THE CITY COLLEGE OF NEW YORK

Unlocking the Secrets of Emotion:

A Breakthrough in Facial Expression Recognition on Video and Image

Abstract

Effective communication among human beings involves the combined use of facial expressions, verbal language, and text. As social beings, humans rely on social interactions for our well-being. Emotions play a crucial role in these interactions, as they convey meaning alongside verbal communication to those we interact with. Automatic facial expression recognition is a method used to automatically detect, identify, and comprehend emotions displayed by others. Numerous techniques have been suggested to enhance the precision of emotion recognition from facial cues. Convolutional Neural Networks (**CNNs**) have shown encouraging outcomes in this field. Nevertheless, the majority of existing **CNN** models demand extensive computational resources for training and processing emotional recognition and do not have high accuracy due to the complexity of the problem and limitations associated with existing datasets. In this report, I will introduce a new simple neural netowork for facial expression recognition that increases the accuracy of previous researched models. I conducted my training on the **FER-2013** dataset and on the The Extended Cohn-Kanade (**CK+**) dataset that contain static facial images in pixels and in grayscale. I implemented a Convolution Neural Network Long Short-Term Memory Network (**CNN LSTM**) to classify the images into seven different facial expressions (angry, disgust, fear, happy, neutral, sad, and surprise), yielding an accuracy of **82.45%** on the test data. I made all the materials publicly accessible at: https://github.com/georgiosioannoucoder/fervi.

Georgios Ioannou

CSC 44700

Professor Erik K. Grimmelmann

19 May 2023

TABLE OF CONTENTS

1. Introduction 5
2. Facial Recognition and Facial Expression 7
3. CNN 9
4. LSTM 11
5. Datasets 12
6. Solving Issue With Limited Data 18
7. Proposed Model Architecture 21
8. Evaluate Model 24
9. Deploying Model 27
10. Methodology Behind Web Application and Deploying Model 27
11. Conclusion 28

LIST OF FIGURES

1. General Overview of FER System 5
2. Six Universal Facial Expressions 7
3. FER-2013 Dataset Information 13
4. FER-2013 Data Frame 13
5. FER-2013 Data Frame With emotion_label Column 14
6. The Extended Cohn-Kanade (CK+) Dataset Information 14
7. The Extended Cohn-Kanade (CK+) Data Frame 15
8. The Extended Cohn-Kanade (CK+) Data Frame With emotion_label Column 15
9. Image For Each Emotion From The Extended Cohn-Kanade (CK+) Dataset 16
10. Emotion Classes Count Column Chart 16
11. Emotion Classes Count Pie Chart 17
12. Image Files Used By Usage Type 17
13. Geometric Transformation 19
14. Photometric Transformation 19
15. Adding/Removing Glasses 20
16. Face and Pose Alignment 20
17. Neural Network Summary 22
18. Neural Network Architecture 23
19. Accuracy Graph 25
20. Loss Graph 25
21. Accuracy of Model on Test Data and Validation Data 26
22. Confusion Matrix 26
23. Classification Report 26
24. Misclassifications 27
25. Steps Needed to Detect a Facial Expression in Real-Time 28

LIST OF TABLES

1. Motion Cues and Pseudo-Muscles Used For Each Universal Expression 8

Unlocking the Secrets of Emotion:

A Breakthrough in Facial Expression Recognition on Video and Image

INTRODUCTION

The human face serves as the psychological core of an individual, expressing an array of subtle signals that can greatly enhance human-machine interactions when properly interpreted by computers. Facial expression recognition (**FER**) plays a crucial role in non-verbal communication, allowing human-machine interface (HMI) systems to understand intimate human emotions and intentions.

Through computer vision, machines are empowered to perceive and comprehend objects much like humans do, making **FER** a classification task. This involves inputting characteristic sets derived from the facial image into a classifier. **Feature extraction** methods such as Gabor wavelet transform, Haar wavelet transform, Local Binary Pattern (LBP), and Active Appearance Models (AAM) are based on static images, while dynamic-based approaches consider the temporal association within sequences of facial expressions. Figure 1 illustrates how **FER** systems work in general.

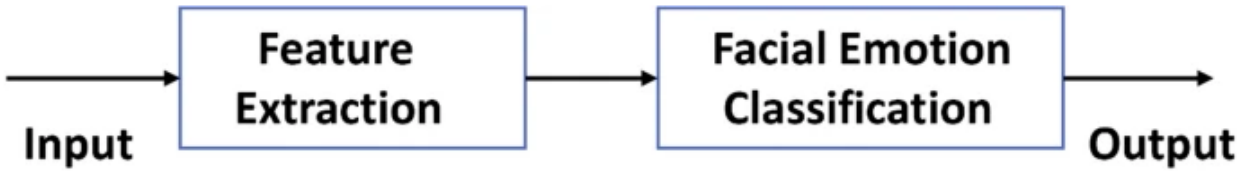

Figure 1. General Overview of FER System

Widely used schemes for **FER** include Support Vector Machine (SVM), Hidden Markov Model, AdaBoost, and Artificial Neural Networks (ANN), but Convolutional Neural Networks

(**CNNs**) have gained popularity due to their high accuracy in solving this complex problem. However, the use of deep learning poses challenges, requiring large amounts of data to train successful models. While **CNN** algorithms have shown improvements in facial expression identification, challenges such as long training times and low recognition rates in complex environments still exist.

Existing databases pose two major challenges for deep learning in **FER** methods: a limited number of images and images captured in structured conditions. This report focuses on determining emotions from input facial images, encompassing phases such as face detection, preprocessing, feature retrieval, alignment, and identification. Geometric attribute extraction, which describes the facial organ's location, is widely used for feature-based approaches.

The paper specifically emphasizes Facial Expression Recognition (**FER**), which involves classifying a person's emotion based on their facial expressions. Advances in **FER** have led to improvements in face detection, recognition, tracking, and cognitive detection, with applications ranging from assisting individuals with autism, remote monitoring of the elderly's health, human-computer interaction, fatigue detection in self-driving cars, security, marketing, clinical psychology, psychiatry, neurology, pain assessment, lie detection, intelligent settings, acting, and more.

To achieve high accuracy in classifying expressions into distinct categories, computers need to learn and understand specific features associated with each expression, necessitating a database with a vast collection of images encompassing diverse expression features.

FACIAL RECOGNITION AND FACIAL EXPRESSION

Before getting deeper into the topic let us first distinguish the difference between facial recognition and facial expression. Facial recognition and facial expression recognition are two distinct but interconnected aspects within the field of computer vision and affective computing. Facial recognition primarily focuses on the identification and verification of individuals based on their unique facial features, such as the arrangement of eyes, nose, and mouth. It aims to match a face in an image or video frame with a pre-existing database of known identities. On the other hand, facial expression recognition is concerned with the analysis and interpretation of facial movements and expressions to discern the emotional state or intention of an individual. It involves detecting and classifying specific facial muscle movements, such as eyebrow raises, smiles, or frowns, to determine the six universal expressions of anger, disgust, fear, happiness, sadness, surprise and also neutral. The six universal expressions are shown in Figure 2. Table 1 illustrates the motion cues and pseudo-muscles used for each emotion.

