2019 Machine Learning II
Kannada Digits Recognition

Jingshu Song, M.S.

The George Washington University

1. Introduction

For this final project, we worked on the classic MNIST problem which to recognize the Kannada digits. Kannada is the official and administrative language of the state of Karnataka in India. 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. The following picture shows the figure of Kannada.

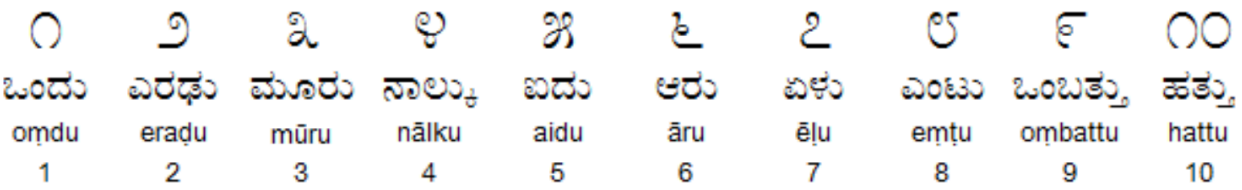

Figure 1

One of reasons we choose this dataset because it is not as simple as Arabic dataset, also not hard to implement deep learning algorithm by us. Another important reason is that this model will be helpful to Karnataka people and scholars who interest in Kannada culture.

For this project, we used Keras as framework and CNN as deep network. We choose Keras because of its simplicity and readability. Also, considering the size of dataset, we don't need to worry about the problem of speed. We choose CNN because it shows high performance than simple MLP with two dimensional images. Our 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. After process image, we built the model and compile it, and we present our results.

For the part of the shared work, we found the Kaggle competition and figured out the model together.

2. Description of your individual work

I worked on the visualization part and finished the group report in this project.

For this project, we used Keras as framework and CNN as deep network. We choose Keras because of its simplicity and readability. Also, considering the size of dataset, we don't need to worry about the problem of speed. We choose CNN because it shows high performance than simple MLP with two dimensional images. The following two paragraphs provide the background knowledge of keras and CNN.

Keras is an open-source neural-network library written in Python. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing Deep Neural Network code.

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. The figure1 shows the architecture of the CNN.

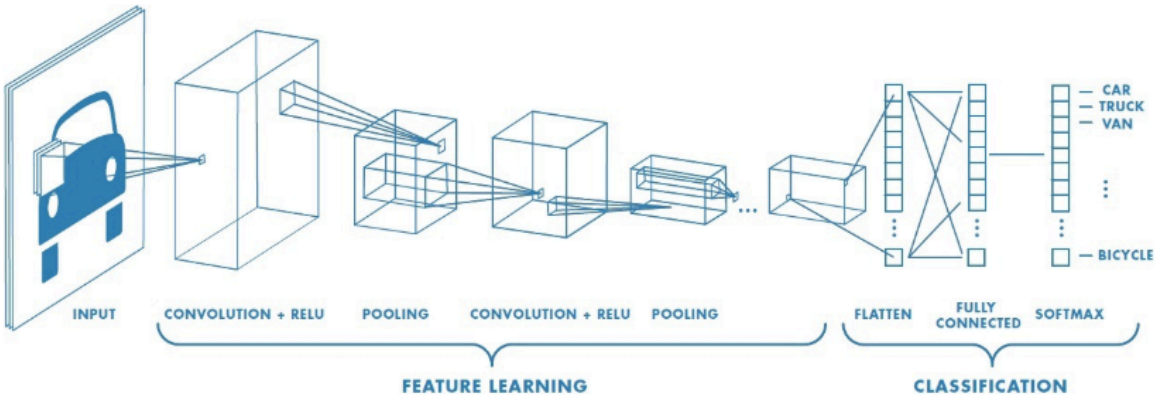

Figure 2

From the figure, we can notice that the input is the pixel matrix of the image and it will preprocess and transform to the convolution layer. Convolutional layer applies convolution operation on the input layer, passing the results to next layer. A convolution operation is basically computing a dot product between their weights and a small region they are connected to in the input volume. After the convolution with Relu activation function, it will transform to

the pooling layer which performs a down-sampling operation along the width in order to reduce dimensions. Then after some combination of previously defined architecture, flattening layer is used to flatten the input for fully connected layer. Next to these layers, the last layer is the output layer. The flatten layer will convert the 3-dimensions (height, width, depth) into a single long vector to feed it to the fully connected layer or Dense layer. It connects every neuron in one layer to every neuron in another layer. Fully Connected Layer and Output Layer Fully connected layers are the same hidden layers consisting of defined number of neurons connected with elements of another layer.

