Semi-Supervised Learning via Offline Pseudolabel Generation and Consistency Regularization

Mohamad Qadri
mqadri@andrew.cmu.edu

Maggie Collier
macollie@andrew.cmu.edu

1 Introduction

Semi-supervised learning (SSL) allows the training of machine learning models on a small amount of labeled data and a large amount of unlabeled data with the goal of circumventing the need of labeling entire datasets, which is often a time-consuming and error-prone process. One approach in SSL is self-training in which a network is trained in a fully-supervised fashion with the labeled examples and then retrained on its own raw predictions of the unlabeled examples. However, training on both the labeled data and unlabeled data without any further pseudolabel quality inspection sometimes does not improve performance relative to training with only the labeled portion of the dataset. One way this problem can be addressed is to investigate how pseudolabels are generated, selected, and/or modified. To explore this idea, this work uses several SSL aproaches which use different methods in pseudolabel generation for an image classification task. In addition, this work proposes a method of pseudolabel selection that uses the confidence in a teacher network's prediction on unlabeled data to determine if the pseudolabeled sample should be included in the next training iteration. With these confidence metrics, the approach is shown to perform better than the baseline, indicating that pseudolabel selection via confidence metrics may be a compelling strategy worth further investigation.
Project code available at https://github.com/mqadri93/semi-supervised-training

2 Data

This project is a proof of concept; therefore, two benchmarked image classification datasets were selected. CIFAR-10, which includes 60,000 32×32 images: 10,000 test images and 50,000 training images labeled with one of ten classes. In our experiments, different subsets from the training dataset were selected to train a fully-supervised network and disregard the labels for all remaining samples in the training dataset. Experiments were conducted on 3 subsets (splits) of the training set: 500, 2,000, and 5,000 labeled examples.
Tests were also done on STL-10, which is a dataset used as a benchmark for unsupervised and semi-supervised approaches. It contains 10 classes, and the images are of a higher resolution (96×96) than in CIFAR-10. STL-10 contains 500 training images (10 pre-defined folds) for a total of 5,000 training samples, 800 test images per class, and 100,000 unlabeled images for unsupervised learning and semi-supervised training.

3 Background

In the midway report, a baseline had been implemented on CIFAR-10, and the upper and lower bounds on the testing accuracy were established in the case that only 10% of CIFAR-10's training set is used as labeled data. (ResNet 18 used for all experiments.) Through supervised learning on all 50,000 training samples in CIFAR-10, the upper bound was found to be approximately 90.96% (as shown in Table 1). Supervised learning on 10% of CIFAR-10's training set established the lower bound to be 76.49%.

Table 1: Previous results on CIFAR-10.

Method	Test Accuracy (%)
Supervised (50,000)	90.96
Supervised (5,000)	76.49
Raw Prediction	79.43

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

To motivate this work, an approach based on [1] was implemented on CIFAR-10, in which 10% of the training set was considered labeled. After excluding the portion of [1] that injects noise into the data, the implemented approach simply takes the teacher's raw predictions as the pseudolabels. The "Raw Prediction" row of Table 1 demonstrates that this strategy barely improves the testing accuracy above the lower bound, indicating that the teacher's raw predictions are not informative enough to boost performance relative to the lower bound. This outcome motivated further investigation into how these raw predictions can be modified or used to produce more informative pseudolabels.