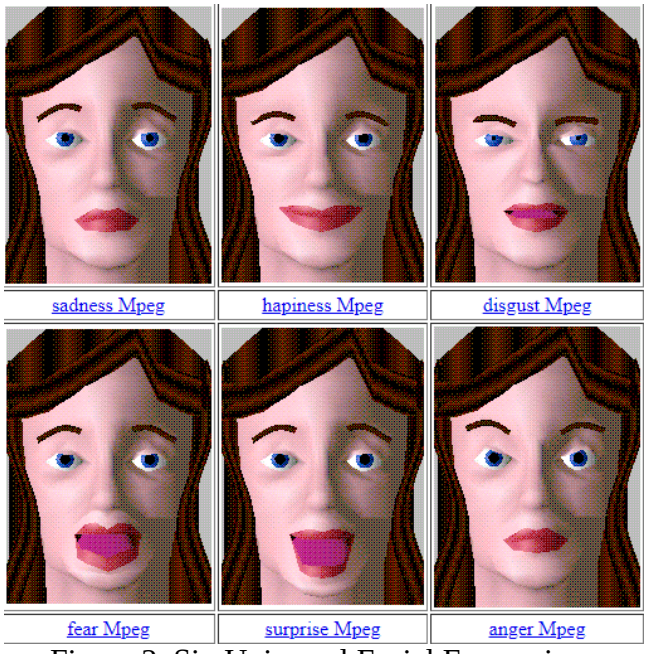

Figure 2. Six Universal Facial Expressions

EXPRESSION	MOTION CUES	PSEUDO-MUSCLES USED
Happiness	raising and lowering of mouth corners	6 linear muscles
Sadness	lowering of mouth corners raise inner portion of brows	6 linear muscles
Surprise	brows arch eyes open wide to expose more white jaw drops slightly	3 linear muscles
Fear	brows raised eyes open mouth opens slightly	5 linear muscles 1 sphincter for the mouth
Disgust	upper lip is raised nose bridge is wrinkled cheeks raised	6 linear muscles
Anger	brows lowered lips pressed firmly eyes bulging	4 linear muscles 1 sphincter for the mouth

Table 1. Motion Cues and Pseudo-Muscles Used For Each Universal Expression

While facial recognition aims to establish identity, facial expression recognition aims to discern emotional or cognitive states. However, there is an inherent relationship between the two, as facial expression recognition can benefit from facial recognition techniques to identify individuals and track their expressions over time. In fact, real-time **FER** systems require a face detection model to detect a face and capture the image so that it can then be fed into the model to process the image and predict the emotion. Understanding the differences and nuances between facial recognition and facial expression recognition is crucial for developing accurate and robust systems for emotion analysis, human-computer interaction, and various applications related to affective computing.

CNN

Since this report will utilize a Convolutional Neural Networks (**CNN**) let us explain briefly what they are, how they are used, and why they are used in this report. A Convolutional Neural Network (**CNN**) is a class of deep learning methods designed for processing grid-like data, such as images, and is predominantly used in various computer vision tasks. **CNNs** consist of an input layer, hidden layers, and an output layer, with the hidden layers including one or more layers that perform convolutions. These convolutions involve applying a filter (or kernel) to the input, which results in an activation map or feature map that summarizes the presence of detected features in the input. The filters can be handcrafted or learned during training in the context of a specific prediction problem. **CNNs** are inspired by biological processes, particularly the connectivity pattern between neurons in the animal visual cortex, and require relatively little pre-processing compared to other image classification algorithms. The architecture of a **CNN** typically includes convolution layers, pooling layers, fully connected layers, and normalization layers A key advantage of **CNNs** is their ability to automatically and adaptively learn spatial hierarchies of features through backpropagation algorithms, which reduces the need for human intervention in feature extraction. **CNNs** have been successfully applied in various domains, including image classification where they manipulate pixels and understand patterns. Since **CNNs** are notoriously famous for performing well on images they were chosen for this report's problem.

Convolution Layer

In Convolutional Neural Networks (**CNNs**), the convolution layer plays a crucial role in extracting features from input images. By convolving the input data with learned filters, this

layer captures spatial relationships between pixels. **CNN** terminologies such as 'filter,' 'kernel,' and 'feature detector' refer to the filters used. The result of sliding the filter across the image and computing the dot product is referred to as the 'Convolved Feature,' 'Activation Map,' or 'Feature Map.' Parameters like filter size and stride are chosen for this layer. The output of the convolutional layer is determined by the formula $(W–F + 2P)/S + 1 = O$, where F, P, and S represent the spatial extent, padding, and stride, respectively. Properly choosing the stride ensures that neurons align with the input, and zero padding is employed to achieve this alignment.

ReLu Layer

The ReLU layer eliminates negative values in the filtered image, replacing them with zero. This activation function prevents values from saturating at zero. The Rectified Linear Unit (ReLU) only activates a node if the input surpasses a threshold; otherwise, the output is zero. Notably, ReLU has a gradient equal to 1 facilitating effective error propagation during back-propagation.

$$\mathbf{f(x)=max(0,x)}$$

Pooling Layer

The pooling layer reduces the spatial dimensionality of the feature maps, resulting in more compact feature extraction. It produces a pooled feature map as its output. The two common pooling strategies are maximum pooling and mean pooling. Maximum pooling selects the maximum value within each pool, effectively down sampling the network. The formula $(I + 2P - F)/S + 1 = O$ is used to calculate the pooling layer, where I, S, P, and F represent the input matrix, stride, padding, and filter, respectively.

Fully Connected Layer

The fully connected layer transforms the pooled feature map from a two-dimensional structure into a one-dimensional vector, commonly known as a feature vector. This flattened feature vector is then utilized as input for classification tasks, serving as a conventional fully connected layer.

Softmax

The softmax layer is typically placed just before the output layer in a neural network. It ensures that the outputs of the network sum up to 1, making it suitable for multi-class classification problems. The number of nodes in the softmax layer matches the number of classes in the problem domain.

Batch Normalization

Batch normalization is a technique employed in **CNNs** to expedite the training process and maintain stable activations. It achieves this by normalizing the mean activation close to zero and the standard deviation close to one during training, aiding in more efficient gradient propagation.

LSTM

Since this report will also utilize Long Sort-Term Memory Networks (**LSTM**) let us discuss briefly what they are, how they are used, and why they are used in this report. Long Short-Term Memory (**LSTM**) networks are a type of recurrent neural networks (RNN) that are designed to handle and process time series data with long-term dependencies more accurately than conventional Recurrent Neural Networks (**RNNs**). Unlike RNNs, **LSTMs** can remember and

recall previous long-term time-series data and have automatic control to retain relevant features or discard irrelevant features in the cell state. **LSTMs** are widely used in various applications such as speech recognition, speech synthesis, and natural language processing. **LSTM** models converge quickly and give state-of-the-art performance for relatively small-sized models.

LSTMs have three gates to control features: the input gate, forget gate, and output gate. The input gate controls new information to flow into the cell state. The forget gate removes previous unimportant information from the cell state. The output gate regulates extracted information from the cell state and then decides the next hidden state. An **LSTM** model can automatically save or remove stored memory using these gates. The forget gate in **LSTMs** is represented as a linear identity function. It allows the network to forget some of the previously stored information that may not be relevant for future predictions. This feature is useful in processing sequence data for predictions where the network needs to capture only the relevant features and discard the irrelevant ones. For instance, in this report's problem we need to only focus on the pixels and areas of the image that will help us identify the emotion as shown in the Table 1.

