Visual and Lingual Emotion Recognition Using Deep Learning Techniques

A Project Report

Presented To

Dr. Chris Pollett

Department of Computer Science

San Jose State University

In Partial Fulfillment

of The Requirements for the Class

CS297

By

Akshay Kajale

December 2020

ABSTRACT

Emotion Recognition is one of the most researched topics in modern day machine learning arena. We are developing a prototype of emotion recognition system by making a hybrid model using computer vision and natural language processing techniques. Our goal hybrid system would use video feeds of different facial expressions and speech features to recognize emotions. Our prototype will operate on videos created in Unity 3D humanoid model performing different facial expressions. Currently the unity model is expressing seven expressions namely angry, disgust, fear, happy sad, surprise and neutral. In the future, model will also articulate words and sentences for the deep learning model to predict the emotions.

This semester we divided the initial phase into four deliverables which will help us towards development of our final system. The first deliverable was to develop two deep learning models which can recognize emotions based on facial expressions using different techniques namely sequential and depth-wise convolution. Depth-wise convolutional model was selected for the final prototype purpose based on accuracy and number of calculations. In the second deliverable, we generated a synthetic dataset using Unity. The dataset contains 51 videos, each of 20 seconds in which our Unity 3D model is carrying out various facial expressions. This dataset will be used to train and test our deep learning model in the future. We developed an Android application in the third deliverable which can access both cameras simultaneously. The deep learning model will be deployed on this application so that we can detect emotions of person facing both front and back camera. Lastly in the fourth deliverable, model developed in the first deliverable was deployed on the application for initial testing. The keras model was converted into tensorflow-lite for the deployment. The model was tested on static images on the application. The model predicted accurate expression on static image. In the future, we will work on getting live camera frames from android application and start working on lingual feature detection of the model.

Index Terms: Deep Learning, Sequential, Lingual Features, Sequential Model, Depth wise convolutional model.

INTRODUCTION

Facial emotion recognition takes part as a key role in human interaction and communication process [3]. Since the expression contains abundance of information and the state of mind of person, it can be helpful in different field included from machine learning to healthcare. Though it is very hard tasks, but it gives lots of information which is helpful so much in various application in the era of recent emerging technology [4].

It is possible to identify emotions based on non-facial attributes like verbal accent, tone of speech etc. An important issue in speech emotion recognition is the extraction of efficient speech features that characterize the emotional content of speech [5]. To solve this problem, we aim to develop a hybrid model which will consider both facial and lingual features to accurately predict the emotion of a person. To further enhance this, we will deploy this model on an Android application which can access both front and back cameras simultaneously. We will also provide speech recording feature which can extract the features from voice and will be used as an input to deep learning model to predict emotion.

The remaining document is organized as follows:

In deliverable 1, we introduce two deep learning models developed for emotion recognition based on facial expression. We also discuss the results of both models. Next in deliverable 2, we discuss the synthetic dataset generated which will be used to train and test the model. In deliverable 3, we discuss about the Android application developed which can access both cameras simultaneously. In this section, we also discuss about the significance of this application in the future work. Deliverable 4 section displays the code snippet for the deployment and initial testing of the model on static image.

According to [6], human emotions can be categorized into seven different classes namely angry, disgust, fear, happy, sad, surprise and neutral.

Usually, it is observed that Happy or Neutral are the constant emotions in human life. Other emotions are situational [6]. Intensity of these emotions also vary based on the person. In this project, I will build a deep learning model that can consider the facial and lingual features to identify the emotions. In the future, it can be also customized based on the person whether he is more expressive in terms of face or in terms of speech.

DELIVERABLE 1

DEVELOP A SEQUENTIAL AND DEPTH WISE CONVOLUTIONAL MODEL TO DETECT FACIAL EXPRESSIONS.

The Aim of this deliverable is to develop a sequential and depth wise convolutional model to detect the facial expressions and finalize the model for CS298. The finalized model will be deployed on Android application to detect emotions.

Model Architecture:

- **Sequential Model**

```
Model: "functional_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 48, 48, 1)]  0
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 46, 46, 8)    72          input_3[0][0]
__________________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, 46, 46, 8)    32          conv2d_11[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 46, 46, 8)    0           batch_normalization_22[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, 44, 44, 8)    576         activation_10[0][0]
__________________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, 44, 44, 8)    32          conv2d_12[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 44, 44, 8)    0           batch_normalization_23[0][0]
__________________________________________________________________________________________________
separable_conv2d_12 (SeparableC (None, 44, 44, 16)   200         activation_11[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, 44, 44, 16)   64          separable_conv2d_12[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, 44, 44, 16)   0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
separable_conv2d_13 (SeparableC (None, 44, 44, 16)   400         activation_12[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, 44, 44, 16)   64          separable_conv2d_13[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 22, 22, 16)   128         activation_11[0][0]
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 22, 22, 16)   0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, 22, 22, 16)   64          conv2d_13[0][0]
__________________________________________________________________________________________________
add_5 (Add)                     (None, 22, 22, 16)   0           max_pooling2d_5[0][0]
                                                                 batch_normalization_24[0][0]
__________________________________________________________________________________________________
separable_conv2d_14 (SeparableC (None, 22, 22, 32)   656         add_5[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, 22, 22, 32)   128         separable_conv2d_14[0][0]
__________________________________________________________________________________________________
activation_13 (Activation)      (None, 22, 22, 32)   0           batch_normalization_28[0][0]
__________________________________________________________________________________________________
separable_conv2d_15 (SeparableC (None, 22, 22, 32)   1312        activation_13[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 22, 22, 32)   128         separable_conv2d_15[0][0]
__________________________________________________________________________________________________
```

