Ahmed Refaay

900141806

Deep Machine Learning

Assignment 2

10/30/2017

Fully Connected Neural Network

**Project Description**:

This project implements a fully connected neural network using the Keras and Tensorflow frameworks on Python 2.7.

**Code Description**:

The code is composed of three main parts: preprocessing, neural network implementation, and accuracy presentation.

1. Preprocessing:

The data is loaded from the Keras framework using the load function. It is converted to float32 to ease float calculations. Then it’s zero-centered and normalized. The labels are converted to one-hot encoding. The “ImageDataGenerator” function is used to generate augmented data to get better training for the neural network.

1. Neural Network:

The neural network is a four layer network.

* Input layer: size = length of flattened data (40000).
* First hidden layer: 2500 neurons, activation function = relu, used batch normalization to normalize data for all neurons, and a drop out percentage of 0.2.
* Second hidden layer: 1500 neurons, activation function = relu, used batch normalization to normalize data for all neurons, and a drop out percentage of 0.35.
* Third hidden layer: 1000 neurons, activation function = relu, used batch normalization to normalize data for all neurons, and a drop out percentage of 0.2.
* Output layer: 10 neurons and activation function = softmax.

ReduceLROnPlateau: reduces the learn rate to avoid saturation.

fit\_generator: runs the neural network for training and validation.

model.evaluate: evaluate accuracy on the testing set.

1. Accuracy presentation:

Summary function prints the neural network formation.

TensorBoard function: makes files for graphs than can be viewed in an internet browser.

Both the test loss and accuracy are printed.

Calculating CCRn by using “predict\_classes” function and comparison with the test labels.

The program prints the accuracies for each class.

**Network architecture and hyper-parameters’ tuning**:

1. Choosing number of layers:

The number of layers was chosen to avoid linearity and overfitting at the same time. By testing one, two, three, four, and five layers, changing the numbers of layers and making the other parameters constant, it was found out that three hidden layers get the best accuracy.

1. Choosing the number of neurons per layer:

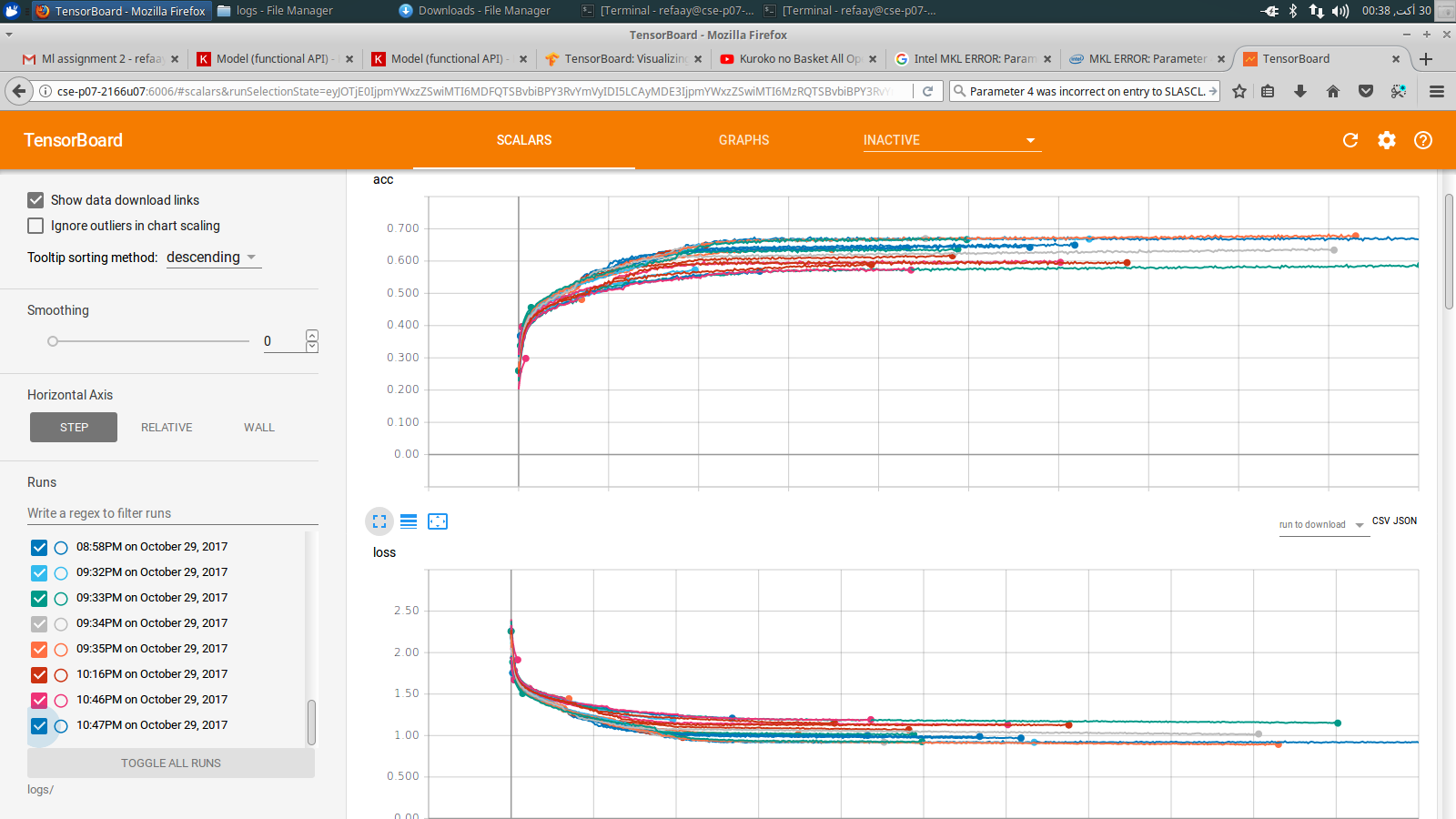
The limits are the sizes of the input and output layers to make them functional and avoid bottle-necking. Each layer also needs to be larger than the one that follows it. Doing many experiments by those aspects, the numbers given were chosen.

