

# CS583A: Course Project

Mengjiao Zhang

December 1, 2019

## 1 Summary

I participate in an active competition of classifying a new handwritten digits-dataset, termed Kannada-MNIST [1].

The final model I choose is CNN, a 10 layer deep convolutional neural network architecture, which takes  $28 \times 28$  images as input and outputs the class labels.

I implement the convolutional neural network using Keras and run the code on a PC with 1 NVIDIA GeForce RTX 2080Ti GPU.

Performance is evaluated on the classification accuracy.

In the public leaderboard, my score is 0.98480; I rank 310 among the 947 teams. The result on the public leaderboard is not available until December 17, 2019.

## 2 Problem Description

**Problem.** The problem is to classify a new handwritten digits-dataset. Instead of using Arabic numerals, this uses a recently-released dataset of Kannada digits. It also contains 10 classes, including 0 to 9. So, this is a multi-class classification and image recognition problem.

The competition is at <https://www.kaggle.com/c/Kannada-MNIST>.

**Data.** The main Kannada-MNIST dataset that consists of a training set of 60000  $28 \times 28$  gray-scale sample images and a test set of 10000 sample images uniformly distributed across the 10 classes. The ten classes are shown in Figure 1.

**Challenges.** One of the challenges mentioned in the original paper is that *some of the images either have only a partial slice of the glyph/stroke or have an appearance where it can be argued that they could potentially belong to either of two different classes*. With regards to these images, we want to design a effective classifier to capture the more distinguishing features.

## 3 Solution

**Model.** The model I finally choose is a 10 layers deep convolutional neural network. I partition this model into 4 blocks of convolutional parts and followed by 2 dense layers. More details are shown in Figure 2.

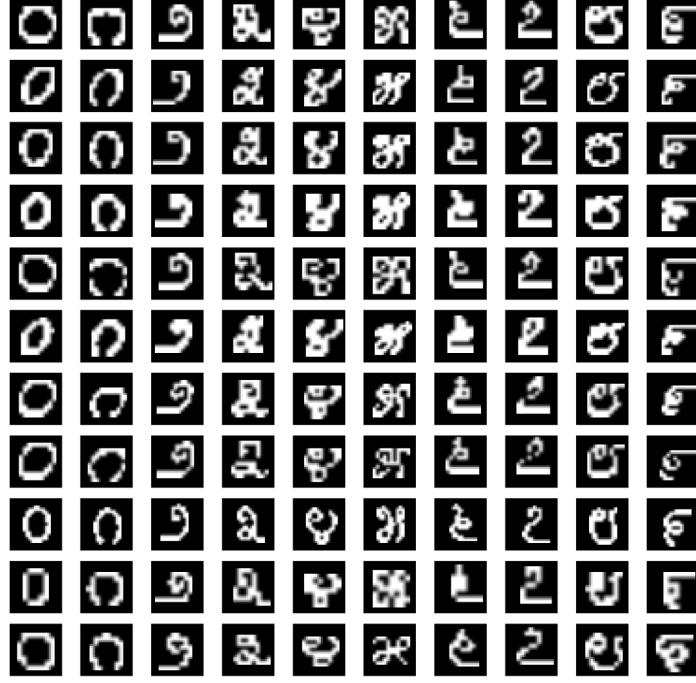


Figure 1: MNIST-sized renderings of the 0-9 Kannada numerals in 11 modern fonts [1]

**Implementation.** I implement the 10-layer CNN model using Keras with TensorFlow as the backend. My code is available at <https://github.com/MengjiaoZhang/CS583A/tree/master/project/code>. I run the code on a PC with an NVIDIA GeForce RTX 2080Ti GPU. It takes 8 minutes to train the model.

**Settings.** The loss function is categorical cross-entropy. The optimizer is Adam. The learning rate is 0.005. I train the model for 50 epochs and the batch size is 512.

#### Advanced tricks.

- Data augmentation
- Batch Normalization
- Dropout

**Cross-validation.** I simply partition the training data to 90%-10% for hyperparameter tuning.

Figure 3 plots the convergence curves on 90% training data and 10% validation data. The accuracies and losses on training and validation are basically the same, because I use data augmentation, batch normalization, dropout in the CNN model, which prevent the model from over fitting even if there are a lot of parameters in the model.

