Digit Recognizer using Convolutional Neural Network

For Machine Learning: CS6375.502 F18

Submitted By:

Rutuja Kaushike (RNK170000)

Jeel Christian (JSC180001)

Table of Contents

[Introduction 3](#_Toc531523925)

[Dataset 3](#_Toc531523926)

[Pre-processing Technique 4](#_Toc531523927)

[Proposed Solution and Methods 4](#_Toc531523928)

[Experimental result and analysis 5](#_Toc531523929)

[Conclusion 6](#_Toc531523930)

[Contribution of Team Member 6](#_Toc531523931)

[References 6](#_Toc531523932)

# Introduction

* This project is the solution for an ongoing Kaggle competition named as ‘Digit Recognizer’ and can be found at <https://www.kaggle.com/c/digit-recognizer>.
* The goal of this project is to identify digits from a dataset of tens of thousands of handwritten images.
* The solution has been implemented using the Sequential Convolutional Neural Network algorithm along with the Keras and TensorFlow.

# Dataset

* The dataset used for the project is MNIST dataset.
* The dataset is provided by Kaggle, which is present at :

<https://www.kaggle.com/c/digit-recognize>

* The dataset contains two data files namely ‘train.csv’ and ‘test.csv’.
* The dataset in train.csv and test.csv is MNIST ("Modified National Institute of Standards and Technology") which is the subset of the entire dataset available at NIST.
* The ‘train.csv’ has a set of 60,000 examples which are used to train the algorithm, whereas the ‘test.csv’ has set of 10,000 examples which are used to test the algorithm.
* As each image has 28\*28 pixel size, i.e. the height of the image is 28 pixels as well as the width of image is 28 pixel. So in total there are 784 pixels.
* Every pixel has value associated with it, the value can range from 0 to 255 indicating the darkness or lightness of the pixel. Higher number in the pixel indicates higher brightness.
* There are 785 columns in the .csv file, 784 for the pixel values of the image and the one is ‘label’ which represents the digit that is handwritten.
* The data in the .csv file is represented as follows:



# Pre-processing Technique

* Pre-processing techniques for example resizing the image, normalizing the pixels of image are implemented in this project. Below are the steps:
  + Step 1: The data is separate into predictors (784 columns that gives information about the image that is labelled as X) and label (the predicted output value that is labelled as Y).
  + Step 2: The data values are converted to float as we need to further divide it by 255 and obtain the value in the range 0.0 -1.0. Here the data is divided by 255 as RGB (Red, Green, Blue) are 8 bit colour each so the range for an individual colour is 0-255 (i.e. 2^8 = 256). Here 0 means 0(0x00) and 1 means 255(0xFF)
* As each row contains the data of single image, the images are flatten. Hence, we need to reshape them to get the image matrix.
* To reshape it, this solution converts pandas ‘data frame’ to NumPy ‘matrix’.
* The label Y should be convert to categorical format. That is, the label Y is converted into vector of form [0,0,1,1,0,1,0,1,0,0], the vector has 10 values.
* In this project, the training data further split into training data and validation data to train the model.

# Proposed Solution and Methods

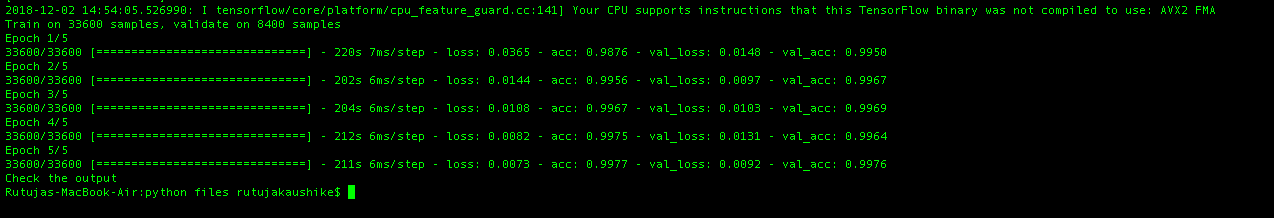
* In deep learning with neural network, one of the most powerful supervised deep learning techniques is the Convolutional Neural Networks (CNN). The structure of a CNN is to Regular Neural Networks (RegularNets) where there are neurons with weights and biases.
* The main structural feature of RegularNets is that all the neurons are connected to each other. For example, if the images are 28 X 28 pixels greyscale, it has 784 (28 x 28 x 1) neurons in a layer which is fine. But, normal images have more pixels and they are not grey-scaled, which is not fine for the RegularNets. Hence, RegularNets are not scalable for image classification. This leads to the idea of Convolutional Neural Nets.
* Hence, convolutional neural network is used for the image classification.
* Here a 2-D convolution neural network followed by MaxPooling and full connected layer a softmax layer is used.
* To extract features of different types, at each convolutional layer we have used multiple filters.

