Deliverable 3

1. **Problem Statement**:

   Emotion Detection using Convolutional Neural Networks.
   We want to build a model that takes images as data and predicts the emotion that matches closely with the image provided.

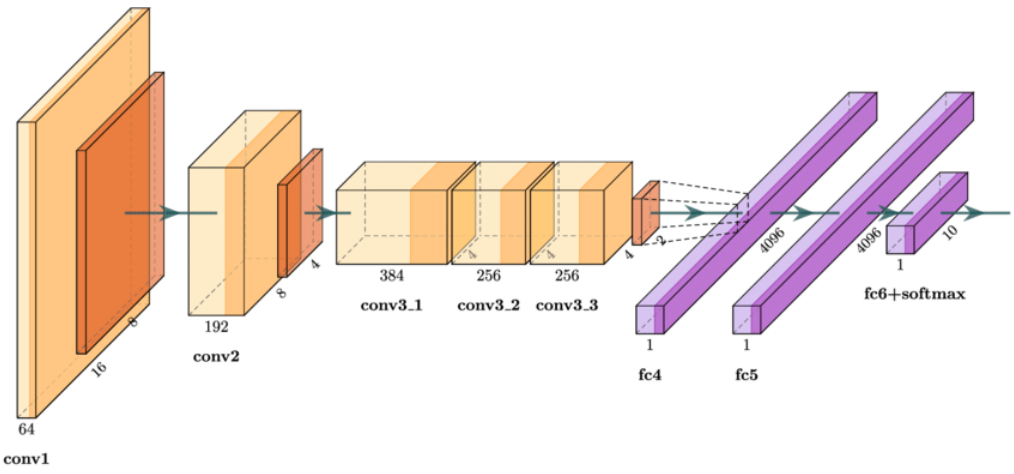
2. **Data Preprocessing**:

   - ➔ Dataset : https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer
   - ➔ The dataset contains 35,685 examples of 48x48 pixel gray scale images of faces divided into train and test dataset. Images are categorized based on the emotion shown in the facial expressions (happiness, neutral, sadness, anger, surprise, disgust, fear).
   - ➔ The data preprocessing methods we used are: resizing, face detection, cropping, adding noises, and data normalization. Face detection alone can achieve very high accuracy and combined with other preprocessing techniques we hope to boost our accuracy further.

3. **Machine Learning Model**:

   For our model, we are using the *AlexNet DCNN*.

   A. The model has a total eight layers: the first five are convolutional layers, some of which are followed by max-pooling layers. The last three layers are fully connected layers. We are using Tensorflow and Keras libraries for our code.



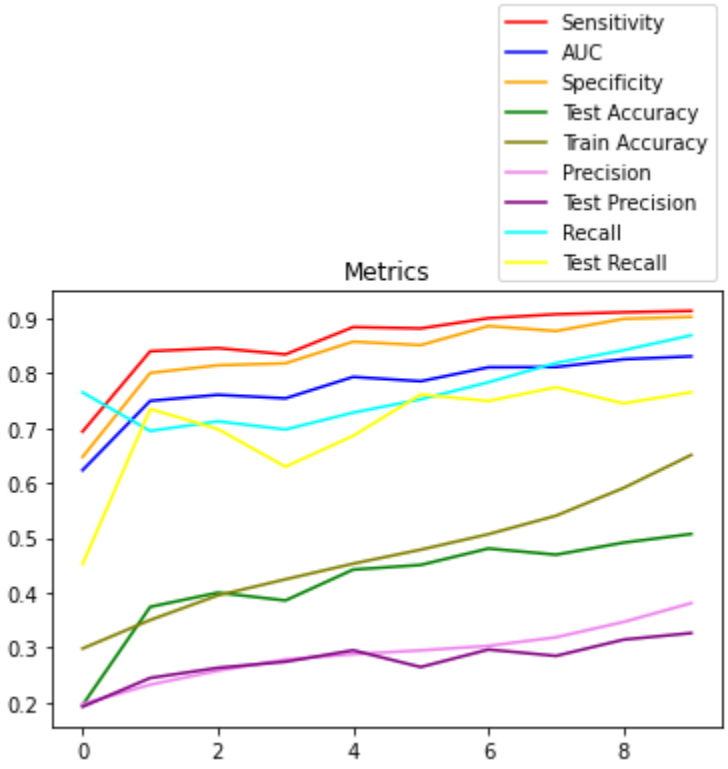
   B. To avoid overfitting, we are using dropout as our regularization technique. The results obtained using the dropout model have low losses and the accuracy of the model is higher than other techniques like L1/L2 regularization.
   The parameters learning rate, batch size, training ratio, number of epochs, image preprocessing method and optimizer were chosen after testing with different values. We

