Make Full Use of Unlabeled Data

Qianhong, Zhou. 2017111706

Abstract. I combine self-supervised learning and semi-supervised learning in a largely unlabeled dataset. Firstly use deep clustering [3] and modify its initialization mechanism a little to extract high quality features, then improve the loss function proposed by Lee[4] to do semi-supervised learning. By the combination of self-supervised learning and semi-supervised learning I make full use of the information of the unlabeled data, achieving good prediction accuracy by using only 1% labeled data on CIFAR10.

Keywords: self-supervised learning, semi-supervised learning

1. Introduction

A very important factor for the improvement of convnets is the quality of the dataset. Nowadays, it is very convenient for us to get a huge amount of images as dataset, but putting manual annotations on them costs a lot. Therefore, methods of dealing with real datasets where no or only small part of the data is labeled have been widely studied. Self-supervised learning and semi-supervised learning are main methods in this filed.

Self-supervised learning learns a neural network in a supervised way. But the label is created by the data itself instead of being annotated by humans. Lecun once said that if artificial intelligence is a piece of cake, then most of the cake is self-supervised learning. Partly from his word we can see the importance of self-supervised learning. Many excellent have been done. For example, Zhang et al. train a convnet to colorize gray scale images, using the grayed images as input and the original colored images as output[1]. He holds the belief that when the model is able to restore the grayed image to colored picture, the model learns well about the shape and relative-location of the objects, meaning that the model is a good feature extracter. The similar method is used by Gidaris et al. [2], who rotate a image by 90, 180, 270 degrees and label the original image and the three rotated images as 0,1,2,3 separately. By training a convent to successfully discriminate the rotate degree, the model learns the shape and form of the objects well. Caron et al. put forward a more novel way of self-supervised learning, called deep clustering [3], which is to alternate between clustering of the image descriptors and updating the weights of the convnets by predicting the cluster assignments. The cluster assignments are used as pseudo labels for training. These self-supervised learning are good feature extractors and they can be used to improve the performance on down-stream tasks.

Semi-supervised learning describes the situation where we want to apply a machine learning algorithm in a datasets where unlabeled data are much more than the labeled data. There are many different ways to solve this problem, such as co-training, simple self-training, etc. A more efficient way is to let neural network work in semi-supervised fashion. Just as Lee Does [4], he put train labeled data and unlabeled data together, and use the predict result as the pseudo label to train the unlabeled data, the true label to train the labeled data. The total loss is the weighted sum of the labeled data loss and unlabeled data loss, where the weight of the unlabeled data loss is increasing from 0 to a certain value as epoch increases.

It is a very simple but efficient way and it uses part of the information of the unlabeled data to evaluate the network. Rasmus puts forward ladder networks, also working in semi-supervised fashion and using skip connections to improve past methods [5].

As discussed above, self-supervised learning and semi-supervised learning both can be applied to no or partly labeled dataset. What I am thinking is whether they can be used together to address the problem of predicting the largely unlabeled dataset. For example, use self-supervised learning first to learn powerful features. I assume these features to be powerful because by self-supervised learning the neural network is trained with all data, both labeled and unlabeled, instead of a little bit of labeled data. Then use semi-supervised learning with the extracted features, and the result is expected to be better as the high quality features are used. Finally the experimental results prove that using self-supervised learning and semi-supervised learning in combination is better than using either alone or none of them.

My main achievements in this project is to make a small improvement to one existing self-supervised learning method [3] and one existing semi-supervised learning method [4], and then use them in combination to predict a dataset with only a little bit of labeled data.

In the following sections, I describe my methodology in 2, provide experimental results in 3, and finally conclude in 4.

2. Methodology

2.1 Data

We choose the dataset of CIFAR 10, with 50000 samples and 10 classes. In order to do experiments on dataset with only a little bit of labeled data, only the labels of 500 of the 50000 samples are kept, and the labels of the left 49500 samples are dropped. Our goal is to use the 500 labeled samples and 49500 unlabeled to train a model that can discriminate the unlabeled samples well. It is worth pointing out that my objective is to directly do prediction on the 49500 unlabeled samples instead of using the unlabeled sample to enhance the generalization of the model, so it can be called transductive learning, too.

2.2 Self-Supervised learning for feature extracting: deep clustering

As proposed by Caron [3], we use clustering algorithm to assign samples to different clusters, then the cluster assignments are used as pseudo labels for the model to train. The idea is that if the model can discriminate the clusters assignments ("pseudo-labels") well, then it has certain ability to discriminate the true labels, for we admit that there are some connections between clusters assignments and true labels. Here we use k-means as clustering algorithm while other methods of clustering are also applicable.

