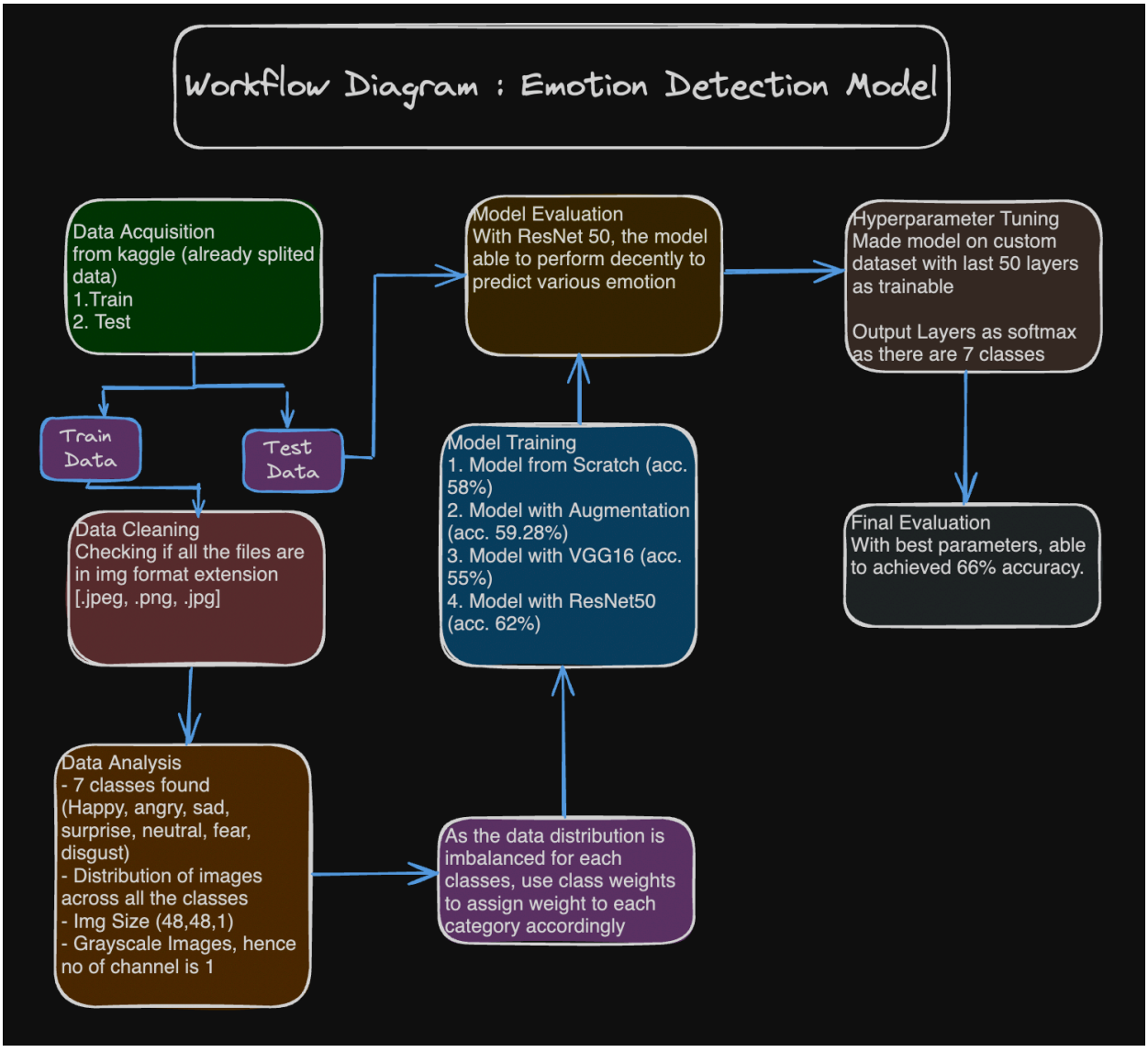
Emotion Detection Model

Problem Statement:

The objective is to create a deep learning model that can correctly classify a set of inputs into one of several predefined emotion categories, such as happiness, sad, anger, fear, surprise, neutral, and disgust given a set of inputs that are photos that depict human emotions. With the ability to identify and categorize people's emotional states from their facial expressions, the model should be able to be used for a variety of purposes, including sentiment analysis, human-computer interaction, and the provision of emotional context for user-generated content.

Solution Workflow Diagram:



Challenges face:

i) Challenges in this dataset: -

The following are the main ideas taken from the text that was provided:

1. **Low Quality photographs**: The 48x48 size of the photographs may make it more difficult for the model to recognize them correctly.

2. **Grayscale Pictures**: Since the photos are grayscale, they lack colour information, which might give more context and distinguishing characteristics to the classification.

3. **Insufficient Colour Details**: The model's performance may be impacted by the grayscale photos' limited feature set compared to colour images.

4. **Influence of Age:** The participants' ages have an impact on how accurately complex emotions are recognized.

5. **Impact of Education and exercise:** The literature presented makes reference to the impact of education and exercise on the precision of emotion recognition, but it does not go into detail.

ii) Challenges in real world application

Here are the key points extracted from the provided text:

1. **Focus on Face:** Most of the images in the dataset focus primarily on the face.

2. **Need for Cropping:** In real-world scenarios, images need to be cropped to focus on the face before training a model.

3. **Inconsistent Image Sizes:** The images in the dataset have different sizes.

4. **Standardizing Image Sizes:** Before training any model, all images need to be resized to the same dimensions.

Usage:

The capacity to recognize non-verbal cues, such as facial expressions, is essential for understanding others' feelings and intentions. FRT can decrease the need for human interaction and thus increase efficiency during, for example, border checks in airports. FRT can even be employed in medicine, such as in identifying subtle facial traits to determine genetic disorders. A large number of tech companies are developing facial biometric systems.

Detailed Dataset Description:

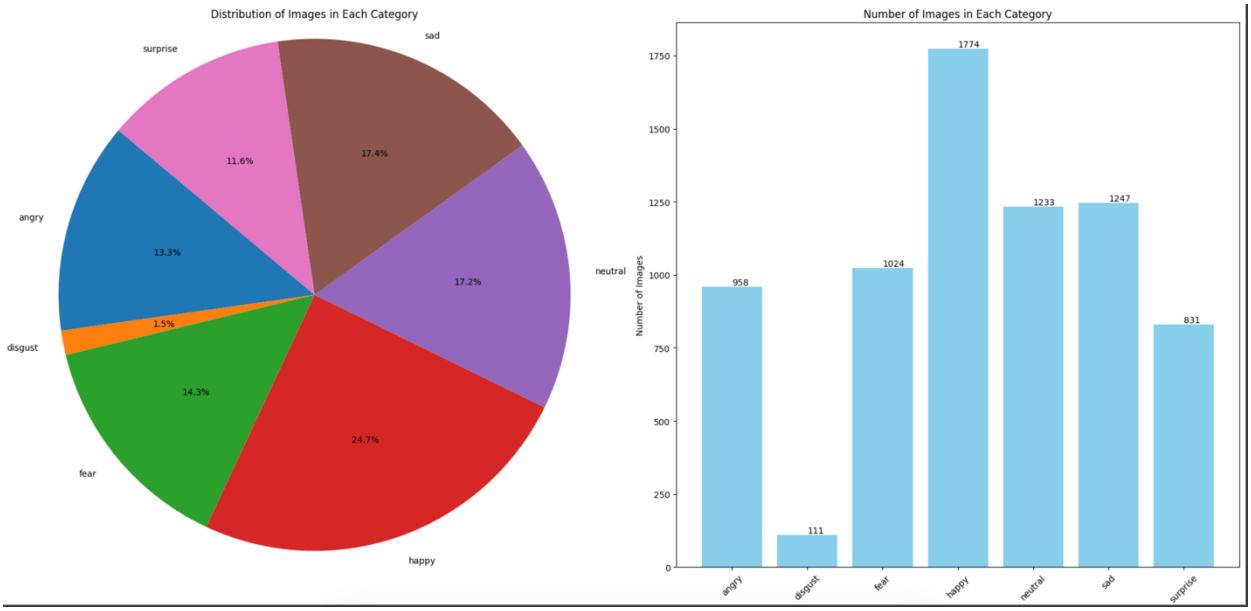
Kaggle - https://www.kaggle.com/datasets/msambare/fer2013

