Individual Final Report

Junchi Tian

1. Introduction

For this final project, we worked on the classic MNIST competition we're all familiar with, and we used a recently released dataset of Kannada digits. 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. The following picture shows the figure of Kannada.

Figure 1

2. Description of your individual work

For this project, I mainly build a CNN model to recognize digits and try to improve this performance by hyperparameter tuning methods. This idea and proposal is completed by me. And I also make contributions to visualization, presentation and report.

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

To be specific, I write about 80% code including building model, compiling model, hyperparameter tuning, two visualization graph (loss and accuracy) and submitting result to Kaggle. I also find this dataset from Kaggle and write the proposal. I also contribute to slides for changing style and words. For the report, I mainly responsible for the result and try to explain what we get from graphs.

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.

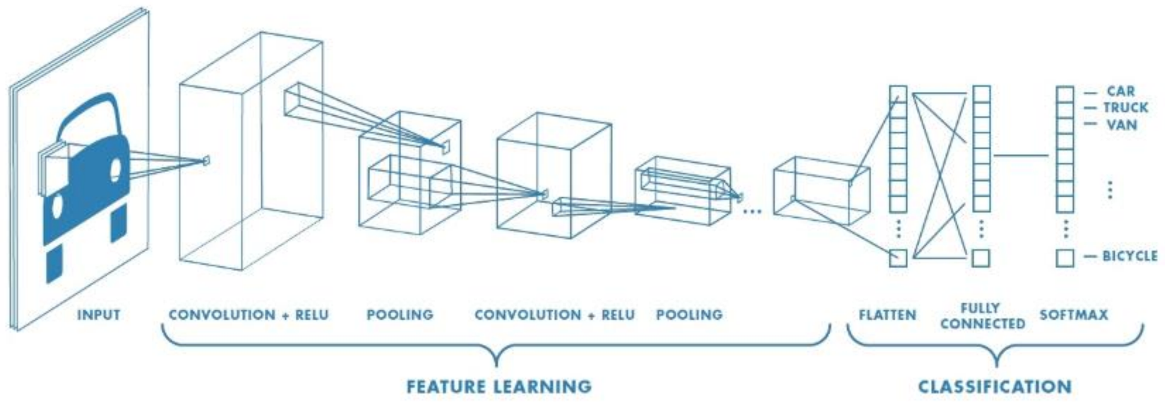


Figure 2

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.

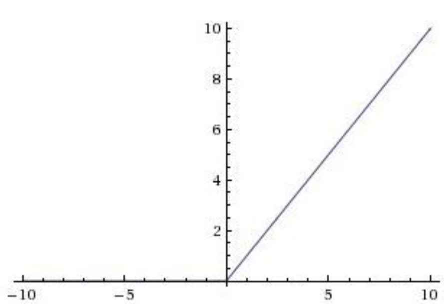


Figure 3

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.

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is better for our model because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network 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.