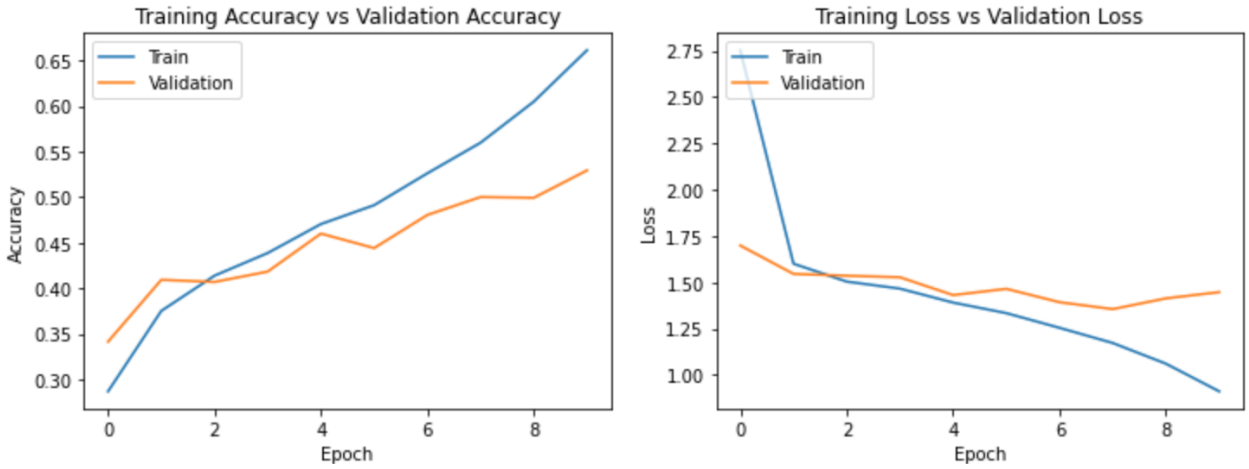
chose Adam as our optimizer and a batch size of 32. Our image dimensions were chosen to be 227x227

C. For the validation step, we used the basic train and test split. We used an 80/20 split for our model.
Training data = 28709 images belonging to 7 classes.
Testing data = 7178 images belonging to 7 classes.

4. **Final Results:**

The metrics we are using are accuracy, precision, recall, and loss. (see below for details)





The above three figures are what we received for our preliminary results.

