

CS583A: Course Project

Gopi Miyani

December 1, 2019

1 Summary

I have participated in an active competition of identifying digits from a MNIST dataset of tens of thousands of handwritten images. I tried several different methods like Logistic Regression, Fully Connected Neural Network, Supervised Autoencoder and Convolutional Neural Network to solve this problem. The final model I chose is convolutional neural network, which takes 28×28 images as input and outputs the class labels. I implemented the convolutional neural network using Keras and run the code on a MacBook Pro with one Intel i7 CPU and 16 GB memory. Performance is evaluated on the categorization accuracy of my predictions (the percentage of images I get correct). In the public leaderboard, my score is 0.99485; My rank is 471 among the 2289 teams. The result on the private leaderboard is not available until the competition end.

2 Problem Description

Problem. The problem is to correctly identify digits from a MNIST dataset of handwritten images. This is a multi class classification and image recognition problem. The competition is at <https://www.kaggle.com/c/digit-recognizer>. The goal of this problem is to take an image of a handwritten single digit, and determine what that digit is. The predictions need to be submitted in a file with ImageId and Label.

Data. The data are in two files train.csv and test.csv, which contain gray-scale images of hand-drawn digits, from zero through nine. Each image is of 256×256 pixels. The number of training samples is $n = 42,000$. The number of classes is 10. The training set is balanced. The number of test samples is $n = 28,000$.

Challenges. The Challenge here is to get higher accuracy greater than 0.9900 to achieve high rank in the competition because this competition is active, on going and quite big, involving 2289 teams right now.

3 Solution

Model. The model we finally choose is a sequential convolutional neural network.

Implementation. We implement the convolutional neural network model using Keras with TensorFlow as the backend. Our code is available at <https://github.com/gopimiyani/CS-583-Deep-Learning/tree/master/Git%20Final%20Project>. I ran the code on a MacBook Pro with one Intel i7 CPU and 16 GB memory. It took around 5 hours to train the model.

Settings. The loss function is categorical cross-entropy. The optimizer is RMSprop. The learning rate is 0.001, epochs are 5 and batch size is 64.

Advanced tricks. I used data augmentation and batch normalization to alleviate over fitting.

Cross-validation. I partitioned the training data to 90%-10% for hyperparameter tuning. Here, the training accuracy and validation accuracy both are increasing and not much difference in between them. The training accuracy is 99.94% and validation accuracy is 99.50%

4 Compared Methods

Logistic Regression. I used Logistic Regression, a linear model provided by SKlearn. The accuracy score I got is 90.90%. This model I used as baseline to compare the results of other methods.

Fully-connected neural network. I implemented a 3-layer fully-connected neural network. The width of the layers (from bottom to top) are respectively 600, 600, and 10. The training and validation accuracies are respective 100.00% and 98.10%, which are higher than baseline model (logistic regression).


















Supervised Autoencoder I used supervised autoencoder model provided by keras. I used 4 dense layers in encoder network, one bottleneck layer, 5 layers in decoder network and 3 classifier layers. The training and validation accuracies are respective 98.57% and 96.26%.

Advanced tricks. Our finally adopted method is a convolutional neural network with 4 convolutional layers, 2 max pooling layers and 2 dense layers. We applied the following tricks.

- Data augmentation. Without data augmentation, the training and validation accuracies are respective 98.90% and 98.70%. With data augmentation, the training and validation accuracies are respective 99.94% and 99.50%.
- Batch Normalization I used batch normalization to do faster convergence. I added 4 batch normalization layers in CNN model. Using batch normalization, accuracy got increased.

5 Outcome

I participated in an active competition. My score is 0.99485 in the public leaderboard. My rank is 471/2289 in the public leaderboard. The result on the private leaderboard is not available until the competition end. The screenshot is in Figure 1.

	Overview	Data	Notebooks	Discussion	Leaderboard	Rules	Team		My Submissions	Submit Predictions
464	mostatatahsin7								0.99485	1 2mo
465	Agnimitra Sen								0.99485	3 1mo
466	Peng Sun								0.99485	1 1mo
467	Jake Cherrie								0.99485	1 14d
468	Vyom Pathak								0.99485	6 13d
469	Ogura								0.99485	5 11d
470	Hechenye								0.99485	1 5d
471	Gopi Miyani								0.99485	10 15h
Your Best Entry  Your submission scored 0.97614, which is not an improvement of your best score. Keep trying!										
472	Remi Calizzano								0.99471	1 2mo
473	Rohit Gadhwari								0.99471	1 2mo
474	Udbhav Pangotra								0.99471	1 2mo
475	fsc2016								0.99471	11 1mo
476	Rishabh Nimje								0.99471	1 1mo
477	Jack Zhang #2								0.99471	4 1mo
478	Anudeep peela								0.99471	1 1mo
479	Maciej Rogala								0.99471	3 1mo

(a) Public leaderboard.

Figure 1: My ranking in the public leaderboard.