100 with 2500 labeled samples to prevent overfitting on limited labeled training data. Also the epochs for CIFAR-10 with 250 labeled samples are chosen to be less than the epochs for CIFAR-100 with 2500 labeled samples because it has 10 classes as compared to CIFAR-100 which has 100 classes. Moreover, the values for T1 and T2 are set to 0 in the case of 250 and 2500 labeled samples in CIFAR-10 and CIFAR-100 respectively as we noticed that the model is not able to predict a good amount of pseudo labels and was running only on the labeled dataset, thus leading to overfitting. The values of T1 and T2 for CIFAR-100 with 10000 labeled samples are increased because we want the model to learn on the large number of labeled data for a significant number of epochs.

For Task I, we report the error rates of our implementation of the Pseudo-Labeling algorithm on CIFAR-10 (for 250 and 4000 samples) and CIFAR-100 (for 2500 and 10000 samples) in Table 1. The visualizations of the training metrics are shown in Figures 3 and 4. As we can see in the figures, the training metrics for CIFAR-10 with 4000 samples and CIFAR-100 with 10000 samples have random peaks. This is because of the hyperparameter *alpha weight* that is used to give weightage to the unlabeled data.

4.3 Semi-supervised classification using Virtual Adversarial Training

We outline the experimental details for Task II in this section. In this task, we used the same WRN architecture as in Task I. Here, we experimented with different values of ϵ and ξ. We trained the models for $\epsilon = 2.5, 6, 8, 10$ and $\xi = 1 \times 10^{-6}, 10$. Out of these combinations, the best model came out to be on $\epsilon = 6.0$ and $\xi = 1 \times 10^{-6}$. The adversarial examples generated with this configuration are shown in Figure 2. The result is also supported by the plot shown in Figure 1. The higher value of ϵ makes sense because the model is trying to add more noise to the image so as to change the labels but keeping the identifying details of the image same. As we choose a very high value of ϵ, the model completely changes the image by adding noise such that the original image cannot be brought closer to this newly generated one. Additionally, we also trained two models with same configurations; one with VAT loss for unlabeled data and another with VAT loss + conditional entropy loss. We noticed a 5% increase in the model accuracy by using the latter model.

For Task II, we report the error rates of our implementation of the Virtual Adversarial Training algorithm on CIFAR-10 (for 250 and 4000 samples) and CIFAR-100 (for 2500 and 10000 samples) in Table 1. The visualizations of the training metrics are shown in Figures 5.



(a) (b) (c) (d)

Figure 2: Task II adversarial images for $\epsilon = 6.0$. Figures 2a and 2c are the sample images from the dataset. Figures 2b and 2d are the corresponding adversarial images generated by the model.

4.4 Semi-supervised learning using Pseudo-Labeling and Siamese Network

In this section, we list out different set of experiments performed for Task III. While this algorithm sounded good in theory, in practice, the training of a Siamese network took a long time to converge and the triplet loss required a large amount of negatives to learn well. First we started out with only

Table 1: Test error for different Semi-Supervised Learning algorithms

	CIFAR-10		CIFAR-100	
SSL Algorithm	**250 labels**	**4000 labels**	**2500 labels**	**10000 labels**
Pseudo-Labeling ($\tau = 0.60$)	62.86	16.94	78.20	43.81
Pseudo-Labeling ($\tau = 0.75$)	66.59	**14.17**	75.43	**40.35**
Pseudo-Labeling ($\tau = 0.95$)	62.40	19.22	**72.87**	41.31
Virtual Adversarial Training (VAT)	**62.05**	22.58	78.74	48.26
Pseudo-Labeling ($\tau = 0.75$) + Siamese	71.94	31.65	84.68	55.57

one negative image per anchor but it gave unsatisfactory results. Therefore, to increase our number of triplets, we tested out with 2, 4, and 8 negative images per anchor. This increased out model's performance but only by 4%. At last, for every positive we created as many triplets as there are negatives in the mini-batch. However, such huge tensors require large memory and gave CUDA out of memory error. Because of this, we decided to use a training batch size of 4 which then increased our training time. Due to time constraint and limited GPU resources, we could not train our models for higher epochs. The hyperparameters used for this task are listed in Table 5 The next experiment we did is with the number of additional linear layers for the Siamese Network. We experimented with 1, 2, and 3 linear layers and got the best performance with 3 linear layers. The embedding of an image generated by the Siamese Network was of size 1 x 64. The final experiment we did is for the number of epochs for the training of the last classification layer. Epochs below 10 had an adverse effect on the model. The optimal number of epochs we got were 50 for the last layer.