The data consists of **48x48** pixel **grayscale** images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories **(0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)**

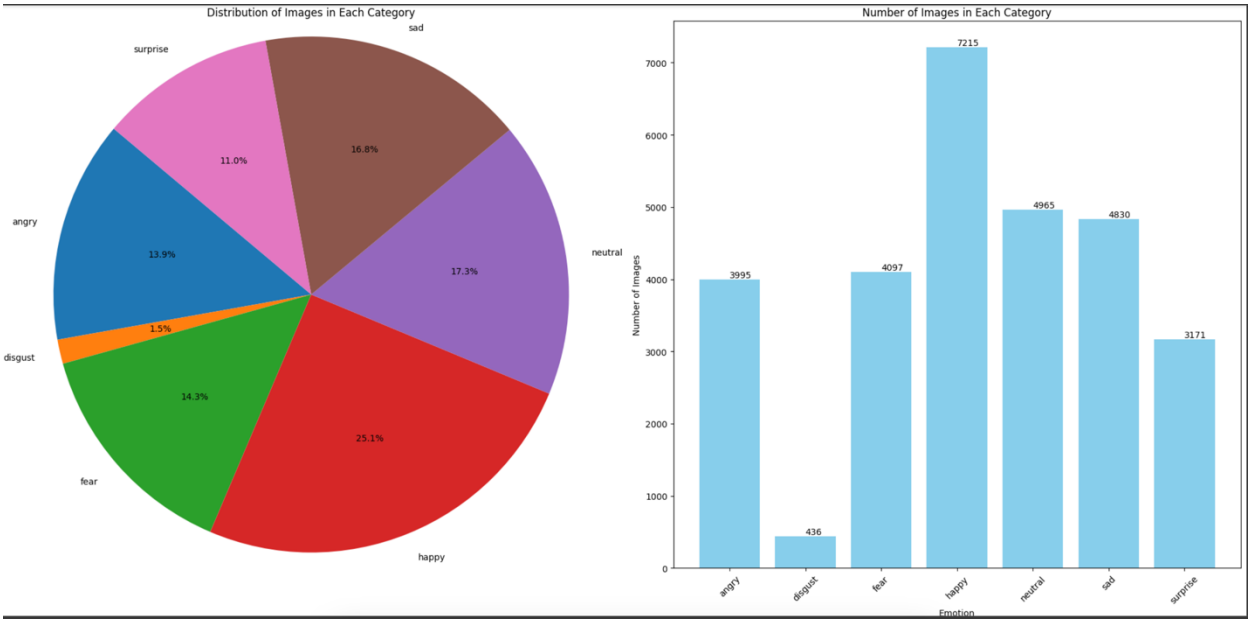
In this dataset, we have two directory, train and test directory. Each directory contain 6 other directories, which indicate 6 different class names. (i.e 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)

Image Distribution in train and test dataset:



Train dataset

In the train directory, it can be observed that the data is unevenly distributed. Disgust expression facial images contribute only 1.5% to the overall images whereas happy emotion contribute to mostly 1/4th of the total images

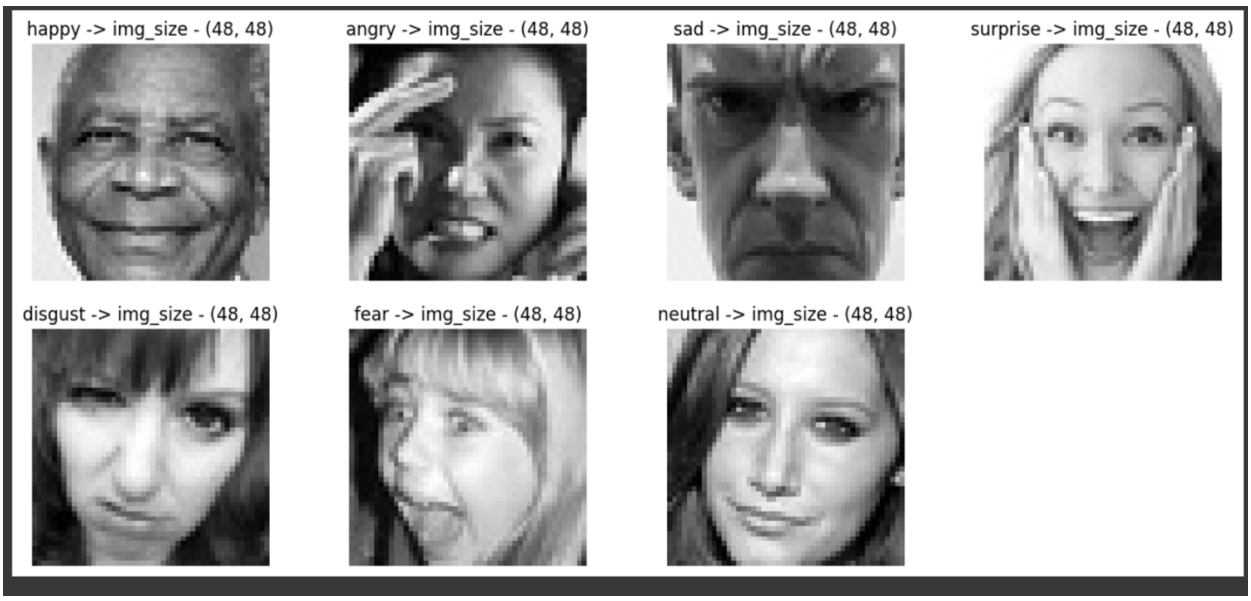


Test dataset

In test directory, it can also be observed that the data is unevenly distributed. Disgust expression facial images contribute only 1.5% to the overall images whereas happy emotion contribute to mostly 1/4th of the total images

