COMP551 Project 4

Reproducibility in ML

Xiaoman Sun, Chuqi Wang, Yubai Zhang

17th April 2022

Abstract

Random Forest[5] and Convolutional neural network(CNN) are the two widely used machine learning algorithms which has become a fundamental part in modern scientific methodology. In this work, we integrate different machine learning approaches, including random forest classifiers, convolutional neural networks (CNN) with support vector machine (SVMs) for classification of digit images in the MNIST datasets. The purpose of the study is to compare the performance of the Random Forest model and the CNN model which is replaced the traditional softmax layer with a linear support vector machine on various training data size with their best hyper-parameters or conditions and to explore meaningful extensions of the models to improve the model performance. Our findings suggested that CNN-SVM model have a better performance on MINIST dataset with an accuracy of 99.33% compared to 97.2% of Random Forest model.

Contents

1. Introduction

As of recent, machine learning and deep learning have been widely applied to achieve outstanding performance on a wide variety of tasks such as speech recognition, image classification, natural language processing, and bioinformatics. In the era of data, images contribute to a significant amount of global data production in the form of images or videos. Image classification technology has been an area of constant exploration and renewal. Both Random forests and CNN are considered as two most effective image data classifiers owing to their simplicity and relatively good performance.

In this study, we generally implemented Random Forests model and CNN model with support vector machine(SVMs) on MNIST dataset, a handwritten-digits-image database. Random Forest Model was optimized by experimenting various number of decision trees, different complexity of each decision tree and feature selection implementation. In addition, we also investigate the difference on testing and training accuracy between using traditional CNN-Softmax model and CNN-SVM model. Additionally, we made some change in the hyperparameter from the original paper and finally obtained an improvement on the model performance of CNN-SVM model.

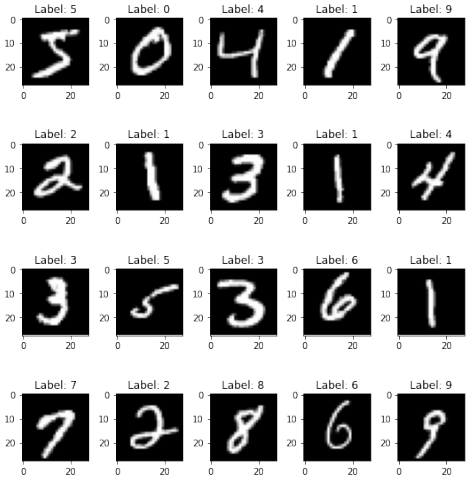

Figure 1. First 20 images of MNIST training data visualization

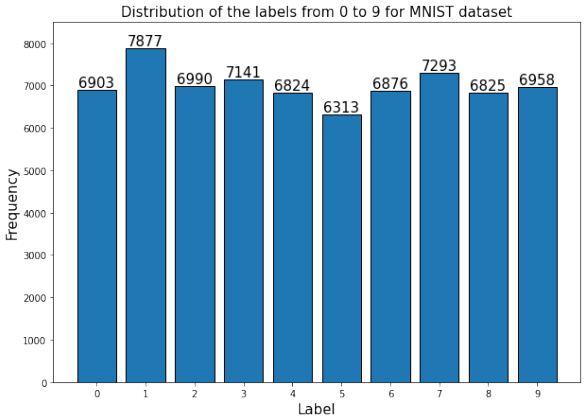

Figure 2. The distribution of 10 labels for MNIST dataset

2. Datasets

2.1 MNIST database

MNIST database [4] is a large database consist of 70000 handwritten digits images from different people, where the size of each black and white images are 28*28 = 784 pixels. Each pixel is encoded as an integer from 1 (white) to 255 (black): the higher this value the darker the color. In addition, the dataset were randomly split into 60000 training images and 10000 test images. In order to understand the MNIST dataset and visualize it, we displayed first 20 images from training set. The visualization [6] of the first 20 images from training set is shown in Figure 1. There are 10 labels from digit 0 to 9 in the MNIST dataset, the distribution of them is shown in Figure 2. From the distribution plot, we could conclude that label 1 is the most frequently occurring digit in the MNIST dataset whereas label 5 has the smallest number of occurrences in the dataset. And the other 8 label digits both have around 7000 frequency of occurring in the MNIST dataset. In order to fit the training images to CNNs and Random Forest models, we reshaped each 28*28 pixels format images to a 784 dimensional vector and then applied normalization for both train and test images. Therefore, the features of both training and test images are 784 pixels. Furthermore, one-hot encoding was applied for CNNs instead of Random Forest model.

