CSE 535 Mobile Computing - Project 3 - Group 14

Team Members

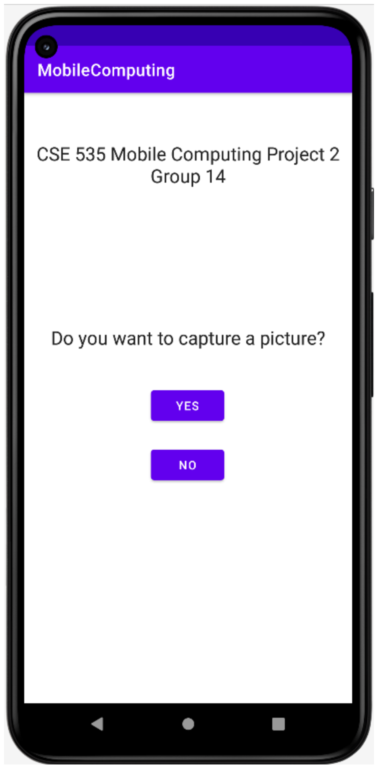
Sai Krishna Reddy Cheruku	1222300703	scheru13@asu.edu
Mahesh Chandra Yayi	1224141347	myayi@asu.edu
Fenny Zalavadia	1221443561	fzalavad@asu.edu
Shilpitha Gandla	1224631798	sgandla2@asu.edu
Ajay Kannan	1219387832	akanna14@asu.edu

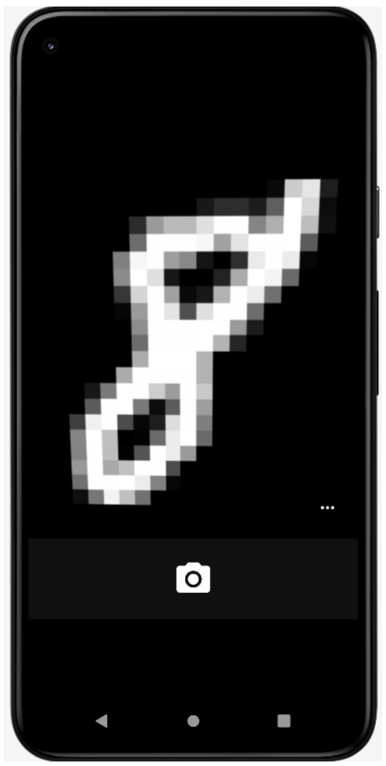
Overview

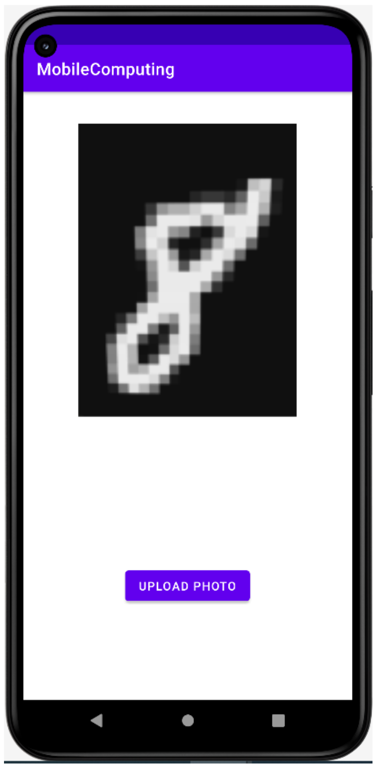
The master device captures an image. It divides the image into 4 quadrants. Each quadrant is sent to 4 client devices. The main server has the model that trains on MNIST quadrants and generates a trained model which is saved to the client devices. Each client device performs a forward pass using the trained model which gives the confidence values for each class. Hence, confidence value is an array of 10 probabilities and the prediction value is the class that has the highest confidence value. These confidence values are sent to the master device and the master device calculates the final prediction value from the four sets of confidence values. This is calculated by summing the confidence matrix across all the classes (i.e, column-wise matrix operation) and then calculating the maximum value of the resulting array. This result is sent to the main server and it saves the images into the respective class folders.

Android Application

The functionality of the android application developed in the previous assignment has been extended to accommodate offloading to client servers for ML-based prediction for the quadrants of the MNIST digit images. To reiterate the functionality of the previous version of the application, it consists of two pages. In the first page the user is asked whether they want to capture a photo or not. The user is provided with two buttons namely yes and no. If the user clicks on the yes button, they are navigated to the camera to capture a photo. If the user clicks the no button they are exited from the app. These activities are implemented with the help of listeners which are initialized when the app is started. So whenever the user clicks on these buttons the corresponding methods are triggered. After the user clicks on the yes button they are taken to the camera functionality.[1] Here they can capture a photo. Once they capture a photo they have the option to confirm or recapture. Once they confirm the photo they are taken to the second page of the application. The camera functionality is provided with the help of the underlying camera application which is in the operating system. This is done with the help of camera intent. This photo is initially loaded into a bitmap class variable. Later the bitmap photo is converted into a byte array which is sent to the server for prediction of the digit using an ANN pre-trained on the MNIST dataset.

