Comp 551 Reproducibility Challenge: CNN + SVM

Sarah Zhou
shi.q.zhou@mail.mcgill.ca
Ziqing Liang
ziqing.liang@mail.mcgill.ca
Zuchan Fang
zuchan.fang@mail.mcgill.ca

Reproducibility Summary

Scope of Reproducibility

This study mainly refers to the previously published research Tang (2015). This previous work, published in 2015, mainly challenged the routine operation: the Softmax function is generally the classifier used at the last layer of the deep learning networks. As the authors demonstrated a small but consistent advantage of replacing the softmax layer with a linear support vector machine, they concluded that using L2-SVM showed that by simply replacing softmax with linear SVM gave significant gains on certain deep learning datasets. In this study, we selected a subset of experiments of the original work for reproducing, aiming to confirm the same subject.

Methodology

In this study, we mainly followed the idea of the previous work done by D.Tang Tang (2015), and investigated the performance of two combined CNN models (CNN-Softmax and CNN-SVM) on classifying images to the appropriate classes on the classic hand-written digit dataset: the MNIST dataset. of Standards and database Since there is no publicly available code for the previous work, in the specific implementation, we partially used the code of another work Agarap (2017) that is also based on the same previous work. Briefly, the model we used had some structural differences with the one described in the original work in detail, and we have also made some additional attempts in terms of hyperparameters tuning.

Results

Through our experiments, we verified that CNN-SVM is indeed a very strong image processing model. Compared with the traditional CNN-Softmax model, the training time of the CNN-SVM model took shorter. However, based on our current experiments, we believe that the test accuracy of CNN-SVM and CNN-Softmax are very close to one another, with accuracy of 0.993 and 0.991 respectively. That is, the former does not significantly improve the classification accuracy compared to the latter. This conclusion seems to be different from the original work's conclusion. The reason may come from the deviation from the specific implementation method and the limitation of data set to which the conclusion is applicable. The specific analysis will be discussed at the end of this study.

What was easy

In this study, we found that the theoretical logic behind the author's methodology was easily understandable. We were able to relate the technicalities mentioned to what was learned in the frame of this course. Additionally, the chosen dataset, MINST, is a easy dataset to manipulate.

What was difficult

We found, however, that the some specific implementations of the model used in the original work was not fully described and the original code was not given. Consequently, in the experimental process, we need to add some specific implementation processes based on our understanding of the original theory, including the adjustment of CNN layers, the method of weight update and optimization, etc. In addition, we found that for image data sets of different complexity, the CNN model needs to be adjusted accordingly to the size of the input data. In this study, we have also made full efforts to adjust the CNN model.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

Convolutional neural networks (CNNs), as one of the most powerful deep learning models, is widely employed when classifying the image data by mapping between raw image pixels and their class scores. This kind of model is generally composed of a set of feature recognition layers and a classification layer, which is most often implemented by softmax. In this study, we mainly reproduced the previous work Tang (2015), and investigated the use of combined model method in the last layer of CNN to replace softmax with a support vector machine (SVM) as the non-linear classifier.

2 Scope of reproducibility

Consistent with the purpose of the original work, our study aimed to study whether replacing a traditional softmax top layer with a linear SVM layer will be beneficial for the accuracy of CNN's deep learning capabilities for image data. In the original work, Dr. Tang made the following claims on the above subject through demonstrating experiments on 3 distinguish image datasets: MNIST, CIFAR-10, and the ICML 2013 Representation LearningWorkshop's face expression recognition challenge.

- Claim 1: By simply replacing softmax with linear SVM as the top layer gives significant gains on the 3 specific popular deep learning datasets.
- Claim 2: Performance gain is largely due to the superior regularization effects of the SVM loss function, the loss function of SVM makes the loss of the model more stable.

In terms of specific experimental methods, the original work carried out a more sophisticated model and a series of more complicated experiments. Our study this time is also aimed at verifying the above claims, we selected one of the data sets used in the original work, the MNIST dataset, for experimentation on comparing the two combined models. Therefore, our final conclusions are only applicable to the MNIST data set, which is hereby explained. In addition, although our model structure is based on the same principles as the original, there are actually some structural differences in the specific implementation, so there may be some contrast with the original results.