4 Related Work

The literature relevant to this work mainly focuses on self-training, teacher-student approaches, and consistency regularization. Radosavovic et al. [2] investigated omni-present learning which utilizes all available labeled data plus large-scale unlabeled data with the goal of surpassing state-of-the-art, fully-supervised baselines. The paper proposed generating pseudolabels using data distillation which ensembles the results of the teacher model run on different image transformations. Such data augmentation is known to improve the test accuracy of deep learning models indicating that they provide rich and new information to the network [3]. Hinton et al. [4] used model distillation to transfer the generalization ability of an ensemble of networks, initialized with different parameters, to a smaller neural network by training the latter on soft labels taken as the geometric mean of each neural network output in the ensemble. They demonstrated on a speech recognition task how a distilled model trained on such aggregated soft labels performed better than a single model trained on hard labels only. Xie et al. [1] trains a teacher network on labeled ImageNet images. The teacher network is then used to produce pseudolabels for a large number of unlabeled images. A student network is trained by minimizing the cross entropy loss on labeled and unlabeled images. The process is iterated by treating the newly trained student as the next teacher network. Noise, in the form of dropout and data augmentation, is injected during the learning of the student which is shown to improve the generalization and accuracy of the predictions. Mixmatch [5] is one of the state-of-the-art approaches for semi-supervised classification tasks. It uses MixUp [6] which involves training a network with a convex combination of examples and their labels encouraging a linear behavior in between training examples. MixMatch generates pseudolabels by augmenting an unlabeled sample K times, averaging the output distributions over the classes and finally sharpening the distribution to obtain the final soft label. Sajjaddi et al. [7] uses the idea that labels should remaining unchanged under different data augmentations and introduced a transform/consistency loss which minimizes the L2 difference between different passes of the same (augmented) sample. Along the same line, Laine et al. [8] uses a similar loss to encourage the prediction of an augmented sample to be close to this sample's temporal average prediction during training. Bank et al. [9] uses self-training to train a classifier based on its own predictions if these predictions are generated with high confidence. They use the softmax layer's probabilities as a confidence measure and explored different methods to assess the quality of pseudolabels such as dropout consensus and bagging.
Our work leverages the data distillation approach introduced in [2], soft pseudolabel generation for unlabeled samples from [4], and the loss function from [5]. Similar to Bank et al. [9], our work also introduces and investigates an approach for high confidence pseudolabel selection.

5 Methods/Model

5.1 Baseline and Self-Training

Our baseline is based on the work of [1], but excludes the noise injection process. A teacher network is first trained on the labeled data only and is then used to make predictions on the entire unlabeled training dataset. These raw predictions are taken as the pseudolabels for the unlabeled data. A student network is then trained on both the labeled and unlabelled samples. This student network becomes the teacher network for the next iteration. In each iteration, the teacher network generates pseudolabels offline (outside the training procedure) and a student is trained with the existing labeled samples and the new pseudolabels for the unlabeled data. This project focuses on this pseudolabel generation step and studies ways to select unlabeled samples that satisfy a confidence metric. These teacher-student iterations are repeated until no further significant improvement is seen when comparing the student and the teacher networks. (See Algorithm 1 lines 14-17 and 21-23).
We view our network as a probabilistic function $f(y|z, \theta)$ conditioned on the the network weights θ and the input z. We borrow the notation used in [5], we use x for labeled samples, u for unlabeled samples, p for a hard label associated with a labeled sample x and q for a pseudolabel associated with unlabeled sample u. We refer by $\mathcal{D}_{\sqcap}$ to the set of all unlabelled samples available for training and by

$\mathcal{D}_{\S}$ to the set of all labeled samples available for training We refer to the set of labeled data used to train the network as $\mathcal{X}$ and the set of selected unlabeled data used to as $\mathcal{U}$. We note that $\mathcal{U}$ is a subset of $\mathcal{D}_{\sqcap}$. (All experiments used the Resnet-18 architecture shown in Fig. 1.)

