Digit Recognizer in Kaggle

Wun-Sian Chen, Cho-An Lee, Jin-Yu Lin, Anatole Papin
CSIE, National Central University

*computer science, National Central University, Zhongda Rd., Taoyuan, Taiwan.

Abstract

We attend a Kaggle competition - Digit Recognizer. We aim to correctly identify digits from a dataset of tens of thousands of handwritten images. We discover that using simple CNN models[1] can achieve very high accuracy on the MNIST dataset. In this survey paper, we will introduce the method we implemented and comparison between models with different parameters.

Keywords: CNN, MNIST, computer vision

1 Introduction

MNIST handwritten digit recognition data set is one of the most basic data used to test the performance of neural network models and learning techniques. By using a Convolutional neural network (CNN), we can achieve over 99% accuracy easily. In this paper, we use the model without complex structural aspects or learning techniques [1] to achieve even higher accuracy. Augmenting training data by rotating and translating original images can also increase the rate. In the end, we have some confusion that if we put different datasets in our trained model, does it work ? Thus, we put emnist dataset to ensure whether our model can be executed.

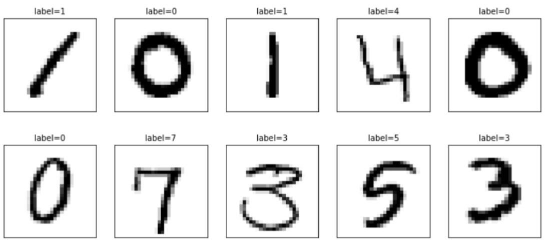


Fig. 1 Instance of mnist dataset

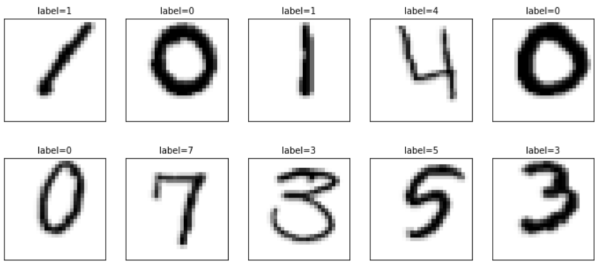


Fig. 2 Instance of mnist dataset after rotation and translation

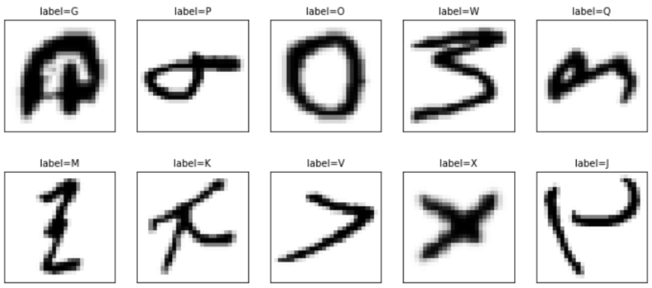


Fig. 3 Instance of mnist dataset

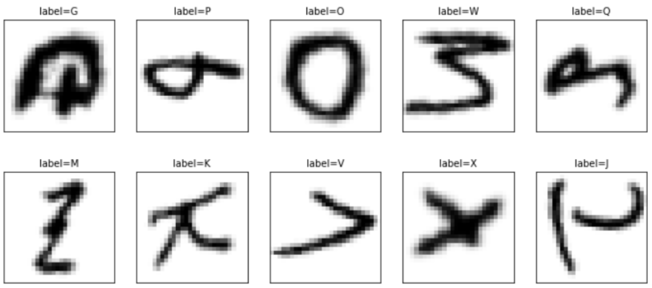


Fig. 4 Instance of mnist dataset after rotation and translation

2 Method

2.1 preprocessing on data

Test data on Kaggle contains 42000 figures. We take 30000 as training data, and others as testing data. And we do some processes to our data, to extend data. we rotate figures randomly in each direction with less than 20 degrees and shift figures randomly with up to 20% of the image size in each direction. Now, our training data become 300000 figures. And we divide each value of pixel 255 before training.