1. Learning rate:

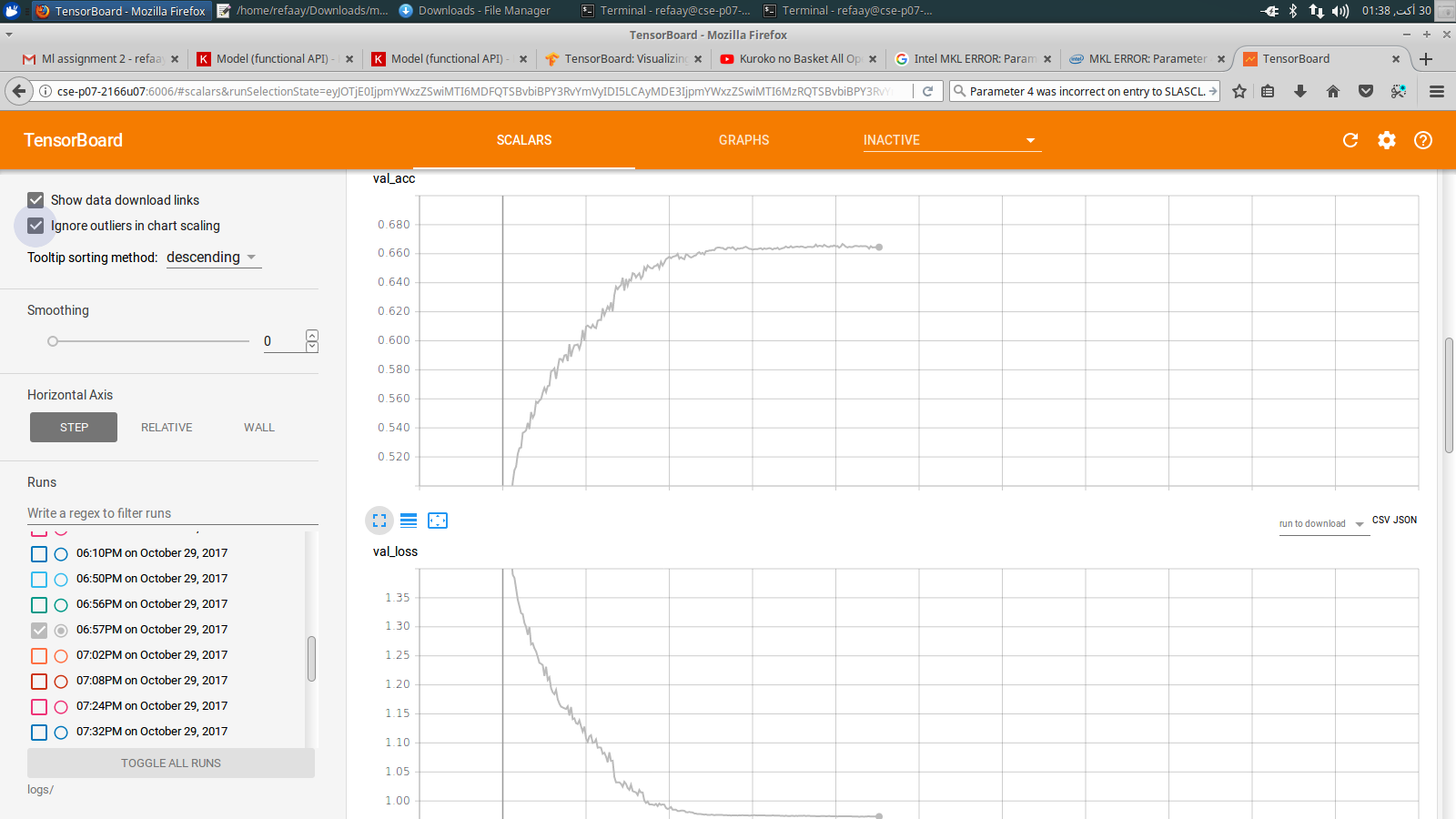
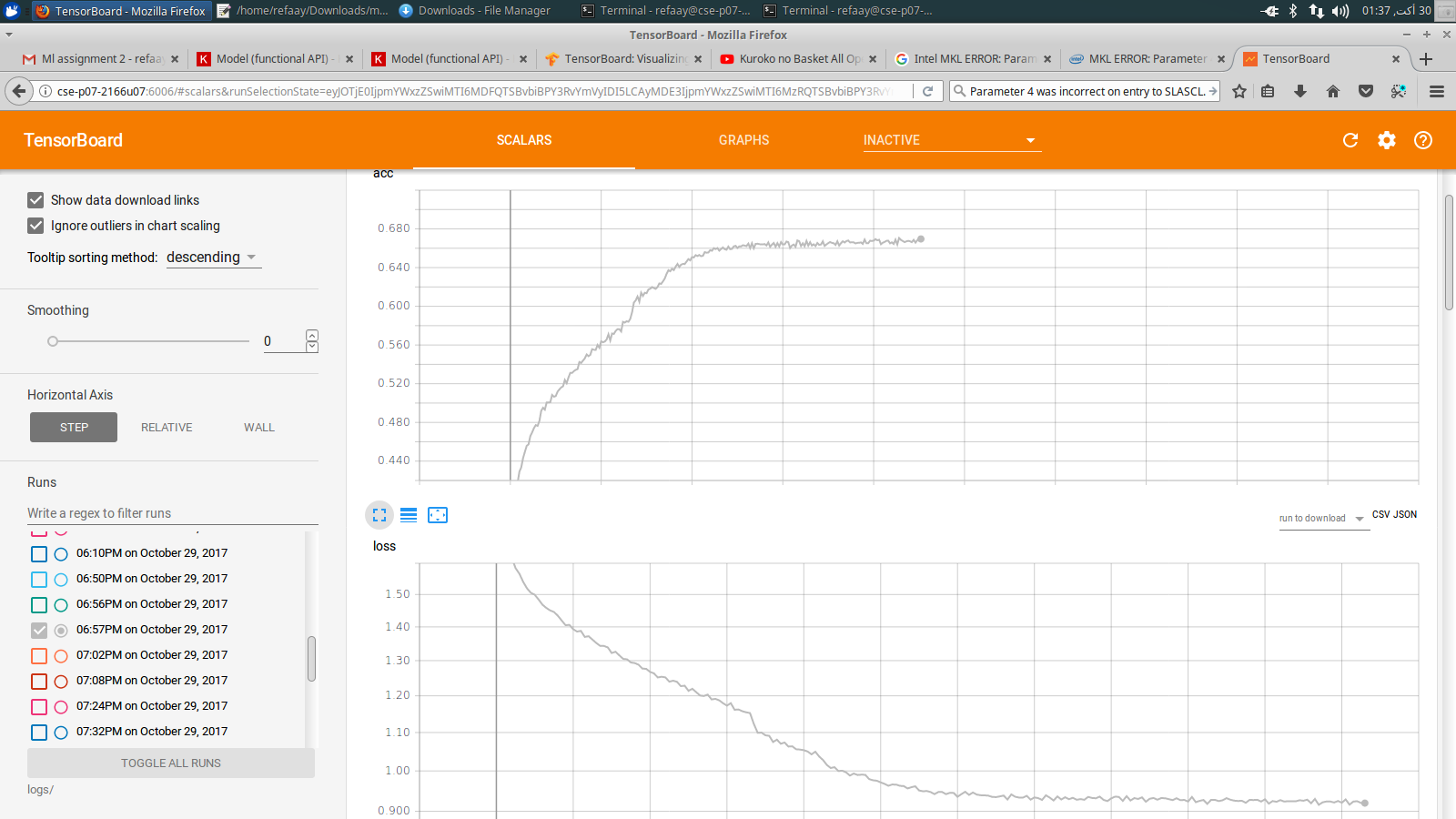
Like the previous parameters, a large learning rate will cause divergence and a very small one will make it slow and not functional. Chosen by experiments.

1. Other parameters also by testing and tuning, like batch size, dropouts, decay rates, … Test data are in the attached screenshots.

**Training and validation accuracies and losses plots (overall)**:



**Best accuracy and losses plots (for around 225 epochs)**:



**NN CCRns**:

CCrn of airplane is 0.709000  
CCrn of automobile is 0.783000  
CCrn of bird is 0.513000  
CCrn of cat is 0.463000  
CCrn of deer is 0.504000  
CCrn of dog is 0.561000  
CCrn of frog is 0.732000  
CCrn of horse is 0.720000  
CCrn of ship is 0.733000  
CCrn of truck is 0.732000

**KNN CCRns**:

Class 1 = 0.535000

Class 2 = 0.070000

Class 3 = 0.438000

Class 4 = 0.089000

Class 5 = 0.493000

Class 6 = 0.165000

Class 7 = 0.289000

Class 8 = 0.142000

Class 9 = 0.664000

Class 10 = 0.107000

Got 2992 / 10000 correct => average correct accuracy: 0.299200

Got 2992 / 10000 correct => accuracy2: 0.299200

**NN ACCR**:

Test accuracy = 0.64500000000000002

**Some experiments secrrenshots**:

