Spring 2023

# Enhancing Facial Emotion Recognition Using Image Processing with CNN

Sourabh Deokar
*San Jose State University*

# Enhancing Facial Emotion Recognition Using Image Processing with CNN

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Sourabh Deokar

May 2023

The Designated Project Committee Approves the Project Titled

Enhancing Facial Emotion Recognition Using Image Processing with CNN

by

Sourabh Deokar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2023

| | |
|---|---|
| Dr. Nada Attar | Department of Computer Science |
| Dr. Wendy Lee | Department of Computer Science |
| Dr. William Anderopoulos | Department of Computer Science |

## ABSTRACT

Enhancing Facial Emotion Recognition Using Image Processing with CNN

by Sourabh Deokar

Facial expression recognition (FER) has been a challenging task in computer vision for decades. With recent advancements in deep learning, convolutional neural networks (CNNs) have shown promising results in this field. However, the accuracy of FER using CNNs heavily relies on the quality of the input images and the size of the dataset. Moreover, even in pictures of the same person with the same expression, brightness, backdrop, and stance might change. These variations are emphasized when comparing pictures of individuals with varying ethnic backgrounds and facial features, which makes it challenging for deep-learning models to classify. In this paper, we provide a simple yet efficient way for recognizing facial expressions that combines a CNN with certain image pre-processing techniques. We conducted our experiments on a combination of MUG, JAFFE, and CK+ datasets. To improve the performance of CNN, we experimented with various image pre-processing techniques such as face detection and cropping, image sharpening using Unsharp Mask, and normalization techniques like Global Contrast Normalization, Histogram Equalization, and Adaptive Histogram Equalization. Furthermore, we also examined data augmentation techniques such as image translations and adding noise to images to enhance performance of the deep learning model. Our custom CNN-based FER model achieved a maximum average accuracy of 93.3% (6 classes) and 91% (7 classes) after cross-validation. Our experimental results show that our proposed method can effectively enhance the accuracy of facial expression recognition.



**Keywords: Convolutional Neural Network, Data Augmentation, Deep Learning, Facial expression recognition, Image Processing, Normalization**

## ACKNOWLEDGMENTS

It is with immense gratitude and admiration that I express my sincerest appreciation to Dr. Nada Attar for her exceptional leadership, unwavering inspiration, and steadfast support throughout this academic journey. Her guidance has been invaluable, and I am deeply fortunate to have had the privilege of working with her. Her remarkable ability to impart knowledge and her passion for the subject matter have not only enriched my academic experience but have also inspired me to aim higher and work harder. I will always be grateful for her unwavering support and mentorship, which have been integral to my personal and academic growth. Thank you, Dr. Attar, for your invaluable contribution to this project.

I extend my heartfelt gratitude to my esteemed defense committee members, Dr. Wendy Lee and Dr. William Anderopoulos, for their unwavering support during the rigorous defense process. I am also immensely grateful to the exceptional faculty members of the Computer Science department, whose extensive knowledge and expertise have been instrumental in shaping my academic journey over the past two years.

## TABLE OF CONTENTS

**CHAPTER**

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

## Introduction

Emotions are a significant factor in human communication. Basic emotions include disgust, anger, happy, fear sadness, contempt and surprise [7]. Facial Expressions are used for the identification of an individual's emotion. It may be defined, in accordance with Li and Jain [8], as facial changes made in reaction to an individual's internal emotional state, objectives, or social communication. Numerous tasks, including security monitoring, multimodal human-computer interface (HCI), intelligent environments, lie detection, customer satisfaction identification, e-learning, emotion and paralinguistic communication, and smart card applications, benefit from facial expression recognition (FER) [9] [10] [11]. The three primary components of the conventional emotion recognition system, according to Tian et al. [11], are the identification of faces, the extraction of facial features, and the classifier construction. After the identification of faces, the facial features caused by facial changes need to be extracted. The most common traditional feature extraction methods include Principal component analysis [12], Bezier curves [13], Independent Component analysis [14], Local binary patterns [15], two-directional 2D Fisher principal component analysis [16], clustering techniques [17] and facial landmarks [2]. The extracted features need to maximize inter-class variance while minimizing the intra-class variation of expressions [15]. Because images of various people with the same expression are spread out in the pixel's space, it is difficult to minimize the intra-class variance of expressions. Also, since images of the same individual with various emotions may be quite close to one another in the pixel's space, maximizing the inter-class variation is similarly challenging [18].

