

# Machine Learning Contest 2016 Solution Document

## Team : *SCP*

### Team Leader:*Eric H Chang*

### Team Members: *Jesse Yeh, Paxion Tang, Winsdom Chen, Bruce T Wang, Doma Chiang*

# Abstract

*(The short description of works)*

*At beginning, we build a docker image for team member to quick start and share the code. We choice SVM (support vector machine) as our first machine learning model, it has good performance that we can get result fast and submit first result as third place. The log loss score is 0.136538.*

*After we do more research, we found that CNN (convolutional neural networks) are the best models for image recognition. So we change model from SVM to CNN. We used 2 convolution networks at starting. And got log loss score 0.078297. The we try to use 4 convolution networks and got better score to 0.0566.*

*For get better score, we start to use Keras. Keras has good image processing function, so we can extend our training data as many as we want. And we got score to 0.034669.*

*The last improvement is vote. We save better weights to HDF5 file format. And later we pick up 3 best weighs to produce result. Then we use 3 result to vote each image and produce final result for submission. The vote improve score to 0.030193.*

# Platform

*(Cloud platform or local? packages? programming language)*

*We use docker container and running at local.*

*We use Keras, Theano, numpy and pandas.*

*We use Python as our programming language.*

# Features

*(Including the description and generating method of each feature)*

*We load image from file and convert to CSV file format. The CSV file contained image id, label and all 784 pixels.*

*Our features are all 784 pixels of image, and convert to 28x28 2D array as input of model.*

# Models

*(Including algorithm name, parameter setting of each model)*

model = Sequential()  
  
model.add(Convolution2D(32, 3, 3, border\_mode=**'same'**,  
 input\_shape=(img\_channels, img\_rows, img\_cols)))  
model.add(Activation(**'relu'**))  
model.add(Convolution2D(32, 3, 3))  
model.add(Activation(**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(drop\_one))  
  
model.add(Convolution2D(64, 3, 3, border\_mode=**'same'**))  
model.add(Activation(**'relu'**))  
model.add(Convolution2D(64, 3, 3))  
model.add(Activation(**'relu'**))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Dropout(drop\_two))  
  
model.add(Flatten())  
model.add(Dense(input\_dim=64\*7\*7, output\_dim=128))  
model.add(Activation(**'relu'**))  
model.add(Dropout(0.5))  
model.add(Dense(input\_dim=128, output\_dim=nb\_classes, W\_regularizer=l2(0.01) ))  
model.add(Activation(**'softmax'**))

# Training process

*(CV? Feature importance? Anything worth to mention during this work.)*

*Steps*

1. *Convert train/test images to CSV file format.*
   1. *python conv\_test\_data\_to\_csv.py test\_data.csv test/*
   2. *python conv\_train\_data\_to\_csv.py train.csv train\_data.csv train/*
2. *Start first round training by 27000 extended training images and save better weights to hdf5 files according to validation loss.*
   1. *python CNN\_preprocess.py ../train\_data.csv ../test\_data.csv predicted.csv 100 1000 3 0.5 0.5 30 0.2 0.3 0.1 0.1 128 > /dev/null*
3. *Repeat step 2 for several times*
4. *Pick up the best weight of round 1 to do second round training by 54000 extended sample.*
   1. *python CNN\_preprocess.py ../train\_data.csv ../test\_data.csv best\_of\_round1.hdf5 100 1000 6 0.5 0.5 30 0.2 0.3 0.1 0.1 128 > /dev/null*
5. *Repeat step 4 for several times*
6. *Pick up the top 3 weights of round 2 to produce result*
   1. *python CNN\_preprocess.py ../train\_data.csv ../test\_data.csv top1\_of\_round2.hdf5 0 0 0 0.5 0.5 30 0.2 0.3 0.1 0.1 128 > /dev/null*
   2. *python CNN\_preprocess.py ../train\_data.csv ../test\_data.csv top2\_of\_round2.hdf5 0 0 0 0.5 0.5 30 0.2 0.3 0.1 0.1 128 > /dev/null*
   3. *python CNN\_preprocess.py ../train\_data.csv ../test\_data.csv top3\_of\_round2.hdf5 0 0 0 0.5 0.5 30 0.2 0.3 0.1 0.1 128 > /dev/null*
7. *Get voting result from 3 results of step 3.*
   1. *csv=" top1\_of\_round2.hdf5.csv top2\_of\_round2.hdf5.csv top3\_of\_round2.hdf5.csv"*
   2. *count=3*
   3. *result=predicted\_a.csv*
   4. *python aggregation2.py $count $csv $result > /dev/null*
8. *Submit voting result “predicted\_a.csv”*

# Conclusion

For digit recognize CNN is the best choice. Beside a good learning algorithm, extended training sample is also important especially when training data is not enough. CNN is good, but it couldn’t learn great always, so we have to save the weights into files and test them to find best weight to predicted test data. Voting is the another method that can increase accuracy.

Investigate test data is necessary, because you have to understand why you can’t classify some images. Then adjust your model to more accuracy. For example, the second test data released on 5/9. We got bad score by original model. After investigate on the images that we can’t classify correctly. We found there are some images have many noises. So we decide to remove noise for all images including train/test data, then re-training our model. And we got much better result by new model.

I think there are many methods could increase accuracy, we just need to learn and try.

# Reference

1. <https://www.youtube.com/playlist?list=PLXVfgk9fNX2I7tB6oIINGBmW50rrmFTqf>
2. <http://neuralnetworksanddeeplearning.com/chap6.html>
3. <https://www.kaggle.com/c/digit-recognizer>