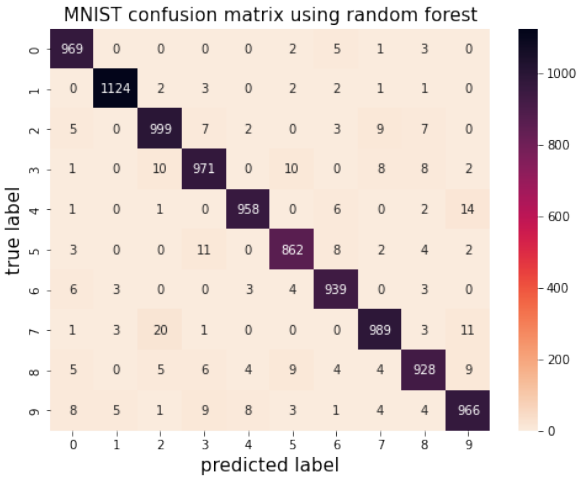

Figure 3. MNIST confusion matrix using random forest

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.96	0.97	0.97	1032
3	0.96	0.96	0.96	1010
4	0.98	0.98	0.98	982
5	0.97	0.97	0.97	892
6	0.97	0.98	0.98	958
7	0.97	0.96	0.97	1028
8	0.96	0.95	0.96	974
9	0.96	0.96	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

Figure 4. Classification Report on MNIST database

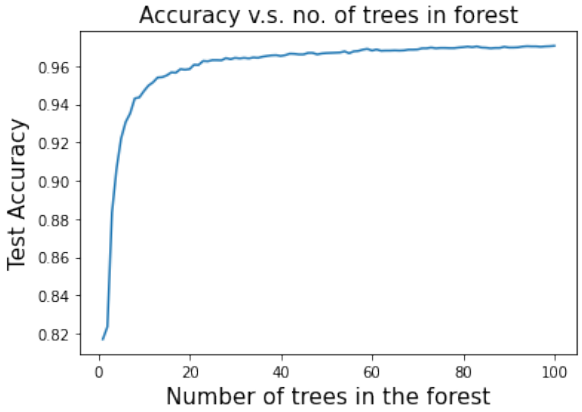

Figure 5. The impact of number of trees on accuracy

2.2 Methodology

For both model, we implemented according to the code the author provided in github[1]. For CNN model, we decided to create two sections for models and running independently for both CNN-SOFTMAX and CNN-SVM.

3. Results

3.1 Random Forest Model

By using a random forest classifier from skelarn.ensemble package [7] and only setting random_state be 0, the test accuracy for MNIST database is 97.05% which is pretty impressive. The confusion matrix is shown in Figure 3 which also indicates random forest works very well on the MNIST digits recognition. Moreover, by looking at classification_report in Figure 4, we could conclude that the random forest model has a very good prediction on digits 1 and 4 (0.99 and 0.98 accuracy respectively) compared to other digits. To investigate how the number of decision trees in the forest affects the test accuracy, we simply set the parameter n_estimator from 1 to 100 when creating the random forest classifier. The line chart is shown in Figure 5 which shows the same relationship as we expected before. As the number of tree increases from 1 to 20, the test accuracy rapidly rise from about 82% to 96%, and then if we keep raising the number of trees, the test accuracy rise very slowly and almost stay flat.

Not only considering the influence of the number of trees in the random forest, but also the complexity of each decision trees might affect the test error and accuracy. Thus, we retain the number of trees be 10 and change the maximum depth of each trees in the forest, the result is shown in Figure 6. From the line chart we could say that as the maximum depth of each tree increases from 1 to 10, the mean square error extremely dropped from 9.4 to about 1, and then if we keep increase the maximum depth of the trees, the MSE would stay flat around between 0.9 and 1. In conclusion, if each decision trees are complex, the test error of the model will decrease, increasing the test accuracy. In addition, more number of trees we have in a forest, more accurate we get on the test data for the model. Feature selection is an important method for supervised learning models in machine learning. To select the most important features