For our preliminary results our model was not very accurate. Emotions such as ‘angry’, ‘disgusted’, ‘neutral’ and ‘fearful’ were being categorized as ‘sad’. While emotions such as ‘surprised’ were being categorized as ‘happy’. One improvement that we made was to increase the epochs from 10-25. This greatly improved our accuracy, precision and reduced our loss as seen below. We also received a decreasing function for our new validation loss around epoch 22-25.

```
Epoch 1/25
898/898 [==============================] - 148s 159ms/step - loss: 0.0738 - accuracy: 0.9795 - precision: 0.7109 -
Epoch 2/25
898/898 [==============================] - 142s 158ms/step - loss: 0.0717 - accuracy: 0.9804 - precision: 0.7397 -
Epoch 3/25
898/898 [==============================] - 133s 148ms/step - loss: 0.0674 - accuracy: 0.9813 - precision: 0.7424 -
Epoch 4/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0529 - accuracy: 0.9843 - precision: 0.7466 -
Epoch 5/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0669 - accuracy: 0.9820 - precision: 0.7340 -
Epoch 6/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0650 - accuracy: 0.9834 - precision: 0.7340 -
Epoch 7/25
898/898 [==============================] - 133s 148ms/step - loss: 0.0581 - accuracy: 0.9828 - precision: 0.7584 -
Epoch 8/25
898/898 [==============================] - 135s 150ms/step - loss: 0.0537 - accuracy: 0.9843 - precision: 0.7682 -
Epoch 9/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0613 - accuracy: 0.9833 - precision: 0.7601 -
Epoch 10/25
898/898 [==============================] - 133s 149ms/step - loss: 0.0612 - accuracy: 0.9838 - precision: 0.7865 -
Epoch 11/25
898/898 [==============================] - 144s 160ms/step - loss: 0.0395 - accuracy: 0.9889 - precision: 0.7934 -
Epoch 12/25
898/898 [==============================] - 135s 150ms/step - loss: 0.0396 - accuracy: 0.9881 - precision: 0.7982 -
Epoch 13/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0498 - accuracy: 0.9862 - precision: 0.7927 -
Epoch 14/25
898/898 [==============================] - 133s 148ms/step - loss: 0.0426 - accuracy: 0.9874 - precision: 0.8053 -
Epoch 15/25
898/898 [==============================] - 133s 148ms/step - loss: 0.0554 - accuracy: 0.9844 - precision: 0.7958 -
Epoch 16/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0430 - accuracy: 0.9881 - precision: 0.7717 -
Epoch 17/25
898/898 [==============================] - 133s 148ms/step - loss: 0.0515 - accuracy: 0.9868 - precision: 0.7571 -
Epoch 18/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0457 - accuracy: 0.9888 - precision: 0.7736 -
Epoch 19/25
898/898 [==============================] - 134s 149ms/step - loss: 0.0343 - accuracy: 0.9899 - precision: 0.7970 -
```

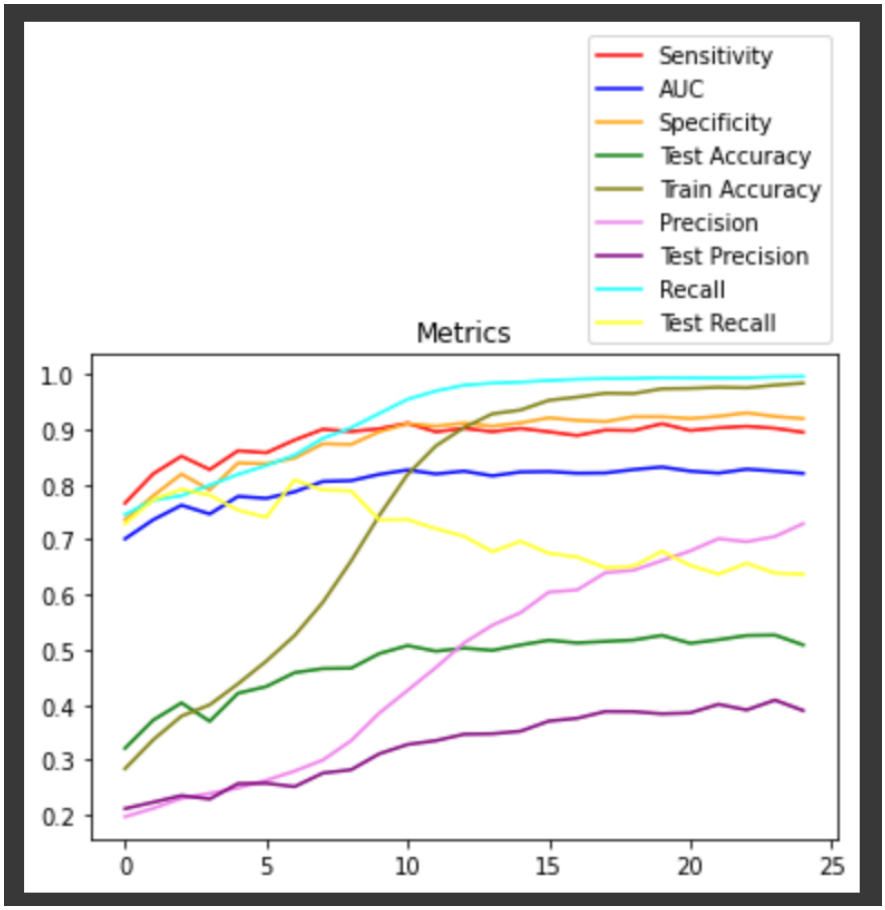
Problems:

