Computer Vision Project Semi-Supervised Image Classification

Basavaraj Rajendra Sunagad
7015644
M. Sc. Computer Science student
Saarland University
basu00001@stud.uni-saarland.de

Abhijith Srinivas Bidaralli
7015642
M. Sc. Data Science and AI student
Saarland University
abbi00001@stud.uni-saarland.de

Abstract

To achieve reasonable success using deep neural networks we need large datasets to train them. This has led to explore ways to train these networks using limited annotated data, Semi-Super learning (SSL) is one such approach. The key idea behind SSL is to develop training strategies that learn from both labeled and unlabeled data. In this project we implement common techniques for SSL and attempt to beat a benchmark result for semi-supervised image classification.

1 Introduction

The success of deep neural networks is partly attributed to their scalability, i.e., training them on bigger datasets results in better performance. Often deep networks achieve their strong performance through supervised learning, which needs a labeled dataset. The performance benefit gained by the use of a bigger dataset comes at a significant cost, as labeling data often requires human labor. This cost is especially extreme when labeling requires an expert (for example, a doctor in medical applications).

This is where semi-supervised learning proves to be extremely beneficial, it provides an effective means of leveraging unlabeled data to improve a model's performance greatly. Since acquiring unlabeled data has minimal effort, the scope for good SSL methods is huge in deep networks.

In our tasks we understand and implement different classes of SSL methods namely, Pseudo-labeling, VAT and FixMatch. Pseudo-labeling is a method of producing artificial or pseudo labels for unlabeled images based on few labeled images, and using the pseudo labeled data for training the network. Virtual adversarial training (VAT) is a simple SSL method in which we deliberately add noise to an image in such a way that the network misclassifies the image. In adversarial training we add adversarial noise to the images and continue training our network. The network will be robust to test images perturbed with adversarial noise [5].

FixMatch is a state of the art SSL algorithm in which, a supervised model is trained on labeled images with cross-entropy loss.Each unlabeled image undergoes two augmentations, strong and weak. The weakly augmented image is used to get pseudo-labels, the strongly augmented image is passed through our model to get a prediction over classes. This is compared to ground truth pseudo-label using cross-entropy loss. Both the losses are combined and the model is tuned [6].

In our attempt to improve the performance of FixMatch we propose to use cosine-embedding loss [9] in to enforce the model to get similar representations for weak and strong augmented versions of same image. We achieve a minor performance boost using the proposed method compared to our implementation of FixMatch.

2 Implementation

2.1 Pseudo-labeling

The main goal of pseudo-labeling is to enable usage of large amounts of unlabeled data along with unlabeled data. Although it's a naive approach, pseudo-labeling has proven to be extremely beneficial when dealing with large unlabeled data. The technique is as follows, first train the model on a batch of unlabeled data, use the trained model to predict labels on a batch of unlabeled data, consider the predicted labels as ground truth labels (pseudo-labels) to calculate loss on unlabeled data, combine labeled loss with unlabeled loss and backpropagate [4].

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L({y'}_i^m, {f'}_i^m) \tag{1}$$

In our implementation we first train the model only using labeled data for few epochs and then make predictions on unlabeled data. To get better results, the pseudo-labels are only accepted if the predictions are over a certain threshold and the rest are discarded for that iteration. We compute loss using the pseudo-labels and combine it with labeled loss with a weight as proposed by [4]. The weighing factor is a linear function which increases with number of epochs, this helps in better convergence.

2.2 VAT

For this task, we had to implement the Virtual Adversarial Training. Local distributional smoothness (LDS) can be defined as the smoothness of the output distribution of the model, with respect to the input. LDS regularization inclines towards the smoothening of output distribution. Smoother the distribution, better will be the generalization. This will result in model giving similar outputs for unseen data as the training data.

Virtual Adversarial Training is an effective technique for LDS. We take an input, transform this by adding a small perturbation 'r'. This perturbation must be in the adversarial direction, which means, the perturbed input should give different output than the output of original input. We then calculate the KL divergence between the two then update the model parameters through backpropagation, to minimize the KL divergence.

In our implementation, we calculate this VATLoss by first calculating the model output on unlabelled image and then calculate the model output on the perturbed unlabelled image. We then calculate the KL divergence between two images and add this to the classification loss.

$$\Delta_{KL}(r, x^{(n)}, \theta) \equiv KL[p(y|x^{(n)}, \theta)||p(y|x^{(n)} + r, \theta)] \tag{2}$$

$$r_{v-adv}^{(n)} \equiv arg_r max[\Delta_{KL}(r, x^{(n)}, \theta); ||r||_2 \leq \epsilon] \tag{3}$$

$$LDS(x^{(n)}, \theta) \equiv -\Delta_{KL}(r_{v-adv}^{(n)}, x^{(n)}, \theta) \tag{4}$$

$KL[p, p']$ is a non-negative function that will measure the KL divergence between the two input distributions.

