Individual Report

Yiming Dong

Introduction:

In this final project, we choose the Kannada MNIST Dataset as our input data. The reason why we pick up Kannada MNIST is that we hope it could be helpful to people and scholars who are interested in Kannada culture. Kannada is the official and administrative language of the state of Karnataka in India with nearly 60 million speakers worldwide. The language has roughly 45 million native speakers and is written using the Kannada script. Distinct glyphs are used to represent the numerals 0-9 in the language that appear distinct from the modern Hindu-Arabic numerals in vogue in much of the world today.

Description of individual work:

In the preparation part, searching for the CNN model algorithm and Keras framework. Data loading and data-preprocessing is my work. As for our dataset contains 10 number, so it is should be a multiclassification model. Dataset file contains four csv files which are train, test, sample submission and Dig MNIST. The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine, in the Kannada script. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive. The training data set has 785 columns. The first column, called label, is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image. The test data set is the same as the training set, except that it does not contain the label column. The dig MNIST contains an additional labeled set of characters that can be used to validate or test model results before submitting to the leaderboard. It is an additional handwritten dataset (with 10k images) which can serve as an out-of-domain test dataset.

This dataset is a large database of handwritten digits (0 to 9). The first step is to load the dataset. Considering the file format of our dataset, it can be easily done through “pandas”. When we looked at the train data, the first column is the label, so it is selected as y train and the rest column is the x train. We reshaped the images with 28 * 28 *1 in put image. 28 *28 means the size of the image and 1 means the color of the image is white and black. After doing the necessary processing on the image, the label data, like y train and y test need to be converted into categorical formats for model.

```
X=train.drop('label',axis=1)
Y=train.label

Id = test['id']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20, r
andom_state=42)
x_train = x_train.values.reshape(-1,28,28,1)/255.0
x_test = x_test.values.reshape(-1,28,28,1)/255.0
y_train = np_utils.to_categorical(y_train,num_classes=10)
y_test = np_utils.to_categorical(y_test,num_classes=10)
```

Figure1

Description of individual work in details:

After separating the train and test data, we worked on the image enhancement and we used Image Data Generator to increase the size of the train data in order to reduce the problem of overfitting. Figure 4 shows the steps we did for preprocess image. rotate the images’ angle (the value of the angle is 10), float the images with fraction of total width (20%), float images with fraction of total height (20%), random zoom image with 20%.

```
imagegen = ImageDataGenerator(
            rotation_range = 10,
            width_shift_range = 0.2,
            height_shift_range = 0.2,
            shear_range = 0.1,
            zoom_range = 0.2
             )
imagegen.fit(x_train)
```

Figure 2

Result:

Test accuracy 98.54% implies the model is trained well for prediction.

The figure 7 and figure 8 show the performance of our best model in the train data set. Form the pictures, we can notice that our accuracy is more than 98% and the loss is lower than 0.1 finally. With more epochs, the performance is better and the difference between train and validation is smaller.