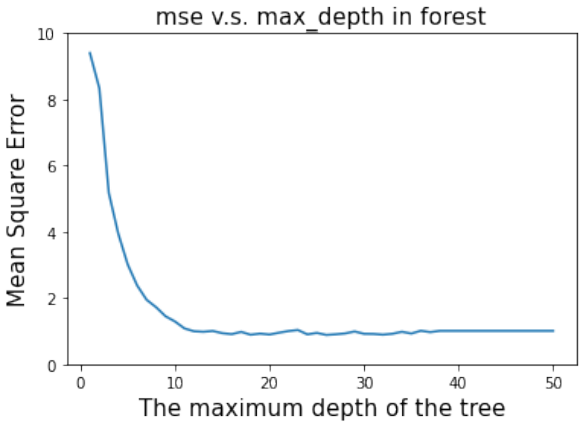

Figure 6. The impact of the maximum depth of trees on accuracy

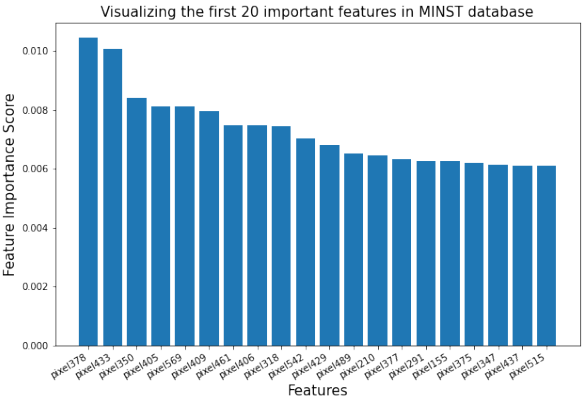

Figure 7. The 20 most important features in MNIST dataset

which are 784 pixels in the MNIST dataset, we used feature_importances_ method from sklearn.ensemble package [3] to visualize the importance of each features(Figure 7). There are 118 features having the feature importance score equaling to 0, if we drop those features and retain other parameters to be the same, the test accuracy became 97.1% which didn't have too much difference compared to the previous accuracy of 97.05%. To investigate how the number of important features affect the test accuracy on MNIST database, we altered the number of the most important features and then fit them in the random forest model, the result are shown in Figure 8. The line chart illustrated that if we drop half of the least important pixels, the test accuracy would stay around 97% instead of increasing. However, if we only keep less than the 200 most important features, the accuracy would rapidly

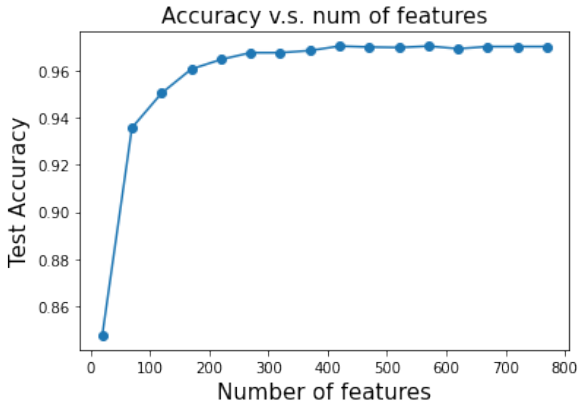

Figure 8. The impact of the number of the most important features on accuracy

drop to about 85%. As a result, for MNIST database, feature selection wouldn't improve the performance on accuracy of random forest model whereas dropping too many features would decrease the test accuracy obviously. To improve the accuracy of random forest model, the most common methods are getting more training data or increasing the number of decision trees in the forest. By increasing the number of trees to 1000 in the forest for MNIST dataset, finally the test accuracy was increased to 97.2% which is the best accuracy we got for random forest model.

3.2 CNN-SOFTMAX And CNN-SVM

3.2.1 Result From the Original Paper

	CNN-Softmax	CNN-SVM
Batch Size	128	128
Dropout Rate	0.5	0.5
Learning Rate	0.001	0.001
Epochs	10000	10000
SVM C	N/A	1

Table 1. Hyperparameters used for CNN-Softmax and CNN-SVM models

We started the reproduction work on this section based on the direct experiment the author

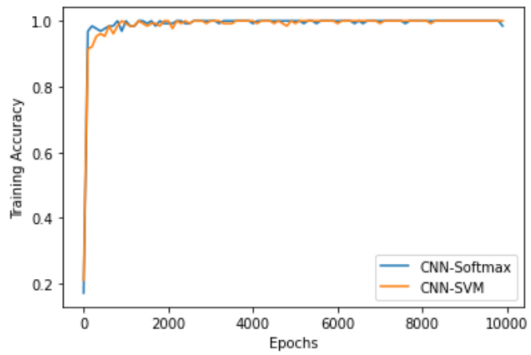

Figure 9. The training accuracy of CNN-Softmax and CNN-SVM model

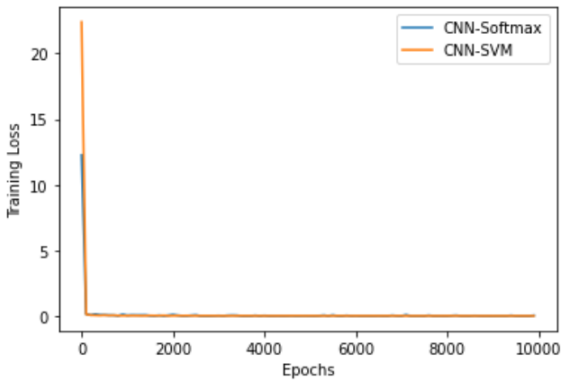

Figure 10. The training loss of CNN-Softmax and CNN-SVM model

did on MNIST dataset[2]. By setting the hyperparameter manually shows in table 1, we run the experiment on MNIST dataset.