DATASETS

The datasets that were used in this report to train and test the model were the **FER-2013** dataset and the The Extended Cohn-Kanade (**CK+**) dataset. The more data the better however I had an issue in finding datasets that were publicly available and therefore I was restricted to just two datasets. The **FER-2013** datasets contains 35,887 images that are 48x48 pixels and in grayscale. I chose to work with pixels from the start of the process as it is easier to work with pixels which are a value from 0 to 255 and manipulate them with numpy arrays. Therefore, the **FER-2013** has 35,887 rows and three columns. The first column is the target emotion labeled from 0 to 6 for the

six universal facial expressions and neutral emotion also. The second column is a string containing the pixels of the image. The third column indicates whether the image will be used for training, or public testing, or private testing. Figure 3 illustrates the information of the **FER-2013** dataset and Figure 4 displays few of its rows. For readability purposes I added another column named emotion_label to match the integer target emotion to string. This is shown in Figure 5.

```
fer_2013_df.shape = (35887, 3)

Unique emotions = [0, 1, 2, 3, 4, 5, 6]

# of Unique emotions = 7

3    8989
6    6198
4    6077
2    5121
0    4953
5    4002
1     547
Name: emotion, dtype: int64

Unique Usage = ['PrivateTest', 'PublicTest', 'Training']

# of Unique Usage = 3

Training       28709
PublicTest      3589
PrivateTest     3589
Name: Usage, dtype: int64

0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral
```

Figure 3. FER-2013 Dataset Information

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training
...	...	...	...
35882	6	50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...	PrivateTest
35883	3	178 174 172 173 181 188 191 194 196 199 200 20...	PrivateTest
35884	0	17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...	PrivateTest
35885	3	30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...	PrivateTest
35886	2	19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...	PrivateTest

35887 rows × 3 columns

Figure 4. FER-2013 Data Frame

	emotion	pixels	Usage	emotion_label
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training	Angry
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training	Angry
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training	Fear
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training	Sad
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training	Neutral
...	...	...	...	...
35882	6	50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...	PrivateTest	Neutral
35883	3	178 174 172 173 181 188 191 194 196 199 200 20...	PrivateTest	Happy
35884	0	17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...	PrivateTest	Angry
35885	3	30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...	PrivateTest	Happy
35886	2	19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...	PrivateTest	Fear

35887 rows × 4 columns

Figure 5. FER-2013 Data Frame With emotion_label Column

The Extended Cohn-Kanade (**CK+**) dataset is extremely similar to the FER-2013 dataset in the sense that it also contains 48x48 pixel grayscale images. However, this dataset is small with only 902 images. Figure 6, Figure 7, and Figure 8 shows the Extended Cohn-Kanade (**CK+**) dataset information.

```
ck_df.shape = (902, 3)

Unique emotions =  [0, 1, 2, 3, 4, 5, 6]

# of Unique emotions = 7

6    593
5     83
3     69
1     59
0     45
4     28
2     25
Name: emotion, dtype: int64

Unique Usage = ['PrivateTest', 'PublicTest', 'Training']

# of Unique Usage = 3

Training       720
PrivateTest     93
PublicTest      89
Name: Usage, dtype: int64

0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral
```

Figure 6. The Extended Cohn-Kanade (CK+) Dataset Information

	emotion	pixels	Usage
0	6	36 39 35 25 19 11 8 7 3 13 15 9 21 57 75 90 10...	Training
1	6	88 74 19 4 5 5 3 12 8 21 15 21 15 18 24 29 32 ...	Training
2	6	9 2 4 7 1 1 1 0 7 29 49 76 115 141 156 169 177...	Training
3	6	104 106 108 104 95 50 60 61 58 83 126 133 139 ...	Training
4	6	68 72 67 67 6 2 1 1 1 1 1 14 24 24 38 65 79 94...	Training
...	...	...	...
915	5	87 86 88 92 92 127 231 248 251 253 254 254 254...	PrivateTest
916	5	21 24 26 28 27 28 30 8 0 0 0 0 0 0 1 4 37 42 4...	PrivateTest
917	5	76 40 31 38 28 34 38 36 41 36 46 38 44 26 45 5...	PrivateTest
918	5	114 87 16 29 17 25 30 34 37 35 45 93 63 80 73 ...	PrivateTest
919	5	101 102 99 96 98 42 23 18 15 17 27 34 17 24 29...	PrivateTest

902 rows × 3 columns

Figure 7. The Extended Cohn-Kanade (CK+) Data Frame

	emotion	pixels	Usage	emotion_label
0	6	36 39 35 25 19 11 8 7 3 13 15 9 21 57 75 90 10...	Training	Neutral
1	6	88 74 19 4 5 5 3 12 8 21 15 21 15 18 24 29 32 ...	Training	Neutral
2	6	9 2 4 7 1 1 1 0 7 29 49 76 115 141 156 169 177...	Training	Neutral
3	6	104 106 108 104 95 50 60 61 58 83 126 133 139 ...	Training	Neutral
4	6	68 72 67 67 6 2 1 1 1 1 1 14 24 24 38 65 79 94...	Training	Neutral
...	...	...	...	...
915	5	87 86 88 92 92 127 231 248 251 253 254 254 254...	PrivateTest	Surprise
916	5	21 24 26 28 27 28 30 8 0 0 0 0 0 0 1 4 37 42 4...	PrivateTest	Surprise
917	5	76 40 31 38 28 34 38 36 41 36 46 38 44 26 45 5...	PrivateTest	Surprise
918	5	114 87 16 29 17 25 30 34 37 35 45 93 63 80 73 ...	PrivateTest	Surprise
919	5	101 102 99 96 98 42 23 18 15 17 27 34 17 24 29...	PrivateTest	Surprise

902 rows × 4 columns

Figure 8. The Extended Cohn-Kanade (CK+) Data Frame With emotion_label Column

Let us visualize a sample image for each emotion to get an idea with what we are goinf to work with. Figure 9 displays an image for each emotion. These images are from The Extended Cohn-Kanade (**CK+**) dataset however the pose of the face, the size, and the color of the image is the same in the FER-2013 dataset. Now it is a great point to mention that the majority of the images of the datasets **do not include people wearing glasses** and **faces have**

the same pose. Therefore, since the model has not been fed with images in which people wear glasses or have a different pose (sideways), it is expected to perform less accurate in those cases. Once again we can see how important the input data is.

Figure 9. Image For Each Emotion From The Extended Cohn-Kanade (CK+) Dataset

After combining the two datasets, inspecting and removing NULL and duplicate values the column chart in Figure 10 and Figure 11 are produced. As it can be seen from the column chart the model will be biased for the emotion disgust. This is because there were not enough images for the disgust emotion to give the model the ability to learn the patterns of the disgust emotion. Moreover, in general by feeding unbalanced classes to a model, the model will do a lot better for the classes that have more data. Therefore, the amount and type of data that is inputted into the model is extremely important in making sure that the model will be able to generalize and learn patterns for each class. To solve this imbalanced in the disgust emotion I performed random over-sampling on the disgust emotion using Imbalanced-learn.

