Image Classification Based On Semi-supervised Learning

Xiaowei Zhu
xzhu339@wisc.edu

Kyungjin Cho
kcho36@wisc.edu

Yuning Wu
ywu483@wisc.edu

Abstract

In the experiment, our goal is to use semi-supervised learning to complete the task of image classification. There are 200 classes in our dataset, with 50 labeled images and 450 unlabeled images in each class. For the dataset, we deal with the training set, unlabeled data, validation set, and test set separately. At the same time, we use the Pseudo-Label[7] method for semi-supervised learning. When comes to model selection, our neural network model uses the DenseNet model[4]. We got the result with the training of our data on Nvidia 2080Ti, and meantime we saved the model to the personal computer to predict the class label of our test data. The final predicted class labels are saved to the corresponding file.

1. Introduction

Image classification is a task a machine sets the label to unlabeled data and extracts its features in order to sort the image by class. Data scientists usually train the model to a machine for the most economical design with minimal human error. Image classification is an essential need of the deep learning field since millions of data are used for various applications in today's society.

As mentioned above, image classification is not a new topic in the machine learning field; it has been around for the past decade, but still remains to be a steam hot topic. Despite its wide application in multiple fields such as healthcare, education, surveillance, geology, etc., many researchers are still working to improve existing methods and inventing new ones for better performances.

For our image dataset, we employed the Pseudo-Label[7] method proposed by the author of an article titled "Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". This semi-supervised learning method trains the labeled and unlabeled data simultaneously. It works by choosing the classes with "maximum predicted probability" and use them as if they were "true labels". Such method achieves the same effect as Entropy Regularization. By favoring a low-density separation equipped with Denoising Auto-Encoder and Dropout, the author states that it outperforms the conventional methods for semi-supervised learning when applied to the MNIST handwritten digit dataset. A more detailed description of this method can be found in the Related Work section.

Our group examined the random dataset with 200 classes for training our model. Each of them contained 50 images of the same class. The dataset contained images of animals, objects, or trees. We were intrigued to examine the Pseudo-Label and DenseNet combo's performance to the training dataset. After we trained our data, we moved to the validation of our results with 100,000 images.

Software tools we used to train our data contained followings: VGG16, ResNet and, and DenseNet. When it comes to choosing the best model, in the end, we decided to implement DenseNet[4] due to the better performance result that it has yielded. For the Hardware, our experiment was trained on Nvidia 2080Ti. More information about these tools is written in the report Section 4.

2. Related Work

Our idea behind this project comes from Dong-Hyun Lee's article[7] written in 2013 about the efficient method of using semi-supervised learning on deep neural networks. Basically, Dong-Hyun proposes a way to train the dataset with both labeled and unla-

beled data. He mentions that if unlabeled data are considered as if they were true labeled data, then choosing the maximum predicted probability class would yield better accuracy when training the data. It will also provide a simple and robust rule-of-thumb comparing to the other methods such as minimizing the combined loss function of each layer's weights.

In this project, we are going to implement the method of using DenseNet and Pseudo-Label proposed in Dong-Hyun's article to train our datasets of 200 classes with 50 labeled images and 450 unlabeled images. Yet, unlike the article proposed, our group is going to use pseudo-label in the fine-tuning phase.

According to the article, the method of choosing the maximum predicted probability for each pseudo label sample has earned 2nd place in the "Black Box Learning Challenge"[2]. This contest was held in the ICML 2013 workshop which tests the test accuracy and efficiency of classifying the image of Street View House Numbers. Our group, therefore, believes that the image classification based on this semi-supervised learning method yields high accuracy and a robust rule-of-thumb method.

METHOD	100	600	1000	3000
NN	25.81	11.44	10.7	6.04
SVM	23.44	8.85	7.77	4.21
CNN	22.98	7.68	6.45	3.35
TSVM	16.81	6.16	5.38	3.45
DBN-RNCA	-	8.7	-	3.3
EMBEDNN	16.86	5.97	5.73	3.59
CAE	13.47	6.3	4.77	3.22
MTC	12.03	5.13	3.64	**2.57**
DROPNN	21.89	8.57	6.59	3.72
+PL	16.15	5.03	4.30	2.80
+PL+DAE	**10.49**	**4.01**	**3.46**	2.69

Figure 1. Classification Error with 100,600,1000, and 3000 Labeled data. Image source:[7]

Another reason why we decided to refer to the method presented in this article is that it provides significant accuracy when using a pseudo-label method. Referring to Figure 1, Dong-Hyun provides the classification error rate with diverse methods used in 100, 600, 1000, and 3000 labeled data. The +PL(Pseudo-Label) section supports the high accuracy of classification with the second least error rate when there are small sets of data.