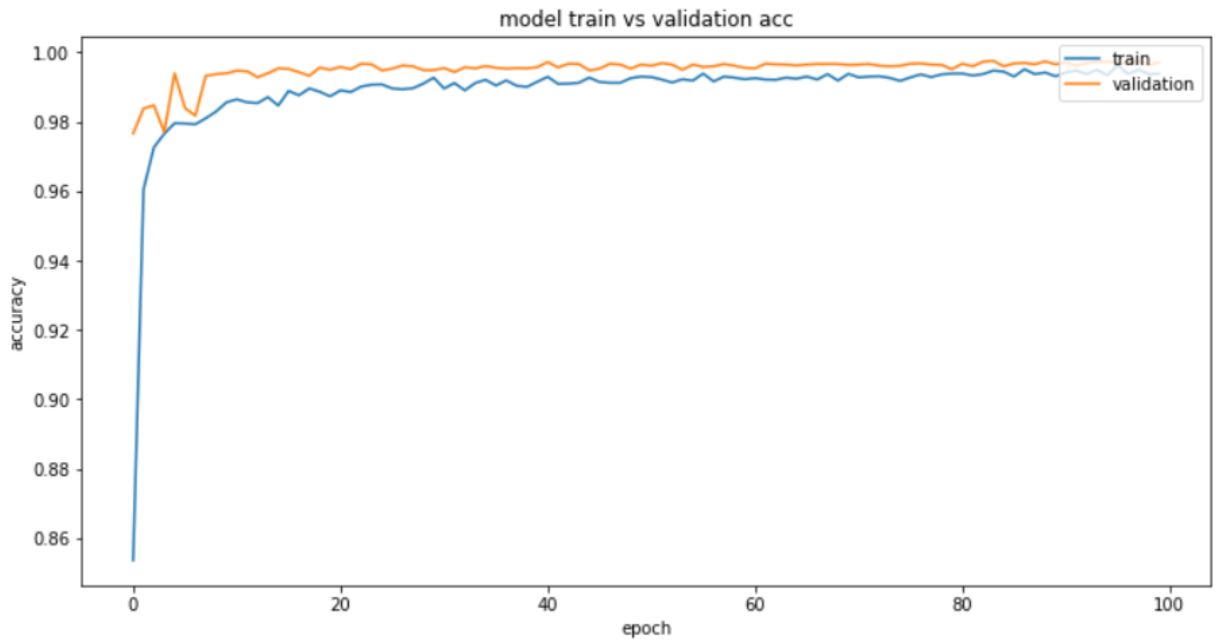


Figure 4

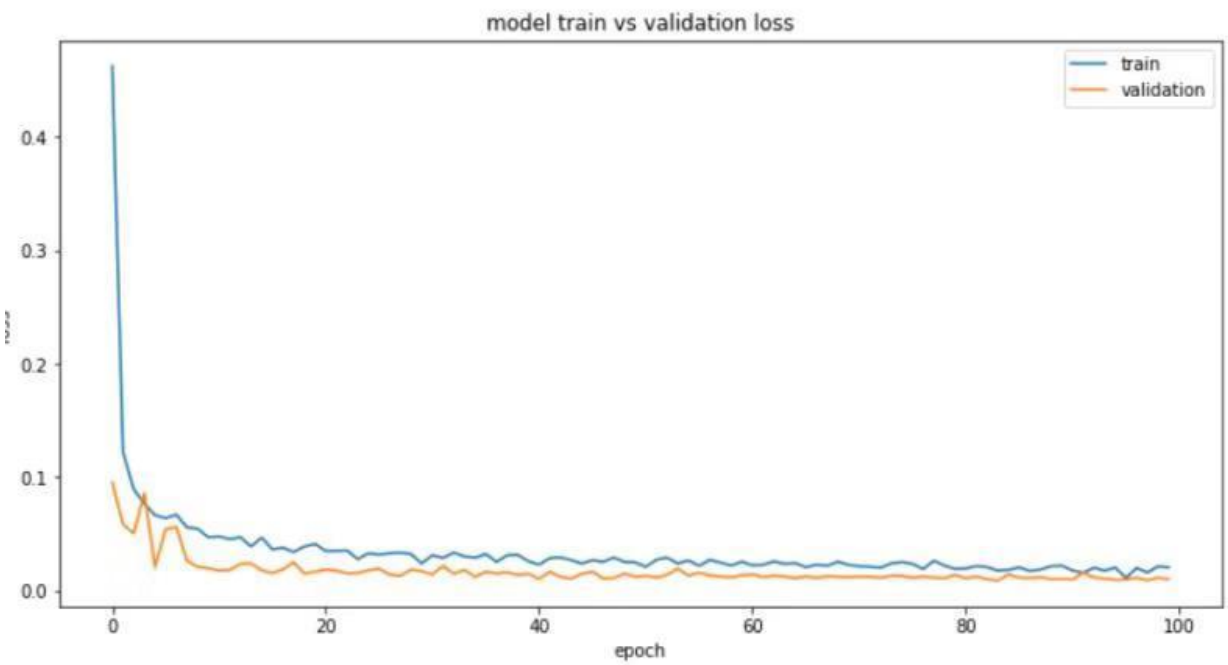


Figure 5

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 6

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.

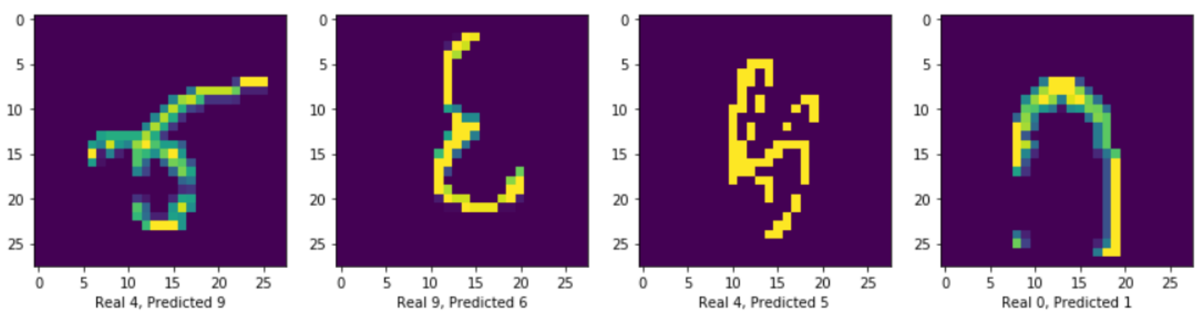
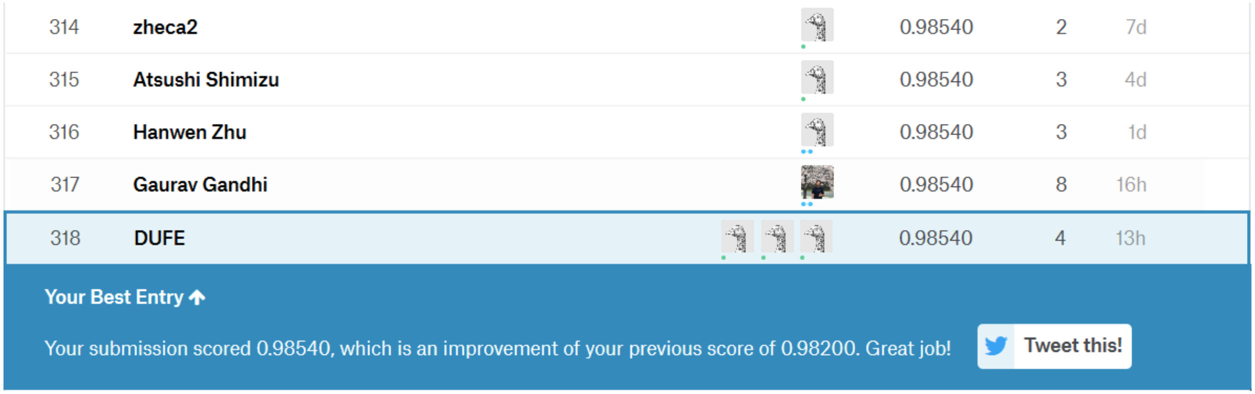


Figure 7

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 8

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.

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.

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

Part of code we get from internet is in the visualization part, learning rate reduction function and how to deal with batch normalization in the modeling. We learn the theory and how to implement these methods by python code from Kaggle notebook. So in total, about 18% code is from Kaggle notebook.

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