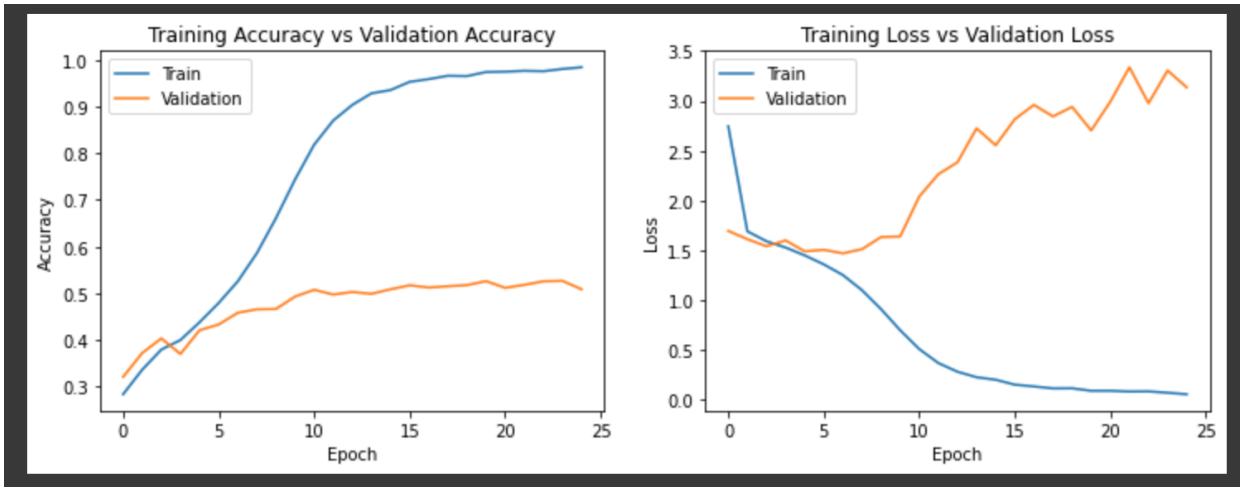
For the newly trained model with 25 epochs, fortunately we managed to save the weights and graphs. However, we lost our history when the runtime got disconnected. We can not construct graphs from saved models so we decided to retrain the model to redraw the graphs. However, our runtime got disconnected at epoch 21. To make up for this we decided to add our own accuracy metric. We used the saved model to predict the testing data for each emotion and compiled the data in a dictionary. For example, out of all happy test images, the saved model accurately identified 1234 of them and the rest were misclassified.

```
[ ] print(my_dict) # for happy

    {'Surprised': 62, 'Angry': 102, 'Neutral': 136, 'Happy': 1234, 'Disgusted': 3, 'Fearful': 91, 'Sad': 146}

    print(my_dict) # for sad

    {'Surprised': 38, 'Angry': 183, 'Neutral': 181, 'Happy': 160, 'Disgusted': 5, 'Fearful': 151, 'Sad': 529}

[ ] print(my_dict) # for surprised

    {'Surprised': 575, 'Angry': 58, 'Neutral': 44, 'Happy': 44, 'Disgusted': 3, 'Fearful': 66, 'Sad': 41}

[ ] print(my_dict) # for fearful

    {'Surprised': 72, 'Angry': 151, 'Neutral': 109, 'Happy': 107, 'Disgusted': 2, 'Fearful': 404, 'Sad': 179}

[ ] print(my_dict) # for neutral

    {'Surprised': 52, 'Angry': 144, 'Neutral': 515, 'Happy': 185, 'Disgusted': 1, 'Fearful': 119, 'Sad': 217}

[ ] print(my_dict) # for disgusted

    {'Surprised': 2, 'Angry': 23, 'Neutral': 6, 'Happy': 11, 'Disgusted': 54, 'Fearful': 5, 'Sad': 10}

[ ] print(my_dict) # for angry

    {'Surprised': 34, 'Angry': 397, 'Neutral': 126, 'Happy': 120, 'Disgusted': 4, 'Fearful': 104, 'Sad': 173}
```

