Final Assignment Report

Yifan Zhao ID: 862188932

1. Implementation Method

Method: Convolutional Neural Network

Implementation: Use Keras of Tensorflow as API for building and training the CNN model. I use the sequential model as the basic model, add convolutional layer and pooling layer. In the training process, use the cross-validation as the loss function to optimize the model.

2. Result and Accuracy

Figure 1 shows the model of CNN of training. After running the train.py, we get the weight of the model. We set the epoch to be five. We can see from the result of figure 2 that the accuracy reaches to about 0.99.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

Figure 1

```
Train on 42000 samples
Epoch 1/5
42000/42000 [==============================] - 20s 477us/sample - loss: 0.1748 - accuracy: 0.9448
Epoch 2/5
42000/42000 [==============================] - 19s 446us/sample - loss: 0.0542 - accuracy: 0.9828
Epoch 3/5
42000/42000 [==============================] - 19s 445us/sample - loss: 0.0392 - accuracy: 0.9877
Epoch 4/5
42000/42000 [==============================] - 19s 444us/sample - loss: 0.0292 - accuracy: 0.9908
Epoch 5/5
42000/42000 [==============================] - 18s 433us/sample - loss: 0.0231 - accuracy: 0.9923

Process finished with exit code 0
```

Figure 2

Then we use the weight we get from the train.py for testing of the data from test.csv. Output the predicted value for the data in test.csv and save it as test_submission.csv.

3. Scatterplot

Use the PCA method to reduce the features to two principal components, PC-1and PC-2. Figure 3 shows the scatterplot of the data using different color according to their labels.

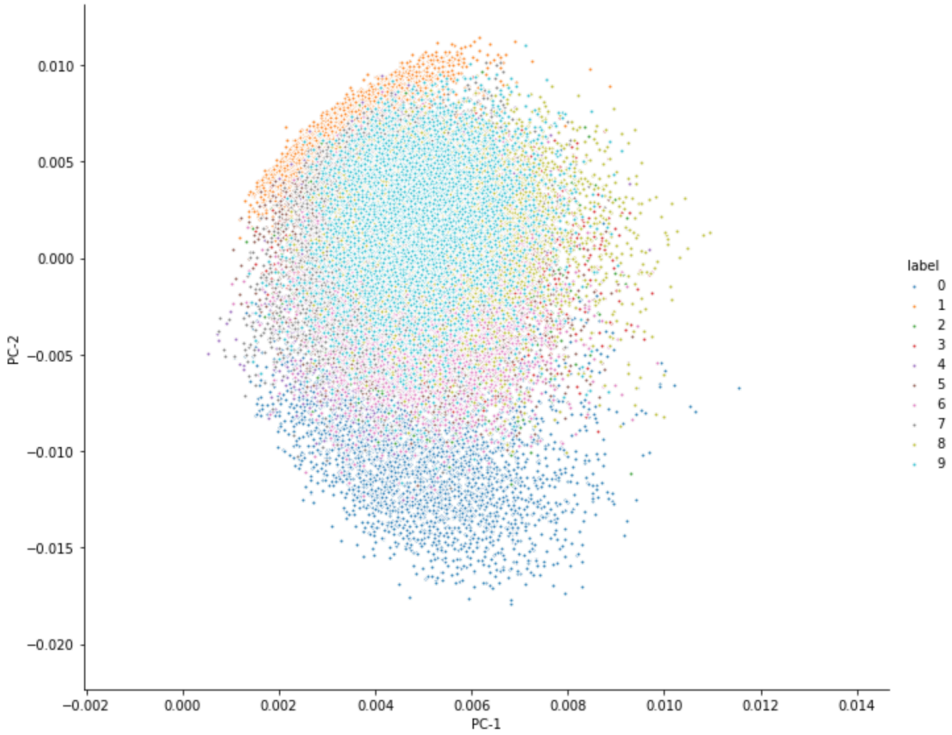

Figure 3