Following feature extraction, classifiers such as Naive Bayes and Decision trees [19], Support vector machines [10], k-nearest neighbors [20], Hidden Markov models

[12], etc. are used to infer facial emotion. The disadvantage of the typical machine learning approach is that increasing system performance is difficult because of the separation between feature extraction and classification phases. Deep neural networks employ end-to-end learning, which combines the feature extraction and classification steps into a single phase to address the problems with traditional methods [21] [22] [23] [24] [25] [26]. The most effective algorithms for image classification problems are convolution neural networks (CNN). The automatic feature extraction offered by this CNN is one of its main advantages. However, the amount of data affects how well it performs. Facial expression dataset sizes are still limited for deep learning to be used. Moreover, images in highly controlled settings in the existing available FER datasets exhibit acted expressions rather than spontaneous ones. Consequently, an additional challenge associated with facial expression recognition is the model might not perform well if the training photos differ greatly from testing images in terms of the setting and subject ethnicity. Combining several datasets with potential subjects from various ethnic groups to train and test models is one way to assess facial expression recognition in these settings, which we follow in our study. Additionally, noisy and deviated images in the dataset also hamper the performance of CNNs. These drawbacks motivate the use of image pre-processing and data augmentation methods to enhance deep learning performance.

By combining image pre-processing techniques like region of interest selection, image sharpening, normalizations, and data augmentation by synthetic training-sample generation with deep learning, we attempt to address the aforementioned limitations in our study while maintaining a straightforward solution. By merging three separate datasets (CK+, JAFFEE, and MUG) [27] [1] [28] and conducting cross-validation, we carry out a thorough validation. The four primary contributions of this paper are, in brief: *a real-time, simple, and effective method for recognizing*

*facial expressions; a research on how image pre-processing techniques influence face expression recognition; a series of specific pre-processing steps that help deal with the lack of data and minimize the requirement for controlled environments; a study of the effectiveness of a FER system trained on subjects from various racial and cultural backgrounds and samples from varying settings (multi-database evaluation).*

# CHAPTER 2

## Project Roadmap

We observed how pre-processing methods including the region of interest extraction, global contrast normalization, histogram equalization, and data augmentation by introducing salt and pepper noise enhanced the performance of CNN in earlier research by Pitloka et al [23]. Pitloka et al. findings serve as a motivation for our research. The objective is to evaluate other image pre-processing techniques, such as image sharpening using Unsharp Mask and normalization using Adaptive Histogram Equalization along with the techniques from [23]. Sharpening an image improves the borders of important features like the lips and eyes, which are crucial for identifying emotion. We'll try out a different noise addition technique called Gaussian noise to generate synthetic images. In contrast to [23], where the generated images were present in both training and test datasets, causing data overlap, we will only generate the synthetic images for the training the model. Additionally, image translations like rotation, zoom, height shift, width shift, shear, and flip will be employed for data augmentation. Unlike [23], cross-validation will be used for each model to produce precise results. Finally, we will establish the ideal pre-processing pipeline for enhancing the performance of FER models.