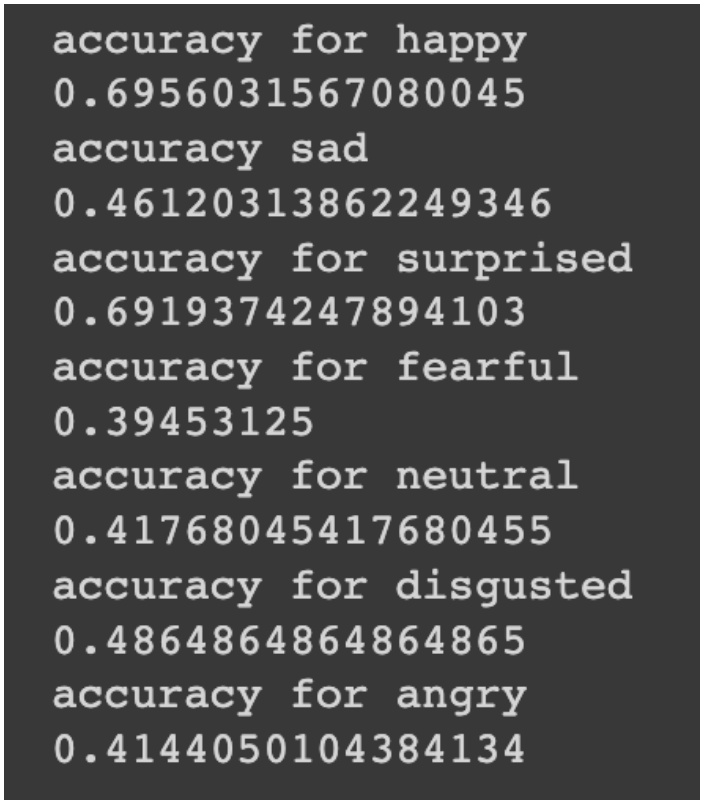
Solution

We managed to retrain the model with 25 epochs and obtained the following graphs.





The following picture shows our own metric for accuracy. We predicted images using the testing data for each emotion and the amount of images predicted correctly vs the rest of the images are as follows.



This is still a very good metric because a lot of the images could easily have been misclassified by humans. Our team was also surprised by some of the images which were classified as happy or sad when they clearly looked different. Also images for 'angry', 'fearful', sad and 'disgusted' had similar attributes which why they had comparatively lower accuracies.

5. **Next Steps: Web app deployment**

For the web app, we plan to use flask for the backend and HTML CSS and Javascript for the frontend. We only have basic experience with frontend development and no experience in backend development which is why we will need to learn how to deploy our ML Model on the web app. We plan to use the existing github repository provided by MAIS and then build our web app using that as a skeleton.

For our emotion detection ML model we need a webcam to take images as input in real time and display the emoji corresponding to the emotion. For that we will be using CV2 for video and image processing as well as MediaPipe.

Link to collab file:
https://colab.research.google.com/drive/1DBirn4AUAKVMLUcMJHGTQd4YvDfGB6WW?authuser=1#scrollTo=I7kJmcA27LTg