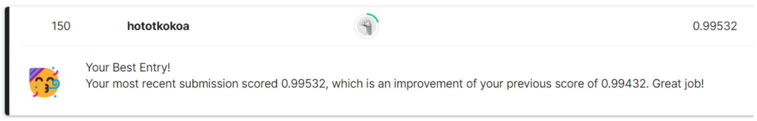
2.2 model architecture

Our model includes three models with different kernel sizes marked as M3, M5, and M7 respectively. Each sub-model has different deep convolution layers, 10, 5, and 4 respectively.

For every convolution layer, we use relu as an activation function and add a batch normalization layer after. Following convolution layers, we add one dense layer with a dropout rate of 0.25 and a softmax layer. Regarding optimizer, we use Adam optimizer with cross-entropy loss function, the learning rate starts from 0.001, and exponentially decays with decaying factor $\gamma = 0.98$.

For the train parts, we trained 50 epochs in each mode with batch size 128. M3, M5, and M7 have about 2 million trainable parameters.

Finally, our best submission got 99.532% accuracy on Kaggle hidden data, reaching 150th of the contest.



150 hototkokoa 0.99532

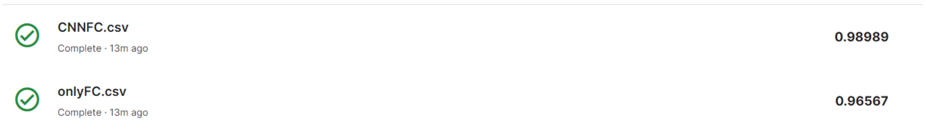
Your Best Entry!
Your most recent submission scored 0.99532, which is an improvement of your previous score of 0.99432. Great job!

3 Experiment

In the below sections, we will show our experiment outcomes. Identify whether our work is effective. We operate one factor to compare with, although the accuracy doesn't have big differences, still maintains around 99 %, but if testing more essential experiences, the small difference in accuracy rate played an important role at all.

3.1 convolution layers

In the Kaggle discussion area, we found that people usually use deep convolution layers before shallow fully connected layers instead of only use fully connected layers. We build a smaller model to verify why we should use convolution layers. In the test data on Kaggle, using convolution bring out a big difference to the model with only use fully connected layers. (from 96.567% to 98.989%)



CNNFC.csv
Complete · 13m ago
0.98989

onlyFC.csv
Complete · 13m ago
0.96567

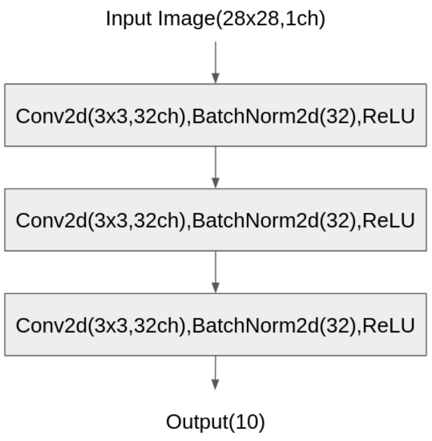

Input Image(28x28,1ch)
Conv2d(3x3,32ch),BatchNorm2d(32),ReLU
Conv2d(3x3,32ch),BatchNorm2d(32),ReLU
Conv2d(3x3,32ch),BatchNorm2d(32),ReLU
Output(10)