For example: First filter helps in detecting the circle, second filter helps in detecting line and so on.

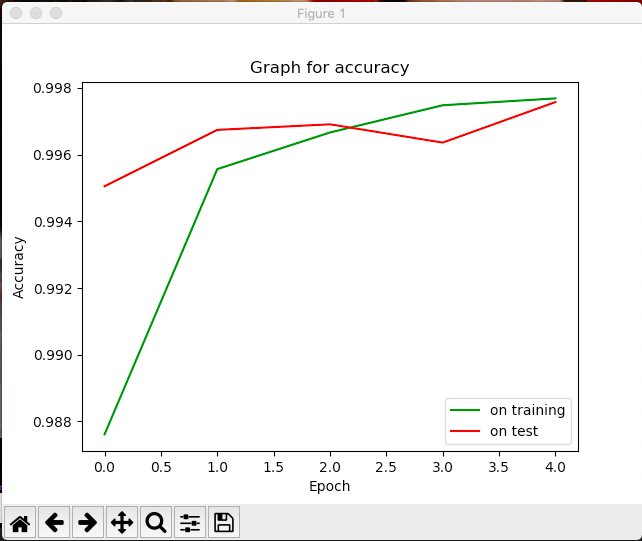
* For regulation to reduce the overfitting in model, dropout has been introduced before softmax layer.
* Then, as the model is ready, it is compiled.
* Loss function used here is categorical\_crossentropy as our problem is multi class classification.
* Performance metric is preferred for accuracy as all the labels have equal weight.
* To optimize the parameter the RMSprop optimizer is used, as it is the best optimizer for recurrent neural network.
* Now the model has been defined and compiled, it is trained by passing the training data (the training data and validation data) so that the model is trained to recognise the hand written digit.
* Hence, the training it fitted to the model.
* Epoch means one forward and one backward pass on the training data.
* Batch size indicated the number of data to be included in one forward or one backward pass.

# Experimental result and analysis

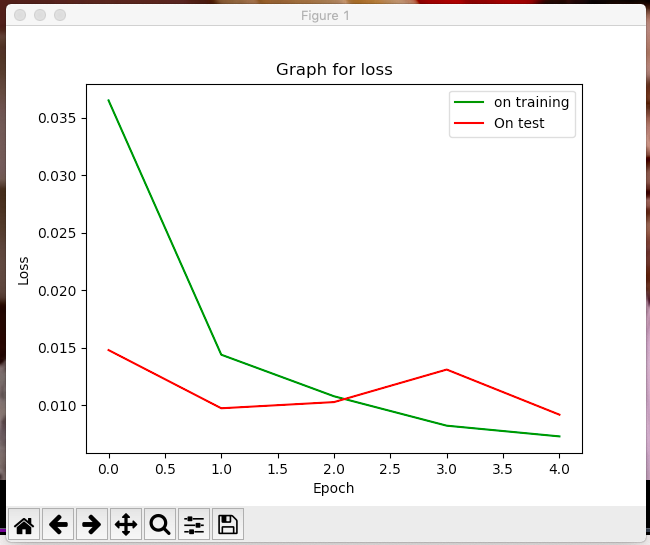
The training data output log is as follows:



The accuracy graph:



The loss graph:



# Conclusion

* The model is trained very well for the prediction as the test accuracy is 100%.
* The loss and accuracy on the training and test of the model converges as the number of epochs increases

# Contribution of Team Members

* Both the team member were included in all the following task:
* Understanding the project topic
* Building the project timeline and plan
* Analyze the data
* Pre-processing the data
* Working on the selection of the project model and tunning parameters
* Creation of Project Status report
* Creating the final project report

# References

* <https://keras.io/optimizers/>
* <https://www.coursera.org/learn/convolutional-neural-networks/home/info>
* <https://cs231n.github.io/convolutional-networks/>
* <https://www.tensorflow.org/tutorials/estimators/cnn>
* <https://scikit-learn.org/stable/index.html>
* <https://www.kaggle.com/c/digit-recognizer>