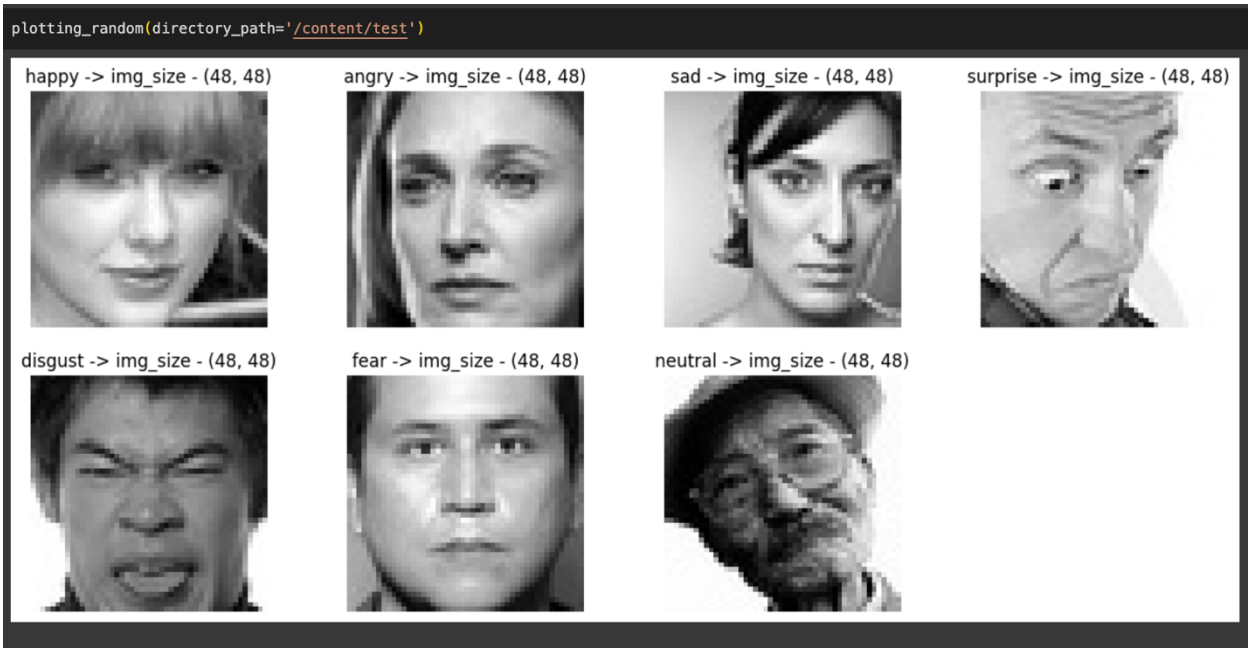
Since, the data distribution in each class is imbalanced, it is recommended to apply class weights according to the number of images in each category.

Image Size details for each classes:



Train data

Randomly displaying images along with image size from train directory to check if all the images are of same size.



Test Data

Randomly displaying images along with image size from train directory to check if all the images are of same size.

Displaying Random images from each category:

Random Images of each class from train directory

Model Building:

We are building 4 CNN models

1. CNN Model from Scratch
2. CNN model with Image Augmentation
3. CNN model using Transfer Learning (VGG16)
4. CNN model using Transfer Learning (ResNet50)

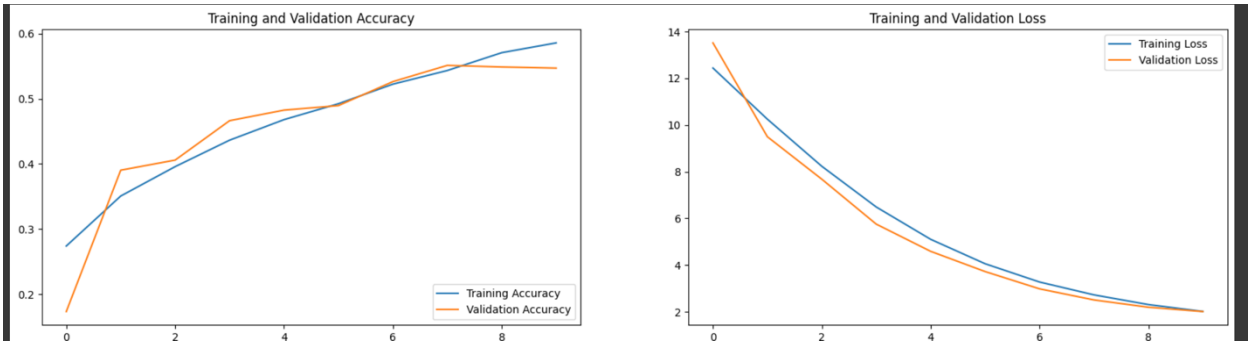
whichever is performing best we can pick that model and do prediction on test data using that model.

1. CNN Model from Scratch

CNN Model from Scratch Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 activation (Activation)     (None, 48, 48, 32)        0

 conv2d_1 (Conv2D)           (None, 48, 48, 64)        18496

 activation_1 (Activation)   (None, 48, 48, 64)        0

 batch_normalization (Batch  (None, 48, 48, 64)        256
 Normalization)

 max_pooling2d (MaxPooling2  (None, 24, 24, 64)        0
 D)

 dropout (Dropout)           (None, 24, 24, 64)        0

 conv2d_2 (Conv2D)           (None, 24, 24, 128)       73856

 activation_2 (Activation)   (None, 24, 24, 128)       0

 conv2d_3 (Conv2D)           (None, 22, 22, 256)       295168

 activation_3 (Activation)   (None, 22, 22, 256)       0

 batch_normalization_1 (Bat  (None, 22, 22, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 256)       0
 g2D)