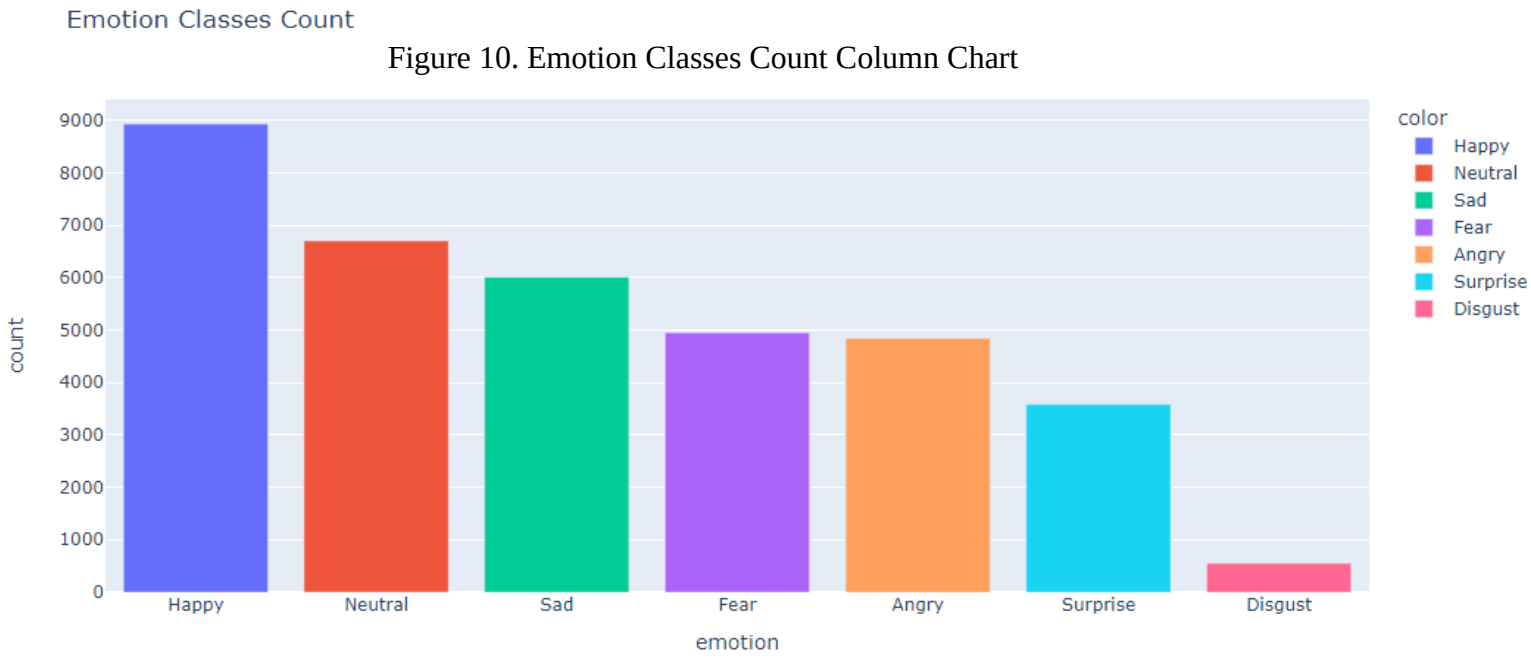

Figure 10. Emotion Classes Count Column Chart

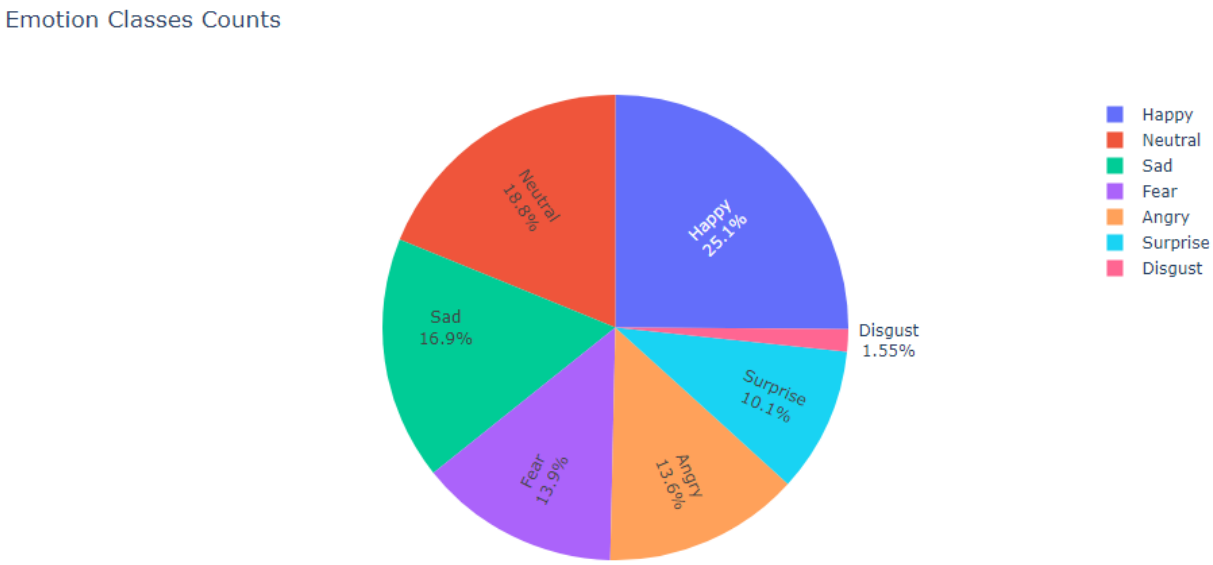

Figure 11. Emotion Classes Count Pie Chart

Figure 12 shows the amount of images that were used for each type of testing and training. **80%** was used for testing, **10%** was used for validation, and **10%** was used for testing.