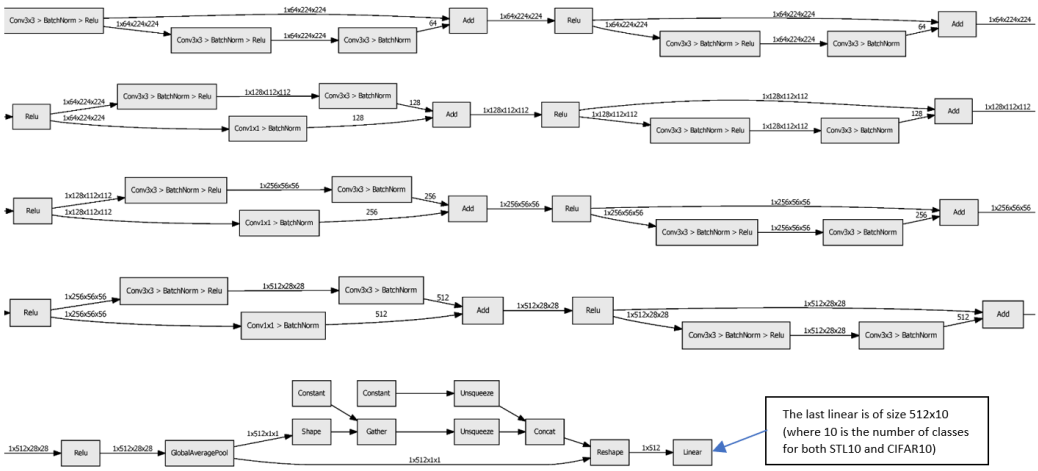

Figure 1: Resnet-18 network used in all experiments in the report. (Figure generated with Hiddenlayer python libary). We used the raw output from the last linear layer in the data distillation and confidence measure pipelines. Residual networks [10] use identity layers and residual shortcuts to ease the training of deeper networks and prevent the vanishing gradient problem

5.2 Data Augmentation and Data Distillation

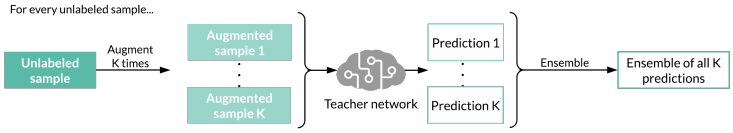

Figure 2: Data distillation block

Data augmentation is leveraged in two different ways—first in data distillation: at the beginning of each teacher-student iteration i, pseudolabels are generated offline for the unlabeled data (see Fig. 2). For each sample u in the unlabeled set $\mathcal{D}_{\sqcap}$, K augmentations ($K = 10$ in our experiments) are performed to get augmented versions u_k for $\forall k \in 1..10$. In the *data distillation* and *data distillation + confidence measure* experiments, each u_k is passed to the teacher network (which is the student trained in loop $i-1$) to generate the corresponding output. A pseudolabel is generated by taking the mean of all K outputs over each class distribution $q = \frac{1}{K}\sum_{k=1}^{K} f_{i-1}(y|u_k, \theta)$ (See Algorithm 1 (lines 9-13)).
Standard data augmentation is also performed for each training example. Specifically, for each sample x, an augmented version $x_a = \text{Augment}(x)$ is generated and used for training. All augmentations (during training and data distillation) consisted of random cropping (with a padding of 4) and random horizontal flips.

5.3 Confidence-measure-based pseudolabel selection

We hypothesized that careful selection of pseudolabeled training samples based on a confidence metric should boost the testing accuracy of trained networks compared to using all samples for training indiscriminately. This metric is to be computed and assigned to each unlabeled sample and, more importantly, should separate "good" and "bad" pseudolabels. To find a possible metric, we searched for patterns in the raw prediction of a neural network for different augmented versions of the same sample and engineered features which might be predictive and descriptive of the quality of a pseudolabel. (These raw predictions are the output of the last fully connected layer output. This output is not a probability distribution since no softmax layer is applied.)

Definition 1. *We define a "good" pseudolabel as a vector q whose maximum value is assigned to the index corresponding to the correct class label z (i.e:* $argmax(q) = z$*).*

Of course, the correct class label for the unlabeled samples are not known. Therefore, in order to obtain a confidence metric, we reserve samples from the labeled set/split (X) to be used for confidence metric calculation only. These samples are not used for training (similar to a validation set). In this report, we refer to this subset of data as $\mathcal{V}$.