# CHAPTER 3

## Related Work

Over the past few decades, several facial expression recognition (FER) techniques have been developed, with growing performance [19] [26]. Both machine learning and deep learning methods have been used to build FER approaches. SVM and Logistic Regression models were used by Raut [2] to classify the face images in the CK+ dataset into seven categories: contempt, anger, happiness, sadness, disgust, fear, and surprise. The 68 facial landmark detector, which provides the positions of 68 landmarks on the face, was used to extract the features from the face and create the feature vectors. For the Logistic Regression model and the Linear SVM model, this method's accuracy was 88.2% and 83.23%, respectively. Applying image pre-processing and Recursive Feature Elimination to eliminate any low-weighted features has the potential to increase accuracy. On the CK+ dataset, Dewi et al [29], suggested an Active Appearance Model (AAM) and a semi-supervised Fuzzy C-Means for facial expression recognition. AAM is a feature extraction method that uses template matching and is applied during training. 68 spots were used to analyze the shape of the face. Their model's accuracy was 80.71%. Salmam et al. [19] used a decision tree to create a FER system with seven potential classes on two datasets: COHEN and JAFEE. In order to assess the portions of the face that best depict a facial expression, this model uses the six distances that were previously computed (using the Minkowski, Manhattan, or Euclidian distance) for each face as input to the classifier. In the JAFFE and COHEN datasets, the accuracies achieved were 89.20% and 90.61%, respectively. Other studies have employed machine learning algorithms like Hidden Markov model [30] and k-nearest neighbors [20] for facial expression recognition. Since the processes of feature extraction and classification are isolated in the traditional machine learning approaches, it is challenging to improve

the model performance.

To overcome the issues with conventional machine learning techniques, deep learning networks integrate feature extraction and classification procedures into a single phase. CNN is the most commonly used deep learning technique for image classification. Although CNNs function effectively on their own, preprocessing images and using them as input rather than supplying the CNN with raw input images has demonstrated a considerable boost in accuracy [23]. The earlier versions of CNN architectures were less resistant to deviated and inconsistent facial images because they did not use image data augmentation and preprocessing approaches [31] [32]. A CNN model was created by Wang et al. [21] and trained on the FER-2013 dataset [33] to classify a collection of static images into 7 fundamental emotions. The outcome of the softmax activation function was stacked using SVM. In order to improve performance, they also employed data augmentation and histogram equalization as pre-processing techniques. Their findings show that preprocessing techniques enhance the CNN model's accuracy. Based on earlier research [21], Vepuri [24] was able to show that, although using a relatively simple CNN model, performance can be improved by using a sharpening technique (Unsharp Mask) to preprocess images for a FER model. For the FER13 dataset, they measured a 2% improvement in accuracy following the use of Unsharp Mask. In order to improve FER, Lopes et al. [22] investigated the effects of data pre-processing prior to network training. Before CNN, processes such as data augmentation, cropping, rotation correction, downsampling, and intensity normalization were used. The CNN had two layers of convolution-pooling and two layers of fully linked layers with 256 and 7 neurons, respectively. Three databases CK+, JAFFE, and BU-3DF [34] were used to analyze this approach. According to the results, applying each of these pre-processing processes together was more efficient in improving the model performance than doing it separately.

Influenced by the research in [22], Pitloka et al. [23] improved their 2-layer CNN by implementing data pre-processing techniques such as face detection and cropping, resizing, data normalization (global contrast normalization, local normalization, and histogram equalization), and noise addition (for data augmentation). They used a combination of JAFFE, CK+, and MUG datasets to train the CNN. When compared to other pre-processing steps face detection and cropping as a single pre-processing phase produced a substantial increase in the model accuracy from 62.35% to 87.06% accuracy. The histogram equalization step improves accuracy more than other normalizing methods, although not as much as face detection and cropping. The CNN was able to attain 97.06% accuracy by adding salt and pepper noise to images and augmenting the noisy images to the original dataset. Nevertheless, since there are two versions of the data—the original and noisy data—the likelihood of data overlap between the training set and the testing set is significant. We believe that this is not a fair evaluation of the FER model and that an accurate analysis of the FER model may be attained by only utilizing the training dataset for data augmentation. Without data augmentation by adding noise Pitloka et al. obtained a max accuracy of 90.59% using face detection and histogram equalization as the pre-processing techniques. This study, as well as several other works, did not include any cross-validation procedures. As a result, the accuracy of the results obtained remains unclear. The absence of cross-validation in these studies raises concerns about the reliability and generalizability of their findings. In the absence of cross-validation, it is important to interpret the findings of these studies with caution and to consider the potential limitations of their methodologies.