For the above we generally give p as the true distribution for KL divergence. In case of semi-supervised learning we use virtual labels that are an approximation of this. This is denoted by $p(y|x^{(n)}, \theta)$. This approximation is better if you have more number of labelled samples.

$r_{v-adv}^{(n)}$ is done to obtain the perturbation that gives the maximum KL divergence between the two input distributions.

Finally $LDS(x^{(n)}, \theta)$ is the loss, which is step in negative direction of KL divergence between two input distributions, with one distribution having the added maximum perturbation $r_{v-adv}^{(n)}$ [5].

2.3 FixMatch and proposed modifications

We first started out with implementing FixMatch algorithm and then implement our changes on top of it. FixMatch, produces artificial labels by using pseudo-labeling and consistency regularization. Consistency regularization is one of the most important aspects of state-of-the-art SSL algorithms. It exploits the unlabeled data on the assumption that when fed with perturbed versions of same image the model should output similar predictions [6].

Pseudo labels [4] are produced based on weakly augmented images, the weak augmentation is achieved by a flip-and-shift data augmentation method. Only the pseudo-labels with higher probability than the threshold are retained. Strong augmentation is based on RandAugment [1][reference: FixMatch].

FixMatch loss function [6] basically has two cross-entropy loss terms, a supervised loss term (l_s) and unsupervised loss term (l_u).

$$l_s = \frac{1}{B}\sum_{b=1}^{B} H(p_b, p_m(y|\alpha(x_b))) \tag{5}$$

$$l_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} (max(\hat{q_b}) > \tau) H(q_b, p_m(y|A(u_b))) \tag{6}$$

$$l_{FixMatch} = l_s + \lambda_u l_u \tag{7}$$

Where $q_b = p_m(y|\alpha(u_b))$ is models predicted class distribution given a weakly-augmeted version of unlabeled image, $\hat{q_b}) = argmax(q_b)$ is pseudo-label and $A(u_b)$ is output of strongly augmented version of image. λ_u is set to 1 as per authors of FixMatch [6].

In our approach to modify FixMatch algorithm [6] we propose to enforce consistency regularization not only on the final predicted output but also on inter-mediate representations of the weakly and strongly augmented models. We achieved this by extracting activations form bn1 layer of the network (WideResNet) [7] and comparing the similarity between the two using cosine embedding loss [9](l_c).

Cosine embedding loss measures how similar or dissimilar the given two input vectors are, using the cosine distance [9].

$$l_c = \begin{cases} 1 - cos(x_1, x_2), & \text{if } y = 1 \\ max(0, cos(x_1, x_2) - margin), & \text{if } y = -1 \end{cases} \tag{8}$$

The final loss equation for our proposed model is:

$$l_{Total} = l_s + \alpha(t)(l_u + l_c) \tag{9}$$

$\alpha(t)$ is a linear function of epochs similar to task 1.

3 Results

3.1 Pseudo-labeling

Table 1: Pseudo-labeling

Threshold	Cifar10		Cifar100	
	250 labels	4000 labels	2500 labels	10000 labels
0.95	61.87	22.65	63.04	50.9
0.75	59.34	23.7	70.53	51.25
0.6	57.91	23.99	71.47	51.2

Table 1 shows the error rates in % for different combinations of threshold values and number of labeled samples. We tuned learning rate and model depth on cifar-10 labels and found that learning rate of 0.1[verify] and model depth of 34 gave best results we then kept these two parameters constant for other models to get the comparison of performance.

3.2 VAT

Table 2: VAT results

Hyperparameters	Cifar10		Cifar100	
	250 labels	4000 labels	2500 labels	10000 labels
VAT xi	0.6	0.3	0.3	0.3
VAT epsilon	2.0	7.0	7.0	7.0
VAT iteration	1.0	1.0	1.0	1.0
Learning rate	0.01	0.03	0.03	0.03
Error rate	65.29	23.33	74.87	51.27

Table 2 shows VAT results, we observed that by changing the VAT Xi and VAT epsilon values, we were able to control the perturbations on the unlabelled input images. More the Xi and Epsilon values, more were the perturbations. When we varied the xi values between 0.3 and 0.6 and varied the epsilon values between 2-7, we observed low error rates, with the best values captured in the table. When we increased both beyond this range, this lead to higher perturbations that started to increase the error rate of the models. These findings are also consistent with the observations given in the paper [5].