For Task III, we report the error rates of our implementation of the Pseudo-Labeling + Siamese Network algorithm on CIFAR-10 (for 250 and 4000 samples) and CIFAR-100 (for 2500 and 10000 samples) in Table 1. The visualizations of the training metrics are shown in Figures 6. In figures, the loss is Triplet loss and the accuracy is for the final model with classification layer. As in the figures we can see that the model is not converging. Possible reason for that could be small number of epochs or less number of negative samples.

4.5 Comparative Analysis

A comparison of the test error rates on our different Semi-Supervised Learning algorithms has been depicted in Table 1. According to the results shown, Pseudo-Labeling performed best for CIFAR-10 with 4000 labeled samples and CIFAR-100 for both 2500 and 10000 labeled samples. Virtual Adversarial training worked out to be best in case for CIFAR-10 with 250 labeled samples. The third algorithm that is the mixture of Pseudo-Labeling and Siamese did not show significant results but it can be further modified for better results.

5 Conclusion and Future Work

In this project, we implemented different Semi-Supervised Learning algorithms to solve the task of classification on CIFAR-10 and CIFAR-100 datasets. For Task I, we implemented the Pseudo-Labeling algorithm and achieved better accuracy than other algorithms for three cases. For Task II, we used the method of Virtual Adversarial Training which showed promising results with a faster convergence than other two. For Task III, we explored the idea of using Pseudo-Labeling algorithm and Siamese Network together. The results were not according to our expectation but we believe that this approach can generate better results because of effectiveness of Siamese Network in unsupervised learning. As the possible future scope of this project, we can aim to improve algorithm used in Task III by using a pre-trained network and adding augmentation to generate more data as well as negative samples for the triplet loss.

References

E. Abbasnejad, A. R. Dick, and A. van den Hengel. Infinite variational autoencoder for semi-supervised learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*,

pages 781–790, 2017.

A. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16 (4):373–379, 1970. doi: 10.1109/tit.1970.1054472.

D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. Mixmatch: A holistic approach to semi-supervised learning. *ArXiv*, abs/1905.02249, 2019.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98*, 1998.

J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Int. J. Pattern Recognit. Artif. Intell.*, 1993.

G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 03 2010.

S. Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967. doi: 10.1109/tit.1967.1053952.

Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *CAP*, 2004.

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2014.

G. R. Koch. Siamese neural networks for one-shot image recognition. 2015.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. doi: 10.1145/3065386.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, and K. e. a. Lee. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. doi: 10.1162/tacl_a_00276.

D.-H. Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. 2013.

T. Miyato, S. ichi 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*, 41:1979–1993, 2019.

A. R. Sajun and I. Zualkernan. Survey on implementations of generative adversarial networks for semi-supervised learning. *Applied Sciences*, 12(3):1718, 2022. doi: 10.3390/app12031718.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *ArXiv*, abs/2001.07685, 2020.

A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

X. Yang, Z. Song, I. King, and Z. Xu. A survey on deep semi-supervised learning. *ArXiv*, abs/2103.00550, 2021.

S. Zagoruyko and N. Komodakis. Wide residual networks. *ArXiv*, abs/1605.07146, 2016.

Q. Zhang, L. T. Yang, Z. Chen, and P. Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018. doi: 10.1016/j.inffus.2017.10.006.

X. Zhu and A. B. Goldberg. Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130, 2009. doi: 10.2200/s00196ed1v01y200906aim006.

A Appendix

Table 2: Structure of Wide Residual Networks. From Zagoruyko and Komodakis [2016]

group name	output size	block type $= B(3,3)$
conv1	32×32	$[3 \times 3, 16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Table 3: Hyperparameters used for Task I

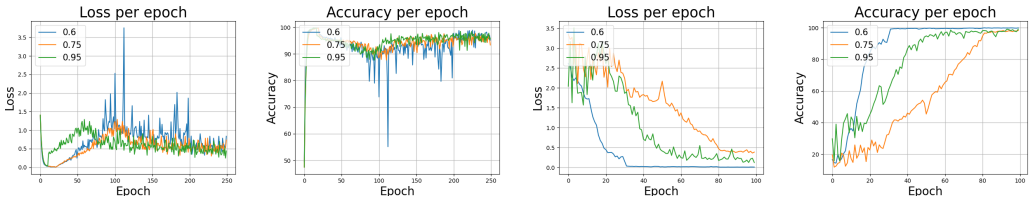
	CIFAR-10		CIFAR-100	
Hyperparameters	250 labels	4000 labels	2500 labels	10000 labels
Iterations per epoch	128	800	256	800
Epochs	150	250	150	250
Learning rate	0.03	0.03	0.03	0.03
Dropout	0.0	0.3	0.0	0.3
T1	-	20	-	60
T2	-	100	-	120