3 Methodology

In this study, all the deep learning models are implemented by the Tensorflow library in python. Since the code of the original work is not public avaliable, the code we mainly implemented in this study was largely inspired by the public code of another studyAgarap (2017). The pre-constructed models and starter code used are from Agarap's Github repository https://github.com/AFAgarap/cnn-svm.

3.1 CNN-SVM Model descriptions

Since the image classification task on MNIST dataset is under the supervised learning category, the CNN model is then applied to learn feature vector representations for the labeled training data. As for the CNN-SVM models, the learned feature vectors are fed to train the SVM classifier at the last step. Such a combined model is expected to combine the advantages of the two parts. Particularly, CNN model on feature learning and SVM model on efficient non-linear classification.

3.1.1 CNN

Convolutional Neural Network (CNN) as one of most powerful deep feed-forward artificial neural networks is widely used in image classification. A typical CNN model usually consists of input layer, convolutional layers, pooling layers, fully connected layers, and a top-level classification layer. The specific CNN architectures of this study are described as follows: We constructed our model with two sets of convolutional filtering + pooling stages, followed by a fully connected layer with 1024 hidden neurons hidden units to meet our needs for the input data. The activation function we use in each convolutional layer is ReLU. At the same time, we set the size of each convolutional layer to $5 \times 5 \times 1$ (Among them, 5 pixels for width and height, and 1 stride for the images are in grayscale). Since a dot product is being calculated in the convolution layer, the output as a 2-D map would be fed into the pooling layer for the next steps. Therefore, we constructed the size of the pooling layers as 2 x 2 x 1. The last layer of the model is usually used to calculate the loss function. We replace the traditional Softmax function (cross entropy loss) with L2-SVM (marginal-based loss) as planned.

3.1.2 SVM: L2-SVM

Based on the conditions given in the original work, in this study we directly use L2-SVM to experiment. The support vector machine (SVM) was developed by VapnikCortes and Vapnik (1995) for binary classification, The essence of this model is to find the optimal hyperplane function to distinguish two different groups. L1-SVM is known as the model learns the parameters w by solving an optimization equation. However, L1-SVM is not differentiable. A popular variation of it is known as the L2-SVM which can minimize the squared hinge loss:

$$min\frac{1}{p}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{p} max\left(0, 1 - y_i'(\mathbf{w}^T\mathbf{x}_i + b)\right)^2$$

Unlike the hinge loss of a standard SVM, L2-SVM is differentiable and imposes a bigger (quadratic vs. linear) loss for points which violate the margin. For example. to predict the class label of a test data x:

$$\arg\max_t(\mathbf{w}^\mathsf{T}\mathbf{x})t$$

3.1.3 Optimization and Regularization

In addition to supplementing some of the implementation steps the original work did not mention, we encountered some bottlenecks when implementing some of the specific optimization and regularization methods mentioned in the original work. In order to solve the problems, we finally chose to replace similar methods to continue implementation. In this regard, we replaced the stochastic gradient descent (SGD) in the original work with Adam as the optimization method. Furthermore, we implemented dropout to deal with the problem of overfitting instead of adding the Gaussian noise terms as described in the original work.

3.2 Datasets

The Modified National Institute of Standards and Technology database (MNIST) is a large database of handwritten digits that is commonly used for training various image processing systems. In this study, the MNIST dataset is retrieved directly from Keras library in Python. The MNIST database contains 60,000 training images and 10,000 testing images.

The MNIST data set consists of images of hand written digits belongs to ten labeled classes from 0 to 9. Each image data is composed of 785 numbers, whose value is between 0 and 255. The first number labels the digit represented by the image data (ranging from integer 0-9), and the following 784 numbers can form a 28×28 matrix, and each value corresponds to the pixel value of the pixel at the position, thus forming a picture with a pixel of 28×28.