- Depth Wise Convolutional Model

```
________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, 11, 11, 32)   512         add_5[0][0]
________________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)  (None, 11, 11, 32)   0           batch_normalization_29[0][0]
________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 11, 11, 32)   128         conv2d_14[0][0]
________________________________________________________________________________________________
add_6 (Add)                     (None, 11, 11, 32)   0           max_pooling2d_6[0][0]
                                                                 batch_normalization_27[0][0]
________________________________________________________________________________________________
separable_conv2d_16 (SeparableC (None, 11, 11, 64)   2336        add_6[0][0]
________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 11, 11, 64)   256         separable_conv2d_16[0][0]
________________________________________________________________________________________________
activation_14 (Activation)      (None, 11, 11, 64)   0           batch_normalization_31[0][0]
________________________________________________________________________________________________
separable_conv2d_17 (SeparableC (None, 11, 11, 64)   4672        activation_14[0][0]
________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 11, 11, 64)   256         separable_conv2d_17[0][0]
________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, 6, 6, 64)     2048        add_6[0][0]
________________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)  (None, 6, 6, 64)     0           batch_normalization_32[0][0]
________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 6, 6, 64)     256         conv2d_15[0][0]
________________________________________________________________________________________________
add_7 (Add)                     (None, 6, 6, 64)     0           max_pooling2d_7[0][0]
                                                                 batch_normalization_30[0][0]
________________________________________________________________________________________________
separable_conv2d_18 (SeparableC (None, 6, 6, 64)     4672        add_7[0][0]
________________________________________________________________________________________________
batch_normalization_34 (BatchNo (None, 6, 6, 64)     256         separable_conv2d_18[0][0]
________________________________________________________________________________________________
activation_15 (Activation)      (None, 6, 6, 64)     0           batch_normalization_34[0][0]
________________________________________________________________________________________________
separable_conv2d_19 (SeparableC (None, 6, 6, 64)     4672        activation_15[0][0]
________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 6, 6, 64)     256         separable_conv2d_19[0][0]
________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, 3, 3, 64)     4096        add_7[0][0]
________________________________________________________________________________________________
max_pooling2d_8 (MaxPooling2D)  (None, 3, 3, 64)     0           batch_normalization_35[0][0]
________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 3, 3, 64)     256         conv2d_16[0][0]
________________________________________________________________________________________________
add_8 (Add)                     (None, 3, 3, 64)     0           max_pooling2d_8[0][0]
                                                                 batch_normalization_33[0][0]
```

```
__________________________________________________________________________________________________
separable_conv2d_20 (SeparableC (None, 3, 3, 128)    8768        add_8[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 3, 3, 128)    512         separable_conv2d_20[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 3, 3, 128)    0           batch_normalization_37[0][0]
__________________________________________________________________________________________________
separable_conv2d_21 (SeparableC (None, 3, 3, 128)    17536       activation_16[0][0]
__________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, 3, 3, 128)    512         separable_conv2d_21[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, 2, 2, 128)    8192        add_8[0][0]
__________________________________________________________________________________________________
max_pooling2d_9 (MaxPooling2D)  (None, 2, 2, 128)    0           batch_normalization_38[0][0]
__________________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, 2, 2, 128)    512         conv2d_17[0][0]
__________________________________________________________________________________________________
add_9 (Add)                     (None, 2, 2, 128)    0           max_pooling2d_9[0][0]
                                                                 batch_normalization_36[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, 2, 2, 7)      8071        add_9[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 7)            0           conv2d_18[0][0]
__________________________________________________________________________________________________
predictions (Activation)        (None, 7)            0           global_average_pooling2d_1[0][0]
==================================================================================================
Total params: 72,631
Trainable params: 70,775
Non-trainable params: 1,856
```

Results Achieved:

- **Sequential Model:**

  80% accuracy for 200 Epochs.

  Highest Individual Emotion Accuracy: 84.3% (Neutral)

  Lowest Individual Emotion Accuracy: 75.32% (Sad)

- **Depth Wise Convolutional Model:**

  85.32% accuracy for 200 Epochs.

  Highest Individual Emotion Accuracy: 88.3% (Neutral)

  Lowest Individual Emotion Accuracy: 79.86% (Sad)

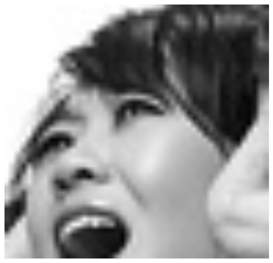
Dataset

The Dataset consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that face is more or less centered and occupies about the same amount of space in each image [2]. The Expressions are categorized in 7 categories (0 = Angry, 1 = Disgust, 2 = Fear, 3 = Happy, 4 = Sad, 5 = Surprise, 6 = Neutral). The training and testing are divided in 80% and 20% respectively.

Angry

Disgust

Fear



Happy

Sad

Surprise

Neutral

DELIVERABLE 2

CREATE A SYNTHETIC DATASET COVERING DIFFERENT FACIAL EXPRESSIONS WHICH WILL BE USED FOR MODEL TRAINING AND TESTING. THE DATASET IS CREATED IN UNITY

Introduction to Unity:

Unity is an engine to create games on multiple platforms. The focus Unity lies in the development of both 3D and 2D games and interactive content.

My Work in Unity:

Unity provides lots of open-source assets to try your hands on. We used an open-source Anime Character provided by unity to create my synthetic dataset that will be used for initial testing of my model. We created the animations of the seven emotions that Anime character could express.

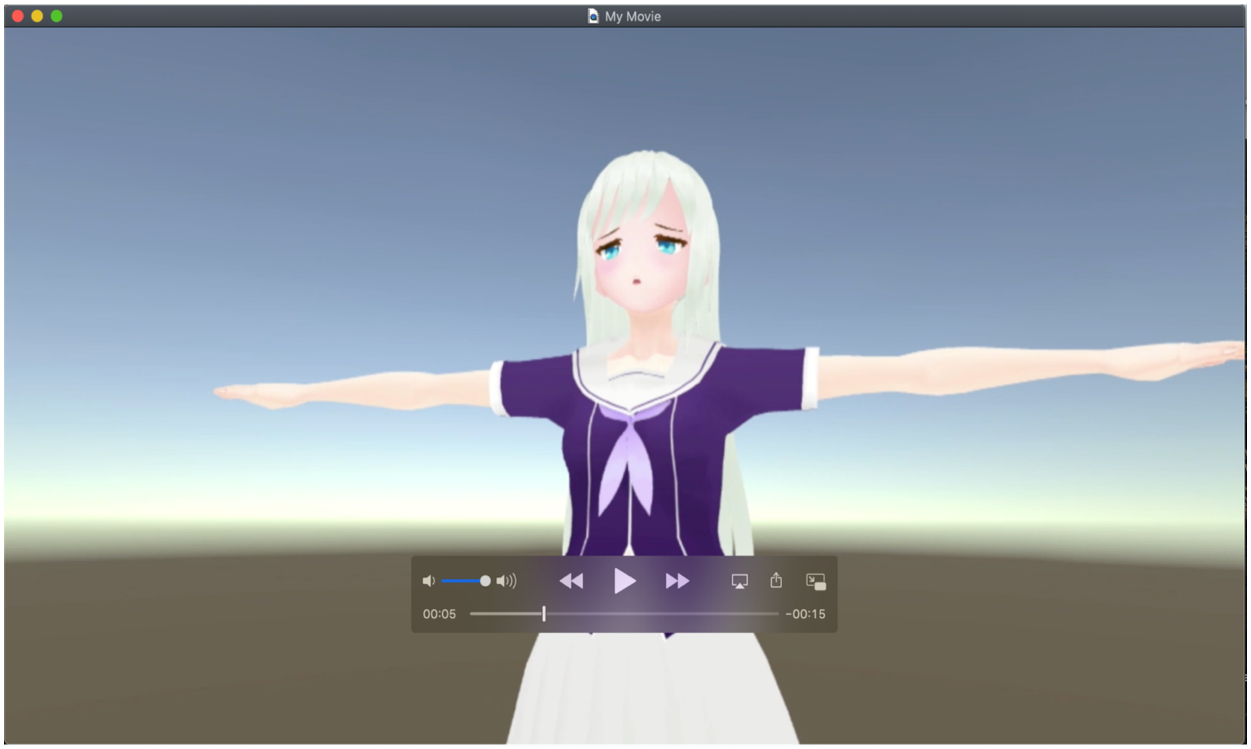
In the deliverable, the scope of the dataset is limited to 51 videos, each of 20 seconds. The scene created in Unity is captured from different angle with character expressing random expressions to improve the variety in the dataset.

In the scene, the Humanoid will express different expressions randomly and the camera angle will be changed after every 20 seconds.

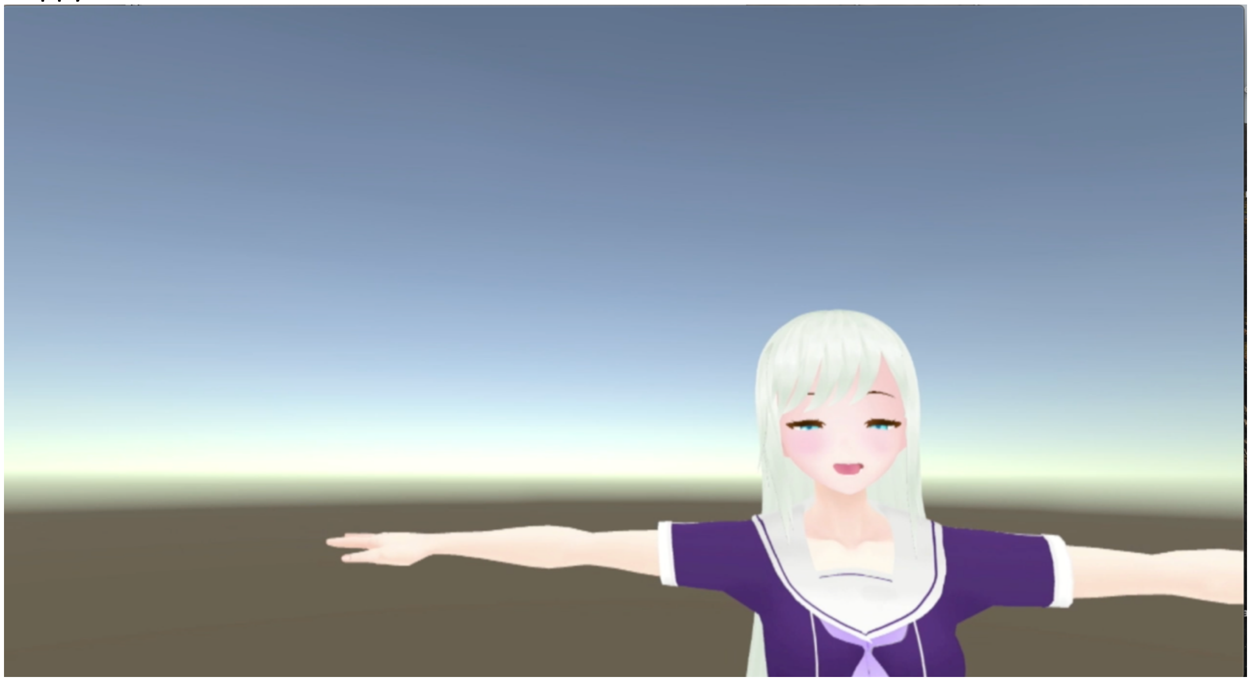
This dataset will be used for the initial training and testing of the model in my project. Model will capture the distinct facial features to predict the emotion.

