Project 2 Report

By: Vaibhav Rao
Email: Vaibhavr@buffalo.edu
Person No.: 50375332

In this project I have created a Deep Learning model using Keras for the dataset of German Traffic Signs (https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign)
The dataset had separate Image sets for Training and Testing purpose.
The project has been enclosed in the zip file as per specifications mentioned and include the file main.py.
NOTE: The code was developed using TF 2.x and Keras 2.x and might throw some errors on lower versions especially with the Keys of the History Object.

TASK 1 (Data Partiotioning)

For Task 1, our dataset already had different training images for Test and Train. We can see the details in Test.csv and Traing.csv.
Further, I have Partitioned my Training dataset of around ~40,000 images into Validation data and Train Data. The partitioning is carried out in the ratio of 20:80 respectively.
Since the File Paths of a particular image set were adjacent (See Train.csv for details) we randomly shuffled the data set before splitting it into Train and Validate at line 33-37.
We have extracted the values from the csv file into a pandas dataframe.
We only required 2 columns, "ClassId" for the value of Y as this is the target class.
And the "Path" of the image file to load it using OpenCV (cv2).

TASK 2 (Design the model)

I have designed and implemented the new model and then used compile and fit_generator to train the model on the X_train data. We also supplied the model with Validation data to for it to correct between epochs.

TASK 3 (Evaluate Performance using Accuracy Scores and Confusion Matrix)

The Scores and matrix are Printed at the End of the model Once we use the trained model to determine the model. My Code has 2 models which I finalized to be the best ones out of the multiple Trial-and-error I tried with various Parameters.
Model Sequential:
3 Convolution Layers with Filters : 32,64 and 128
Trained in 24.13 minutes
The Accuracy and Validation Accuracy Scores are plotted on the graph.
The Loss and Validation Loss scores are also plotted on the graph.
Training Data accuracy: 99.29%, Testing Accuracy: 97.06%

TASK 4:

Now I tried multiple Things to come to the conclusion of the model and Finally achieve an Accuracy of 99% and testing Accuracy of 97% all this on the 24 minutes it took to train the model on my machine.

I also Finalized 1 more model that had a Slight better accuracy and Slightly lower loss (Commented in the code, and log output attached below)
This Model had 3 convolution layers(of filter size =64, 128, 256) and similar Maxpooling/Dense layers. However this took almost double the time with not a considerable increase in accuracy.
Trained in 49.63 minutes
Results at the end of training and using model.evaluate on test data: acc=99.48%, test_acc=97.22%

Now For trial, I first tried loading and plotting the image with different Sized. I found most images in the size of 25-40 as per the statistics in the dataset resource and found the pixel size of 40 to be the best in terms of less pixelated images (Line 51).

I also Found that a smaller Batch Size was working better in terms of accuracy without having any impact on the time of the training. I tried Sizes 32,256,300 for Batch_size and finalized 32. Also having a smaller Batch size resulted in lesser Loss of the model, Larger batch sizes had almost similar accuracy but the loss was more than on smaller sizes

For the model, I tried different layered approaches as below:

Having 4 convolution layers in an adjacent placement (2 + 2), i.e. 2 convolution layers followed by a maxpooling/sampling layer then again 2 conv layers followed by maxpooling and dense layers. However I found this model to take a lot of time (Around 15 minutes for 3 epochs) and hence tried different scenarios where the model was more quicker and more accurate as well.

I also tried similar changes In Activation Functions like using Tanh instead of relu for some layers but I already achieved a 99% accuracy with this model.

During my trial of different models, I found Increasing the 2D convulation layer from 1 initially and I saw better performance with 2 or 3 layers, 3 being the best.
For filter sizes of these layers, I observed that having a smaller filter size (5X5 or 3X3) was sufficient since the images in the dataset were not very large images.

I also tried various values for Filters in the Conv layers. I noticed having same filter sizes across layers did not increase the accuracy and that was getting stagnated at the same value of about 98% near epochs 13-15.

Finally I print the Output of using evaluate_generator. This function return loss at position [0] and accuracy at position [1]. Since for test data we want to evaluate accuracy so have assigned [1] to the test_acc variable and have print that (linne 178-179)

Also to have the confusion matrix We used model.predict_classes. We can see Both these methods would return the same accuracy (Line 187 and line 189).

We have also printed the confusion matrix obtained from this.