 dropout_1 (Dropout)         (None, 11, 11, 256)       0

 conv2d_4 (Conv2D)           (None, 11, 11, 512)       1180160

 activation_4 (Activation)   (None, 11, 11, 512)       0

 conv2d_5 (Conv2D)           (None, 11, 11, 512)       2359808

 activation_5 (Activation)   (None, 11, 11, 512)       0

 batch_normalization_2 (Bat  (None, 11, 11, 512)       2048
...
Total params: 17046535 (65.03 MB)
Trainable params: 17044871 (65.02 MB)
Non-trainable params: 1664 (6.50 KB)
```

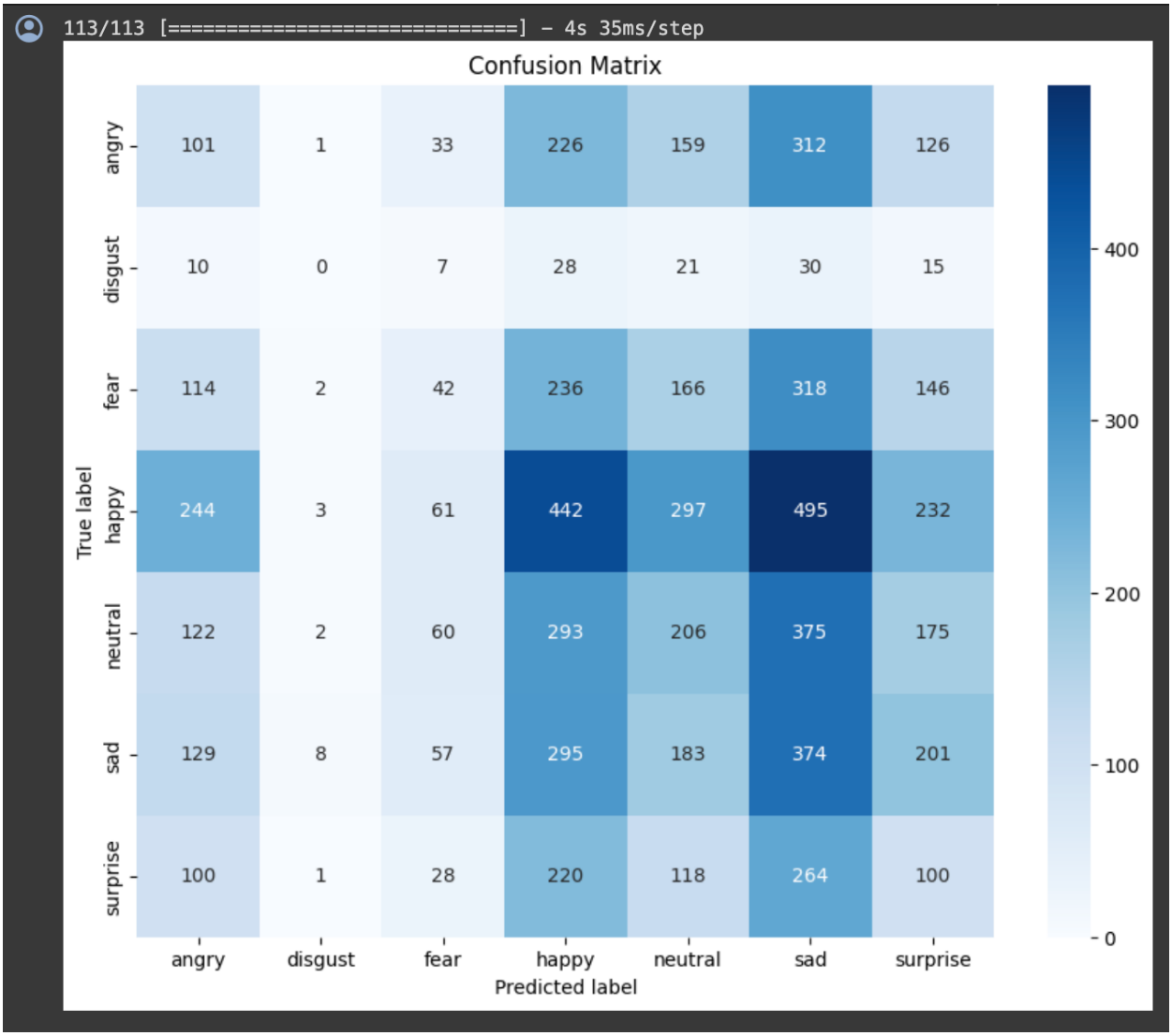
There are **17,046,535** total parameters in the CNN model, out of which **17,044,871** are trainable and **1664** are non-trainable parameters.



Training and Validation accuracy and loss after 10 epochs in CNN model from Scratch.

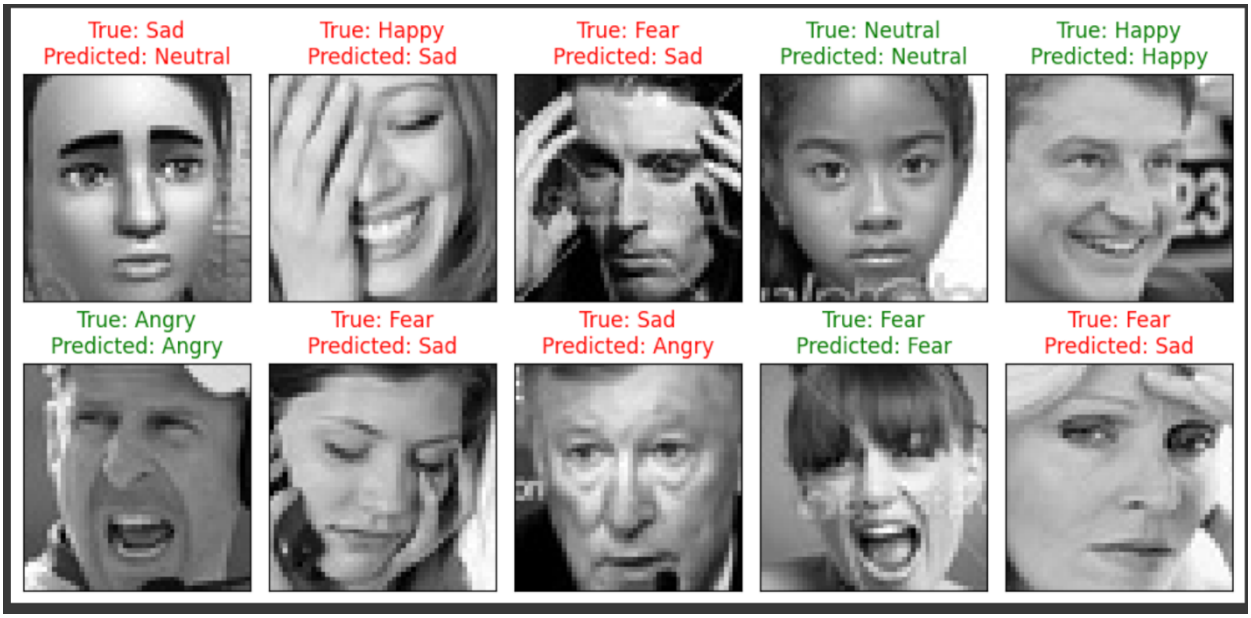
```
359/359 [==============================] - 10s 28ms/step - loss: 1.8045 - accuracy: 0.6311
113/113 [==============================] - 3s 27ms/step - loss: 2.0106 - accuracy: 0.5552
final train accuracy = 63.11 , validation accuracy = 55.52
```

Confusion Matrix:



Confusion matrix of CNN Model from Scratch

It can be observed that for the disgust category the model is unable to predict any of the images. Besides the model is not that accurate in predicting actual classes for a given image.



Actual v/s Predicted Images

These are some of the images from the test data. Clearly, it is seen that the model is making errors in predicting the actual true class for a given image. Therefore, we can conclude that the model is not suitable for us in predicting class.

2. CNN Model with Image Augmentation

What is Image augmentation?

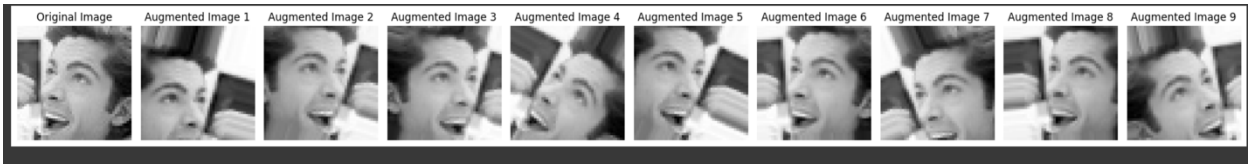
Image augmentation is a technique used in computer vision and image processing to artificially expand the size of an image dataset by applying various transformations to the images in the dataset. This process helps to increase the variability and diversity of the data, which can lead to improved performance of deep learning models, particularly in tasks such as image classification, object detection, and image segmentation.

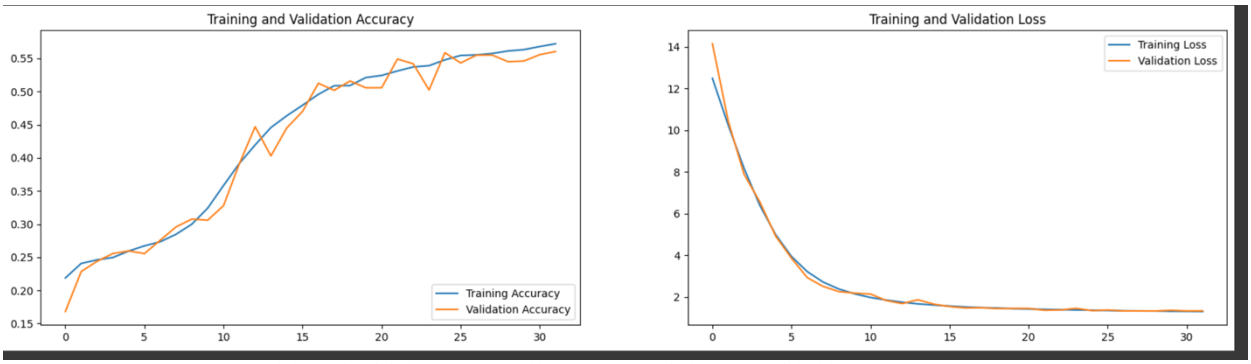
CNN Model with Image Augmentation Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 activation (Activation)     (None, 48, 48, 32)        0

 conv2d_1 (Conv2D)           (None, 48, 48, 64)        18496

 activation_1 (Activation)   (None, 48, 48, 64)        0

 batch_normalization (Batch  (None, 48, 48, 64)        256
 Normalization)

 max_pooling2d (MaxPooling2  (None, 24, 24, 64)        0
 D)

 dropout (Dropout)           (None, 24, 24, 64)        0

 conv2d_2 (Conv2D)           (None, 24, 24, 128)       73856

 activation_2 (Activation)   (None, 24, 24, 128)       0

 conv2d_3 (Conv2D)           (None, 22, 22, 256)       295168

 activation_3 (Activation)   (None, 22, 22, 256)       0

 batch_normalization_1 (Bat  (None, 22, 22, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 256)       0
 g2D)

 dropout_1 (Dropout)         (None, 11, 11, 256)       0

 conv2d_4 (Conv2D)           (None, 11, 11, 512)       1180160

 activation_4 (Activation)   (None, 11, 11, 512)       0

 conv2d_5 (Conv2D)           (None, 11, 11, 512)       2359808

 activation_5 (Activation)   (None, 11, 11, 512)       0

 batch_normalization_2 (Bat  (None, 11, 11, 512)       2048
...
Total params: 17046535 (65.03 MB)
Trainable params: 17044871 (65.02 MB)
Non-trainable params: 1664 (6.50 KB)
```