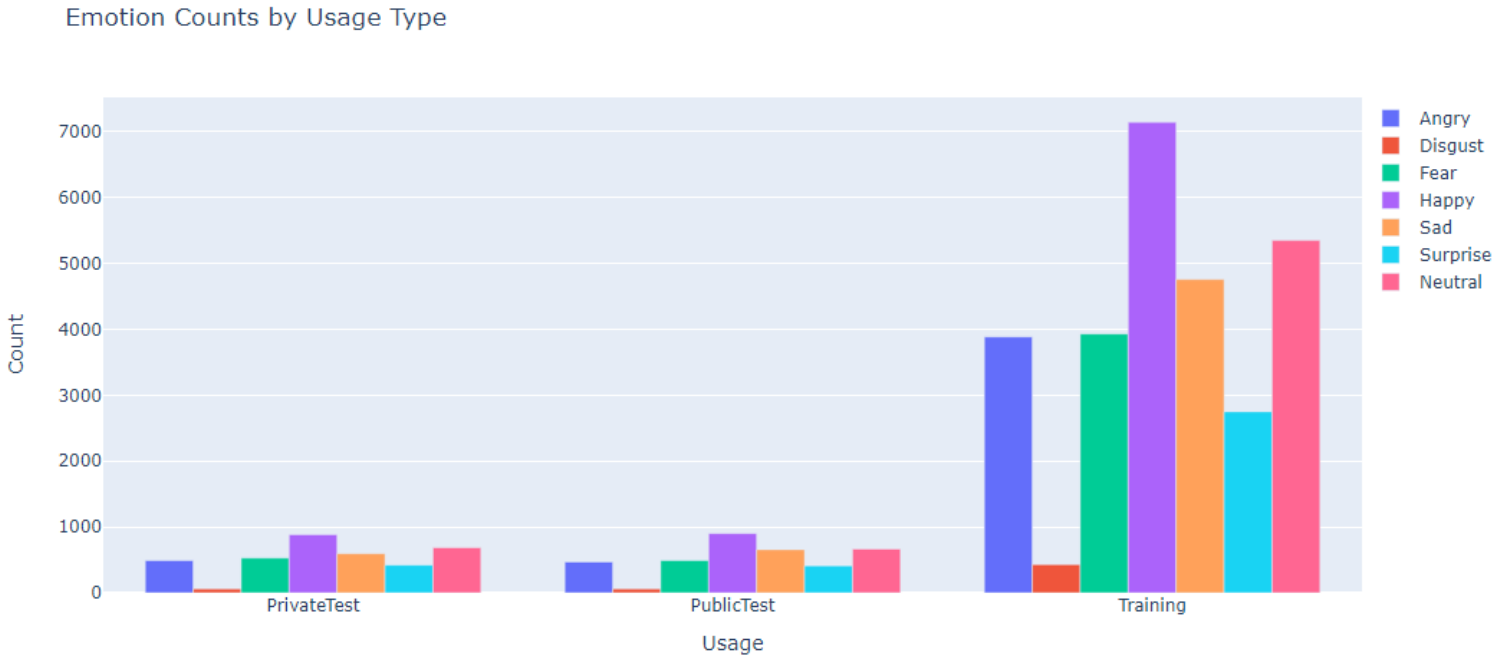

Figure 12. Image Files Used By Usage Type

SOLVING ISSUE WITH LIMITED DATA

One way to solve the issue with limited data when working with images or when you want to increase the size of an unbalanced class is **data augmentation**. The main goal of data augmentation is to make you data more generalized and thus allow your model to generalize and predict with higher accuracy. Moreover, data augmentation allows you to create more data nad therefore feed more data to the model. Transforming the base image dataset is a crucial process that enhances the generalization performance of a model, particularly in the context of facial data. This process takes into account various combinations of an individual's appearance on different days. A brief overview of the techniques used for each transformation type is provided below. In facial data transformation, various techniques are employed to account for the diverse ways an individual may appear on any given day. These techniques are crucial in improving the generalization performance of a model when working with facial data.

Data augmentation techniques can be broadly classified into two categories: **geometric transformations** (Figure 13) and **photometric transformations** (Figure 14). These techniques have been employed in numerous learning-based computer vision tasks. Geometric transformations involve altering the geometry of an image by repositioning its pixels, including operations such as rotation, reflection, translation, and flipping. On the other hand, photometric transformations **modify the RGB channels** by shifting pixel colors to new values. Some of the primary photometric approaches are:

- **Grayscale conversion**: Reducing color channels to black and white shades
- **Color jittering**: Incorporating various manipulations, including inverting, adding, decreasing, and multiplying color channels

- **Filtering**: Applying operations like edge enhancement, blurring, and sharpening
- **Lighting perturbation**: Altering environmental lighting conditions
- **Noise addition**: Modifying pixel granularity
- **Vignetting**: Softening or shading image edges
- **Contrast adjustment**: Applying color fixations as layers

These methods have been shown to improve the performance of machine learning models by providing more diverse training datasets.

Figure 13. Geometric Transformation

Figure 14. Photometric Transformation

We can also use data augmentation to solve the issue of images not having people wearing glasses. The act of removing and wearing various accessories, such as glasses, earrings, and nose rings, is prevalent in everyday life. Figure 15 displays how we can augment images and add or remove accessories such as glasses. The first three columns add glasses to the images and the last three columns remove the glasses from the images.

Figure 15. Adding/Removing Glasses

Finally, we can also use data augmentation to solve the issue of the face pose. Figure 16 illustrates how we can augment images to also make the model predict the emotion of a face sideways more accurate.

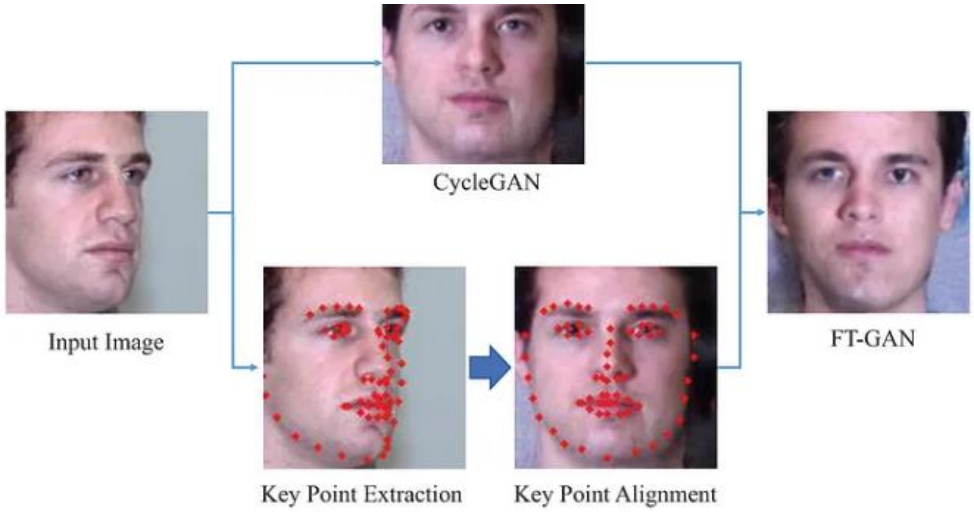

Figure 16. Face and Pose Alignment

PROPOSED MODEL ARCHITECTURE

Let us now turn our attention to the model that was implemented for this report. The model has a total of **457,863** parameters of which 457,031 are trainable and 832 are non-trainable. is a combination of Convolutional Neural Networks (**CNN**) and Long Short-Term Memory (**LSTM**) layers. It is designed for image classification tasks, where the input images have a size of **48x48 pixels** with a single channel (**grayscale**). The model consists of several layers, which I will explain step by step:

1. The model is initialized as a Sequential model.
2. The input layer is created with the shape of (48, 48, 1), which corresponds to the dimensions of the input images.
3. The following layers consist of Conv2D, BatchNormalization, and Activation layers. These layers are responsible for learning the features of the input images. Conv2D layers apply filters to the input, and BatchNormalization layers normalize the outputs of the Conv2D layers along the specified axis. Activation layers apply the ReLU activation function to introduce non-linearity.
4. The model then continues with more Conv2D, BatchNormalization, Activation, and MaxPooling2D layers. MaxPooling2D layers downsample the feature maps by taking the maximum value in each pooling window.
5. The CNN layers are followed by Reshape and LSTM layers. The Reshape layer transforms the output from the CNN layers into the required format for the LSTM layers. LSTM layers are recurrent layers that can capture long-term dependencies in sequential data.