3. Proposed Method

Our group used the following methods to yield the result of our project.

3.1. DenseNet

Traditional convolutional feed-forward networks connect the output of the l-th layer as input to the $(l+1)$-th layer[5], which gives rise to the following layer transition:

$$\boldsymbol{x}_l = H_l(\boldsymbol{x}_{l-1}) \tag{1}$$

ResNets[3] add a skip-connection that bypasses the non-linear transformations with an identity function:

$$\boldsymbol{x}_l = H_l(\boldsymbol{x}_{l-1}) + \boldsymbol{x}_{l-1} \tag{2}$$

Then we introduce the DenseNet[4].

Dense connectivity. In order to further improve the information flow between layers, DenseNet propose a different connectivity pattern: As shown in Figure 2,

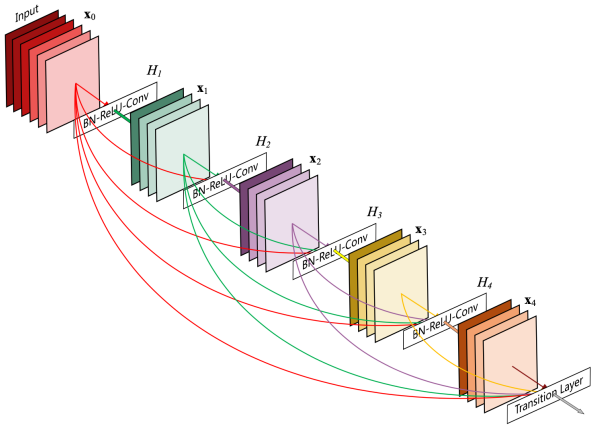

Figure 2. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input. Image source:[4]

the output of the l-th layer not only relate to the $(l-1)$-th layer, but also receives the feature-maps of all preceding layers, the formula can be written as:

$$\boldsymbol{x}_l = H_l([\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}]) \tag{3}$$

where $[\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}]$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, l-1$.

Pooling layers. In DenseNet, it is necessary to connect the feature maps of different layers. Therefore, feature-maps of different layers need to keep the same feature size, which limits the implementation of down-sampling in the network. To facilitate down-sampling, DenseNet is divided into multiple densely connected dense blocks, as shown in Figure 5. In the same dense block, feature size should be kept the same, and transition layers should be set between different dense blocks to implement downsampling.

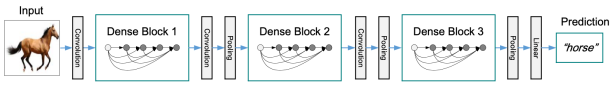

Figure 3. A deep DenseNet with three dense blocks. Image source:[4]

Growth rate. If each function H_l produces k feature-maps, it follows that the l-th has $k_0+k\times(l-1)$ input feature-maps, where k_0 is the number of channels in the input layer. An important difference between DenseNet and existing network architectures is that DenseNet can have very narrow layers. The hyperparameter k is referred to as the growth rate of the network. A relatively small growth rate is sufficient to obtain state-of-the-art results on certain datasets.

Bottle Layers. Although each layer only produces relatively small k output feature-maps. However, since the concatenation between the feature=maps of different layers, it typically has many more inputs. DenseNet introduced a 1×1 bottleneck layer before each convolution to reduce the number of input feature-maps and to improve the computational efficiency.

Compression. To further improve model compactness, DenseNet also reduces the number of feature maps at transition layers. If a dense block contains m feature-maps, the following transition layer will generate $[\theta m]$ output feature-maps, where $0 < \theta \leq 1$ is referred to as the compression factor.

3.2. Pseudo-Label

We adopt the Pseudo-Label method for semi-supervised learning[7]. For unlabeled images, Pseudo-Label are target classes for unlabeled images as if they were true labels. We choose the class with the maximum predicted probability for each unlabeled sample. The formula corresponding to this would be:

$$y_i' = \begin{cases} 1 & \text{if } i = \text{argmax}_{i'}\, f_{i'}(x) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The difference is that we use Pseudo-Label in the fine-tuning phase. The pre-trained model is trained in a supervised fashion with labeled and unlabeled images.