Borgalli and Surve [25], in their work, implemented 10-fold cross-validation on CNN-based FER models, which were trained separately on FER13, CK+, and JAFFE datasets. The accuracy rates for the FER2013, CK+, and JAFFE datasets on the seven types of emotions— happiness, anger, disgust, sadness, fear, contempt, and

surprise—were 86.78%, 92.27%, and 91.58%, respectively. Some other works on FER used transfer learning to tackle the problem of limited datasets. For example, Chowdary, Nguyen, and Hemanth [26] use pre-trained CNNs of MobileNet, VGG19, Inception V3, and Resnet50 to recognize emotions in facial images from the CK+ database. New fully connected layers were introduced that could only be trained to update the weights, removing the existing fully connected layers of the pre-trained models. On the CK+ database, their experiment's accuracy was on average 96%. Most of the studied works in our literature review have either trained models on a single dataset or independently on multiple datasets, but very few like Pitloka et al. [23] have performed a multi-database evaluation. Training models on a single dataset can limit their generalizability to other settings. For example, if the training images are vastly different from the testing images in terms of environment setting or subject ethnicity, the FER model may not perform as well. To address this issue, in our study, we combine several datasets and include potential subjects from various ethnic groups to train and test our FER model. This allows us to evaluate the model's performance in a more realistic and diverse setting.

# CHAPTER 4

## Dataset

To include participants from different ethnic groups and samples with varied environmental conditions, we combine the three datasets below, similar to [23].

### 4.1 CK+

In computer vision and affective computing, the Extended Cohn-Kanade Dataset (CK+) is a facial expression dataset that is often utilized. It was created by including more people and a larger variety of facial expressions in the original Cohn-Kanade dataset. The 593 image sequences in the CK+ dataset, which includes 123 participants, each depict a participant's expression as they carry out a particular job. The dataset includes neutral expressions in addition to the six fundamental emotions of anger, disgust, sadness, fear, happiness, and surprise. The facial muscle movements that underpin face emotions, known as action units (AUs), are labeled for the facial expressions in the CK+ dataset. 35 AUs are annotated in the dataset, enabling researchers to investigate the connection between particular AUs and various emotional expressions. In this investigation, two Panasonic AG-7500 cameras were used to record the facial expressions of 210 adults. Subjects were aged 18 to 50, 69% female, 81% European American, 13% African American, and 6% from other groups. Participants' ages ranged from 18 to 50. Both frontal and 30-degree perspectives were used to record the facial expressions, which were then digitally converted into either 640x480 or 640x490 images of either 24-bit color values or 8-bit grayscale. The dataset offers a thorough collection of facial expressions for a wide range of people, making it an important tool for research into facial expression and emotion identification.

Figure 1: Sample images from CK+ dataset [1]

### 4.2 MUG

The Multimedia Understanding Group developed the MUG Facial Expression Database to address shortcomings of earlier comparable datasets and support research in the area of expression recognition. The collection includes image sequences of 86 subjects, 51 males and 35 women, aged 20 to 35, who are of Caucasian descent. Two 300W light sources were put on supports to distribute the light and prevent shadows while the subjects were being recorded in front of a camera with a blue screen background. The camera took pictures at a rate of 19 frames per second, saving each one as a jpg file with a resolution of 896 x 896 pixels and a size of between 240 and 340 KB. The database is divided into two sections, the first of which contains the seven fundamental emotions—anger, disgust, sadness, fear, happiness, contempt, and surprise—and the second of which comprises emotions that have been artificially created in a lab. The sequences follow the onset, apex, offset temporal pattern, are categorically labeled, and begin and conclude at a neutral state. A brief picture sequence illustrating the neutral condition was captured for each subject. A total of 1462 image sequences that were chosen as the best were made available. Also, other photos had landmark facial points manually and automatically labeled. The second section of the database includes approximately 1000 photos in one image sequence for

each subject that was taken at 19 frames per second with a resolution of 896 x 896 with the intention of eliciting natural reactions in a lab setting. These sequences have not yet been tagged.