There are **17,046,535** total parameters in the CNN model, out of which **17,044,871** are trainable and **1664** are non-trainable parameters. Same no of parameters are above, however, no of images increases due to image augmentation.

Images generated due to Image Augmentation method

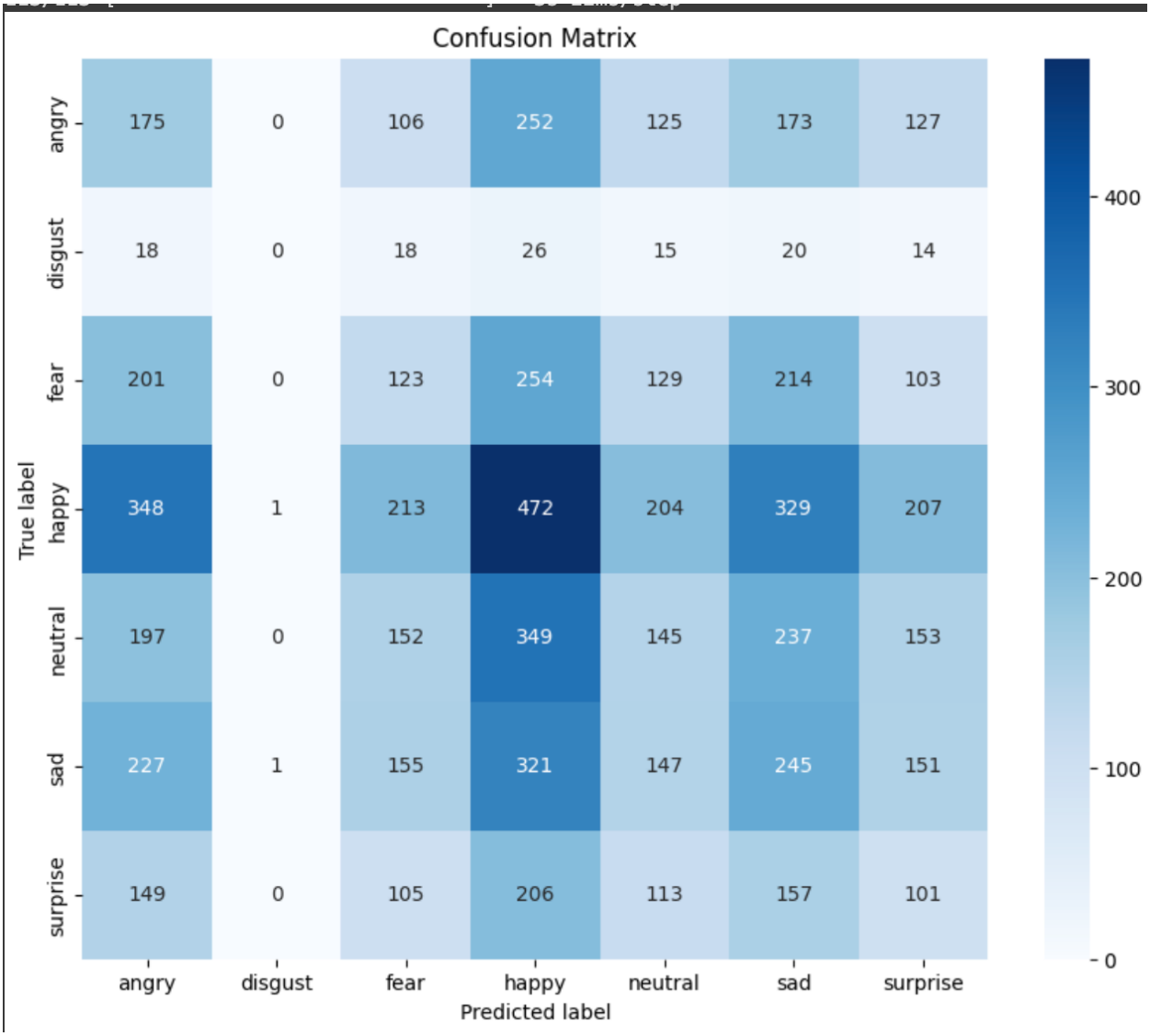




Training and Validation accuracy and loss after 10 epochs in CNN model with image augmentation.

```
359/359 [==============================] - 21s 58ms/step - loss: 1.2739 - accuracy: 0.5764
113/113 [==============================] - 3s 24ms/step - loss: 1.2292 - accuracy: 0.5928
final train accuracy = 57.64 , validation accuracy = 59.28
```

As compare to previous model that is (CNN model from Scratch) the performance of the model increase, as loss decreases. Previously, the model accuracy wad 55%. Now, with image augmentation the model accuracy increases by 4% i.e. 59%.