5.3.1 Minimum Variance Thresholding (MVT)

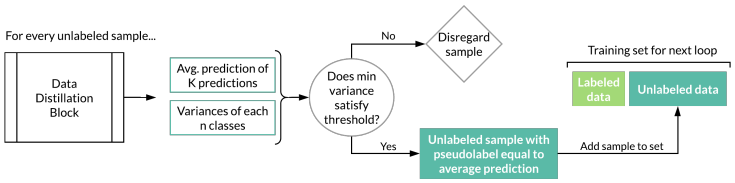

Figure 3: Pseudolabel selection using minimum variance thresholding

The first confidence metric that was successfully applied to the CIFAR-10 dataset is minimum per-class variance over the data distillation output (see Fig. 3). Specifically for each sample in $\mathcal{D}_{\sqcap}$, we extract the output of data distillation which is a $K \times C$ matrix where K is the number of augmentations and C is the number of classes. The columns of this matrix are extracted to obtain C $K \times 1$ vectors each representing the raw predicted output for each of the C classes. We calculate the variance of the C vectors to obtain a set of C variances. The minimum of this set mv, is compared against a threshold. An unlabeled sample is considered as a high confidence sample and used to train the next student network if mv is less than a threshold T ($T = 0.01$ in our experiments). (See Algorithm 1 lines 4-8).
Using MVT increased the final testing accuracy of the trained network (see Results); however, the threshold T required manual tuning. As a next step, we aimed at finding a metric that would require less manual tuning and which is generalizable to other datasets.

5.3.2 A Confidence metric via SVM classification (CSVM)

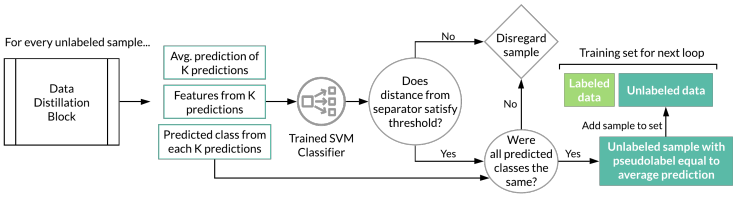

Figure 4: Pseudolabel selection using SVM and augmentation prediction agreement

Unlike MVT, CSVM requires a training step on a subset of $\mathcal{V}$ before the pseudolabels are generated for the unlabeled samples. To train the SVM, we extract the $K \times C$ output matrix from data distillation for each sample v_i in $\mathcal{V}$. The columns of the matrix are extracted to obtain C $K \times 1$ vectors, one for each class. We extract two features from these vectors. The mean of each vector is calculated and the maximum mean mm is used as the first feature. The second feature is the minimum variance mv, calculated as described in the previous subsection. At this point, we have 4 pieces of information for each sample v_i in $\mathcal{V}$: mv_i, mm_i, the averaged pseudolabel q_i from data distillation and the correct one hot hard label z_i. We construct a new dataset $\{\mathbf{X}, y\}$ of size |$\mathcal{V}$| from all samples in $\mathcal{V}$ where $\mathbf{X} = [mv, mm]$ are the features and y is the label. $y = 1$ if $argmax(q_i) = z_i$ and $y = -1$ if $argmax(q_i) \neq z_i$.
We attempt to sample an equal number of positive and negative examples from $\{\mathbf{X}, y\}$ to produce a subset of data for training and testing the SVM. After training the SVM, we obtain a decision function which given a new value $\mathbf{X} = [mv, mm]$ generated from an unlabeled sample in $\mathcal{D}_{\sqcap}$, returns its geometric margin γ from the separating line $L = wx + b$ (i.e the distance D between $\mathbf{X}$ and the separating line in terms $||w||$). Therefore, during pseudolabel generation, a pseudolabel is considered higher confidence if D is greater than a threshold T (in our experiments T =1). In addition, we add a condition before adding the pseudolabeled sample to $\mathcal{U}$ to be used when training the next student.

This condition is based on consistency regularization via data augmentation, which asserts that a network should generate the same label for all augmentations of the same sample u. Based on this idea, we disregard all unlabeled samples for which the teacher does not predict the same class for all K augmentations. To summarize, we use the SVM decision function to infer a confidence score for all samples in the unlabeled set $\mathcal{D}_{\sqcap}$. An unlabeled sample is considered as a high confidence sample if $D > 1$. If in addition, all K augmentations of the pseudolabel agree on the predicted class, the unlabeled sample is added to $\mathcal{U}$ which will be used to train the next student network. (See Fig. 4)