The overall loss function is

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L\left(y_i^m, f_i^m\right) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L\left(y_i'^m, f_i'^m\right) \tag{5}$$

where n and n' are the number of the mini-batch in labeled and unlabeled images respectively, C is the number of classes, f is the network output, y and y' are the label and pseudo-label, $\alpha(t)$ is the balance coefficient, which is a function of the training time t.

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \leq t < T_2 \\ \alpha_f & T_2 \leq t \end{cases} \tag{6}$$

At the beginning, $\alpha(t)$ is 0, and the model is trained with labeled images. When $t \geq T_1$, $\alpha(t)$ starts to increase slowly until it reaches a certain number of iterations and stops increasing. During the whose process, $T_1 = 60, T_2 = 240, \alpha_f = 0.3$.

4. Experiments

4.1. Dataset

In general, the dataset has 200 classes. Class labels are stored in a file called *label.txt*. The format of class labels is **nxxxxxxx**, where x is digit number. Examples of class labels are listed as follows:

Class labels
n02124075
n04067472
n04540053
n04099969
n07749582

Table 1. Name of some class labels

As we are doing the task of semi-supervised learning, the dataset is divided into four parts: training set,

unlabeled from train dataset, validation set, and test set.

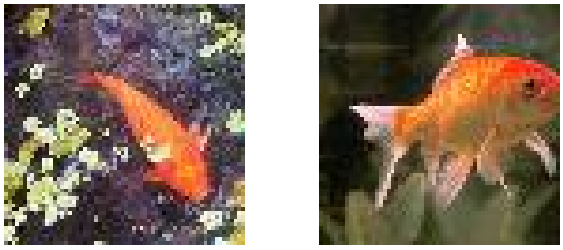

Figure 4. Images in the Dataset(Goldfish)

Training Set. The training set folder contains 200 sub-folders, the name of each subfolder is the name of the class label. Each subfolder contains 50 images. For example, in the subfolder named n01443537, there are 50 goldfish images, as shown in Figure 4.

Unlabeled Set. There are $90000(450 \times 200)$ unlabeled images in the label from the train folder.

Validation Set. The validation set folder contains a folder and a file. The sub folder contains 10000 images and the file called *val annotations.txt* contains the label information of the images. The basic structure is the file is as follows:

Name	Label				
val0.JPEG	n03444034	0	32	44	62
val1.JPEG	n04067472	52	55	57	59
val2.JPEG	n04070727	4	0	60	55
val3.JPEG	n02808440	3	3	63	63
val4.JPEG	n02808440	9	27	63	48
val5.JPEG	n04399382	7	0	59	63

Table 2. Validation images and labels

As shown in table 4.1, the first column is the name of the image, and the second column is the label of the class.

4.2. Software

This experiment mainly uses Python language for programming. In the implementation process, Pytorch, NumPy, and other packages in Python are used.

4.3. Hardware

This experiment was trained on Nvidia 2080Ti. The model was saved on a personal computer for test image class prediction, and the result was saved to the testpredict.txt file

5. Results and Discussion

5.1. Process

5.1.1 Data preprocessing

First, we preprocessed the images. For the training set, unlabeled set, and test set, we adopt three processing strategies.

Training data. For images in the training set, we dealt with them as follows: we first delete the bounding box coordinate file in each class. Then, we use *ImageFolder* to encapsulate them into a dataset. Finally, we use random flip to enhance the dataset, and at the same time, we normalize the images.

Unlabeled data. For the unlabeled images, we create *FlameSet* dataset to read the data.

Validation data. For images in the validation set, we create a *ValiSet* dataset to read the data. And then we use *Valiset* to encapsulate dataset and *vali loader*.

In addition, part of the images in the experiment are not three-channel but single-channel. The solution we adopted for this is to copy the single-channel image into a three-channel image.

5.1.2 Model selection

We tried several networks, such as VGG16, ResNet, and DenseNet. We found that VGG16 has a very simple architecture yet slow in the process because there is a large number of parameters. We also found out that DenseNet yields better performance than ResNet in terms of computation as well as the parameter. Therefore, we decided to use DenseNet to complete the final experiment. Refer to Figure 5 for a more detailed graphic of the comparison between deep neural network architectures and their accuracy.

5.2. Results

The results on the validations sets are shown in Figure 6.