Fig. 5 3 convolutional layers with batch normalization

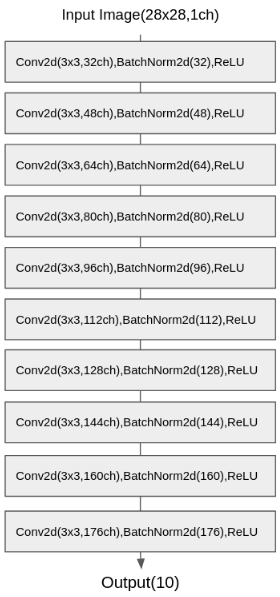
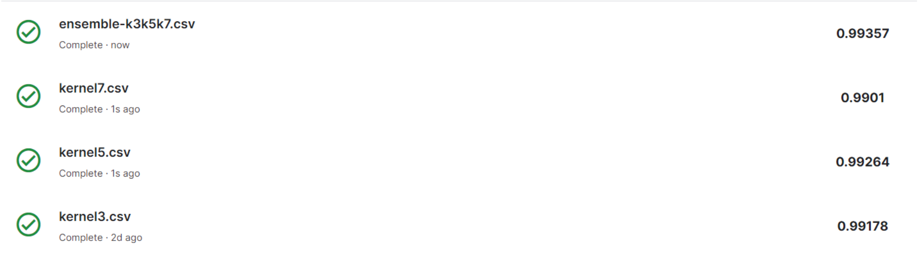

Input Image(28x28,1ch)
Conv2d(3x3,32ch),BatchNorm2d(32),ReLU
Conv2d(3x3,48ch),BatchNorm2d(48),ReLU
Conv2d(3x3,64ch),BatchNorm2d(64),ReLU
Conv2d(3x3,80ch),BatchNorm2d(80),ReLU
Conv2d(3x3,96ch),BatchNorm2d(96),ReLU
Conv2d(3x3,112ch),BatchNorm2d(112),ReLU
Conv2d(3x3,128ch),BatchNorm2d(128),ReLU
Conv2d(3x3,144ch),BatchNorm2d(144),ReLU
Conv2d(3x3,160ch),BatchNorm2d(160),ReLU
Conv2d(3x3,176ch),BatchNorm2d(176),ReLU
Output(10)

Fig. 6 10 convolutional layers with batch normalization

3.2 different kernel size

[1]We test different kernel size CNN layers to train, and also make an ensemble model combining three models, which also enhances the accuracy. In the three basic models, the model with kernel size 5 has better performance than the model with kernel size 3 and the model with kernel size 7.



✓ ensemble-k3k5k7.csv Complete · now	0.99357
✓ kernel7.csv Complete · 1s ago	0.9901
✓ kernel5.csv Complete · 1s ago	0.99264
✓ kernel3.csv Complete · 2d ago	0.99178

3.3 Difference between dataset with rotation and translation

We make a comparison with different sizes of data. In the following tests, we have a clear conclusion, the accuracy advanced with the growth of the dataset.

In each test, we will make a chart to visualize the training loss and validation loss. Training loss indicates that the model is able to accurately predict the correct labels for the training data, while the validation loss indicates that the model is able to generalize well to new data.

—Independent Variable:

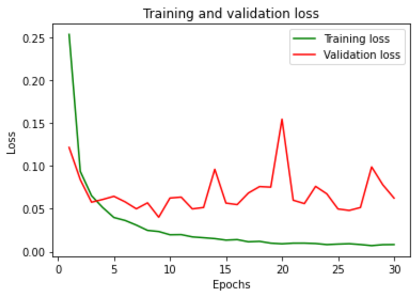
- Rotation
- Translation
- Rotation and Translation

—Control variable:

- 3 convolutional layers with batch normalization
- 10 convolutional layers with batch normalization

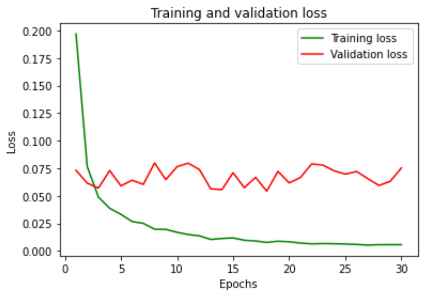
3.3.1 Rotation

1. In 3 convolutional layers, We rotate the data with 20 degrees each, and the dataset get five times larger. (150000 pieces)