5.4 Loss function and batching process

We equally balance the number of unlabeled $|\mathcal{U}'|$ and labeled samples $|\mathcal{X}'|$ in each batch of size$|\mathcal{B}|$ ($|\mathcal{B}| = |\mathcal{X}'| + |\mathcal{U}'|$). Each sample in $\mathcal{B}$ is selected randomly from $\mathcal{X}$ or $\mathcal{U}$ as follows: we generate a random number r from a uniform distribution $\mathcal{U}(0, 1)$ and select a labeled sample if $r < 0.5$ and an unlabeled sample otherwise.
Our loss function is composed of an unlabelled loss term $\mathcal{L}_{\mathcal{U}}$ and a labelled loss term $\mathcal{L}_{\mathcal{X}}$ where: $\mathcal{L}_{\mathcal{U}} = ||f(y|u, \theta) - q||_2^2$ is an L2 loss between the output distribution of the network for sample u and the corresponding pseudolabel which is a vector of continuous values and $\mathcal{L}_{\mathcal{X}} = H(p, f(y|x, \theta))$ is a cross entropy loss for the labeled training sample x.
We balance between the two losses using a balancing factor λ to obtain our final loss function:

$$\mathcal{L} = \frac{1}{|\mathcal{X}'|}\mathcal{L}_{\mathcal{X}} + \lambda\frac{1}{|\mathcal{U}'|}\mathcal{L}_{\mathcal{U}}$$

We experimented with different values of $\lambda \in [0.01, 0.1, 1]$ and used $\lambda = 0.1$ throughout our experiments since it performed best.

5.5 Optimization algorithm

To train our network, we used stochastic gradient descent with momentum=0.9 and weight decay=5×10^{-4}. We used pytorch's optimization package which provides 2 interfaces $loss.backward()$ to compute the gradients and $optimizer.step()$ to perform the parameter update. The generic SGD with momentum parameter update for a model with parameters θ, momentum value equal to 0.9, learning rate η, and Loss $\mathcal{L}$ is

$$v_t = 0.9v_{t-1} + \eta\nabla_\theta\mathcal{L}(\theta)$$

$$\theta = \theta - v_t$$

5.6 Other methods

After implementing our baseline and SVT . We first tested several confidence metrics based on entropy and other statistical analysis using calculated variances and means with various level of success. Manually finding such confidence metric using feature engineering was a time consuming process and such confidence metrics may not be "optimal" . As a result, we attempted to train a fully connected neural network to predict such a decision boundary. More precisely, we extracted N labeled samples from CIFAR-10 (x_i, y_i). For each sample. ran our data augmentation pipeline to generate a $K \times C$ augmentation matrix A using a teacher network. The columns of A are extracted to get C $K \times 1$ vectors v_c. A dataset (A_i, z_i) was constructed where z_i=1 if $v_c = y$ for all c and z_i=0 if $v_c \neq y$ for any c. A fully connected neural net was trained on this dataset using a binary cross entropy loss. Our goal was to train a NN to learn a probability $p(z_i = 1|A_i)$ which will be high if the NN believes that the A_i is a good pseudolabel and predictive of the correct class. This method did not work well or needed more time to analyze. When training this network, the training accuracy increased while the testing accuracy decreased which is indicative that the neural network might be overfitting and might not be learning what we intended it to learn.

6 Results

The following few tables define all of your hyperparameters and variables used to get the final results

Table 2: Parameter Definition

η	Starting learning rate
η_{dec_epoch}	learning rate decay at epochs
η_{dec_rate}	learning rate decay rate
E	Number of epochs
B	Batch size
λ	loss balancing factor
T	Threshold (Used in confidence metrics experiments)
M	Max number of student teacher loops
N	Number of samples reserved for usage by confidence metrics
N_{svm}	Minimum number of samples allowed for training and testing the SVM

Table 3: Parameters used for testing CIFAR-10 (all splits)

Method	η	η_{dec_epoch} [1]	η_{dec_rate}	E	B	λ	T	M
CSVM	0.1	[40, 120, 180]	0.1	250	64	0.1	1	9
MVT	0.1	[40, 150, 350]	0.1	400	64	0.1	0.01	10
data distillation	0.1	[15, 35, 65]	0.1	80	64	0.1	-	5
raw predictions	0.1	[15, 35, 65]	0.1	80	64	0.1	-	6
supervised	0.1	[40, 120, 180]	0.1	250	64	0.1	-	