The class label prediction results of our model in the test set images are saved in the file *testpredict.txt* (the file including 10000 prediction class labels, corresponding to 10000 test images respectively). Some of the results are as shown in Figure 9.

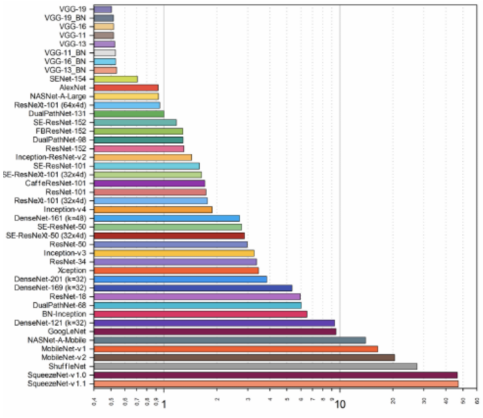

Figure 5. Top 1 accuracy of Deep Neural Network Architecture. Image source:[1]

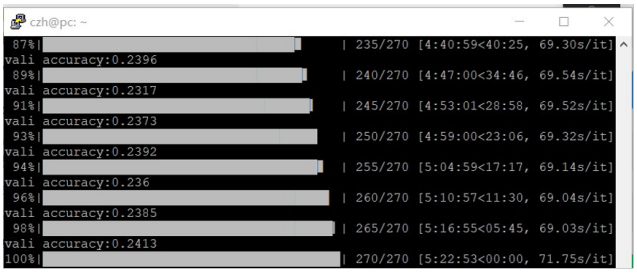

Figure 6. Validation set results

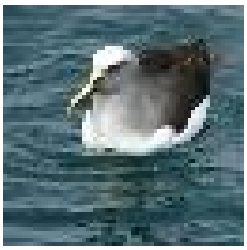

Figure 7. n03444034

Figure 8. n07615774

Figure 9. Images in test set and corresponding class labels

Due to the limitation of the data set (the test set does not have the corresponding true class labels), we can not calculate the accuracy of prediction, which is also what we need to pay attention to when we carry out deep learning practice the next time.

5.3. Discussions

We were able to perform an image classification on the 10,000 images in our data set. When tested using the validation set, we got a 100 percent accuracy at the end of testing. However, as mentioned above, due to lack to true labels in the test data, we were not able to assess the performance of the Pseudo-Label method combined with DenseNet on the test set. Overall, our experiment successfully trained the dataset with the proposed method.

6. Conclusions

In our experiment, we successfully classify images by semi-supervised learning. However, due to the limitation of time and knowledge, we only use the relatively early semi-supervised learning method, Pseudo-Label[7], to train on the data set. However, in the semi-supervised learning image classification task, we can also use other algorithms, such as temporary assembling[6], weight average consistency targets[9], virtual adaptive training[8], which have better performance than the Pseudo-Label method.

7. Acknowledgements

First of all, I would like to thank Professor Sebastian Raschka for arming us with fundamental knowledge in deep learning, which enables us to have enough foundation to comprehend higher-level papers and get to know more different neural networks and training methods. At the same time, I would like to thank my roommate Jiacheng for providing me with the dataset used for this experiment. I would also like to thank Dong Hyun Lee, the author of "Pseudo Label: Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks". Our group refer to his idea of using semi-supervised learning on deep neural networks. Finally, I would like to thank my teammates very much. We worked hard to complete this project.

8. Contributions

Each group member contributed equally to the overall project, including developing the model, writing the project proposal, and preparing a presentation.

Xiaowei contributed to writing the Proposed Method, Results, and Conclusion sections of the project report. Xiaowei helped writing a report and making a visual presentation but mainly focused on training and debugging the model.

Kyungjin contributed to the Related Work, Model Selection, and Contribution sections of the project

report. Kyungjin helped to evaluate the model but mainly focused on making a presentation and writing a project report.

Yuning contributed to the Introduction and Discussion sections of the project report. Yuning helped to tune the model but mainly focused on making a presentation and writing a project report.

References

[1] L. Celona et al. Benchmark analysis of representative deep neural network architectures. volume 4, 2018.

[2] I. J. Goodfellow1, D. Erhan, et al. Challenges in representation learning: A report on three machine learning contests. volume 64, 2013.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[6] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[7] D.-H. 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*, volume 3, 2013.

[8] T. Miyato, S.-i. 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(8):1979–1993, 2018.

[9] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017.