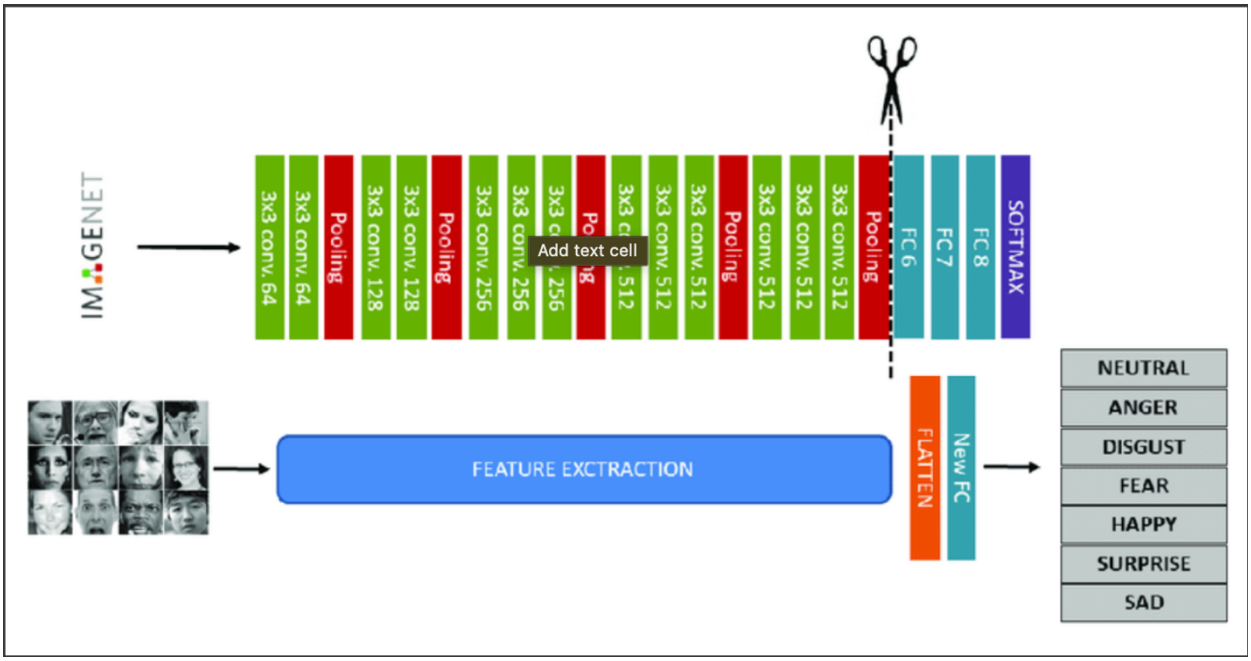
Confusion matrix of CNN Model with Image Augmentation

Here also for the disgust category the model is unable to predict any of the images. Therefore we cannot rely on this model.

3. CNN Model with Transfer learning (VGG16)

What is transfer learning?

The process of applying knowledge acquired from training a model on one task to another that is related to it is called transfer learning. The concept is to use a pre-trained model as a foundation for a new model that performs a different task or functions in a different domain. The pre-trained model has already been trained on a big dataset and has learnt broad patterns or features.



VGG16 Architecture

VGG16 consists of 16 layers, including convolutional layers, max pooling layers, and fully connected layers

i) **Convolutional Layers**: A ReLU (rectified linear unit) activation function comes after each convolutional layer in the network's initial architecture. The convolutional layers of the network are equipped with small 3x3 kernels that have a stride of 1 and a padding of 1, which allow the network to preserve the input's spatial dimensions.

ii) **Max Pooling Layers**: These layers minimize the spatial dimensions of the feature maps and come after a few convolutional layers. They have a 2x2 kernel and a stride of 2.

iii) **Fully Connected Layers**: The network moves to three fully connected layers after a sequence of convolutional and max pooling layers. There are 4096 neurons in each of the first two fully linked layers, and the number of neurons in the final layer is the same as the number of classes in the classification task. Mostly these fully connected layers are customize as per the requirements of the problem.

iv) **Softmax Output**: The last layer outputs the probability for each class using a softmax activation function.

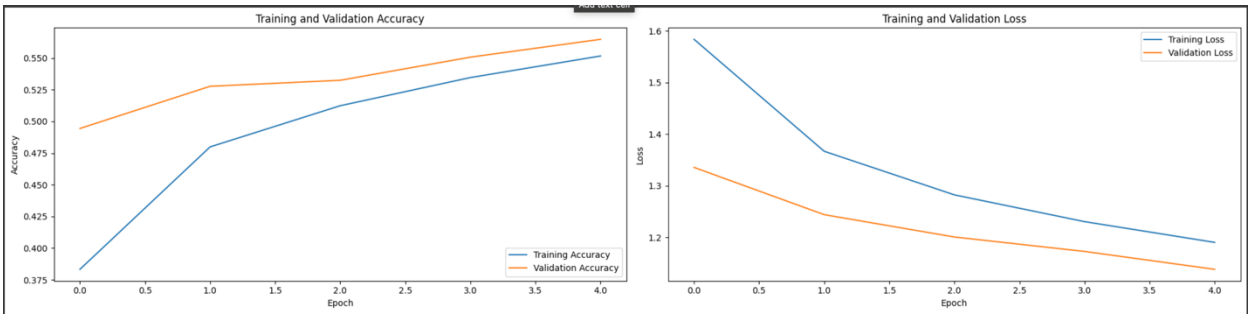
During the train, first of all, we need to change the size of the image from **(48, 48, 1)** to **(224, 224, 3)** in order to train the model, since these pre-trained model only accept images in **(224, 224, 3)** format.

To train VGG16 for custom data, we are removing **last 3 layers** from this model. Instead, we will build our own layers to train the model, according to the problem requirements.

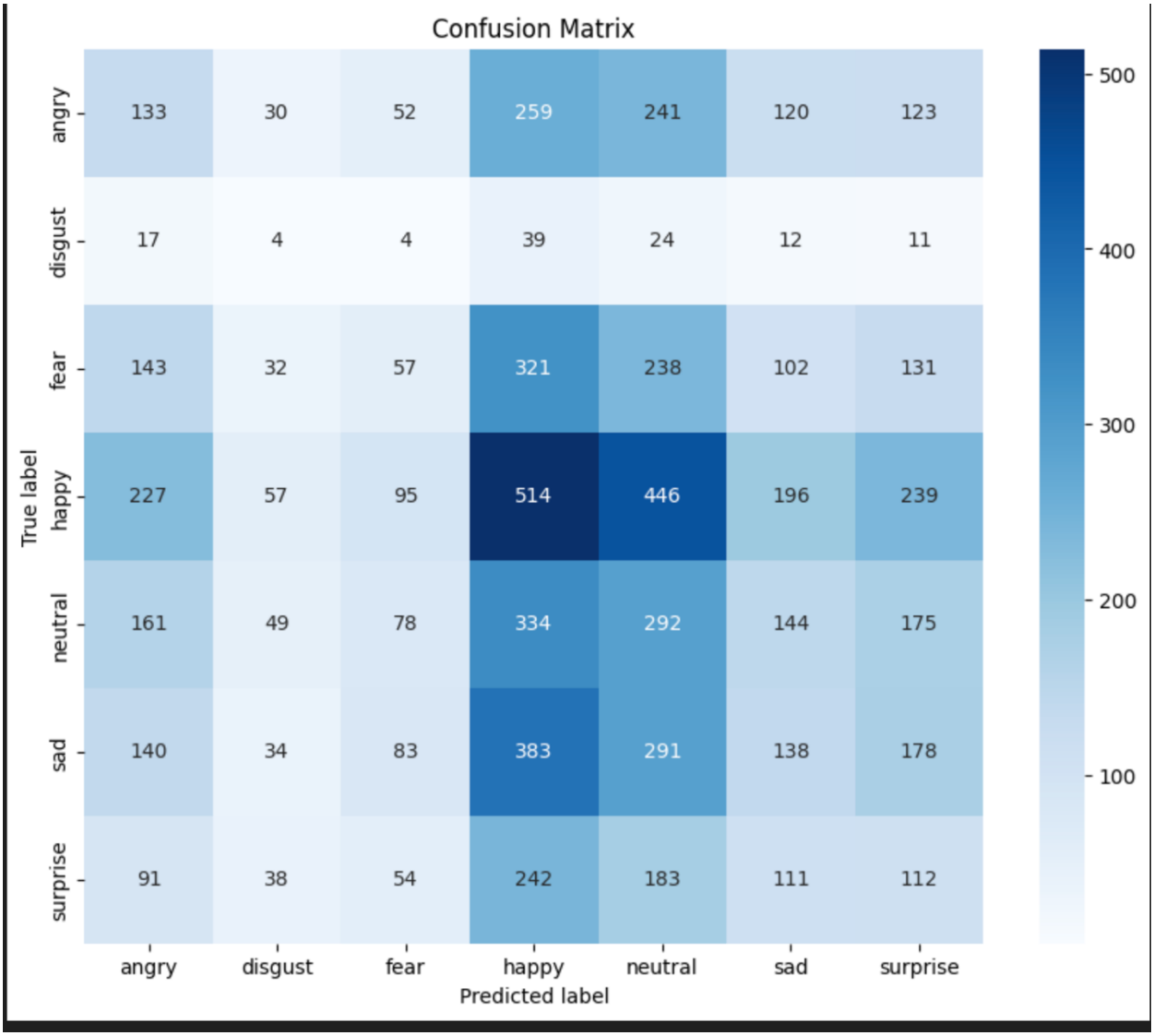
VGG16 Model Summary:

```
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_3 (Dense)             (None, 1024)              25691136
...
Total params: 40934215 (156.15 MB)
Trainable params: 30939143 (118.02 MB)
Non-trainable params: 9995072 (38.13 MB)
```