3.3 Hyperparameters

In this study, The hyper-parameters used for CNN-Softmax and CNN-SVM models are Batch Size, Dropout rate, Learning rate and Epochs. Different from the cross-validation experiments carried out in the original work for the hyper-parameters tuning, we investigated some additional attempts by assigning random values to the hyper-parameters manually. The best combination of the value assignment that resulted in the best model accuracy is: Batch size = 128, Dropout rate = 0.5, learning rate = 0.001, epoch = 10000.

3.4 Experimental setup and code

The author used a simple fully connected model by first performing PCA from 784 dimensions down to 70 dimensions. Two hidden layers of 512 units each is followed by a softmax or a L2-SVM. The data is then divided up into 300 minibatches of 200 samples each. They trained using stochastic gradient descent with momentum on these 300 minibatches for over 400 epochs, totaling 120K weight updates.Tang (2015)

On our side, PCA was not performed. The original data was fitted into the model without normalization nor other processing techniques. The data is then divided up into 300 minibatches of 200 samples each. We trained over 1000 epochs as a first experiemnt, and over 10 000 epochs as a second one.

In this study, 3 measurements were taken to evaluate the experiment results: training accuracy, training loss and test accuracy.

The pre-constructed models and starter code used are from Agarap's Github repository https://github.com/AFAgarap/cnn-svm.

3.5 Computational requirements

In this study, all the experiments were conducted on a laptop computer via Google Colab platform with a standard RAM setting. In terms of the running time, as for per 1000 epochs, the average running time for the experiment with CNN-Softmax model is arounf 8 minutes and 48 seconds, whereas the experiment with CNN-SVM model took shorter, 7 minutes and 13 seconds.

4 Results

In this study, we ended up getting the optimal test accuracies of 0.993 and 0.991 for the CNN-Softmax and CNN-SVM respectively. Which in general, these two numbers are very close in value to the original results. However, contrary to the Claim 1 in the original work, regarding to the specific data set of MNIST, the accuracy of CNN-Softmax model is slightly higher than that of CNN-SVM, and there appears no significant gains when switching the two models. Finally, observing the training loss over 10 000 epochs, the CNN-SVM model had less fluctuations than CNN-Softmax, which aligns with Claim 2 in the original work.

4.1 Results reproducing original paper

4.1.1 Result 1

The test accuracies of the two models on MNIST dataset were very similar. As shown in Table 1 & 2, after a 1000 epochs, the accracies differ by 0.004 and after 10 000 epochs, they differ by 0.002. Even though both models have achieved fairly high accuracy, the accuracy difference between the two models is not large (within 0.01). This phenomenon shows that in our experiments , Mnist data set can be well classified in both Softmax and SVM layers, and the outstanding advantages of the two models are not obvious. Additional to that, in both cases, the L2 regularized CNN-SVM model performed slightly poorer than the CNN-Softmax model, which actually contrast the Claim 1 in the original work.

CNN Softmax	CNN SVM
0.993	0.991

Table 1: Test accuracy after 10 000 epochs

CNN Softmax	CNN SVM
0.989	0.985

Table 2: Test accuracy after 1000 epochs

Steps	CNN Softmax		CNN SVM	
	Train accuracy	Training loss	Train accuracy	Training loss
0	0.140	9.869	0.115	27.010
100	0.955	0.217	0.935	0.100
200	0.985	0.079	0.965	0.068
300	0.970	0.104	0.980	0.028
400	0.990	0.069	0.975	0.045
500	0.985	0.086	0.960	0.044
600	0.995	0.042	0.975	0.033
700	0.985	0.055	0.985	0.030
800	1.0	0.035	0.990	0.022
900	1.0	0.028	0.990	0.016

Table 3: Reported results for models - with learning rate 0.001 and 1000 epochs

4.1.2 Result 2

From a second experiment, the learning rate was kept at 0.001, batch size at 200, but we changed the epoch number to 10 000. We notice more fluctuations in the training loss of the CNN-Softmax model. For instance, at epochs 7400, 7800, 8800, 9200, 9300 and 9900, the loss was around 100 times smaller than at other epochs.