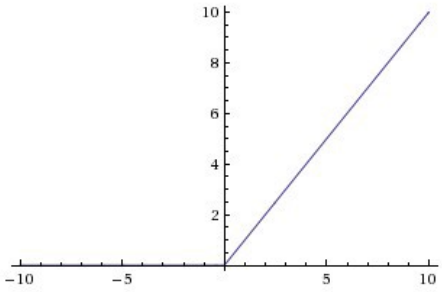

Figure 3

Relu is the most common activation function [A(x) = max (0, x)] and the ReLu function is as shown above(figure 3). It gives an output x if x is positive and 0 otherwise. Compared with other function like Sigmoid, the reason why we choose Relu doesn't need to worry about the vanishing gradient and networks with Relu tend to show better convergence performance in practice. Besides that, ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.

3. Describe the portion of the work that you did on the project in detail

For this project, I am responsible for visualization the result and I finished the report. I also tried to do image enhancement to improve accuracy, but it doesn't work. For the visualization part, first I made two line charts to show the accuracy and loss of our test dataset. It also presents the relationship between epochs and accuracy, loss. And then, the visualization also shows the situation about the actual number and the predicted number.

For the report, I did the research about the background knowledge Kannada and their language, which includes the history, current situation and so on. After that, I descripted our dataset which includes four csv files: train, test, sample submission and Dig MNIST. The first column is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image. And then, I wrote the background knowledge about Keras and CNN, including the definition of framework, how CNN works, the definition of each layers. I also mentioned the two important activation we mainly used: relu and softmax, which includes their definition, advantages, figures and how they use.

After that, I introduced the experimental set up of our model. It includes three parts: data preprocess, build and compile data, train and evaluate data. I descripted the way we preprocess image: Image data generator. I explained the definition of the parameters of our model, like filter, padding and so on. There is an important step we did for our model to improve accuracy: learning rate reduction, so I also explained its purpose, definition and parameters thoroughly. Finally, I presented our result with images and made a summary for our project. In the summary part, I pointed out the three things I learned from the project: image data generator, learning rate reduction and deal with overfitting problem. I also wrote about where we can improve about our model.

4. 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.