There are **40,934,215** parameters, out of which **30,939,143** are trainable parameters and **9,995,072** are non-trainable parameters.



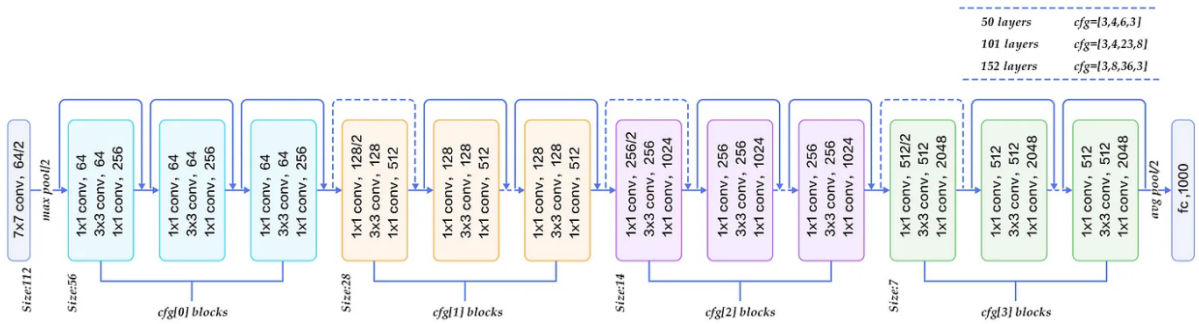
Training and Validation accuracy and loss after 4 epochs in CNN model with transferring learning (VGG16)



Confusion matrix of CNN Model with Transfer Learning (VGG16)

Here the model is able to make some actual prediction for the disgust category. However, there is still a lot of improvement need to be done in order to consider it as a final model.

4. CNN Model with ResNet50



ResNet50 Architecture

Convolutional layers, residual blocks, max pooling layers, and fully linked layers make up ResNet50's 50 layers. This is how the architecture is broken down:

i) **Residual Blocks**: Often referred to as shortcuts or skip connections, residual blocks are the main component of ResNet architecture. These blocks are made up of several convolutional layers connected by a shortcut link that skips a layer or layers. The network can learn residual functions thanks to the shortcut connection, which adds the block's input to its output.

ii) **Convolutional Layers**: A max pooling layer comes after the network's convolutional layer. Subsequently, the network consists of many residual blocks, with convolutional layers using 3x3 filters and ReLU activation functions in each.

iii) **Max Pooling Layers**: Feature maps' spatial dimensions are occasionally reduced and their abstraction level is raised by using max pooling layers.

iv) **Global Average Pooling**: The network employs global average pooling to decrease the feature map size to a single value per channel following the sequence of residual blocks.

v) **Fully Connected Layer**: There are as many neurons in the last fully connected layer as there are classes in the classification task.

vi) **Softmax Output**: The last layer outputs the probability for each class using a soft max activation function.

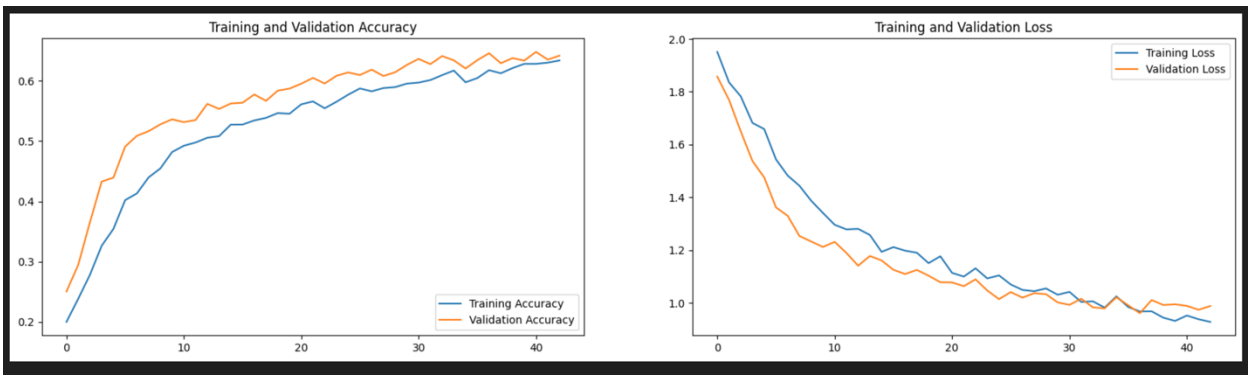
During the train, first of all, we need to change the size of the image from **(48, 48, 1)** to **(224, 224, 3)** in order to train the model, since these pre-trained model only accept images in **(224, 224, 3)** format.

To train ResNet50 for custom data, we are removing last **50 layers** from this model. Instead, we will build our own layers to train the model, according to our dataset/requirements.

ResNet50 Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50v2 (Functional)     (None, 7, 7, 2048)        23564800

 dropout (Dropout)           (None, 7, 7, 2048)        0

 batch_normalization (Batch  (None, 7, 7, 2048)        8192
 Normalization)

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 64)                6422592

 batch_normalization_1 (Bat  (None, 64)                256
 chNormalization)