Figure 2: Sample images from MUG dataset

### 4.3 JAFFE

A collection of facial expressions of Japanese women called the Japanese Female Facial Expression (JAFFE) Dataset has been extensively used in studies on facial recognition and emotion recognition. The collection consists of 213 photos representing each of the seven face expressions: neutral, fear, happiness, sadness, anger, disgust, and surprise. Images of Japanese women of various ages, without makeup, and with their hair pinned back to reveal their faces were taken with a digital camera under controlled lighting settings. Ten distinct people each play a different expression. The JAFFE dataset has been used to train and test emotion recognition models, facial expression recognition algorithms, and other computer vision and artificial intelligence applications. Computer vision and emotion recognition researchers have benefited greatly from the JAFFE dataset. It has helped to improve the reliability and accuracy of face expression detection systems. The JAFFE dataset has also been used by

researchers to examine cultural variations in facial expression, revealing that Japanese people typically exhibit softer expressions than Westerners. The JAFFE dataset does, however, have certain drawbacks, including the small number of people represented and the uniformity of facial characteristics and head postures. The JAFFE dataset continues to be a valuable tool for researchers in the area despite these drawbacks and has paved the path for additional developments in facial recognition and emotion detection technology.

Figure 3: Sample images from JAFFE dataset

Table 1: Comparison between CK+, JAFFE, and MUG datasets

| | MUG | CK+ | JAFFE |
|---|---|---|---|
| # of emotions | 7 Emotions | 7 Emotions | 7 Emotions |
| # of subjects | 86 (35 females & 51 males) | 123 (male and female) | 10 females |
| # of samples | 328 | 593 (327 labeled, 266 unlabeled) | 213 |
| Ethnicities of subjects | Caucasian | 13% African American, 81% European American, and 6% from other groups | Japanese |
| Resolution | 896x896 | 640x490 or 640x480 | 256x256 |
| Format | .jpg | .png | .tiff |

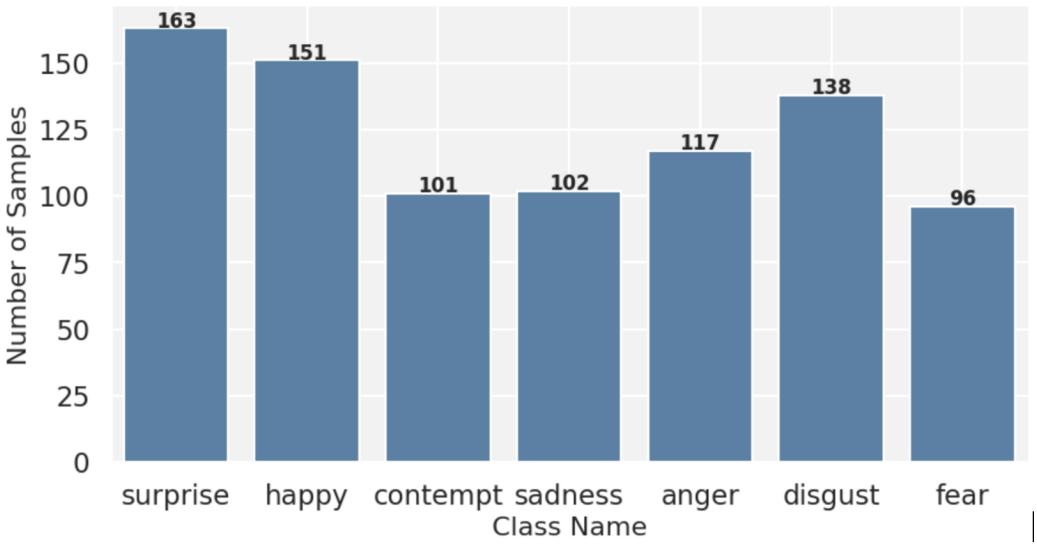


Figure 4: Slass distribution of samples from all 3 datasets

# CHAPTER 5

## Facial Image Pre-Processing

The performance of CNNs can be impacted by a variety of elements, including limited datasets, crowded backgrounds, lighting, and posture deviation. The use of the following preprocessing filters may enhance the classification accuracy of facial expressions.