For $|\mathcal{X}|$ = 500, 2000, 5000, N = 100, 200, 1000, and N_{svm} = 150, 150 and 50 respectively. Tested data distillation + min variance only for $|\mathcal{X}|$ = 5000. η_{dec_epoch} was chosen according to when the training accuracy plateaued

Table 4: Parameters used for testing STL-10

Method	η	η_{dec_epoch} [1]	η_{dec_rate}	E	B	λ	T	M	N_{svm}
CSVM	0.01	[40, 80]	0.1	120	64	0.1	1	10	200
data distillation	0.01	[40, 60]	0.1	80	64	0.1	-	10	-
supervised	0.01	[40, 80]	0.1	120	64	0.1	-	-	-

η_{dec_epoch} was chosen according to when the training accuracy plateaued

We used the testing accuracy as the evaluation metric to compare between different methods. Fig. 5 shows the testing accuracy curves over multiple teacher-training loops for CIFAR-10 using 5,000 labeled samples. Our lower bound is the network trained on the labeled examples only and our upper bound is the fully supervised network trained on all labeled samples in CIFAR-10. MVT (Min Var Threshold) and CSVM (With SVM) both have a final testing accuracy that is around 3% higher compared to using data distillation only and 6.31% higher than our baseline (training student on raw predictions). Note that both MVT and CSVM both surpass the network trained on only the fully labeled split by around 9%. The maximum testing accuracy for these methods is summarized in Table 2. We also tested CSVM and other methods on different labeled data splits: 500, 2000 and, 5,000 labeled samples. These tests are summarized in Fig. 6. We see that the final testing accuracy of CSVM ("with svm" in plot) is higher across all splits. However, we notice that the performance jump from using a confidence metric is lower (compared to training on the teachers' raw predictions) indicating that CSVM is sensitive to the performance of the initial teacher network. We tested CSVM on STL-10 by first training a teacher network on all 5000 labeled samples. This network is our lower bound and had a final testing accuracy of 78.8%. We train a network using data distillation only and another using CSVM. CSVM surpassed data distillation by 1.13% and surpassed the initial teacher network by 4% . (see Fig. 7). The results are summarized in table 5.

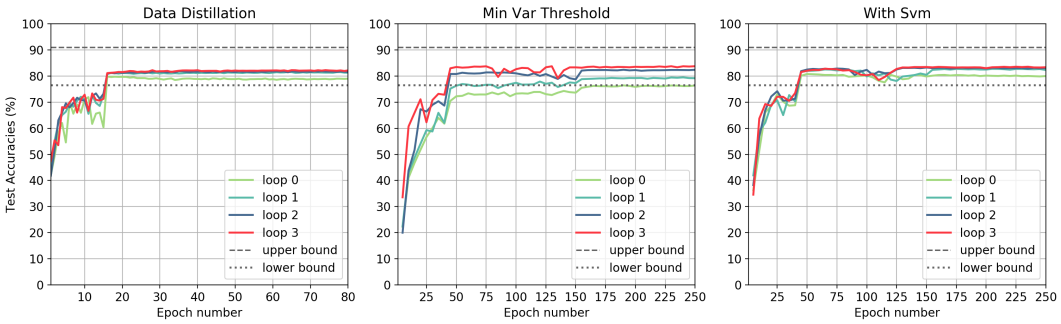

Figure 5: Test accuracies of different methods on CIFAR-10, where 5,000 samples were considered labeled.

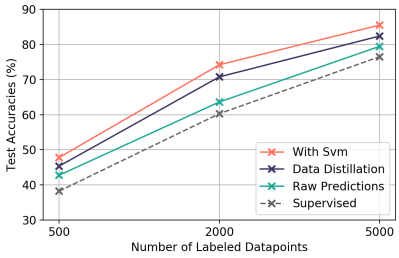

Figure 6: Testing accuracy of methods with different numbers of labeled samples from CIFAR-10.