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 7)                 455

=================================================================
Total params: 29996295 (114.43 MB)
Trainable params: 22779527 (86.90 MB)
Non-trainable params: 7216768 (27.53 MB)
_________________________________________________________________
```

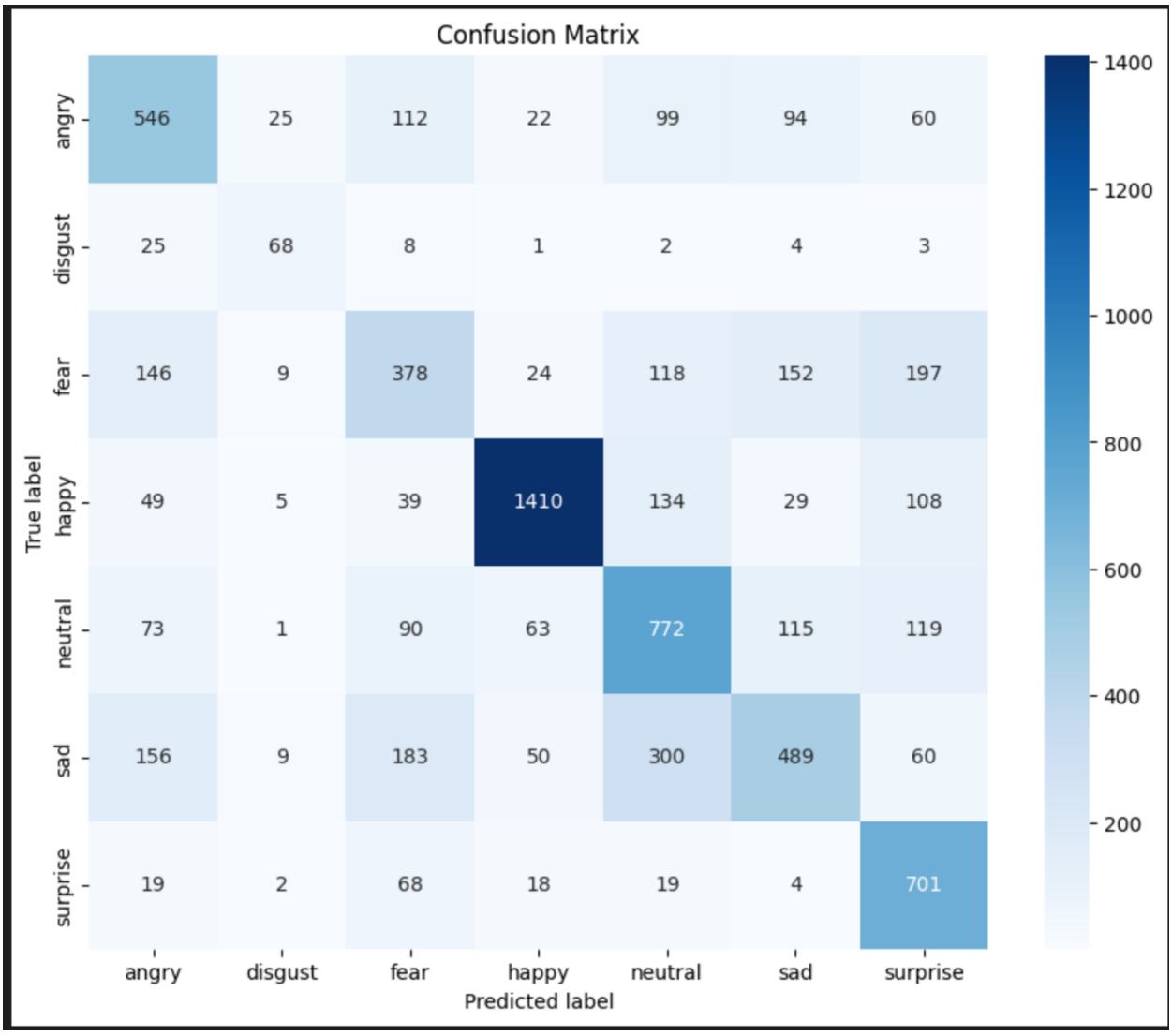
In total, there are **29,996,295** parameters, out of which **22,779,527** are trainable parameters and **7,216,768** are non-trainable parameters in this ResNet50 model.



Training and Validation accuracy and loss after 45 epochs in CNN model with transferring learning (ResNet50)

Classification Report:	precision	recall	f1-score	support
angry	0.54	0.57	0.55	958
disgust	0.57	0.61	0.59	111
fear	0.43	0.37	0.40	1024
happy	0.89	0.79	0.84	1774
neutral	0.53	0.63	0.58	1233
sad	0.55	0.39	0.46	1247
surprise	0.56	0.84	0.67	831
accuracy			0.61	7178
macro avg	0.58	0.60	0.58	7178
weighted avg	0.61	0.61	0.60	7178

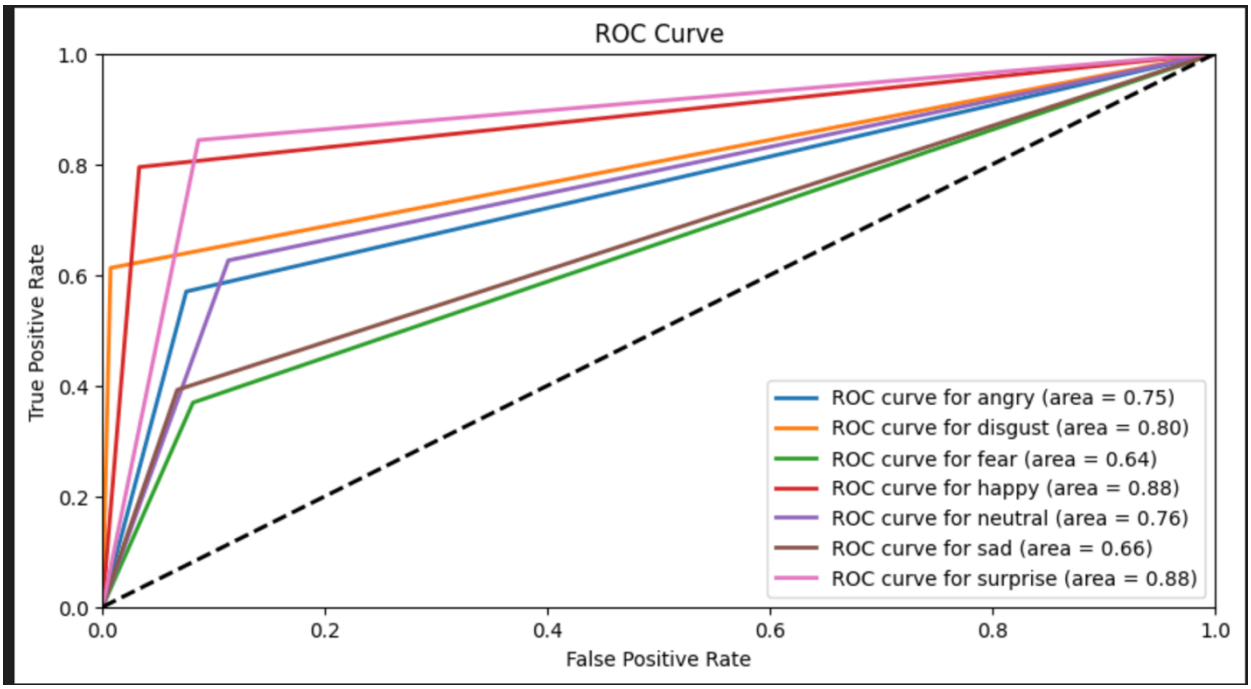
Classification Report



Confusion matrix of CNN Model with Transfer Learning (ResNet50)

Here the model is able to make most of the actual prediction for the disgust category as well as for other classes. It can consider this as our final model.

AUC ROC plot for each class



AUC ROC plot

For happy, surprise and disgust classes, the ResNet50 model is able to predict with more than 80% accuracy. For remaining classes we will the model is able to predict with decent accuracy.

This is one of the many ways one can approach a computer vision problem. However, in real world scenarios, organisations use the transfer learning method to train CNN models, as the institutes do not have enough computational power to train a model from scratch. Besides, funds also play a crucial role in training a model from scratch.