### 5.1 Face detection and cropping

Face detection and cropping is a crucial image preprocessing technique that involves detecting the face region in the image and extracting it for further processing. In this study, we used the Haar Cascade Classifier in the OpenCV library to detect faces in the input image. The classifier is trained to recognize the patterns of facial features such as eyes, nose, and mouth in an image. Once the faces are detected, we cropped the image to only include the face region. The implementation involved setting the scaleFactor which governs the amount of image size that is shrunk at each image scale, and the minNeighbors parameter, which controls the number of rectangles that are retained after the detection [35]. After face detection, we extracted the region of interest (ROI) by using the (x, y) coordinates of the detected face and its width and height (w, h). This ROI can then be further preprocessed with other techniques such as normalization or histogram equalization to enhance the facial features and improve the accuracy of the facial emotion recognition model.

Figure 5: Face detection and region of interest selection

### 5.2 Unsharp Mask

In this study, one of the image preprocessing techniques used to improve facial emotion recognition is the "Unsharp Mask." The Unsharp Mask is a sharpening filter that enhances edges and fine details in an image. It works by subtracting a blurred version of the image from the original image, which enhances the contrast and details of the edges in the image. The Unsharp Mask is particularly useful for improving the clarity of low-resolution images or images with low contrast. We implemented the Unsharp Mask using the OpenCV library in Python. Specifically, we used the cv2.GaussianBlur() function to apply a Gaussian blur to the input image before subtracting it from the original image [36]. The size of the Gaussian blur kernel was set to (0, 0) to automatically calculate the kernel size based on the input image size. We also set the standard deviation of the Gaussian kernel to 1.0 to control the amount of blur applied to the image. After applying the Gaussian blur, we used the cv2.addWeighted() function to subtract the blurred image from the original image and add the result to the original image [33]. The parameters of the function were set to add twice the original image to the blurred image with a weight of -1.0. This resulted in an output image with enhanced edges and fine details that were better suited for facial expression recognition using a convolutional neural network.

### 5.3 Normalization

Normalization is a technique that scales the pixel values of an image to a fixed range to remove the effects of different lighting conditions and contrast levels in the input images. This technique is particularly useful for facial expression recognition as it helps to remove variations in image brightness and contrast, which can affect the performance of the model. We experimented with the following three normalization techniques.

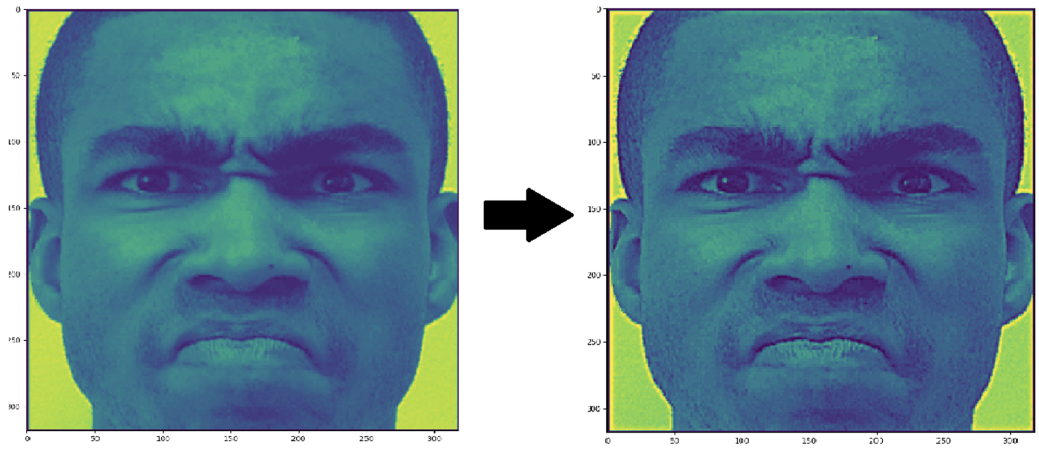

Figure 6: Applying unsharp mask on a sample image

### 5.3.1 Global Contrast Normalization

Global Contrast Normalization (GCN) is a technique that normalizes the pixel values of an image to have zero mean and unit variance to remove the effects of variations in lighting and contrast in the input images. In this study, we implemented the GCN technique by subtracting the mean intensity of the image from each pixel and dividing the result by the standard deviation of the image intensities. This helps to normalize the image contrast across different images and makes it easier for the CNN model to learn meaningful features.