Architecture

Let $X = \{x_1, x_2 ... x_N\}$ be training set of N images. We use VGG-16 convnets f_θ as feature extraction networks, and use a parametrized classifier g_w to predict the correct labels on top of the features $f_\theta(x_n)$.

The algorithm for Deep Clustering is:

Step0: Choosing the number of clusters k (usually larger than 10).

Step1: Initialize f_θ and g_w randomly.

Step2: Use $f_\theta(x_n)$ as features to do k-means (before clustering, PCA is used to reduce dimension), record the clustering loss and cluster assignment.

Step3: Use cluster assignment as pseudo labels to train the network and update θ, w .

Step4: Repeat Step 2 and Step3 until the clustering loss and neural network loss are stable.

Avoiding trivial solutions

Just as Caron Does[3], we take some measures to avoid trivial solutions. After each clustering, we resample each cluster so that each of them has $\frac{N}{k}$ samples to eliminate the impact of category imbalance.

A better method to initialize clusters

The initialization of clusters centroids affects the effectiveness of the k-means algorithm. In this problem, I figure out a better way to initialize them. Noticing that we have 500 labeled samples with 10 classes, use the average of samples in each class as the cluster centroid, and then we get 10 cluster centroids. Afterwards, randomly sample the other (k-10) cluster centroids according to the rule of k-means++. Use this method to initialize cluster centroids at each iteration of k-means.

Since we have 500 labeled samples, the cluster centroids generated by them have higher confidence and can guarantee better convergence properties. K-means++ makes sure that all clusters are distributed as uniformly as possible in sample space.

2.3 Semi-supervised learning: use pseudo labels directly for training

It is a very simple method in semi-supervised learning. Following Lee [4], this method uses predicting results directly as the pseudo labels to train the unlabeled data, the true labels to train the labeled data.

The loss function is

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

Where n is the number of labeled data in a mini-batch and n' is that of unlabeled data, $\mathrm{f}_\mathrm{i}^\mathrm{m}$ is the output of m's sample in labeled data, $\mathrm{y}_\mathrm{i}^\mathrm{m}$ is the label of that, $\mathrm{f}'^\mathrm{m}_\mathrm{i}$ is the output of unlabeled data, $\mathrm{y}'^\mathrm{m}_\mathrm{i}$ is the pseudo label (i.e. the class having the maximum predicting probability) of that, $\alpha(t)$ is a coefficient balancing them, representing the weight of the unlabeled data. $\alpha(t)$ is a very important hyper parameter which should be chosen cautiously. When the model is at its first a few epochs, $\alpha(t)$ should be set to 0 because model is naïve at this time and the model predicting results are not reliable. Then $\alpha(t)$ increases as epoch increases. $\alpha(t)$ can not be set too high in case of disturbing the training of labeled data. By setting a proper scheduling of $\alpha(t)$ we can cautiously borrow some information from the unlabeled data to help train our model.

We set $\alpha(t)$ same as Lee does [4],

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t - T_1}{T_2 - T_1} \alpha_f & T_1 \le t < T_2 \\ \alpha_f & T_2 \le t \end{cases} \tag{2}$$

where T1=10, T2=60 and α_f =0.6, meaning that the maximum influence weight the unlabeled loss has is 0.6 after 40 epochs.

But there is a drawback in this model. It takes every unlabeled data into training, even those with little confidence. To be specific, considering an unlabeled data whose predicting probability of its pseudo label is 0.12 (noticing that for a 10 classes classification task the average is 0.1), meaning that model has little belief in this sample. But the algorithm above will force the model to update towards this very likely wrong direction, which may cause the model to learn in a bad track. So I make a small improvement to this

method, modifying the loss function a little.

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'} I(p'_i > \delta)\sum_{i=1}^{C} L(y'^m_i, f'^m_i) \quad (3)$$

Where p'_i is the predicting probability for the pseudo label and δ is the probability threshold, other symbols the same. This added indicative term means that only when then the predicting probability for the pseudo label of an unlabeled sample is large enough can this sample be used to compute loss. In my experiment, probability threshold δ is set to be 0.25.

3. Experiments

3.1 Deep Clustering

We train deep clustering on all 50000 samples in a self-supervised fashion, with no data augmentation.