Table 4: Hyperparameters used for Task II

	CIFAR-10		CIFAR-100	
Hyperparameters	250 labels	4000 labels	2500 labels	10000 labels
Iterations per epoch	128	800	256	800
Epochs	100	150	100	150
Learning rate	0.001	0.001	0.001	0.001
Dropout	0.0	0.0	0.0	0.0
ξ	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
ϵ	6.0	6.0	6.0	6.0

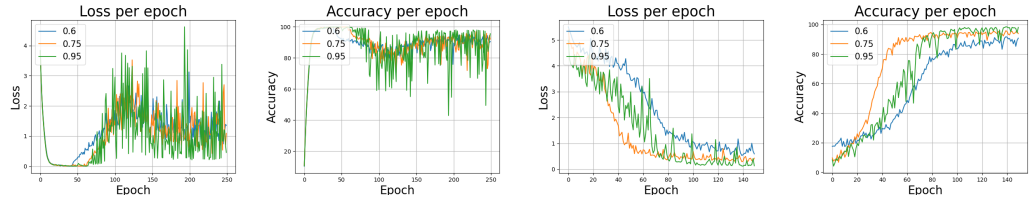
Table 5: Hyperparameters used for Task III

	CIFAR-10		CIFAR-100	
Hyperparameters	250 labels	4000 labels	2500 labels	10000 labels
Iterations per epoch	256	512	256	512
Epochs	400	200	200	200
Learning rate	0.001	0.001	0.001	0.001
Dropout	0.0	0.0	0.0	0.0



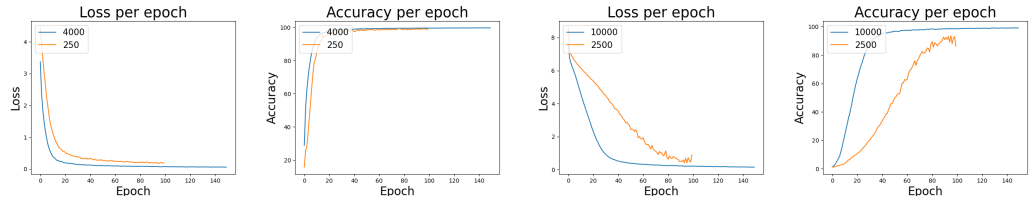
(a) (b) (c) (d)

Figure 3: Task I training metrics for CIFAR-10 for $\tau = 0.60, 0.75, 0.95$. Figures 3a and 3b are for 4000 labeled samples. Figures 3c and 3d are for 250 labeled samples.



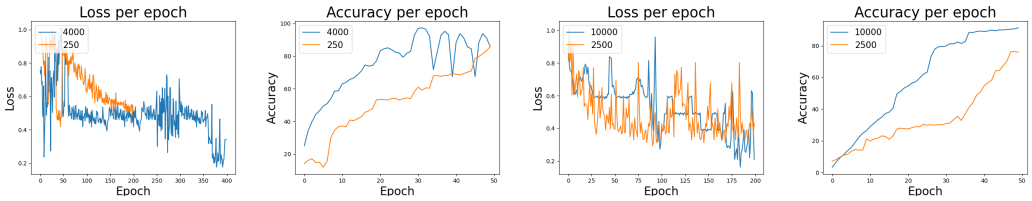
(a) (b) (c) (d)

Figure 4: Task I training metrics for CIFAR-100 for $\tau = 0.60, 0.75, 0.95$. Figures 4a and 4b are for 10000 labeled samples. Figures 4c and 4d are for 2500 labeled samples.



(a) (b) (c) (d)

Figure 5: Task II training metrics. Figures 6a and 6b are for CIFAR-10 for both 4000 and 250 labeled samples. Figures 6c and 6d are for CIFAR-100 for both 10000 and 2500 labeled samples.



(a) (b) (c) (d)

Figure 6: Task III training metrics. Figures 6a and 6b are for CIFAR-10 for both 4000 and 250 labeled samples. Figures 6c and 6d are for CIFAR-100 for both 10000 and 2500 labeled samples.