### 5.3.2 Histogram Equalization

Histogram Equalization (HE) is a commonly used technique in image processing for enhancing the contrast and brightness of images. The technique works by redistributing the intensity values of an image's histogram to improve the overall contrast and brightness. In this process, the intensity values are stretched over a wider range, which results in a more balanced distribution of pixel intensities. Histogram Equalization is particularly useful for images with low contrast, where details may be difficult to distinguish. In our implementation, we used the OpenCV library's cv2.equalizeHist() function to apply the Histogram Equalization technique to the input image [37]. The

function takes the input image as an argument and returns the equalized image with improved contrast and brightness. By enhancing the contrast and brightness of the input images using Histogram Equalization, we were able to improve the accuracy of the Facial Emotion Recognition CNN model. However, it is important to note that Histogram Equalization may also amplify the noise present in an image, which can negatively affect the performance of the model.

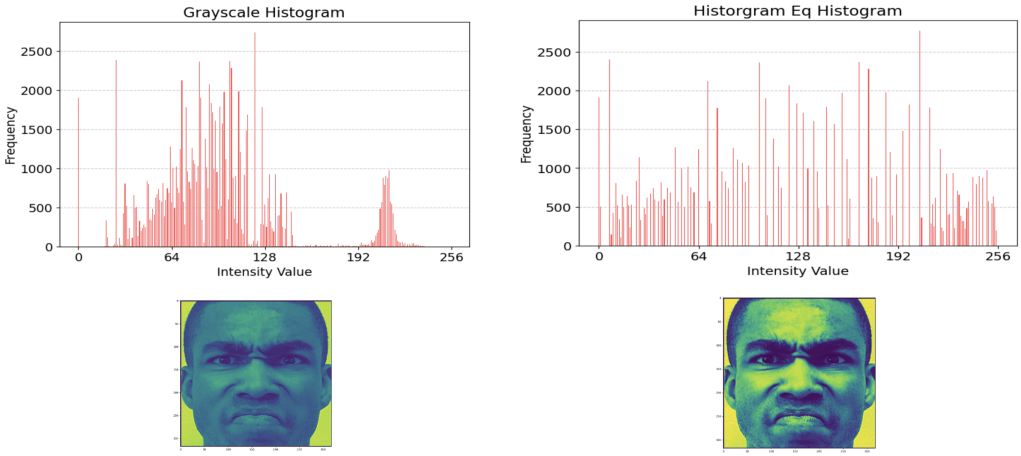


Figure 7: Histogram before and after histogram equalization

### 5.4 Adaptive Histogram Equalization

Adaptive Histogram Equalization (AHE) is a variation of the Histogram Equalization technique that is designed to address some of the limitations of traditional Histogram Equalization. Unlike traditional Histogram Equalization, which applies a global transformation to the entire image, AHE applies the transformation locally to small regions of the image. This allows the technique to better handle images with varying levels of contrast and brightness across different regions. Specifically, AHE divides the image into small tiles and applies Histogram Equalization separately to each tile, using the cumulative distribution function of the pixel intensity values within that tile. This local adaptation prevents the over-enhancement of noise and artifacts seen with Histogram Equalization, while still improving the contrast of the image. In our

implementation of AHE we used the OpenCV library's cv2.createCLAHE() function to create a Contrast Limited Adaptive Histogram Equalization object [38]. We set the clipLimit parameter to 2.0 and the tileGridSize parameter to (8,8) to control the level of contrast enhancement and the size of the local regions, respectively. We then applied this object to the input image using the apply() function, which returns the enhanced image, with the transformation applied locally to different regions of the image.