To monitor the process of deep clustering, we plot clustering loss, NMI and neural network loss in Figure 1.

NMI (Normalized Mutual Information) is used to measure the information shared between two different assignments A and B. If the two assignments A and B are independent, the NMI is equal to 0. If one of them is deterministically predictable from the other, the NMI is equal to 1. Here we compute the NMI between assignments of t-1 steps and assignments of t steps.

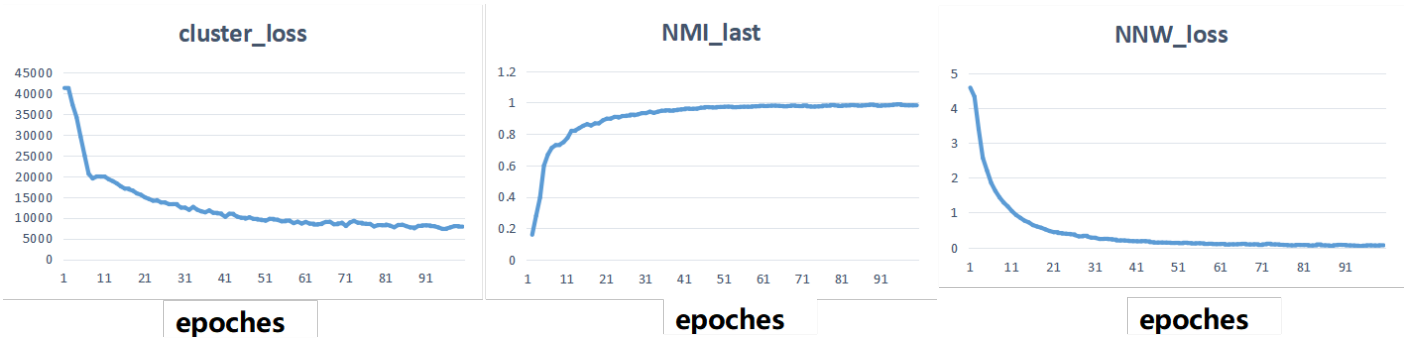

Figure 1: Training stats of deep clustering.

As Figure 1 shows, when epochs increase, the clustering loss and network loss is decreasing and stabilize, while the NMI between t-1 step assignment and t step assignment is steadily approaching to 1, representing that the assignment is not changed after certain epochs. In this example we would choose the model in epoch 40, when the model starts to saturate.

Hyper parameter Selection

Noticing that our goal of self-supervised learning is to extract high quality features, we must have a criterion to judge the quality of the features extracted. Following Zhang et al [6], the method is to train just one linear classifier on top of frozen feature layers, and compute the accuracy on validation set. This method is not hard to understand, because only the features are of high quality can the validation accuracy be high with only a linear classifier. Method having maximum validation accuracy maintains to be best.

As far as our problem, the train dataset is the 500 labeled samples. Firstly we use deep clustering to extract features, and then use the features as inputs, labels as targets to train one linear classifier, finally evaluate the accuracy on 200 validation samples (the validation samples are from CIFAR10-test dataset and has no overlap with our 50000 samples). The validation accuracy is a symbol of the feature extracting ability, so hyper parameters are chosen according to it.

The hyper parameter concerned about are learning rate, number of clusters, k-iter (how much epochs you train your network every time a clustering is run), layer to choose. Layer to choose is the most important parameter. As expected in a self-supervised learning task, shallower layers are naïve and they have not learned too much, while deeper layers are highly task-specified, meaning that they are highly characterized to solve this specific deep clustering task well, so maybe they lack the generation ability to perform well as a feature extractor or be utilized to other downstream task. Therefore, a layer in the middle is preferred as a good feature extractor, and we have to do experiment to find out where it is.

The hyper parameter grid search result is in Figure 2. In this figure No model means that instead of using deep clustering to extract the feature, we just flatten the image into 3072 pixels and train a linear classifier on it. The result of No model helps to illustrate the effectiveness of the feature extraction by deep clustering.

In this figure there is a clear trend that the middle layer is performing well and the two sides layers perform a little poorer, which is consistent with our previous assumption, indicating that the middle layers

work as good features representation.

According to validation accuracy, we choose the model on the left: k-iter=1, learning rate=0.001, layer to choose=5. This model is determined as the ultimate deep clustering model for feature extraction. By the way, the best number of clusters is 100, 10 times as many as actual number of classes, which means some amount of over-segmentation is beneficial, the same as what Caron finds [3]. The detail of choosing the number of clusters is not listed.