In summary, the result is demonstrated in table 2, CNN-Softmax has a better performance on MNIST dataset with a testing accuracy of 99.33% and CNN-SVM has a testing accuracy with 99.09%. Overall, our result obtained by running the experiment is almost the same as the result they produced on the original paper. They obtained a testing accuracy of 99.23% for CNN-Softmax model and 99.04% by using CNN-SVM. This indicates that we reproduce the same result which is CNN-Softmax model has a better performance on MNIST than CNN-SVM model[2].

	Test accuracy
CNN-SOFTMAX	99.33%
CNN-SVM	99.09%

Table 2. Testing Accuracy of CNN-Softmax and CNN-SVM on MNIST

Besides, we also recorded the relationship between training accuracy and epochs graph with the comparison of the two different models as shown in figure 9. Furthermore, we also constructed the relationship between training loss and the epochs graph and it is shown in figure 10.

It is apparent that the above two graphs are very similar to the original result obtained from the paper[2].

3.2.2 Results Beyond the Original Paper

In the previous mini project, we have investigated some aspects of the hyperparameter of a regular CNN-Softmax and how it influences the testing accuracy. As we move forward to focus on this paper, we intended to investigate how the change of the drop out rate of the dense layer influence the CNN-SVM model. We modified the model so that the drop out rate for this CNN-SVM becomes 0.25 instead of 0.5 and here is the result for both models in a single table.

	Test accuracy
CNN-SOFTMAX(dp=0.25)	99.27%
CNN-SVM (dp=0.25)	99.33%

Table 3. Testing Accuracy of CNN-Softmax and CNN-SVM and Random Forest on MNIST

As the table demonstrated, by setting the drop out rate to 0.25, we obtain a better result utilizing CNN-SVM model than the previous one recorded in the original paper with the drop out rate equals to 0.5. This indicates a dramatically improvement of CNN-SVM model.

4. Discussion and Conclusion

In conclusion, we successfully reproduce a subset of the results from two papers respectively. In general, the results we reproduced is very closed to the original paper. Additionally, for

CNN we also investigated how a change in the drop out rate affect our model performance for the new model mentioned in the paper which is CNN-SVM. After running the experiment beyond the CNN paper, we found that by choosing a drop out rate of 0.25 rather than 0.5, we improved the model performance on our CNN-SVM model to having a testing accuracy of 99.33%. In the end, we compared the performance of the best model we determined for random forest model,the standard CNN-Softmax and the CNN-SVM model. The following table demonstrates our result.

	Test accuracy
Random Forest	97.2%
CNN-SOFTMAX(dp=0.5)	99.339%
CNN-SVM (dp=0.25)	99.33%

Table 4. Testing Accuracy of CNN-Softmax and CNN-SVM and Random Forest on MNIST

In summary,it is clear that on MNIST dataset, both two CNN models have a relatively better performance on image classification than the random forest model with an accuracy higher than 99% rather than 97.2%. Additionally, by changing the drop out rate for the CNN models, we found that it is possible to improve the model performance for CNN-SVM. Therefore, in the future, we may continue to run the experiments repeatedly to find the best model for the CNN-SVM in this paper.

4.1 what was difficult

During the process of reproduction of the CNN part. Due to the fact that the code was constructed several years ago, the author utilized the old version of tensorflow to implement the model and running file[1]. We were unable to compile the code in the beginning and tried a variety of approaches. Lastly, we decided to change the version of our current tensorflow to 1.15.2 and we successfully reproduced the experiments.

5. State of Contributions

Xiaoman Sun: Contribution on writing report

Yubai Zhang: Implementation of the CNN-SVM and Reproduction of the relevant paper, contribution on writing report

Chuqi Wang: Implementation of the Random Forest and Reproduction of the relevant paper,contribution on writing report

References

[1] Abien Fred Agarap. *AFAgarap/cnn-svm v0.1.0-alpha*. Dec. 2017. DOI: 10.5281/zenodo.1098369. URL: https://doi.org/10.5281/zenodo.1098369.

[2] Abien Fred Agarap. "An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification". In: *arXiv preprint arXiv:1712.03541* (2017).

[3] *Feature importances with a forest of trees*. URL: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html.

[4] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[5] Gilles Louppe. "Understanding Random Forests: From Theory to Practice". arXiv:1407.7502. PhD thesis. University of Liege, Belgium, Oct. 2014.

[6] Mr. Data Science. *How to plot mnist digits using matplotlib*. Jan. 2020. URL: https://medium.com/the-data-science-publication/how-to-plot-mnist-digits-using-matplotlib-65a2e0cc068.

[7] *Sklearn.ensemble.randomforestclassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.