Therefore, it is reasonable to conclude that the CNN-SVM model is more consistent and had more consistent values compared to the CNN-Softmax model, which aligns well with the Claim 2 given in the original work.

```
step: 7300, training accuracy : 1.0, training loss : 0.00036133622052147985
step: 7400, training accuracy : 1.0, training loss : 9.858342309598811e-06
step: 7500, training accuracy : 1.0, training loss : 0.012674912810325623
step: 7600, training accuracy : 1.0, training loss : 0.0021897703409194946
step: 7700, training accuracy : 1.0, training loss : 0.0009456512634642422
step: 7800, training accuracy : 1.0, training loss : 6.729633605573326e-05
step: 7900, training accuracy : 1.0, training loss : 0.00898569822311401 4
step: 8000, training accuracy : 1.0, training loss : 0.005524829030036926
step: 8100, training accuracy : 1.0, training loss : 0.0010821862379088998
step: 8200, training accuracy : 1.0, training loss : 0.01287543773651123
step: 8300, training accuracy : 1.0, training loss : 0.0032967650331556797
step: 8400, training accuracy : 1.0, training loss : 0.0007473516743630171
step: 8500, training accuracy : 1.0, training loss : 0.002015632577240467
step: 8600, training accuracy : 1.0, training loss : 0.003560899291187525
step: 8700, training accuracy : 0.9921875, training loss : 0.001566399703733623
step: 8800, training accuracy : 1.0, training loss : 3.1692888174831308 4e-05
step: 8900, training accuracy : 1.0, training loss : 0.0072323516942560 67
step: 9000, training accuracy : 1.0, training loss : 0.0028804475441575 05
step: 9100, training accuracy : 1.0, training loss : 0.0011018993100151 42
step: 9200, training accuracy : 1.0, training loss : 3.3900993003044 28e-05
step: 9300, training accuracy : 1.0, training loss : 8.4466126281768 08e-05
step: 9400, training accuracy : 1.0, training loss : 1.5417739632539 45e-05
step: 9500, training accuracy : 1.0, training loss : 0.0066229873336851 6
step: 9600, training accuracy : 1.0, training loss : 0.0001368117809761 3156
step: 9700, training accuracy : 1.0, training loss : 0.0071548903360962 87
step: 9800, training accuracy : 1.0, training loss : 0.0011929182801395655
step: 9900, training accuracy : 1.0, training loss : 6.242115887289401e-06
```

Figure 1: CNN-Softmax training output

```
step: 7300, training accuracy : 1.0, training loss : 0.003659917274489999
step: 7400, training accuracy : 1.0, training loss : 0.004582567606121302
step: 7500, training accuracy : 1.0, training loss : 0.0028887498192489147
step: 7600, training accuracy : 1.0, training loss : 0.00471335556358099
step: 7700, training accuracy : 1.0, training loss : 0.007533218711614609
step: 7800, training accuracy : 0.9921875, training loss : 0.005538024939596653
step: 7900, training accuracy : 1.0, training loss : 0.004264528863132
step: 8000, training accuracy : 1.0, training loss : 0.002693312242627144
step: 8100, training accuracy : 1.0, training loss : 0.003672798862680793
step: 8200, training accuracy : 1.0, training loss : 0.010689831338822842
step: 8300, training accuracy : 1.0, training loss : 0.002320700092241168
step: 8400, training accuracy : 1.0, training loss : 0.004341408610343933
step: 8500, training accuracy : 1.0, training loss : 0.00633067823946476
step: 8600, training accuracy : 1.0, training loss : 0.007152989506721497
step: 8700, training accuracy : 1.0, training loss : 0.004314718768000603
step: 8800, training accuracy : 1.0, training loss : 0.006126332096755505
step: 8900, training accuracy : 1.0, training loss : 0.004209278151392937
step: 9000, training accuracy : 1.0, training loss : 0.00427647028118372
step: 9100, training accuracy : 1.0, training loss : 0.003210180439054966
step: 9200, training accuracy : 1.0, training loss : 0.0025035582948476076
step: 9300, training accuracy : 1.0, training loss : 0.0031589623540639877
step: 9400, training accuracy : 1.0, training loss : 0.004203714430332184
step: 9500, training accuracy : 1.0, training loss : 0.0025754068046808243
step: 9600, training accuracy : 1.0, training loss : 0.002507542259991169
step: 9700, training accuracy : 1.0, training loss : 0.005263316445052624
step: 9800, training accuracy : 1.0, training loss : 0.0029540383256971836
step: 9900, training accuracy : 1.0, training loss : 0.0017149108462035656
```