3.3 Fixmatch and proposed modification

Table 3: Fixmatch and FixMatch with cosine embedding loss

Model	Cifar10		Cifar100	
	250 labels	4000 labels	2500 labels	10000 labels
FixMatch	58.55	18.45	63.69	41.26
FixMatch with cosine	55.48	15.44	61.45	40.75

Table 3 shows error rates comparison of Fixmatch model and our modification to it. As we can observe from the table we were not able to exactly reproduce the results of fixmatch [6], but our modification on top of our fixmatch implementation gave better performance consistent in our tests. By this we can conclude that using cosine similarity to enforce consistency helps to get better performance. After tuning, the learning rates 0.001 for cifar10 and 0.01 for cifar100 gave the best results as tabulated above.

We also experimented with producing pseudo-labels based on strongly augmented images after certain epochs but observed that we were not getting expected results. This could be because of strong augmentations not producing highly reliable labels.

References

[1] Ekin D Cubuk et al. “Randaugment: Practical automated data augmentation with a reduced search space”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020, pp. 702–703.

[2] Chengyue Gong, Dilin Wang, and Qiang Liu. “AlphaMatch: Improving Consistency for Semi-supervised Learning with Alpha-divergence”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 13683–13692.

[3] Zijian Hu et al. “SimPLE: Similar Pseudo Label Exploitation for Semi-Supervised Classification”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 15099–15108.

[4] Dong-Hyun Lee et al. “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks”. In: Workshop on challenges in representation learning, ICML. Vol. 3. 2. 2013, p. 896.

[5] Takeru Miyato et al. “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. In: IEEE transactions on pattern analysis and machine intelligence 41.8 (2018), pp. 1979–1993.

[6] Kihyuk Sohn et al. “Fixmatch: Simplifying semi-supervised learning with consistency and confidence”. In: arXiv preprint arXiv:2001.07685 (2020).

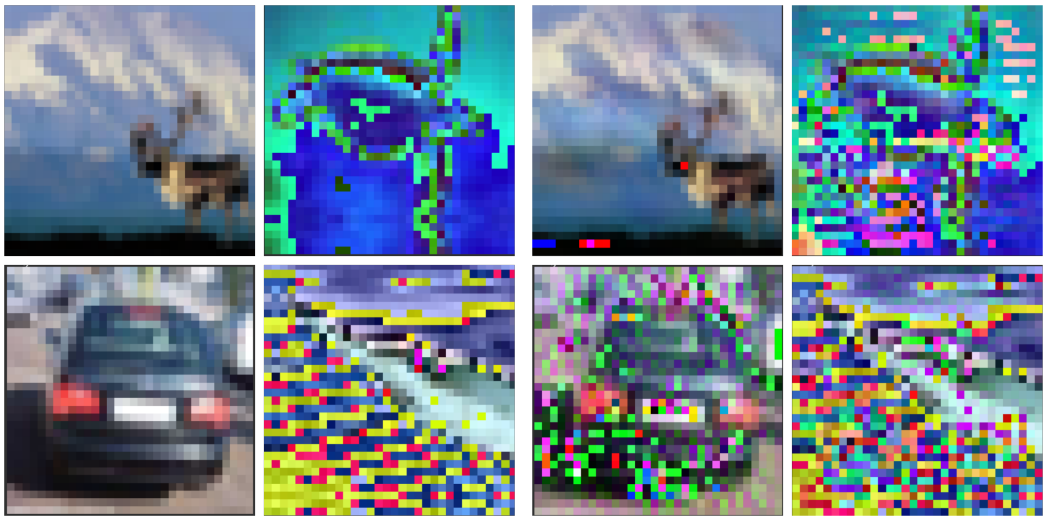
[7] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: BMVC. 2016.

[8] Bowen Zhang et al. “Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling”. In: Advances in Neural Information Processing Systems 34 (2021).