Accuracy: 98.79%

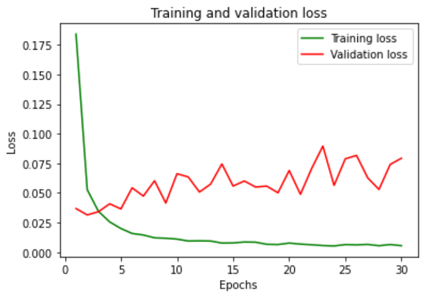
2. In 10 convolution layers, We rotate the data with 20 degrees each, and the dataset gets five times larger. (150000 pieces)



Accuracy: 98.89%

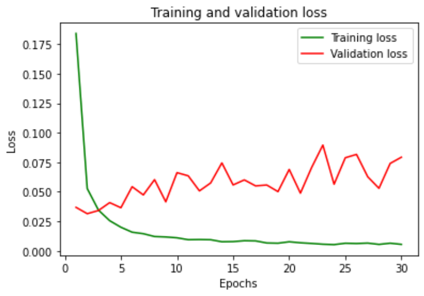
3.3.2 translation

1. In 3 convolutional layers, We shift the data with 20% each, and the dataset gets five times larger. (150000 pieces)



Accuracy: 98.75%

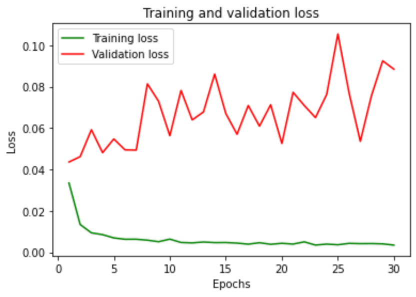
2. In 10 convolutional layers, We shift the data with 20% each, and the dataset gets five times larger. (150000 pieces)



Accuracy: 99.21%

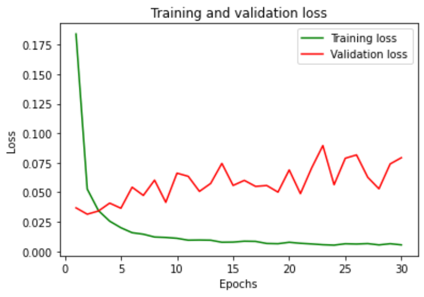
3.3.3 Rotation and Translation

1. In 3 convolutional layers, Our dataset became 27 times larger after translation and rotation. (270000 pieces)



Accuracy: 99.14%

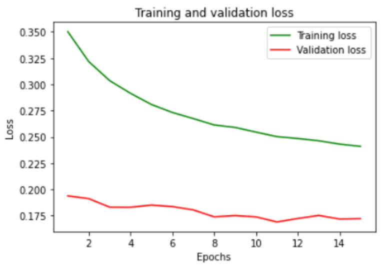
2. In 10 convolutional layers, Our dataset became 27 times larger after translation and rotation. (270000 pieces)



Accuracy: 99.31%

3.4 emnist dataset testing

In 3 convolutional layers, Our dataset became 7 times larger after translation and rotation. (489993 pieces)



Accuracy: 94.39%

4 conclusion

In conclusion, the use of convolutional layers, merging models with different kernel sizes, and data augmentation techniques such as adding rotation and translation to the dataset can improve the accuracy of a deep convolutional neural network (CNN) model. The final model achieved an accuracy of 99.532% on Kaggle's hidden data, placing 150th in the contest.

We also tested our model on the EMNIST dataset and found that it was able to achieve similar accuracy to that on the MNIST dataset.

Overall, these experiments demonstrate the effectiveness of using CNN models for digit recognition tasks and the importance of augmenting training data and using deeper convolutional layers.

References

[1] An, S., Lee, M., Park, S., Yang, H., & So, J. (2020). An ensemble of simple convolutional neural network models for mnist digit recognition. arXiv preprint arXiv:2008.10400.

[2] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[3] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. ArXiv, abs/1502.03167.

[4] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.