# CSE 535 Mobile Computing

# Project 3 : Group 5

Yashwanth Kumar Tirupati: 1225424512, Tejaskumar Patil: 1225501316,
Baibhav Phukan: 1225408392, Pranav Toggi: 1221278773, Aditya Goyal: 1225689049

**Introduction:**

The project's goal is to use the android application created in project 2 to upload images of handwritten digits captured by the camera, divide the image into 4 quadrants and send each quadrant to a different client mobile device. Subsequently, each client mobile classifies their individual quadrants and sends the result to master mobile, master mobile aggregates digits probabilities and classifies the digit, then the classification result is placed into a folder on the master mobile. The project involves building a deep learning model to classify handwritten digits by training it using the MNIST dataset.

The project uses the TensorFlow python library to build a deep learning model and Pillow libraries for image processing. It also uses the NumPy library to perform array operations and tensorflow lite to save the trained deep learning model in a file.

**Application workflow:**

1. The application allows the user to click a photo of the handwritten digit.
2. The image is then preprocessed by first converting to gray scale and then inverting the image and resizing into 28*28 pixels.
3. This image is then split into quadrants of 14x14 each.
4. Then the pixel value are normalize to be in range [0,1] to match the input format of the model
5. The image to be classified is passed to each client mobile along with a different quadrant number.
6. The client mobiles classify the image quadrant using their respective classification models and return 10 probabilities (1 for each digit) to the master client. Thus, the master receives 40 probabilities in total (4 for each digit).
7. The master consolidates the probabilities by summing up the probabilities for each digit. The maximum probability digit is considered the final classification result of the image.
8. The final result and the original image are stored in a folder named as the label is created within the download folder in the master client mobile.

**Communication Between the application and server:**

The main application offloads the task of classifying the captured image of a handwritten digit to four mobile servers running on different mobile phones. When the image is captured and uploaded from the main application, the application splits the image into four equal parts and sends each to one of the four servers using TCP sockets to communicate.

Each mobile server uses socket servers to listen to incoming requests on a predefined port. The main application opens a new socket connection to each server, passes the appropriate image chunk to the server, and receives the deep-learning model predictions for the image as the response.

**Deep Learning Model for classifying handwritten images:**

The model we developed uses convolutional neural networks (CNNs) because they are better at capturing the spatial dependencies in an image. In classifying images, CNNs perform much better than traditional feed-forward neural networks because CNNs can efficiently extract dominant features and suppress noise factors to a great extent. Adam optimizer was used on the categorical cross-entropy loss function for training the model because it has a faster run time and low memory requirements and requires less tuning than any other optimization algorithm. The training was done in input batches of size 256. The model was trained for 20 epochs employed with early stopping using validation loss to prevent the network from overfitting the data. Images of MNIST dataset were split into 4 small images of size 14*14 each and were provided to 4 different identical models (one for each quadrant). The average accuracy of the models when tested on the test data from the MNIST dataset was estimated to be around 85%.

This model is saved to a file so that it can be loaded to the Client Android App to categorize the received images using the tensorflow lite library.

**References:**


1. http://android-er.blogspot.com/2014/08/bi-directional-communication-between.html
2. https://www.tensorflow.org/datasets/keras_example
3. https://www.tensorflow.org/tutorials/images/cnn
4. https://www.tensorflow.org/lite/guide/inference#load_and_run_a_model_in_java
5. https://www.tensorflow.org/lite/models/convert/convert_models#python_api
6. https://developer.android.com/studio/run/emulator-networking