Method	CIFAR-10	STL-10
Supervised	76.49	78.8
Raw Prediction	79, 43	-
Data Distillation	82.36 (loop 4)	81.75 (loop 9)
Min Var Thresholding	85.74 (loop 6)	-
SVM Thresholding	85.48 (loop 8)	82.88 (loop 6)

Table 2: Max testing accuracy (%) of each method on CIFAR-10 and STL-10 with 5000 labeled samples

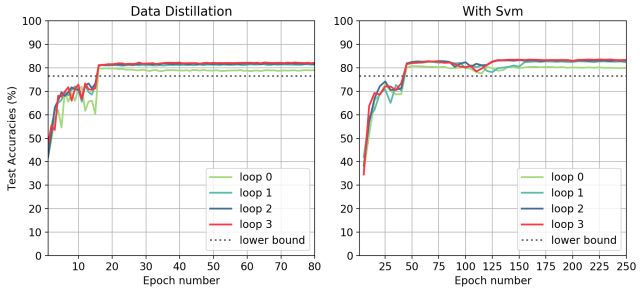

Figure 7: STL-10

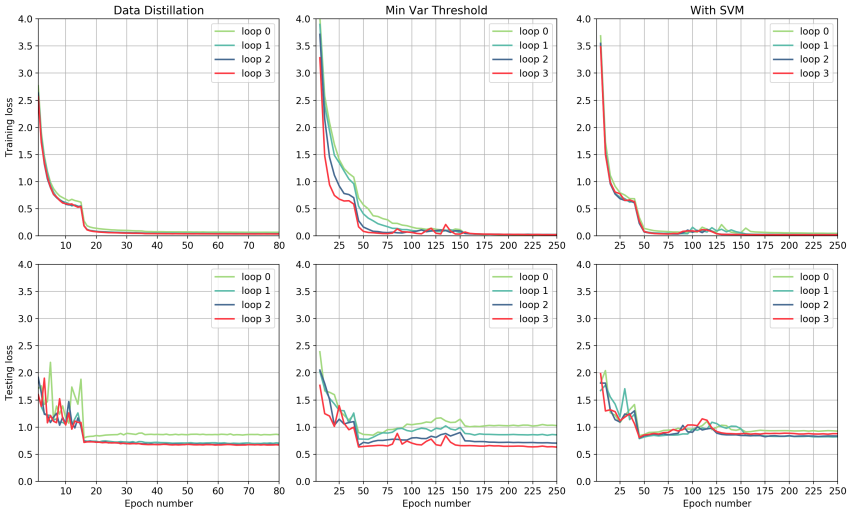

Figure 8: CIFAR-10 Losses. Row 1: training loss. Row 2: testing loss. Note that the training loss is the L2 loss while testing loss is a cross entropy loss

This section gives an analysis of the two confidence metrics introduced in this report: minimum variance and SVM based classification

7 Analysis

7.1 MVT

We used the labeled samples allocated for statistical analysis to calculate the variance and standard deviation of the set of minimum variance thresholds for the samples that are correctly and the samples that were incorrectly classified. Figure 6 shows the two distribution for one of the training runs

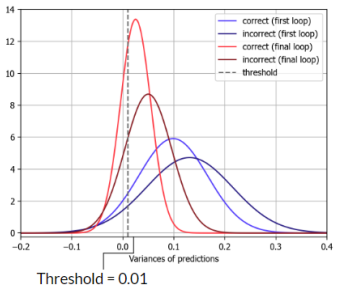

Figure 9: Example distribution of minimum variances for correct and incorrect classification

We see that there is a clear separation between the two sets of minimum variances: The set of correctly labeled examples ($\mu_{correct} = 0.099$, $\sigma_{correct} = 0.067$) have a smaller mean and standard deviation compared to the set of incorrectly labeled samples ($\mu_{incorrect} = 0.13$, $\sigma_{incorrect} = 0.084$). A low and constant confidence value threshold of 0.01 was used which allowed the pipeline to disregard incorrect and low confidence samples with higher probability which allowed the neural network to train on cleaner pseudolabels and as result outperform networks where only data distillation is being used. It is interesting to notice that as the number of teacher-student iterations increase, the means of the two distribution ($\mu_{correct} = 0.025$, $\mu_{incorrect} = 0.05$) shift closer to the selected threshold. The variance of the two distributions also decreases ($\sigma_{correct} = 0.03$, $\sigma_{incorrect} = 0.046$). This allows the network to train on an increasing number of highly confident unlabeled samples in addition to the labeled portion, and as a result, increase its testing accuracy and generalization performance.

