Reproducibility report formatting instructions for ML Reproducibility Challenge 2020

Yichen Huang, Xinyu Wang, Yuhe Fan

Reproducibility Summary

1 Introduction

In this project, we selected a CNN+SVM paper: Deep Learning using Linear Support Vector Machines. We tried to reproduce the result obtained by following the exact method mentioned in this paper. We used online open-sourced implementation of the CNN+softmax model and CNN+SVM model and plugged the hyper-parameters given in the paper into the models. Finally, we found that SVM as the top layer of deep learning performs better than softmax as the top layer.

2 Scope of reproducibility

In our work, we focused on comparing the performance of softmax as the top layer and SVM as the top layer and discuss whether switching from softmax to SVMs would be useful for classification.

The paper claims that DLSVM performs better than softmax by experimenting on three datasets and comparing the performance of the two models. In this project, we will reproduce one of the experiments on the MNIST dataset which is a standard and fundamental image classification dataset. Our result in section 4 will support the conclusion drawn in the paper, that is DLSVM performs better and is useful.

3 Methodology

3.1 Model descriptions

3.1.1 Softmax

For deep learning classification, it is standard to use softmax as the top layer.

$$softmax(z)_c = \frac{e^{z_c}}{\sum_{c'}^{C} e^z c'}$$

In this project, we used a CNN-softmax model with 2 convolution layer and 2 hidden layer which is followed by a softmax layer. The training set has 60000 figures and the testing set has 10000 figures. The CNN-softmax model has 2 convolutional layers and 2 hidden layers. Each hidden layer has 512 units. The top layer of this model is softmax. In our experiment, the models are trained over 400 epochs.

3.1.2 SVM

SVM is used for finding the optimal decision boundary by optimizing the following equation.

$$\sum max(0, 1 - y^n(w^T x^{(n)} + w_0 + \frac{1}{2}\|w\|_2^2)$$

In the class, we introduced the SVM for binary classification and mentioned that it can also be used for multiclassification. The paper pointed out that for the k class problem, we can take out one class and the other classes as one big class and we will train k linear SVM models, which is called the one-vs-rest approach (Vap- Nik, 1995).

In this project, we used a similar model in 3.1.1. The only difference is the top layer is replaced with a multiclass SVM layer.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

3.2 Datasets

In this project, we used the MNIST dataset. This is a handwritten digit image classification dataset with 10 categories (0-9). Each image is represented with 28*28 pixels. It has 60,000 training examples and 10,000 test cases. We applied PCA dimension reduction to the origin data and reduced the dimension from 784(28*28) to 70.

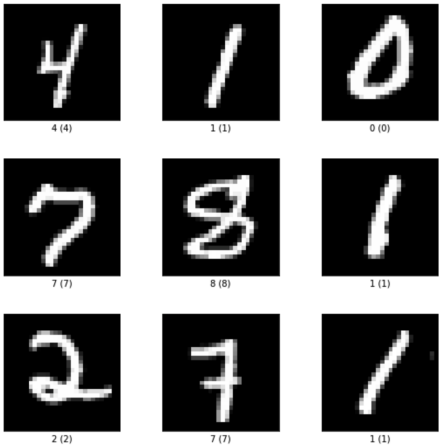

Figure 1: data preview

3.3 Hyperparameters

The hyper-parameters are given in the paper in detail. The batch size is set to 200, the learning rate is set to 0.1, the L2 weight cost on the softmax layer is set to 0.001, and two hidden layers of 512 units each. We followed the original hyper-parameters in the paper except for the learning rate. We also tried to use other hyper-parameters, but the performance got worse except for the learning rate. So we decided to use the original hyper-parameter and replace the learning rate with 10e-3.

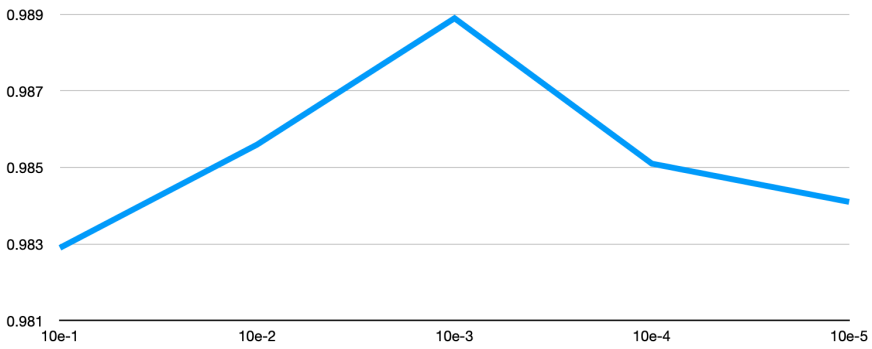

Figure 2: data preview

3.4 Experimental setup and code

Because the model we used was implemented several years ago, the model was implemented with tensorflow v1.15.4 instead or version later than 2.0. Therefore, the user should install exactly tensorflow 1.15.4. A python 3.7 or lower version is needed. The code is attached in the .zip file. Detailed instructions can be found in the README file in the code.zip file.

3.5 Computational requirements

The experiments were conducted on a laptop computer with 2.3GHz 8-core Intel Core i9, Turbo Boost up to 4.8GHz, with 16MB shared L3 cache, and AMD Radeon Pro 5500M 4 GB GPU. However, the creator of this model wrote that his experiments were conducted on a laptop computer with Intel Core(TM) i5-6300HQ CPU @ 2.30GHz x 4, 16GB of DDR3 RAM, and NVIDIA GeForce GTX 960M 4GB DDR5 GPU. His experiment was similar to ours but have much more epochs. Therefore our hardware is way exceeding the requirement. The GPU hour spent is about 0.15 GPU hours which is small. Therefore, these experiments conducted in the paper do not require many resources. For the softmax model with batch size 200, the average running time is 47.84 seconds. For the SVM model with batch size 200, the average running time is 46.38 seconds.