6. Another Reshape and LSTM layer are added for further processing of the features.
7. The Dense layer with 200 units and ReLU activation function is used to learn more complex patterns and relations between features.
8. A Dropout layer with a dropout rate of 0.6 is added to prevent overfitting by randomly dropping out some of the neurons during training.
9. The final Dense layer with 7 units and softmax activation function is responsible for outputting the probabilities for each of the 7 classes.
10. Finally, the model is compiled with the Adam optimizer with a learning rate of 0.0002, categorical cross-entropy loss, and accuracy as the evaluation metric.

Figure 17 displays the summary of the model as described above and Figure 18 visualizes the neural network architecture.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320

batch_normalization (BatchN  (None, 46, 46, 32)        128
ormalization)

activation (Activation)      (None, 46, 46, 32)        0

conv2d_1 (Conv2D)            (None, 46, 46, 64)        18496

batch_normalization_1 (Batc  (None, 46, 46, 64)        256
hNormalization)

activation_1 (Activation)    (None, 46, 46, 64)        0

max_pooling2d (MaxPooling2D  (None, 23, 23, 64)        0
)

conv2d_2 (Conv2D)            (None, 21, 21, 64)        36928

batch_normalization_2 (Batc  (None, 21, 21, 64)        256
hNormalization)

activation_2 (Activation)    (None, 21, 21, 64)        0

conv2d_3 (Conv2D)            (None, 21, 21, 128)       73856

batch_normalization_3 (Batc  (None, 21, 21, 128)       512
hNormalization)

activation_3 (Activation)    (None, 21, 21, 128)       0

max_pooling2d_1 (MaxPooling  (None, 10, 10, 128)       0
2D)

conv2d_4 (Conv2D)            (None, 8, 8, 128)         147584

batch_normalization_4 (Batc  (None, 8, 8, 128)         512
hNormalization)

activation_4 (Activation)    (None, 8, 8, 128)         0

max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)         0
2D)

