CSE 535: Mobile Computing
Assignment -3

Karthik Chadalavada(1225628313)

IMPLEMENTING EDGE COMPUTING TO PREDICT A HANDWRITTEN DIGIT WITH THE USE OF TRAINED MODELS(TO PREDICT PARTS OF IMAGE) ON DIFFERENT SERVERS

Overview:

This project encompasses creation of an android application using Android Studio and Flask server for the backend. Each image from the MNIST Training dataset is divided into 4 parts. 4 different ML models are trained on these 4 parts of the image and stored on the 4 servers respectively. When the android device captures a handwritten digit, it sends the image to the main server which in turn divides the image into four parts and uploads them to four different servers. The servers process the images and send this predicted number to the main server. The main server consolidates all the predictions made by these servers and stores the image in the respective folder.

Application Level:

The android application is developed by reusing the code from the phase 1 of this project. The code has been updated by removing the Category spinner from the application and the HTTP request. Initially, the user is presented with two buttons to select an image from the gallery or to capture an image from the camera. After selecting the image, "Next" button is enabled. After clicking on this button, a new intent is launched to navigate to a new activity which displays the image selected in the previous activity. The user is presented with an "Upload" button in the second screen of the application. If the image is successfully uploaded to the server, a new activity is launched to display that the image is "Successfully uploaded" to the server. If the server is not available, then a Toast message is displayed. After the image is uploaded to the server, the machine learning algorithm predicts the number and saves the image in the respective folder.

Server Level:

After successfully receiving the image from the android device, the image is first preprocessed. The image received from the application is initially saved on the server. This image is then read using the opencv python library. Since the ML model is trained on black and white images, the image is then converted into grayscale using "imread" method. This image is then resized to 28 X 28 pixels. This image is converted into 4 parts of size 14 X 14. These subparts are stored in 4 dictionaries. Then, 4 POST requests are made with these dictionaries as data to the servers which are running on **4 different machines** to imitate EDGE COMPUTING. These 4 servers convert the image into nparray and then reshaped into the input shape(14 x 14) of the model. This n-dimensional image is fed into the model, which is loaded from the saved model by the

deepLearning.py code, which returns the predicted number to the main server. Once the main server receives response from the 4 servers, the most predicted number is considered as the final prediction. Then the server then checks if this folder exists, and if it does, moves this pre saved image into the respective folder. Else, a new folder is created and the image is then saved in that folder.

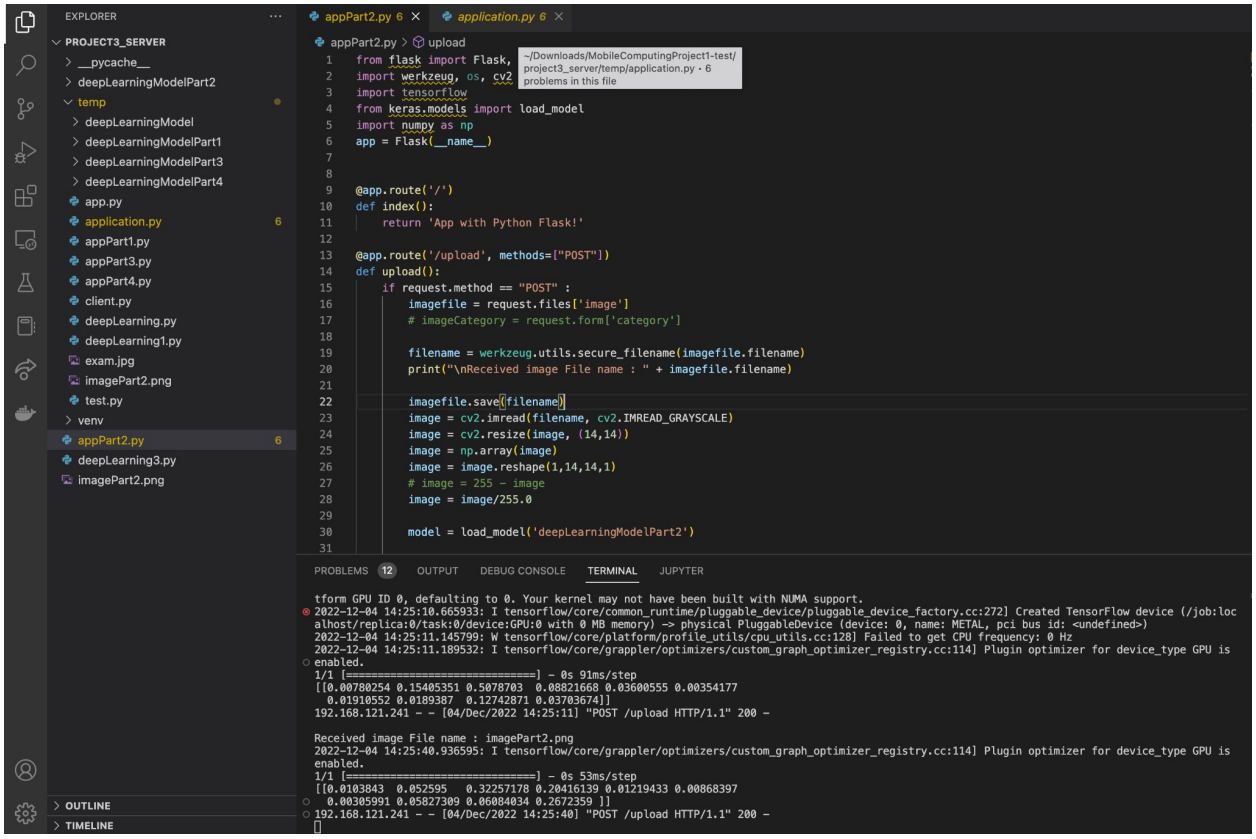
Machine learning:

A machine learning algorithm is implemented to predict the number from the image received. Initially in the machine learning code, the training and test data is loaded from mnist dataset which is imported from **keras** library. This training and test data is then reshaped into 28 x 28 pixels and categorized into 10 classes. Then the training and test images are divided into parts of 14 x 14. This input data is converted into float and then is normalized by dividing each of the pixel data with 255. For this machine learning model, a Sequential model is implemented. We found out that for numOfEpochs = 10 and batch_size = 32, optimal results were obtained. Then with the activation function as 'relu' and optimizer with learning rate as 0.01 and momentum as 0.9, the model is compiled. After compiling the model, the model is fitted with the training data. This part is considered as training the model. After training this model on batches of size 32 and for 10 epochs, the model is trained with an accuracy of 0.92. This model is then saved in the local server as "deepLearningModel1", "deepLearningModel2", "deepLearningMode3", "deepLearningModel4". These 4 models are used by the 4 servers respectively to predict the number from part image. This model is further used by the server to predict the number uploaded by the android application.

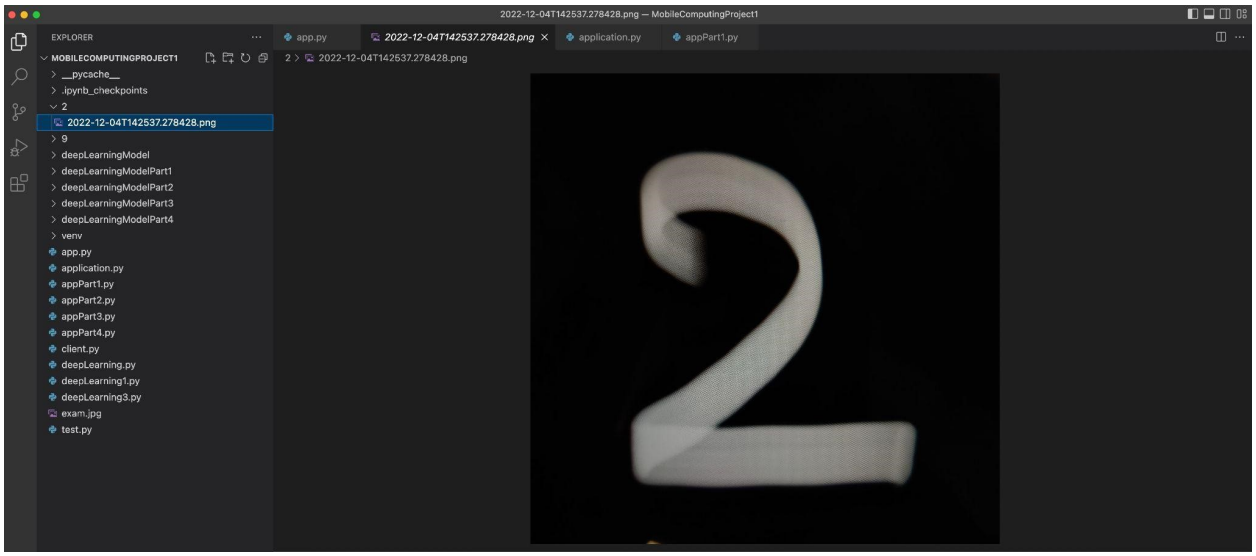
Results:

These are the screenshots of a few numbers classified using the Machine learning model to their appropriate folder.

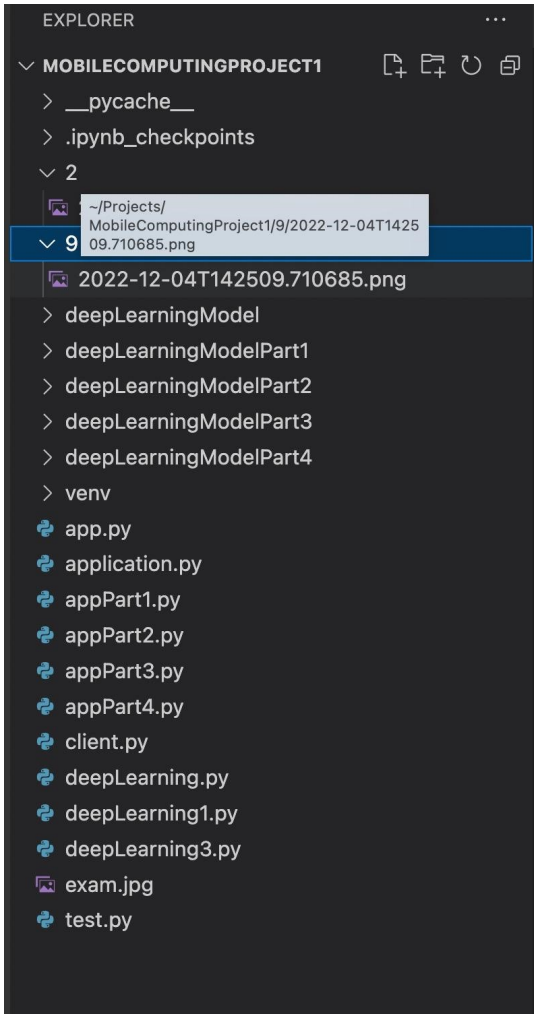
Machine learning model training on the data loaded from mnist dataset.



Server running on one of the 4 devices.



Predicted image on the main server



Folder Structure of the main server.