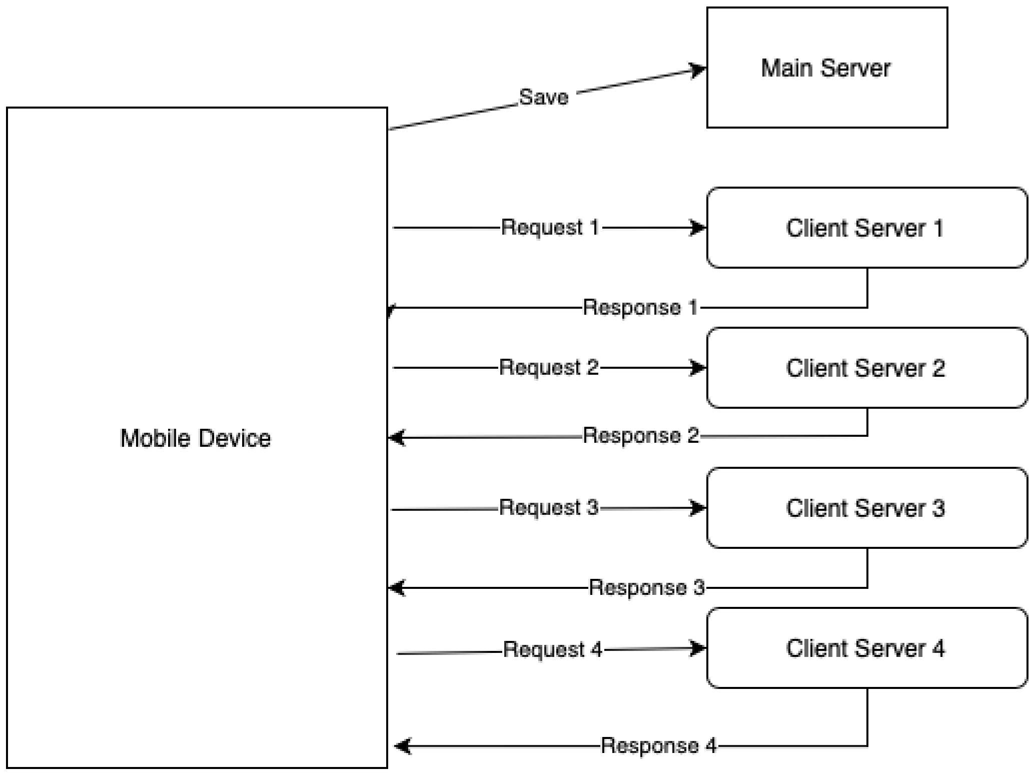

MobileComputing
CSE 535 Mobile Computing Project 2
Group 14
Do you want to capture a picture?
YES
NO





In the second page the user can view the image that they just captured. This is done using image view class. Then the image having shape (28,28) is divided into 4 quadrants resulting in 4 quadrants with shape (14, 14)[5]. Then the application prepares 4 request objects and it sends 4 post requests each having different images. These post requests are sent to 4 different servers located in 4 different machines. After the flask servers apply the model on the image they send back the array of confidence values as the response. After getting response from the 4 servers, we add up the confidence values for each digit.

After adding the 4 confidence values for each digit 0 to 9 from the different servers, we see which digit has the highest. This highest digit tells us which folder we need to save the images in. After this we send the image to a local flask server on the same system. This server is only needed to save the images in the filestorage and has no other function. If the folder does not exist it will create the folder for that category and save inside that folder.



Flask Server

The backend was developed using Flask server and python. Flask server is defined as server software that is capable of running HTTP requests on the public world wide web, private LAN, and private WANs and comprises one or many computers bundled together and dedicatedly working for running the software application on the worldwide web. We have built the Deep learning model using MNIST dataset by splitting an image into 4 halves. The ML model implemented and trained in the flask server is described below.

Specifications of the DL Model

The model architecture comprises a Dense layer of 256 hidden units with ReLu Activation layer followed by another dense layer of 100 units and kernel initializer, final layer with 10 neurons with softmax activation. We fit the model with 20 epochs and 512 batch sizes resulting in test accuracy of 81.8%. The model was saved and used by each client to predict the handwritten image received. The change here is that first the dataset had to be slightly modified. Each image is split into 4 parts each of size 14x14. These splitted images have the same label as the original image.

Client Device

The client device is running a flask server. The client device is where the model will be loaded and used to predict the digit. The server receives a post request from the App via internet - local connection.[3] The server receives a part of an image consisting of a handwritten digit, which is then converted into grayscale image. The conversion is important, because the dataset used to train the model was in grayscale image scale. Now, this image is normalized and converted into feature input on which the trained model is called to predict. Based on the prediction, the array of confidence values will be returned to the user. The android application receives the array of confidence values from the server and stores the image in the desired category folder.

References

1. https://developer.android.com/training/camera/photobasics
2. https://developer.android.com/develop/ui/views/components/spinner
3. https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/
4. https://www.geeksforgeeks.org/dividing-images-into-equal-parts-using-opencv-in-python/
5. https://oleksandrg.medium.com/how-to-divide-the-image-into-4-parts-using-opencv-c0afb5cab10c