reshape (Reshape)            (None, 16, 128)           0

lstm (LSTM)                  (None, 128)               131584

reshape_1 (Reshape)          (None, 2, 64)             0

lstm_1 (LSTM)                (None, 64)                33024

dense (Dense)                (None, 200)               13000

dropout (Dropout)            (None, 200)               0

dense_1 (Dense)              (None, 7)                 1407

=================================================================
Total params: 457,863
Trainable params: 457,031
Non-trainable params: 832
_________________________________________________________________
```

Figure 17. Neural Network Summary

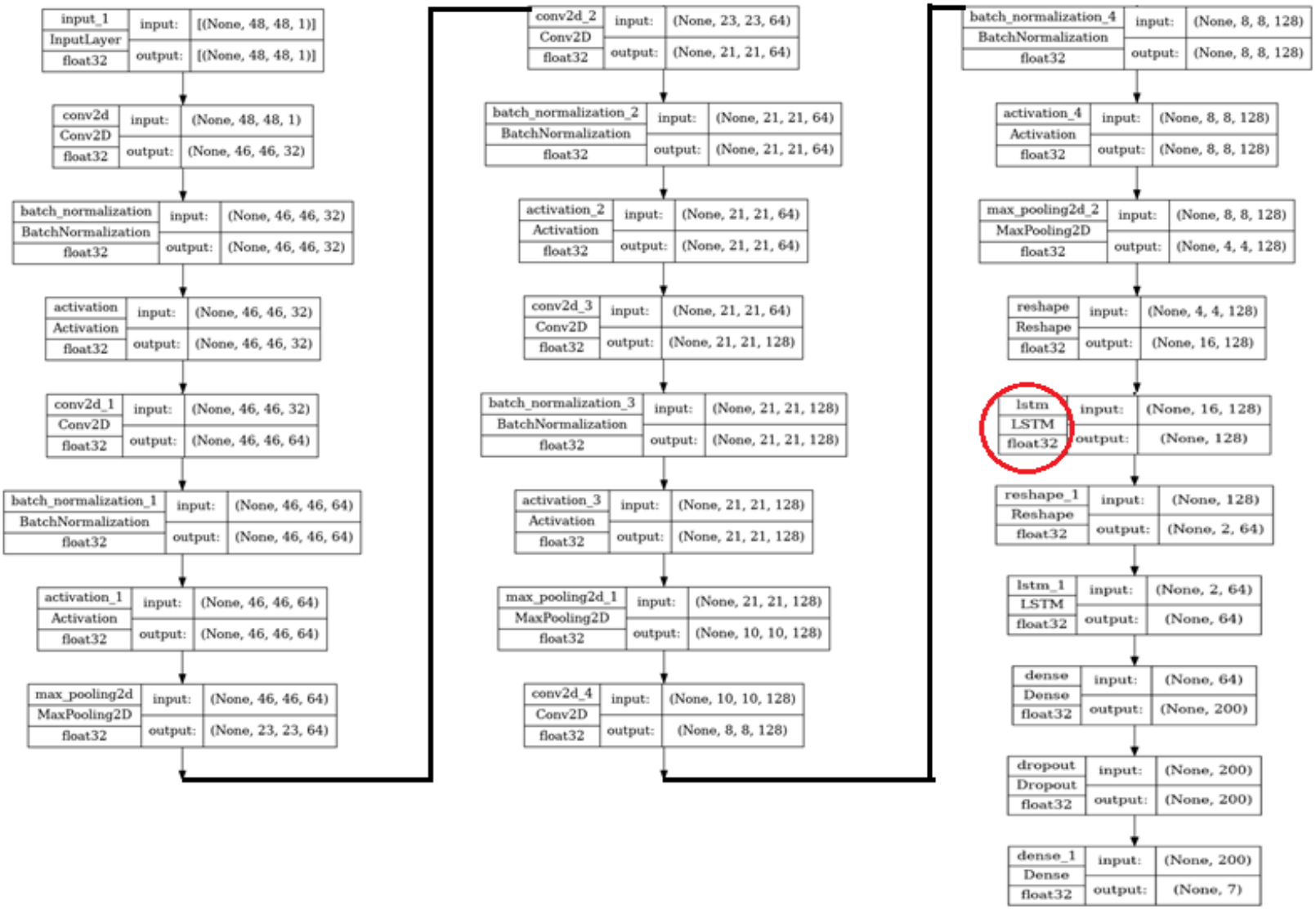

Figure 18. Neural Network Architecture

EVALUATE MODEL

After building and training the model for a total of 48 epochs is time to evaluate it. Figure 19 displays the accuracy graph and Figure 20 displays the loss graph. The validation accuracy is expected to increase and the validation loss is expected to decrease. As we can see the validation accuracy increases however the validation loss does not decrease. The main reason why the validation loss is not decreasing is due to the fact of class imbalance. Even though I over sampled the disgust emotion the model learned to predict the majority classes better, thus increasing accuracy. However, the validation loss does not decrease as the models till struggles with the minority class in this case the emotion disgust. Figure 21displays the accuracy of the model both on the test data and validation data. In Figure 22, we can see the confusion matrix where it shows that the model was mostly confused for the emotion surprise in the last row. Figure 23 displays the classification report. Lastly, the three images in Figure 24, are some of the misclassifications where the model predicted the wrong emotion. The True label is the labeled emotion in the dataset and the Pred label is what the model predicted as the emotion.

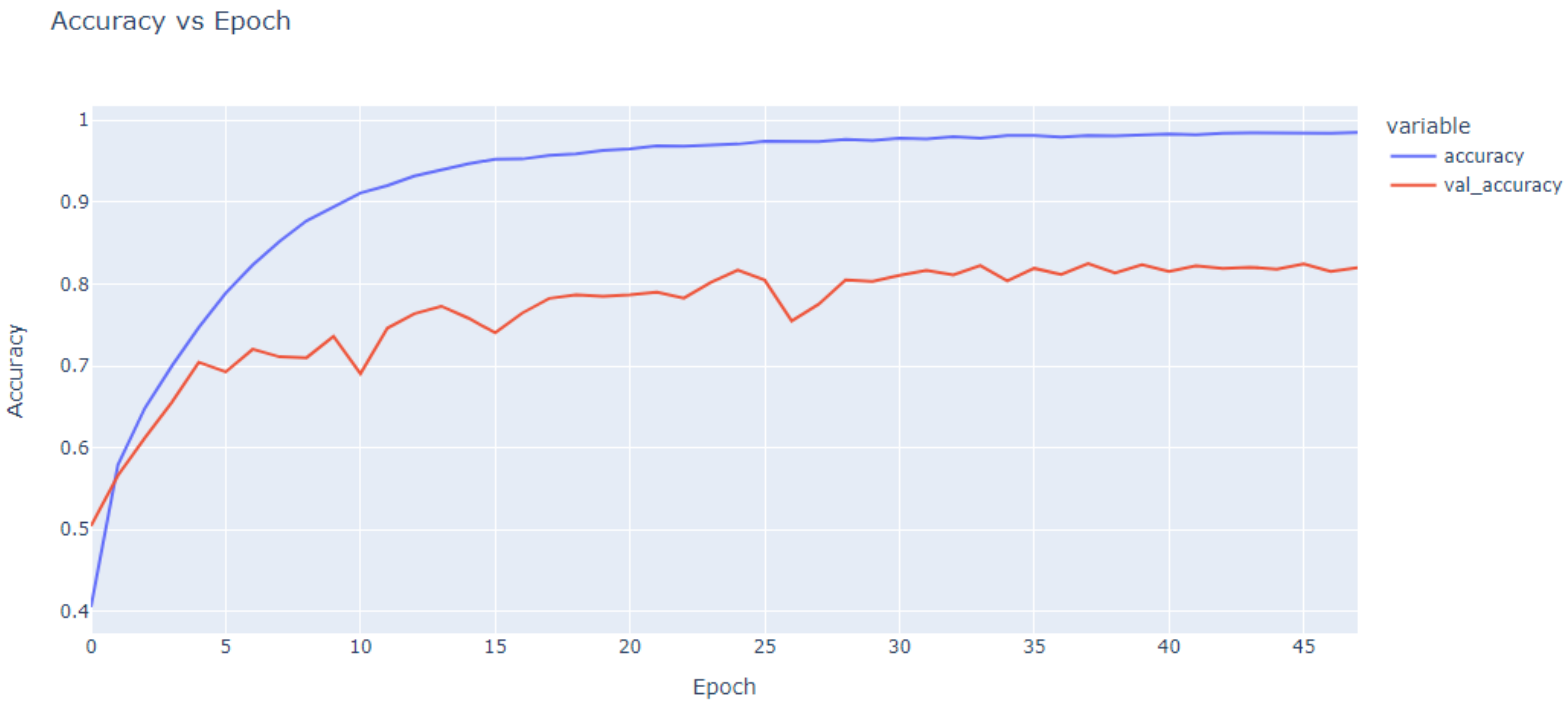

Figure 19. Accuracy Graph

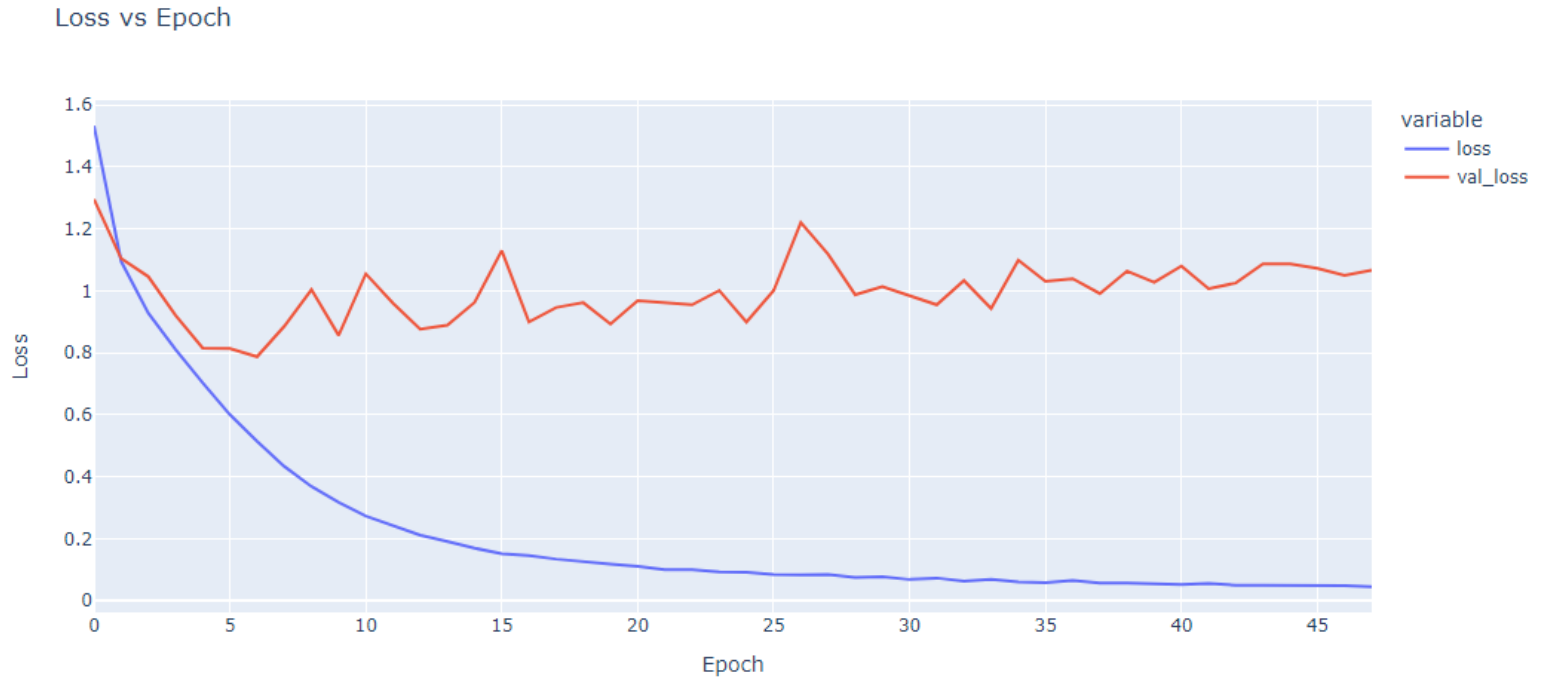

Figure 20. Loss Graph

```
print("Accuracy of model on testing data: ", model.evaluate(X_test, y_test)[1] * 100 , "%", sep = "")
```