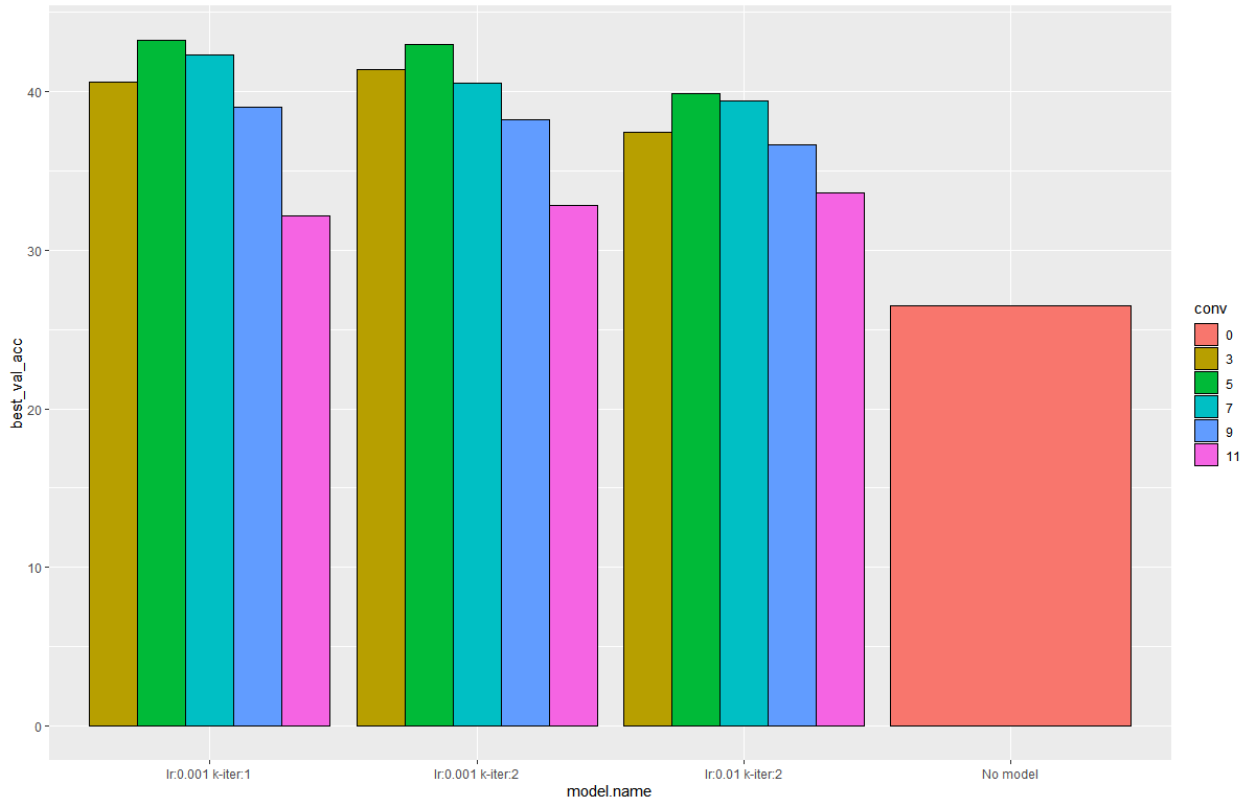

Figure 2: Grid search result for hyper parameter selected, based on one classifier validation accuracy.

3.2 Semi-supervised learning

Now that high quality features have been extracted by deep clustering, we can apply this features in semi-supervised learning afterwards. We expect this semi-supervised learning to perform well because of the good features it uses.

We use the selected best deep clustering model to extract features and freeze these feature layers with no fine-tuning, then adding two classifier layers on top of the convolutional layers (with relu as activation and dropout=0.5). And then use Eq. (3) as loss function to train semi-supervised model. Training stats are

shown in Figure 3.