Facial Expressions on Unity Anime Humanoid

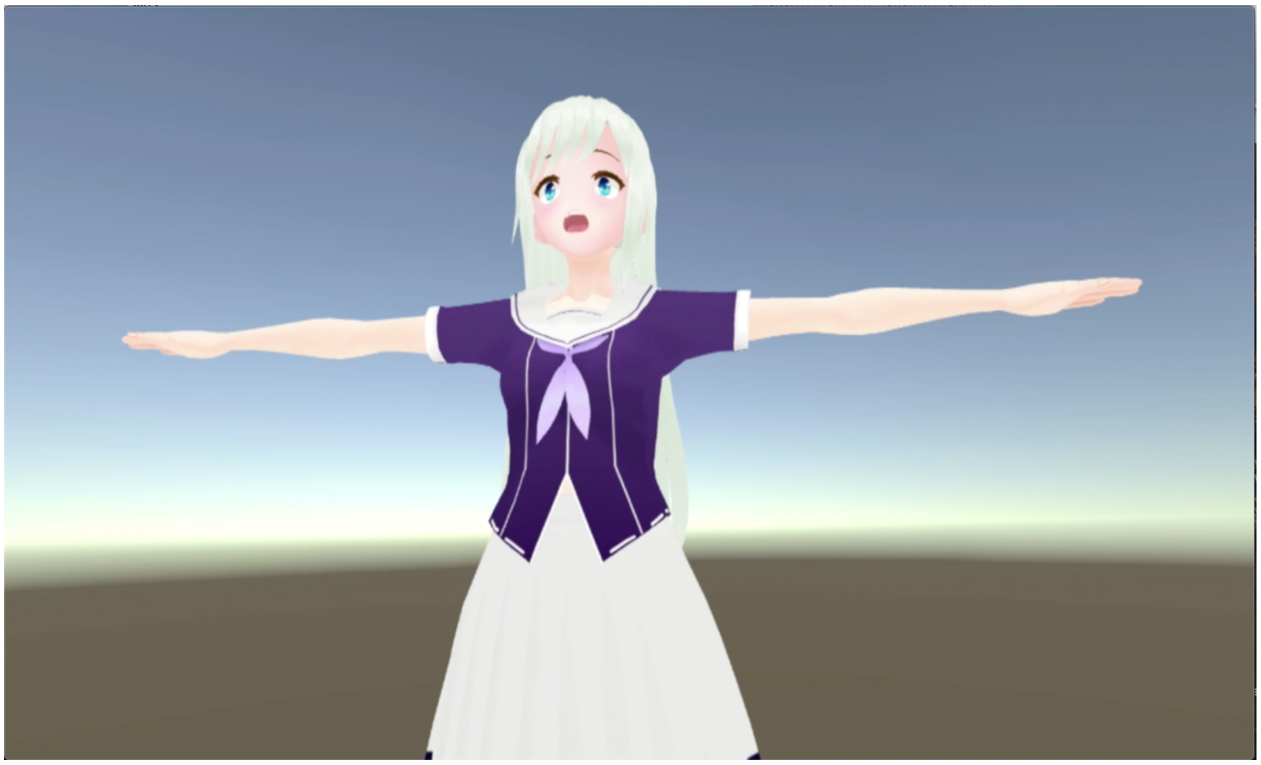
Sad



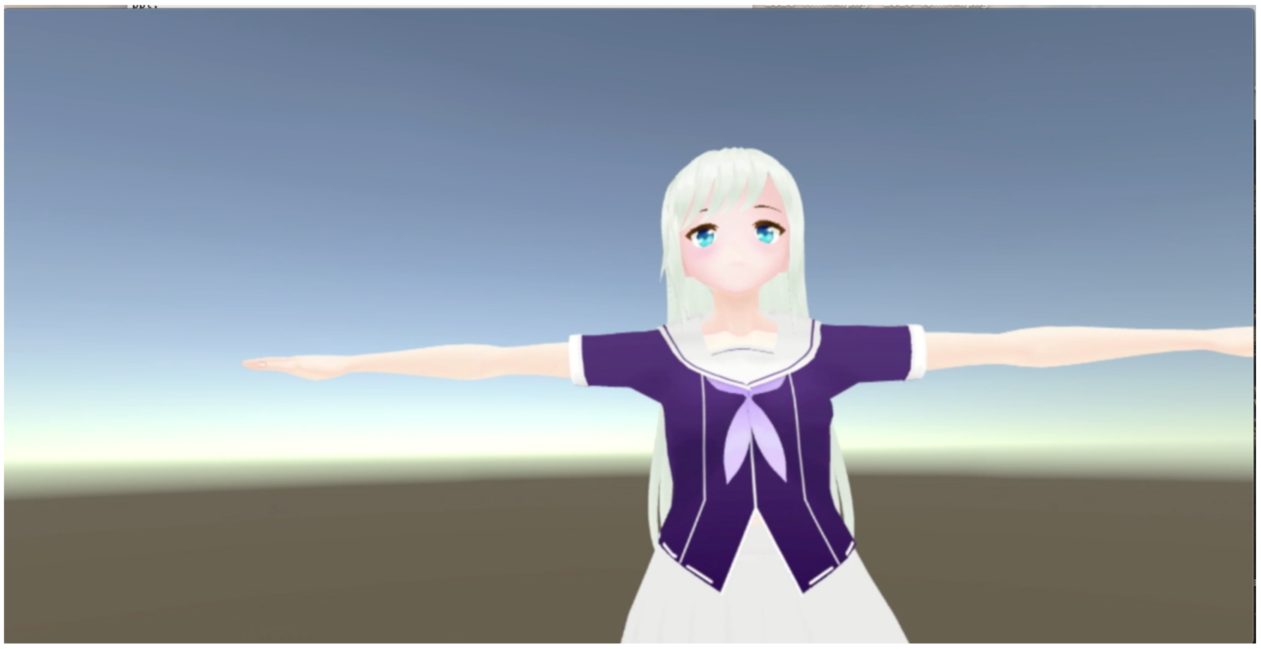
Happy



Fear

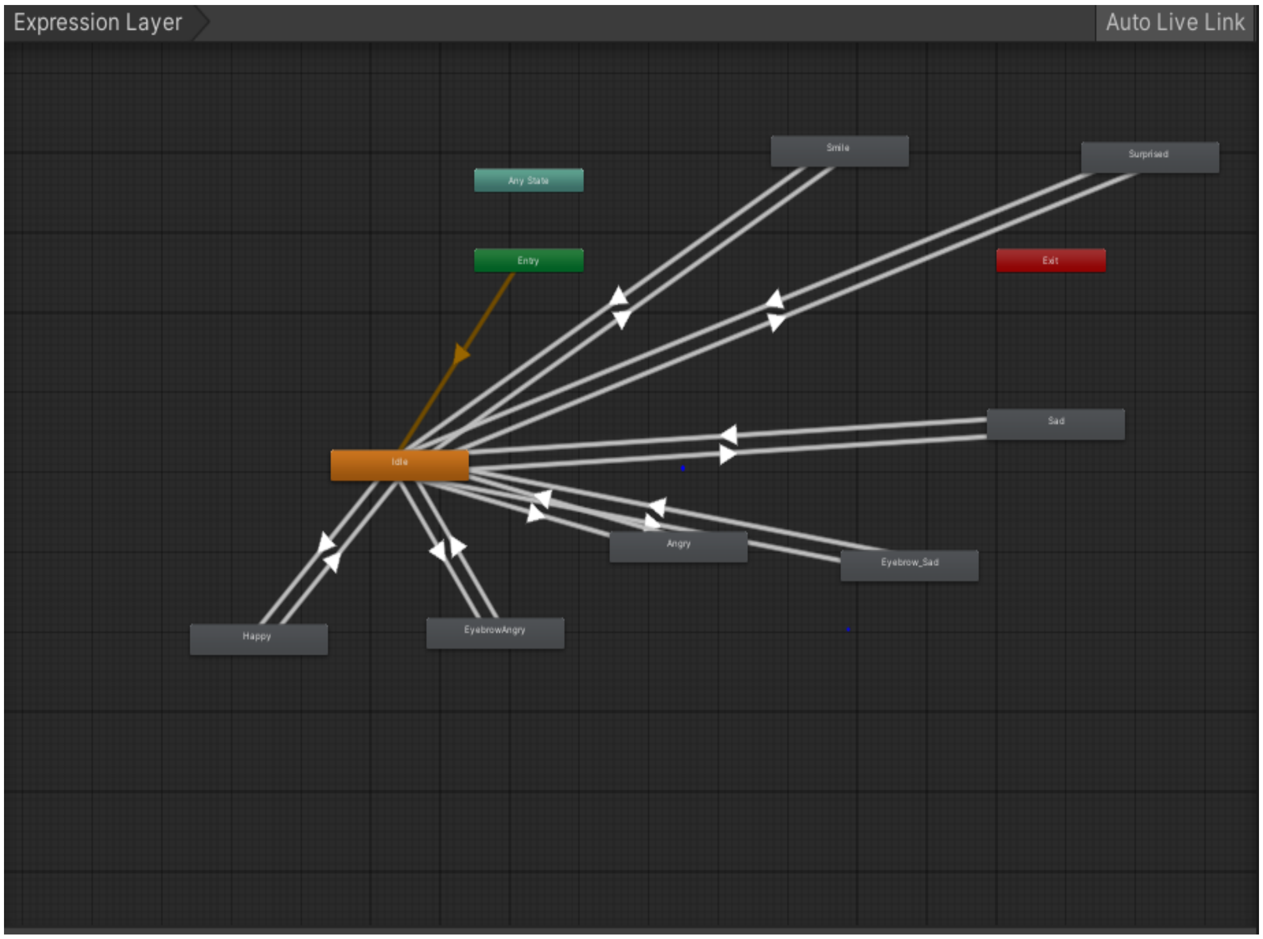


Neutral



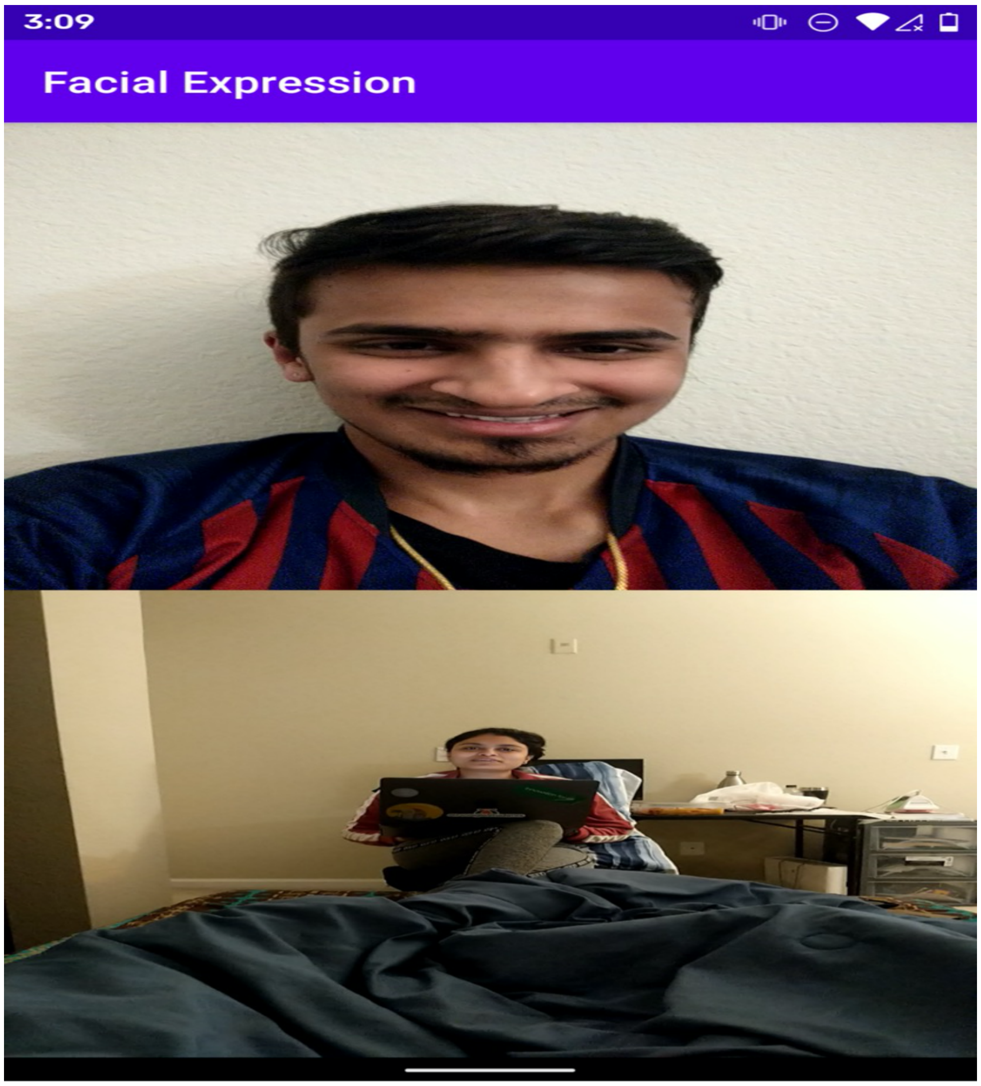
Animator Controller

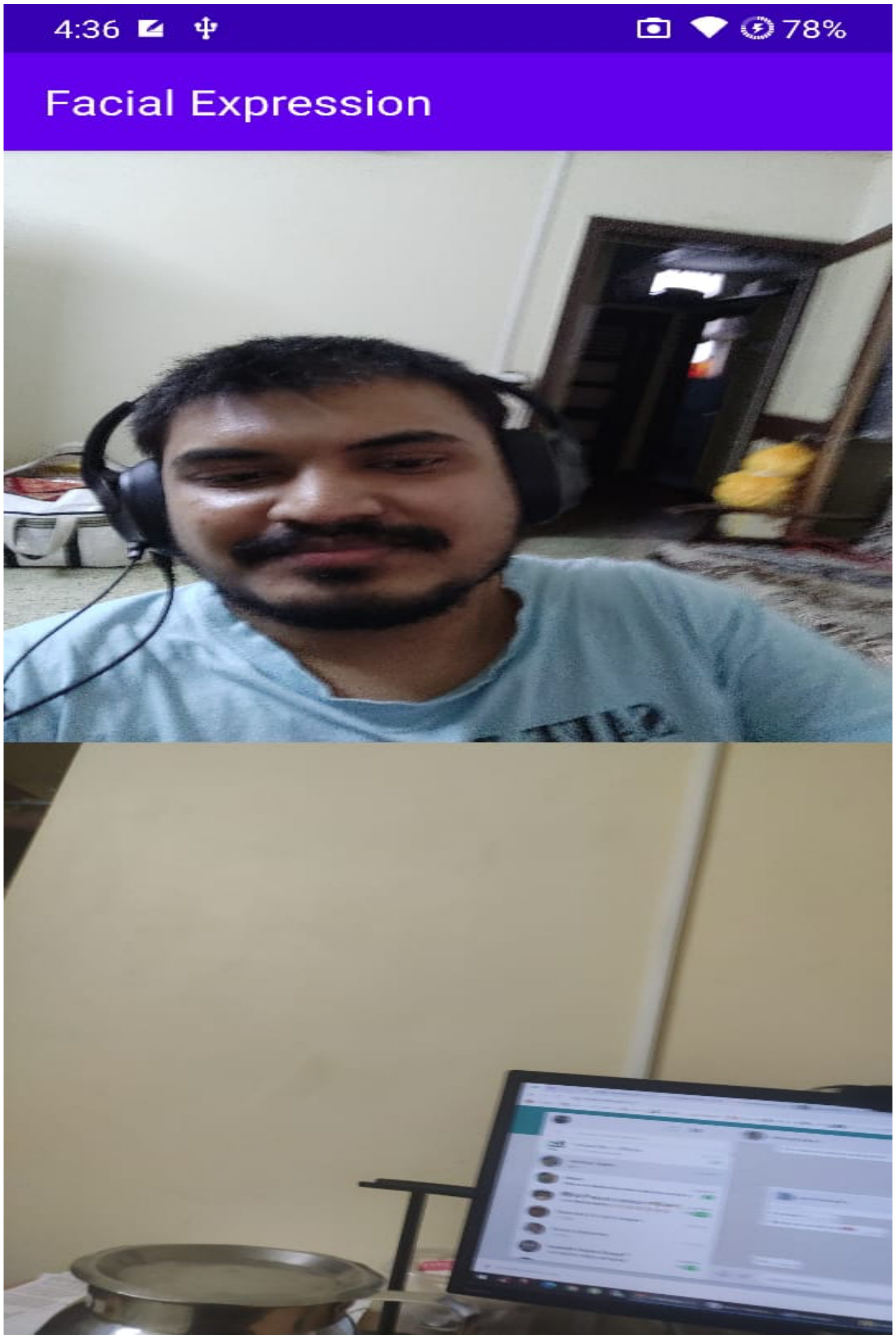
This Animator controller is used to randomly select the animation to express it on Humanoid’s face. The initial state of the humanoid is neutral. After every 20 seconds, the random expression script will pick an expression and display it on the humanoid’s face. This controller expresses the flow of the emotions in real time.



DELIVERABLE 3
DEVELOP AN ANDROID APPLICATION ON WHICH THE DEEP LEARNING MODEL WILL BE DEPLOYED. THE APPLICATION SHOULD BE ABLE TO ACCESS BOTH FRONT AND REAR CAMERA SIMULTANEOUSLY SO THAT IT CAN DETECT EXPRESSIONS OF THE PERSONS WHO ARE FACING BOTH THE CAMERAS.

The Android application is developed in Java. This application can access both front and rear cameras simultaneously. Camera2 API Package is used which provides an interface to individual cameras on Android devices. This dual camera access is dependent on hardware as well. Samsung Devices does not allow this. I am using Google Pixel 3 to use this application.