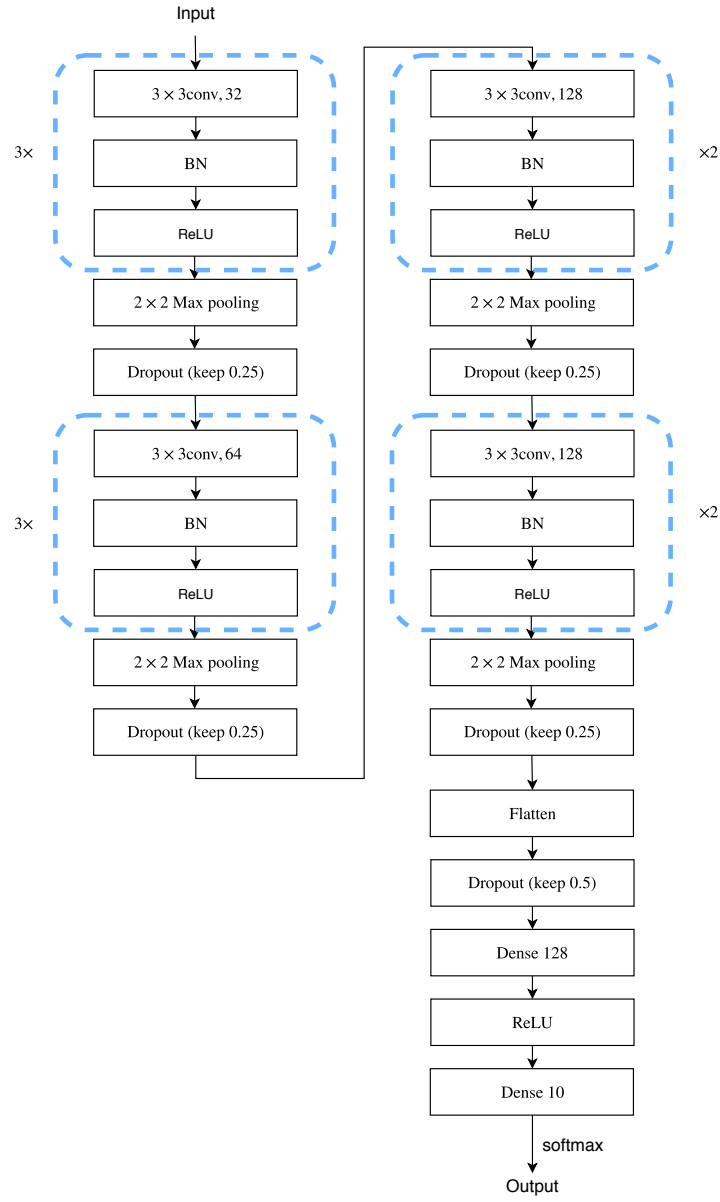
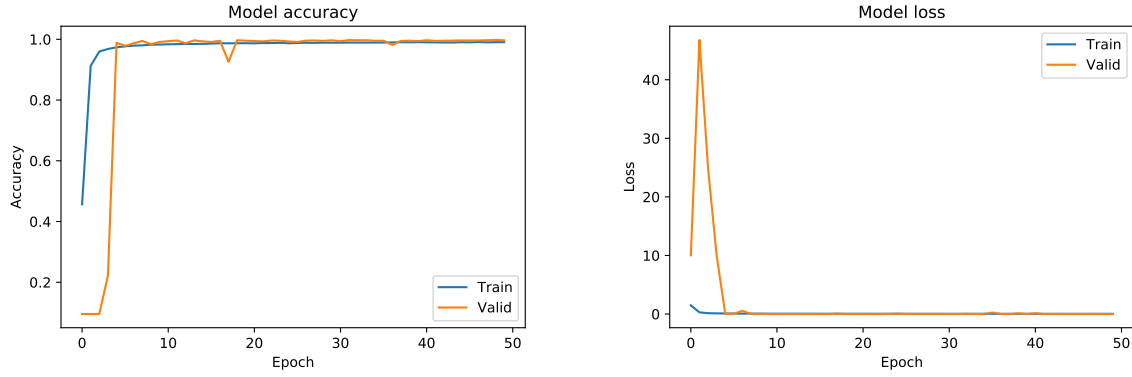


Figure 2: The summary of the model used in the task



(a) The classification accuracy on the training set and validation set. (b) The loss on the training set and validation set.

Figure 3: The convergence curves.

## 4 Compared Methods

**10-layer CNN** I design a 10-layer CNN model with keras. The training and validation accuracies are respective 99.06% and 99.67%.

**Fully-connected neural network.** I implemented a 5-layer fully-connected neural network. The width of the layers (from bottom to top) are respectively 64, 128, 256, 128, 64. I add Batch Normalization to each hidden layer. I train the model for 100 epoches. The training and validation accuracies are respective 95.81% and 98.48%.

**SVM** I use LinearSVM provided by Sklearn. The training and validation accuracies are respective 95.78% and 94.75%.

**Kernel SVM(rbf kernel)** I use SVM with rbf kernel provided by Sklearn. The training and validation accuracies are respective 99.38% and 98.85%.

**Advanced tricks.** My finally adopted method is a CNN with 10 convolutional layers. I applied the following tricks.

- Data augmentation.
- Batch Normalization(BN).
- Dropout

## 5 Outcome

I participated in an active competition. My score is 0.9848 in the public leaderboard. I rank 310/947 in the public leaderboard and the result on the private leaderboard is not available until December 17, 2019. The screenshots are in Figure 4.










306	Igor Tokarev		0.98480	42	14d
307	Shrisha h s		0.98480	8	17d
308	Kernano		0.98480	10	13d
309	Bong K		0.98480	6	14d
310	MengjiaoZhang		0.98480	5	6h
<b>Your Best Entry ↑</b> Your submission scored 0.94440, which is not an improvement of your best score. Keep trying!					
311	Pramod Mehta		0.98460	23	2mo
312	Kush Jajal		0.98460	8	2mo
313	Sujoy K Goswami		0.98460	1	2mo
314	zhou chen		0.98460	5	2mo

Figure 4: My rankings in the leaderboard.

## References

- [1] Vinay Uday Prabhu. Kannada-mnist: A new handwritten digits dataset for the kannada language. *arXiv preprint arXiv:1908.01242*, 2019.