7.2 CSVM

To analyze how well the metric used in CSVM filters high from low confidence predictions, we analyzed the distance between each sample in our validation set and the decision boundary learned in different loops. Fig. 10 shows histograms of these distances for data correctly and incorrectly predicted by the teacher network (class $y = 1$ and $y = -1$ in our SVM respectively). We show these histograms for two different loops for CIFAR-10 and STL-10 respectively: the first loops and the loops in which the maximum test accuracy was achieved. On these histograms, 0 on the x-axis indicates samples located on or very near the decision boundary, and 1 on the x-axis indicates samples located right at our threshold T. For CIFAR-10, the histogram in the first loop indicates that a decent separation was found by our SVM: most of the correctly labeled samples are above the decision boundary, and many of those samples fall at or above our threshold. This indicates that our approach successfully selected high confidence pseudolabels for the first loop on CIFAR-10. By the later loop, our approach still has a decent separation between the correct and incorrectly predicted samples by the teacher, and many of these points fall above our threshold as well. In STL-10, the histogram from the first loop shows that the decision boundary learned was not as successful at separating the data into the correct and incorrect classes as in CIFAR-10; however, the threshold being at 1 enabled our approach to still select samples with correct pseudolabels, as few incorrect predictions surpassed the threshold. In the later loop for STL-10, more of the distribution from the correctly predicted set falls above the decision boundary, allowing our approach to select more of the correctly predicted samples. As in the first loop, our threshold appears to only select a small amount of incorrect pseudolabels. When comparing between the first and later loops for both datasets, the distribution of samples seems to spread out in later loops. This indicates that our teacher is predicting more samples with higher confidence.

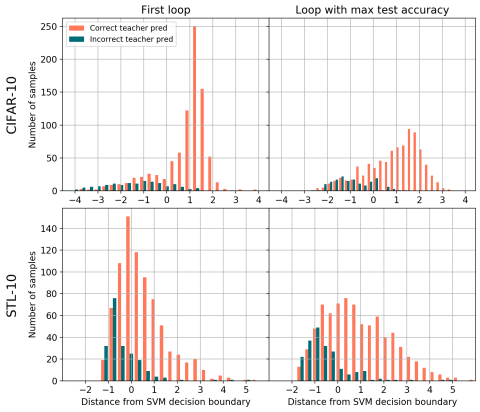

Figure 10: Plot showing the distribution of correct teacher prediction ad incorrect teacher prediction output by the SVM. For CIFAR-10, the data shown is from the trials with 5,000 labeled samples.

8 Conclusion and Future Work

In this project we explored different recent ideas in SSL and constructed confidence metric for offline high confidence pseudolabel generation. We demonstrated on two different datasets how using a confidence metric increased the accuracy and generalization ability of the final model. We note that the performance boost of CSVM in STL10 was less important compared to CIFAR-10. This might be indicative that the features used to train the SVM model might have to be revisited to provide better generalization across datasets. One main assumption and limitations is that we restricted our confidence metric search to metrics that allow linear separation of samples. A direction for future work would be to perform more rigorous analysis of possible features that can be extracted using data augmentation and data distillation . This idea can also be tested using non-linear classification techniques that could capture more interesting patterns in the raw prediction of a neural network.

References

[1] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. 2019.

[2] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data Distillation: Towards Omni-Supervised Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2018.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. 2015.

[5] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 32*, 2019.

[6] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[7] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29*. 2016.

[8] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *CoRR*, abs/1610.02242, 2016.

[9] Dor Bank, Daniel Greenfeld, and Gal Hyams. Improved training for self training by confidence assessments. In *Intelligent Computing*, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.