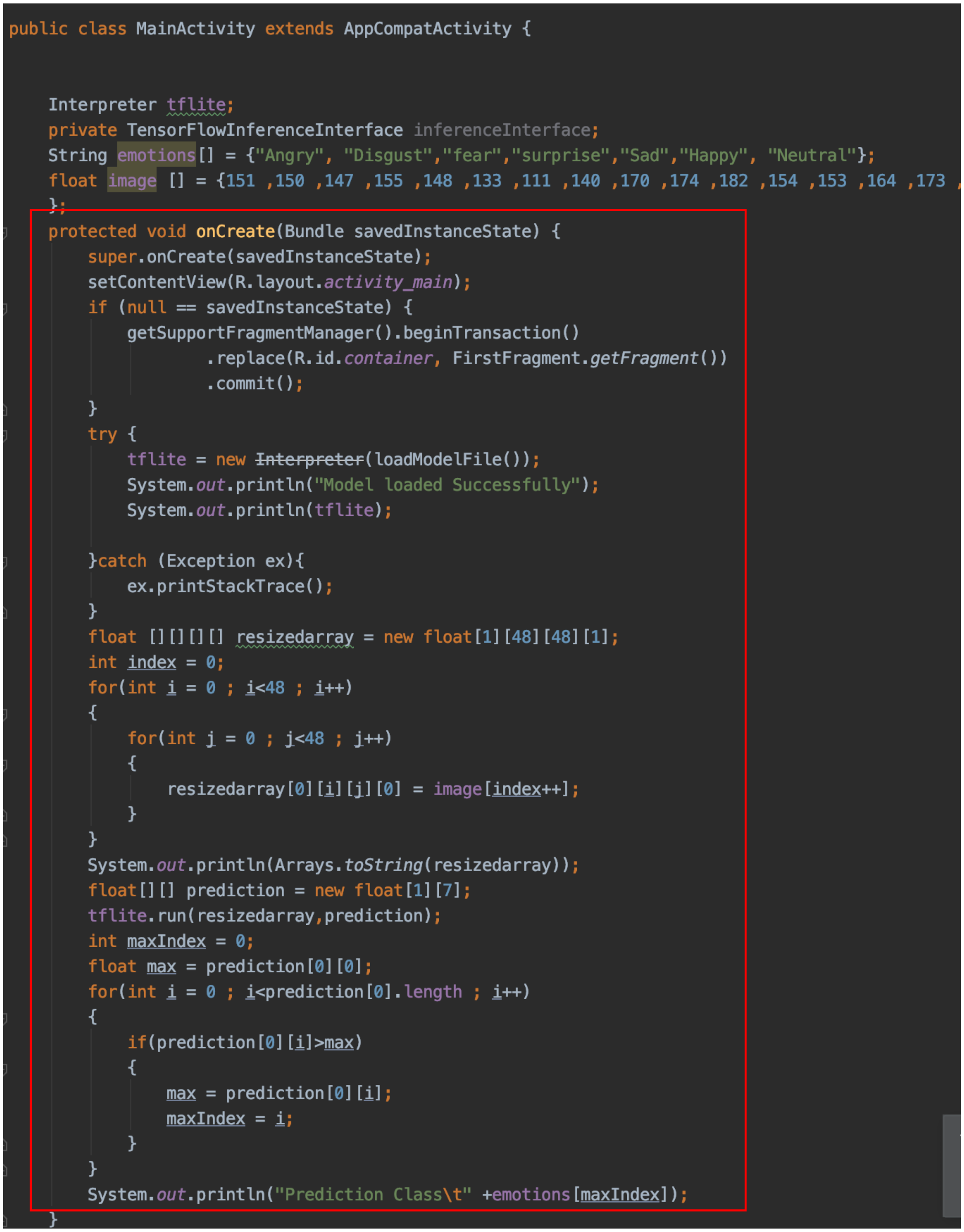



4:36
78%
Facial Expression

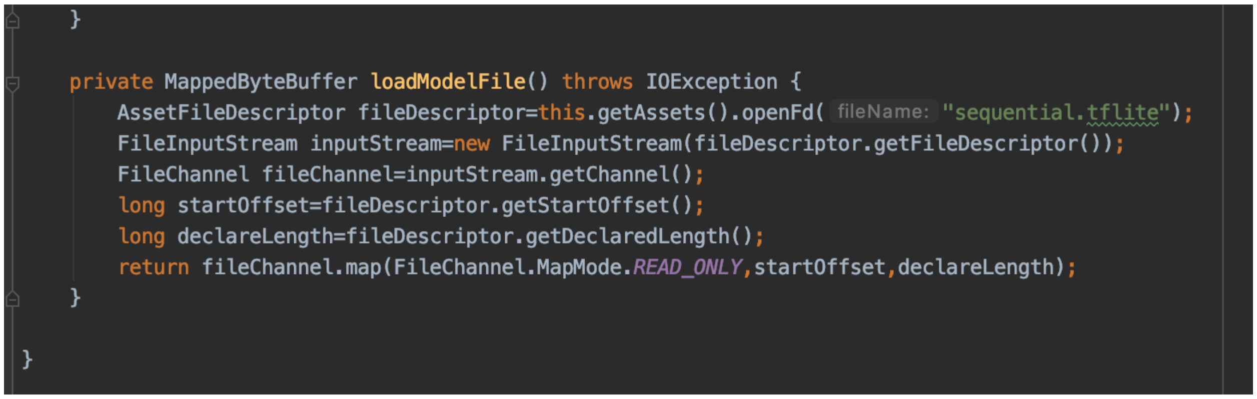
A deep learning model will be deployed on this application that can detect facial expressions from both cameras simultaneously. The application will be further enhanced to support emotion recognition based on speech as well.

DELIVERABLE 4
DEPLOY THE MODEL TRAINED AND TESTED IN DELIVERABLE 1. THE MODEL WAS FIRST CONVERTED INTO A TENSORLITE MODEL AND THEN WAS DEPLOYED ON ANDROID DEVICE.