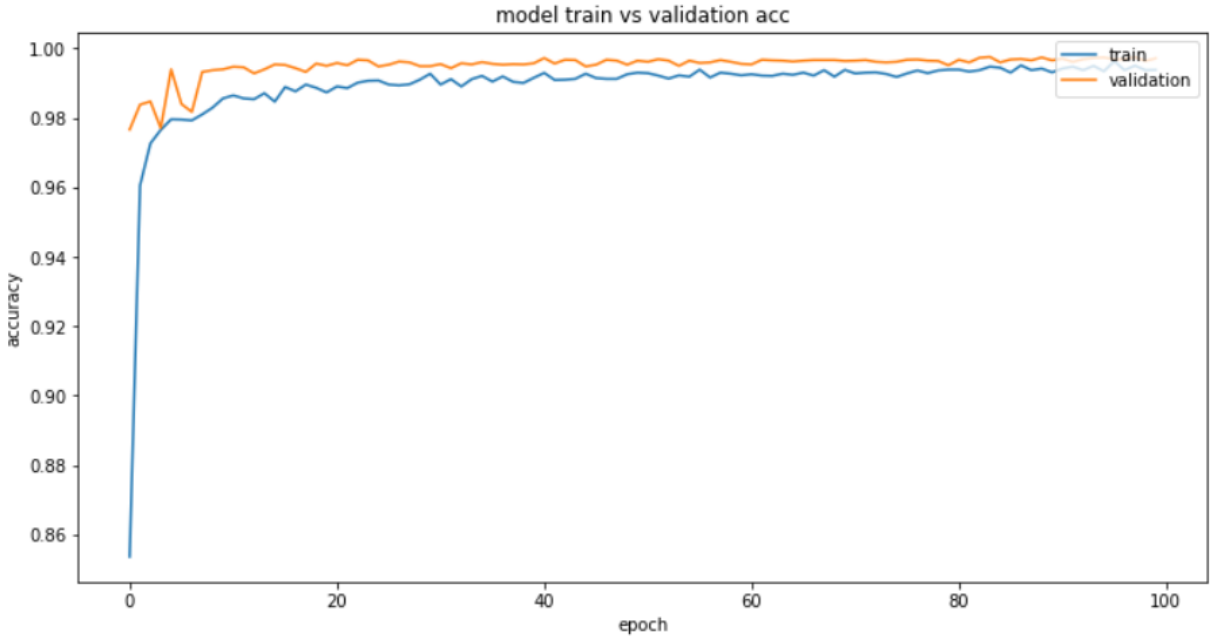

Figure 7

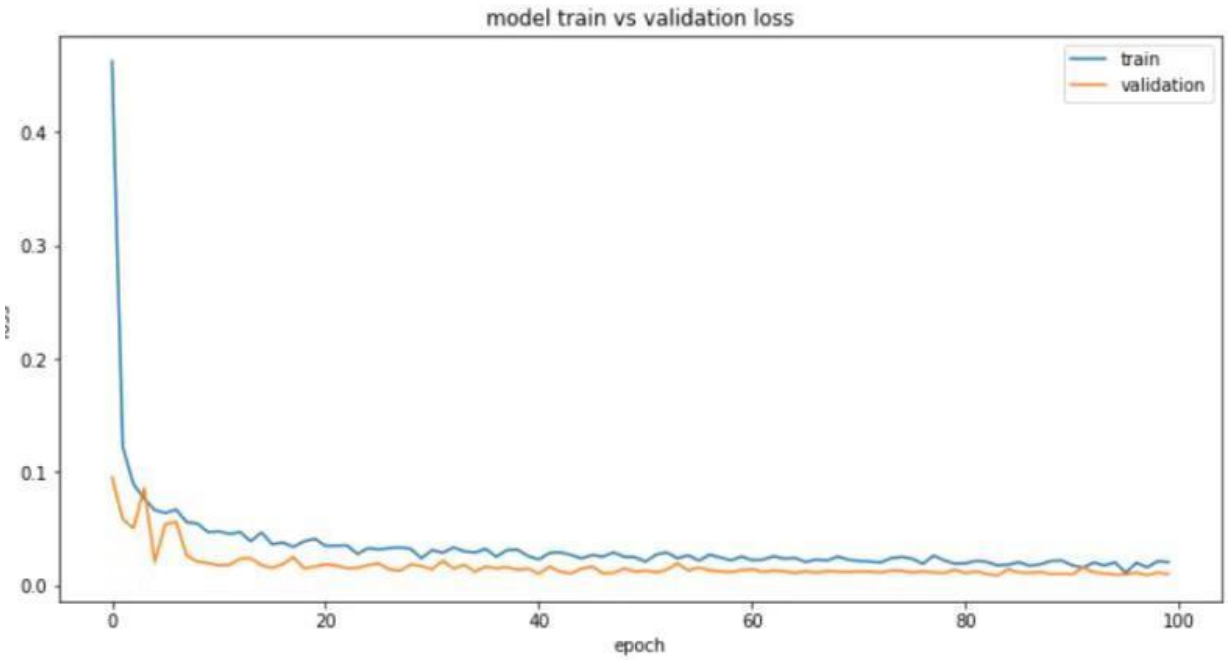

Figure 8

The figure 9 presents the final result of recognition. The diagonal line means the correct number we predict. The other number except 0 means cases the model didn't predict right. The number 0 means that the predicted number is the same as the actual one. For example, the number 13 means that there are 13 images of 1 was predicted as 0. In conclusion, there are highly similarity between 1 and 0, 6 and 9.

	0	1	2	3	4	5	6	7	8	9
0	1162	13	1	0	0	0	0	0	1	0
1	0	1218	0	0	0	0	0	0	0	0
2	1	0	1223	0	0	0	0	0	0	0
3	0	1	1	1177	0	0	0	5	0	0
4	0	0	0	1	1218	2	0	0	0	0
5	0	0	0	2	0	1186	0	0	0	0
6	0	0	0	0	0	0	1156	3	0	10
7	0	1	0	2	0	0	4	1210	0	2
8	0	0	0	0	0	0	0	0	1186	0
9	0	0	0	0	0	0	3	0	0	1211

Figure 9

The figure 10 shows the part results of the prediction that we did the wrong one. Take the first picture as the example. The actual number should be 4, but we predicted as 9.

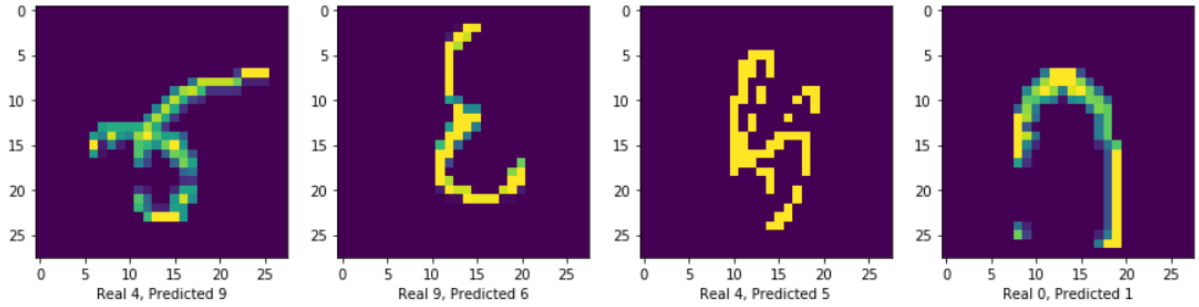

Figure 10

Then, we test our model in the test dataset in the Kaggle competition public board, we got the rank 318/1040 (around top 30%) and the 98.54% accuracy.

Summary and conclusions:

In summary, we used Kannada handwritten digits dataset to recognize Kannada numeric from 0 to 9, so this is the multi-classification problem and we used Keras as framework and CNN as deep network. We load the data and use image data generator to preprocess image, after that, we build 6 convolution layers with Relu activation function followed by pooling layer, a fully connected layer and softmax layer respectively, and we compile the model. Finally, we visualized the result by line charts and shows the score of test loss and test accuracy.
There are three things we learned from this project. The first one is preprocess image by Image data generator, which can rotate, float and reshape images to increase the size of the image and reduce the problem of overfitting. The second thing we learned is learning rate reduce. Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced. The last thing we learned is how to deal with the problem of overfitting. Finally, we got the rank 318/1040 in the Kaggle competition. Form this project, we have a better understanding of deep learning theory and python code. We will apply what learned from this class into the future work.

Percentage of the code that you found or copied from the internet:

20%

Reference:

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

Prabhu, Vinay Uday. "Kannada-MNIST: A new handwritten digits dataset for the Kannada language." arXiv preprint

Vinay Uday's github: https://github.com/vinayprabhu/Kannada_MNIST