Figure 2: CNN-SVM training output

4.2 Results beyond original paper: hyperparameters

In this study, we did attempts on trying out different sets of hyper-parameters. We have tried batch sizes of 128 and 200.

As for the learning rate, we tuned it based on the code by Agarap. A trial learning rate of 0.01 would make the CNN-SVM model fall around 10%. 0.01 was too large and caused the model to converge quickly to a suboptimal solution. The final learning rate was kept to 0.001, same as the one provided by Agarap.

The epoch number provided by Agarap was 10 000, which reached 99% accuracy for noth models. The trial done by us at 1000 epochs found that this change only decreased the accuracy by around 0.4%. This is worth mentioning since the epochs were altered by a factor of 10. This is mainly due to the easy dataset.

5 Discussion

Given the fact that our our result 1 contrast with the original claim 1, and our result 2 aligns well with the claim 2, we finally came to the following discussion.

In the original paper, some of the specific implementation methodology wasn't given. For instance, the author mentioned a weight initialization was performed without saying in which specific technique. We performed a random initialization on a truncated normal distribution. This means that the generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked. API

As we have learned through the course, weight initialization can have a great impact on the observed accuracy. This may be a reason for the discrepancy observed.

In addition, Based on past experience, we suspect that some preprocessing of the data set may also have a great impact on the final result. In the original work, the author performed principal component analysis on the data set to reduce the dimensionality of the data. In our model, we directly feed the original data to the model. We have reason to suspect that retaining more important dimensions can classify features that are more suitable for top-level classifiers.

At the same time, we believe that the complexity of the data set is likely to directly affect the learning quality of CNN. Through previous learning, we know that the MNIST data set is relatively simple and standardized image data. Therefore, an applicable future extension direction of this reproducibility study can be to start with more complex data sets, such as the CIFAR-10 ang the face expression dataset as mentioned in the original work.

Of course, there are many other extension directions. Among them, we are also very interested in other SVM implementation methods. In our implementation, we basically follow the method mentioned in the original work. For 10 classification groups, 10 binary linear SVM classifiers are used for separate processing. However, we believe that this approach is inefficient. In the real world, some scholars also conducted research on the implementation of SVM for multiple classifications, such as the previous work Hsu and Lin (2002). These more efficient implementation methods can be used as directions of further research.

5.1 What was easy

In this study, we found that the theoretical logic behind the author's methodology was easily understandable. We were able to relate the technicalities mentioned to what was learned in the frame of this course. It was easy to verify the majority of original claims about the MNIST dataset. The experiments on MNIST matched the experiments in the paper.

5.2 What was difficult

It was difficult to use the model with other datasets than MNIST as it involves a new construction of the layers and a new transformation of the dataset. For instance, we were not fully able to reproduce the model on CIFAR-10 a with each images of 32x32 and 3 channels for RGB (1x32x32x3) because of the technicalities inside the tensorflow graphs. The transformation requires advanced knowledge of tensorflow graphs.

References

Abien Fred Agarap. An architecture combining convolutional neural network (cnn) and linear support vector machine (svm) for image classification.

Abien Fred Agarap. 2017. An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification.

Tensorflow API. tf.random.truncated$_normal$.

C. Cortes and V. Vapnik. 1995. Support vector networks. *Machine Learning*, 20:273–297.

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

The Modified National Institute of Standards and Technology database. [link].

Yichuan Tang. 2015. Deep learning using linear support vector machines.