4 Results

4.1 Results reproducing original paper

We noticed that in the paper, the author used only one trial to compare the performance of the softmax layer and SVM layer. To avoid serendipity we thought training the model multiple times and using their average performance would be better.

trials	softmax layer	svm layer
1	0.99260	0.99339
2	0.99320	0.99390
3	0.99299	0.99370
3	0.99239	0.99440
3	0.99269	0.99360
avg	0.99277	0.99380

Table 1: Accuracy of the softmax model and DSVM model

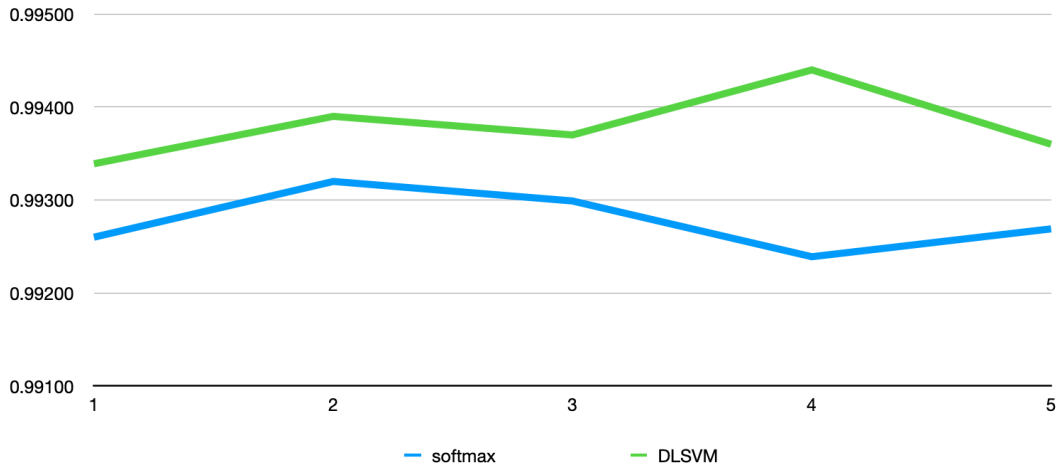

Figure 3: Visualize of the the accuracy of two models

The average accuracy of the softmax model is 0.99277 and the average accuracy of the DLSVM model is 0.99380. We can calculate the error rate for the softmax model and DLSVM model as 0.723% and 0.620% respectively. The accuracy of DLSVM is about 0.1% better than softmax which is similar to the original result in the paper. From figure 3, we can also see clearly that DLSVM performs better than softmax generally in this dataset.

4.2 Results beyond original paper

When tuning the hyper-parameters, we found that the optimal learning rate is 10e-3 which is not identical to the value given by the paper (10e-1) and the paper did not mention how the value is obtained.

5 Discussion

5.1 softmax vs. SVM

The difference between softmax and multiclass SVMs is that the softmax layer minimizes cross-entropy or maximizes the log-likelihood, while SVMs simply try to find the maximum margin between data points of different classes.

From the experiment we conducted, our result shows that as the top layer of CNN, the SVM layer has better accuracy than the softmax layer. Moreover, we recorded the total running time (training and testing), and it shows that the running time of these two models is similar. From the code, we can see that the implementation of both models has a similar level of difficulty. Therefore, we think using SVM as the top layer of CNN would be better.

5.2 What was easy

Give your judgement of what was easy to reproduce. Perhaps the author's code is clearly written and easy to run, so it was easy to verify the majority of original claims. Or, the explanation in the paper was really easy to follow and put into code. The datasets used in the paper are standard datasets which are easy to load, train and test.

5.3 What was difficult

There is no official code for the paper and there is almost no related open-source code. Therefore, it is difficult to find similar models. Finally, we found a model built on the basis of this article but hyper-parameters are different and were implemented a few years ago. Over the years, the package imported in the code has been updated many generations and does not support the latest python version 3.8-3.10, which means we need to create a python 3.7 environment for this code that is several years old and install a 1. x version of tensorflow which we are not familiar with. Moreover, tensorflow 1. x and tensorflow 2.x have lots of different methods not shared with each other which makes it harder to use the model.

5.4 Future investigation

5.4.1 Datasets

In our experiment, we compared the performance of the softmax layer and SVM layer in MNIST. Future research could compare their performance on various datasets and determine whether a type of dataset would affect their performance of them and whether softmax would perform better than SVM in some other datasets.

5.5 Reason behind the experiment result

Our result shows that the SVM layer performs better than the softmax layer. Future work would be exploring where the superiority of SVM is obtained and why we obtained such a result.

5.6 Other multiclass SVM approach

In this project, we used one-vs-rest approach (Vap- nik, 1995) as the approach of multiclass SVM. We could try a different kinds of approaches (e.g. Hsu Lin (2002)) and compare their performance.

References

Deep Learning using Linear Support Vector Machines, Yichuan Tang, 2013

Hsu, Chih-Wei and Lin, Chih-Jen. A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks, 13(2):415–425, 2002.

Vapnik, V. N. The nature of statistical learning theory. Springer, New York, 1995.

6 Statement of Contributions

All works are distributed equally with group participation and discussion.