[9] https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html

A Appendix

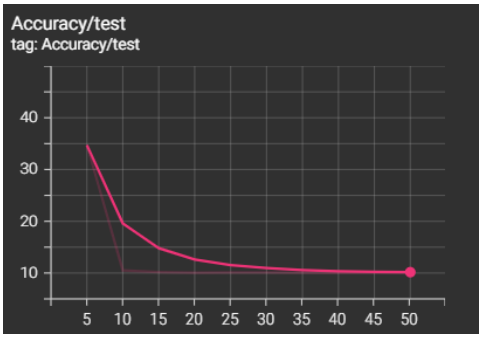
A.1 Figures task 2



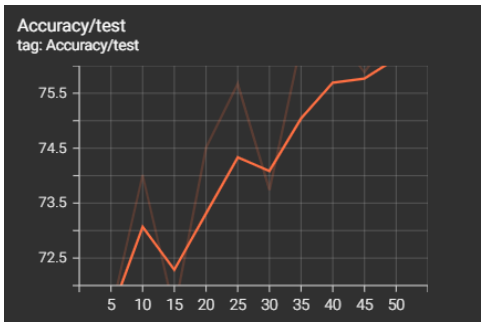
(a) Original images (b) Perturbed images

Figure 1: VAT images

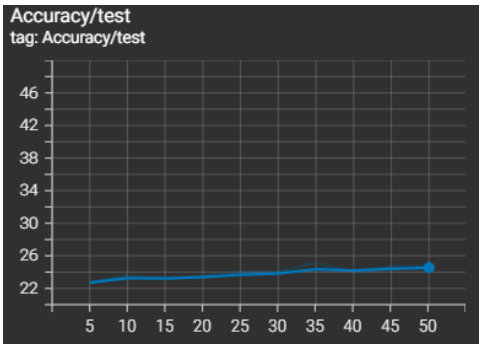
A.2 Figures task 3



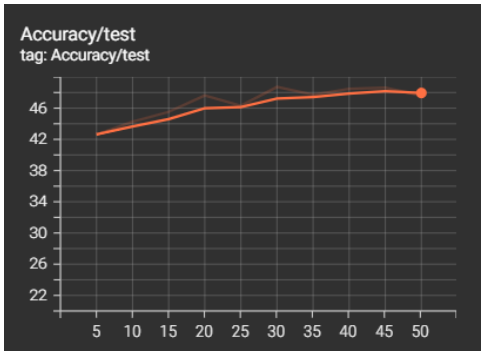
(a) Cifar10 with 250 labeled samples



(b) Cifar10 with 4k labeled samples

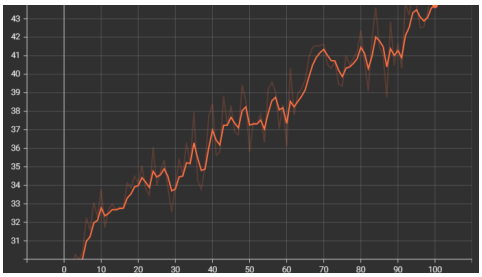


(c) Cifar100 with 2500 labaled samples

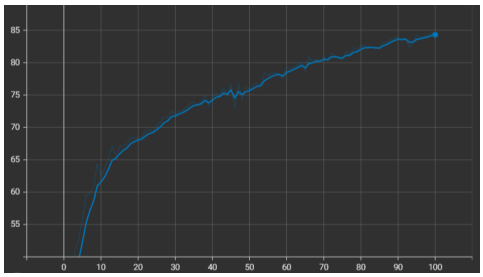


(d) Cifar100 with 10k labeled samples

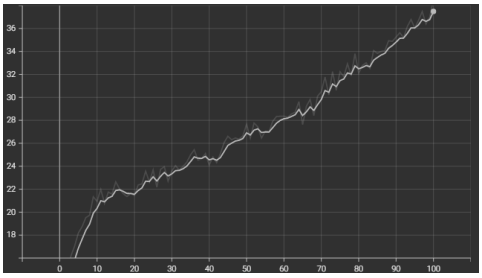
Figure 2: Accuracy vs epoch curves on test data for task 2



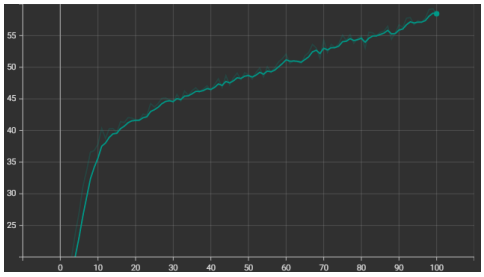
(a) Cifar10 with 250 labeled samples



(b) Cifar10 4k with labeled samples



(c) Cifar100 with 2500 labaled samples



(d) Cifar100 with 10k labeled samples

Figure 3: Accuracy vs epoch curves on test data for task 3