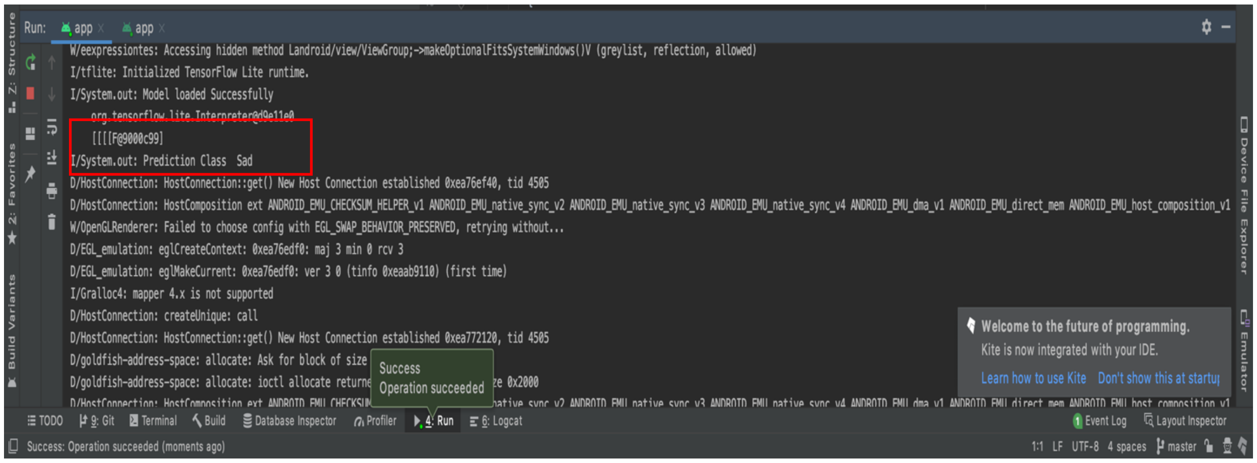
The model which was trained and tested in deliverable 1 is deployed in on the android application. For the initial testing, I tried to test it on static images. Whenever the application starts, the model is loaded on the creation of the instance of the application. The images are reshaped in (1,48,48,1) dimension and passed it to the model. As the last layer is Softmax, the model returns the array of probability of 7 classes (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral). The class having maximum probability is returned as the result. This model will be further enhanced and deployed with functionality of recognizing emotions based on speech as well.



```java
public class MainActivity extends AppCompatActivity {


    Interpreter tflite;
    private TensorFlowInferenceInterface inferenceInterface;
    String emotions[] = {"Angry", "Disgust","fear","surprise","Sad","Happy", "Neutral"};
    float image [] = {151 ,150 ,147 ,155 ,148 ,133 ,111 ,140 ,170 ,174 ,182 ,154 ,153 ,164 ,173 ,
    };
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (null == savedInstanceState) {
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.container, FirstFragment.getFragment())
                    .commit();
        }
        try {
            tflite = new ~~Interpreter~~(loadModelFile());
            System.out.println("Model loaded Successfully");
            System.out.println(tflite);

        }catch (Exception ex){
            ex.printStackTrace();
        }
        float [][][][] resizedarray = new float[1][48][48][1];
        int index = 0;
        for(int i = 0 ; i<48 ; i++)
        {
            for(int j = 0 ; j<48 ; j++)
            {
                resizedarray[0][i][j][0] = image[index++];
            }
        }
        System.out.println(Arrays.toString(resizedarray));
        float[][] prediction = new float[1][7];
        tflite.run(resizedarray,prediction);
        int maxIndex = 0;
        float max = prediction[0][0];
        for(int i = 0 ; i<prediction[0].length ; i++)
        {
            if(prediction[0][i]>max)
            {
                max = prediction[0][i];
                maxIndex = i;
            }
        }
        System.out.println("Prediction Class\t" +emotions[maxIndex]);
    }
```



```java
    }

    private MappedByteBuffer loadModelFile() throws IOException {
        AssetFileDescriptor fileDescriptor=this.getAssets().openFd( fileName: "sequential.tflite");
        FileInputStream inputStream=new FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel=inputStream.getChannel();
        long startOffset=fileDescriptor.getStartOffset();
        long declareLength=fileDescriptor.getDeclaredLength();
        return fileChannel.map(FileChannel.MapMode.READ_ONLY,startOffset,declareLength);
    }

}
```



```
W/eexpressiontes: Accessing hidden method Landroid/view/ViewGroup;->makeOptionalFitsSystemWindows()V (greylist, reflection, allowed)
I/tflite: Initialized TensorFlow Lite runtime.
I/System.out: Model loaded Successfully
    org.tensorflow.lite.Interpreter@d9e11e0
    [[[[F@9000c99]
I/System.out: Prediction Class  Sad
D/HostConnection: HostConnection::get() New Host Connection established 0xea76ef40, tid 4505
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EMU_direct_mem ANDROID_EMU_host_composition_v1
W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
D/EGL_emulation: eglCreateContext: 0xea76edf0: maj 3 min 0 rcv 3
D/EGL_emulation: eglMakeCurrent: 0xea76edf0: ver 3 0 (tinfo 0xeaab9110) (first time)
I/Gralloc4: mapper 4.x is not supported
D/HostConnection: createUnique: call
D/HostConnection: HostConnection::get() New Host Connection established 0xea772120, tid 4505
D/goldfish-address-space: allocate: Ask for block of size
D/goldfish-address-space: allocate: ioctl allocate returne                    ze 0x2000
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM                    native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EMU_direct_mem ANDROID_EMU_host_composition_v1
```

Success
Operation succeeded

Welcome to the future of programming.
Kite is now integrated with your IDE.
Learn how to use Kite   Don't show this at startup

Success: Operation succeeded (moments ago)

image [] represents an array containing 2304 values. Each value represents single pixel of image flattened in one dimension. Each image is of dimension 48x48. OnCreate() function is called whenever the application starts. After starting the application, it creates an object of Interpreter in which tflite model is stored. tflite.run(input, output) method runs the model on the input which is provided. It returns the array of probabilities that the image belongs to which class. Highest class is selected based on the probability.

CONCLUSION

Emotion recognition is a very broad topic which contains multiple techniques to implement. In this project we developed two deep learning models (sequential and depth wise convolutional). We finalized depth wise convolutional model because of high accuracy and lower number of calculations. We also generated synthetic dataset in unity for the initial testing of the model. The dataset contained 51 videos, each of 20 seconds in which humanoid character is expressing different facial expressions. To further extend the idea, we also developed an Android application which can access both front and rear cameras simultaneously. The motive behind the development of this application is to deploy the deep learning model on it so that it can detect the emotions from both the cameras simultaneously. Further, we deployed the initial model developed in Deliverable 1 on the android application for initial testing.

In the second semester of the project, we will enhance the android application in such a way that it can capture real time frames from the camera and pass it to the model for emotion recognition. We will concentrate to work on lingual features on emotion recognition and develop a hybrid model that will consider both facial and lingual features to detect emotions. We will try to deploy this model on the android application to compare the accuracy with the previous model. We will also enhance the application in such a way that it can capture the speech from microphone and send it to the model for further processing. The model, dataset and android application developed in this semester would serve as the basis for achieving the goal of the second semester.

REFERENCES

[1] F. Khan, “Facial Expression Recognition using Facial Landmark Detection and Feature Extraction via Neural Networks”, Department of Electronics and Communication Engineering, NIT Karnataka, Mangalore, India, *IJACSA,* Dec. 10, 2018. [Online]. Available: https://www.groundai.com/project/facial-expression-recognition-using-facial-landmark-detection-and-feature-extraction-on-neural- networks/ [Accessed: Dec. 12, 2018]

[2] L. Zhang, Y. Yang, W. Li, S. Dang and M. Zhu, "Research of Facial Expression Recognition Based on Deep Learning," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 1-4, doi: 10.1109/ICSESS.2018.8663777.

[3] D. Kalita, "Designing of Facial Emotion Recognition System Based on Machine Learning," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 969-972, doi: 10.1109/ICRITO48877.2020.9197771.

[4] T. Huang, Z. Xiong, and Z. Zhang, “Face Recognition Applications,” Handbook of Face Recognition, pp. 371–390.

[5] L. B. Letaifa, M. I. Torres and R. Justo, "Adding dimensional features for emotion recognition on speech," 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, 2020, pp. 1-6, doi: 10.1109/ATSIP49331.2020.9231766.

[6] Ekman P. Darwin's contributions to our understanding of emotional expressions. *Philos Trans R Soc Lond B Biol Sci*. 2009;364(1535):3449-3451. doi:10.1098/rstb.2009.0189