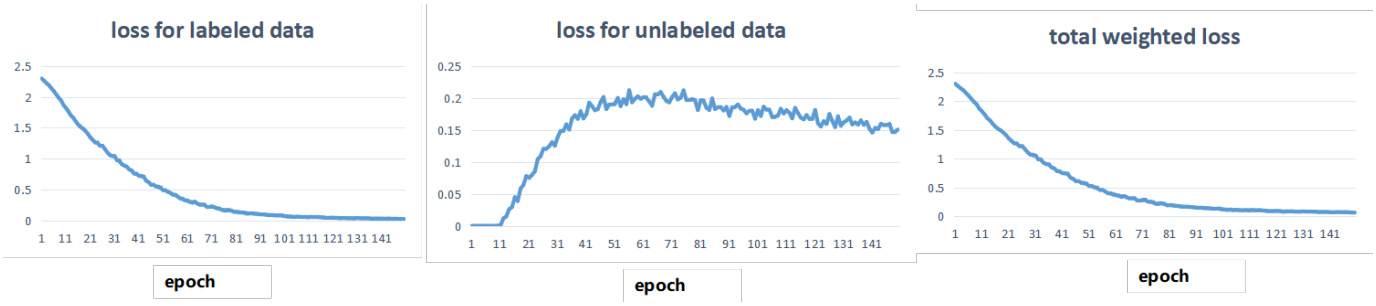

Figure 3: Training stats of semi-supervised learning. Loss of both labeled and unlabeled data are provided.

In this Figure, loss for labeled data and total weighted loss is decreasing. Loss for unlabeled data (this loss is computed according to the pseudo label, with no true label used) is 0 at first because there is no probability of pseudo label passing threshold δ (seeing in Eq. (3)), then it starts to increase because more and more samples are used. Their information is borrowed to train this semi-supervised network. Finally we save the model in epoch 80 when the total weighted loss starts to stabilize. So here is my final model, which is, firstly use convolutional layers trained by deep clustering to extract features, secondly use classifier layers trained by semi-supervised learning to do prediction.

3.3 Results

The following will show the result of my final model. We must remember that we treat the 49500 data as unlabeled samples and train them, just as mentioned above, using NONE of their label information. Now we want to evaluate the power of our model, so we get the 49500 labels back and use them to check how much accuracy can we get on these 49500 pretended unlabeled samples.

To prove the usefulness of my method, I set several models for comparison.

- Null model. Use 500 labeled samples to train VGG-16+2 classifier layers from scratch, tune parameter on 200 samples validation set and predict on 49500 samples.
- Semi-supervised model. Train VGG-16+2 classifier layers from scratch on 50000 samples in a semi-supervised fashion, using Eq. (3) as loss function, and predict on 49500 samples. Deep clustering is NOT used.
- Deep clustering model. Use selected best deep clustering trained before to extract features and add 2 classifier layers on the frozen convolutional layers. Train the 2 classier layers using 500 samples,

tune parameter on 200 samples validation set and predict on 49500 samples. Semi-supervised is NOT used.

- Deep clustering+ semi-supervised model: My proposed model. Use selected best deep clustering trained before to extract features and add 2 classifier layers on the frozen convolutional layers. Train the 2 classier layers on all 50000 samples in a semi-supervised fashion, using Eq. (3) as loss function, and predict on 49500 samples.

The prediction accuracy on the 49500 samples for different models is listed in Table 1.

Table 1: Evaluation of model's power by measuring prediction accuracy on 49500 samples.

Model	prediction accuracy on the 49500 samples
Null model	30.80
Semi-supervised model	32.03
Deep clustering model	43.55
Deep clustering+ semi-supervised model	**45.87**

In this table, My proposed model has an obvious advantage. Both deep clustering and semi-supervised model can enhance the ability of model on largely unlabeled dataset, compared to Null model. And using the combination of them brings the best result, proving the power of my proposed model.

4 Conclusion

Images are not hard to get, but images with labels are. How to use the unlabeled data is an important topic. By using slightly improved deep clustering, a method of self-supervised learning, to do feature extraction and semi-supervised learning, we can make full use of the information of the unlabeled data, and achieve a good result in the transductive learning task: predicting 49500 unlabeled samples in CIFAR10 in an accuracy of 45.87 with only 500 labeled samples.

Rerferences

1. Zhang, R., Isola, P., & Efros, A. A. (2016, October). Colorful image colorization. In European conference on computer vision (pp. 649-666). Springer, Cham.
2. Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728.
3. Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 132-149).
4. Lee, D. H. (2013, June). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Workshop on challenges in representation learning, ICML (Vol. 3, No. 2).
5. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., & Raiko, T. (2015). Semi-supervised learning with ladder networks. In Advances in neural information processing systems (pp. 3546-3554).
6. Zhang, R., Isola, P., & Efros, A. A. (2017). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1058-1067).