```
196/196 [==============================] - 1s 5ms/step - loss: 0.9956 - accuracy: 0.8245
Accuracy of model on testing data: 82.44799971580505%
```

```
print("Accuracy of model on validation data: ", model.evaluate(X_valid, y_valid)[1] * 100 , "%", sep = "")
```

```
196/196 [==============================] - 1s 6ms/step - loss: 0.9917 - accuracy: 0.8248
Accuracy of model on validation data: 82.48000144958496%
```

Figure 21. Accuracy of Model on Test Data and Validation Data

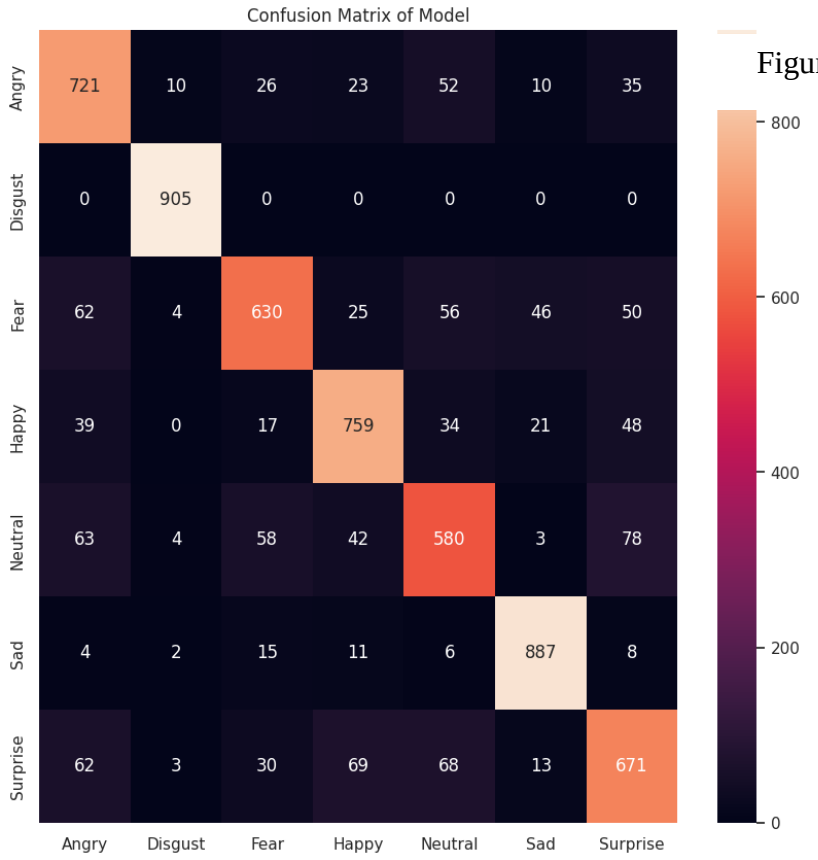

Figure 22. Confusion Matrix

	precision	recall	f1-score	support
0	0.76	0.82	0.79	877
1	0.98	1.00	0.99	905
2	0.81	0.72	0.76	873
3	0.82	0.83	0.82	918
4	0.73	0.70	0.71	828
5	0.91	0.95	0.93	933
6	0.75	0.73	0.74	916
accuracy			0.82	6250
macro avg	0.82	0.82	0.82	6250
weighted avg	0.82	0.82	0.82	6250

Figure 23. Classification Report

Figure 24. Misclassifications

DEPLOYING MODEL

Once the evaluation of the model is done, it is time to deploy it and make a web application where users can test and interact with the model. For deploying the model and making the web application several technologies and frameworks were used such as Flask, HTML, CSS, JavaScript, and Heroku. The name of the web application is Facial Expression Recognition on Video and Image(**FERVI**). Please feel free to visit the web application and test the model at https://fervi.herokuapp.com/. The web application has the following four options:

1. Detect the facial expression from an image both single and multiple people.
2. Detect the facial expressions of multiple people in real-time video.
3. Acting practice where a user must try to match the requested emotion.
4. Detect the facial expressions of a single people in real-time video and generate charts based on the emotion.

METHODOLOGY BEHIND WEB APPLICATION AND DEPLOYING MODEL

As I have previously mentioned, when we want to predict the emotion in real-time we need to have two models. The first model is responsible for detecting the face and the second model is responsible for detecting the expression. To do that we capture **video frames** continuously.

Think of video frames as **images**. These video frames are **sent to the first model to predict if a face has been detected**. If no face has been detected then go to the next video frame. However, **if a face was detected, then resize the video frame to 48x48 pixels** as our model was trained on 48x48 images. Once the video frame has been resized **convert it to grayscale** by normalizing the RGB values as our model was trained on grayscale images. At this step we can choose to **either do feature extraction on the video frame or not** depending on how we trained our model to get the input image. Since I did not use any feature extraction when I trained the model listed above this step is skipped. Finally, we **input the video frame into the model for classification**. Once the model makes the prediction, we **send the predicted emotion to the front-end**. All these steps are illustrated in Figure 25.

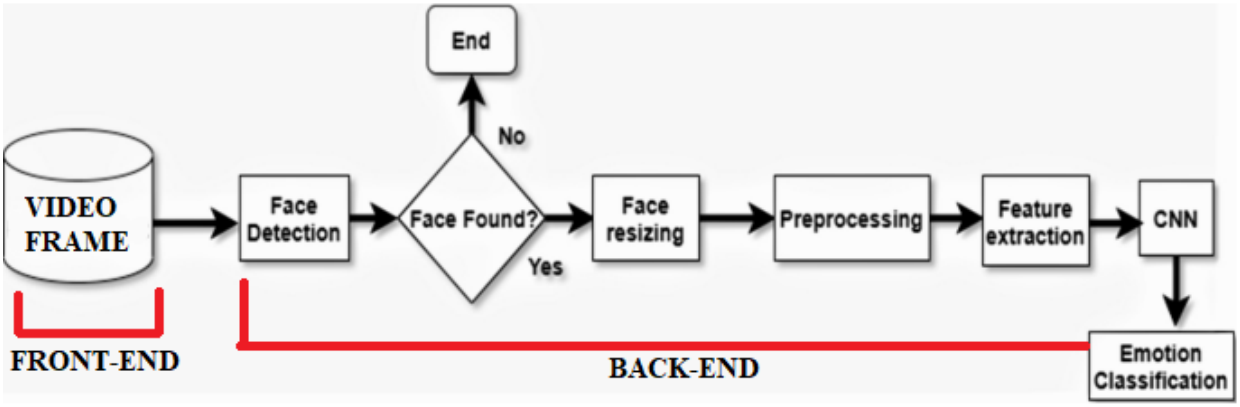

Figure 25. Steps Needed to Detect a Facial Expression in Real-Time

CONCLUSION

Overall, we used neural networks to solve a real and current problem that many companies and people face around the world which is Facial Expression Recognition (**FER**). Facial expression recognition is a fascinating and rapidly evolving field. Through my research, I have learned that it has numerous practical applications and I also learned how sophisticated and complex this

problem is. Moreover, I now have a greater appreciation of the availability of large datasets that have made it possible for me to train a model with a high accuracy score of **82.45%**. However, data is never enough and thus more data is required if we want to make our models more accurate.