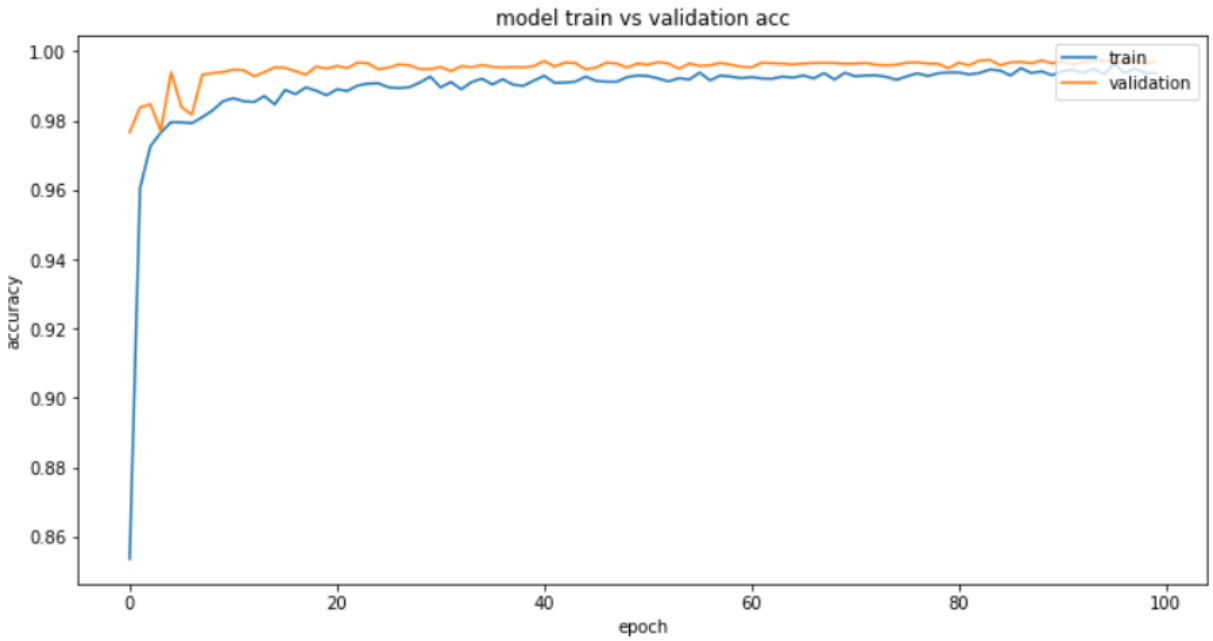

Figure 7

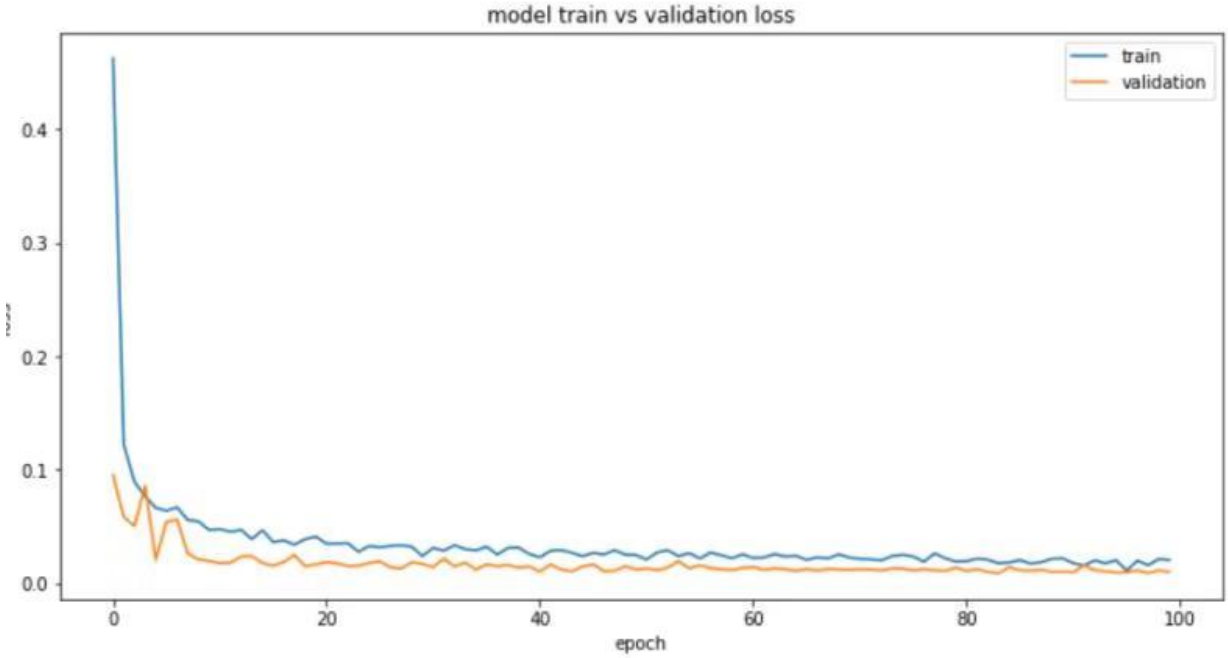

Figure 8

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 that the model did the wrong prediction. Take the first picture as the example. The actual number should be 4, but we predicted as 9.

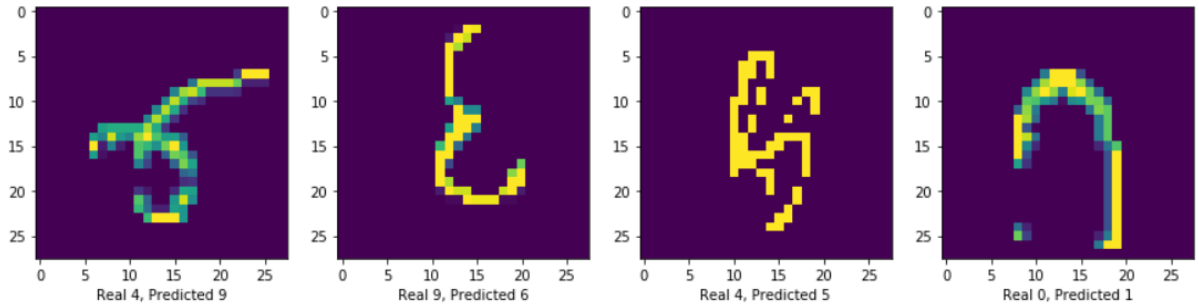

Figure 10

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 (shown on Figure 11)

314	zheca2	0.98540	2	7d
315	Atsushi Shimizu	0.98540	3	4d
316	Hanwen Zhu	0.98540	3	1d
317	Gaurav Gandhi	0.98540	8	16h
318	DUFE	0.98540	4	13h

Your Best Entry

Your submission scored 0.98540, which is an improvement of your previous score of 0.98200. Great job! Tweet this!

Figure 11

5. Summary and Conclusion

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.

In terms of where can be improved, we could do a more complicated model and select more appropriate parameters to improve accuracy. Besides that, for some special situation, I think we should deal with them differently.

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.

6. Calculate the percentage of the code that you found or copied from the internet

Based on the formula of the project requirement, the proportion of code we found on the internet is 18%.

7. 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