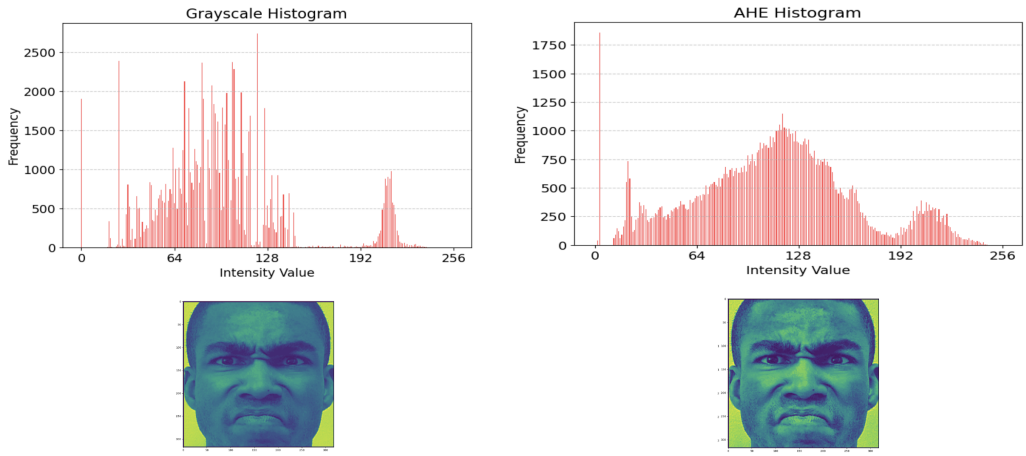


Figure 8: Histogram before and after adaptive histogram equalization

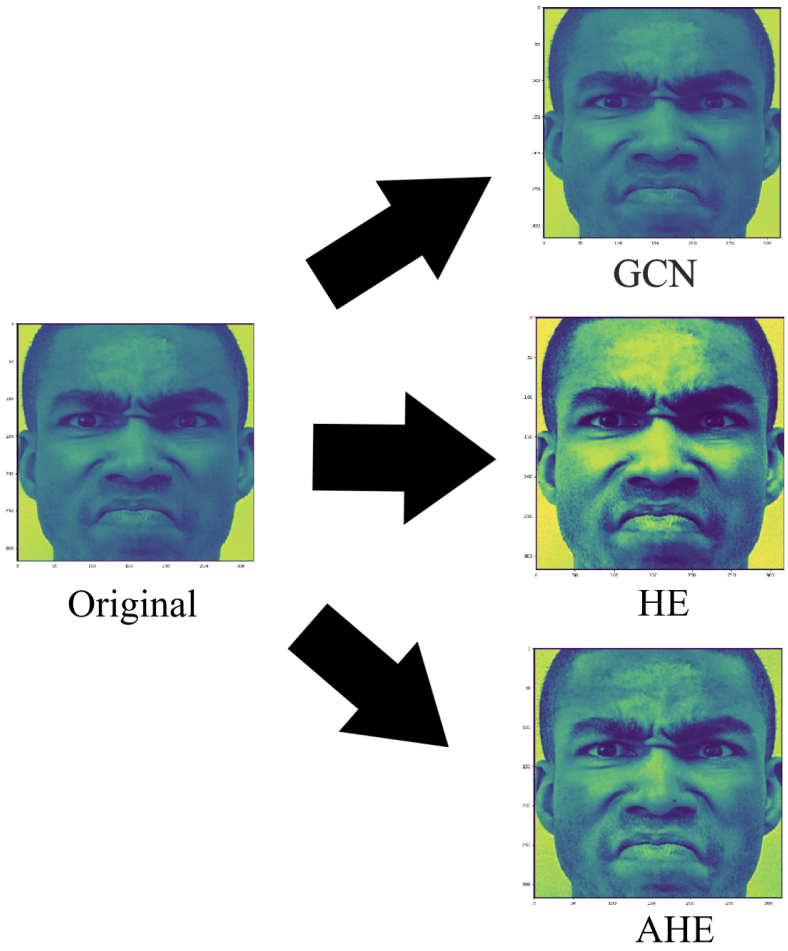


Figure 9: Applying image normalizations on a sample image

### 5.5 Image Augmentation/ Synthetic Image Generation

#### 5.5.1 Image Augmentation by image translations

Image augmentation is a powerful technique used to increase the diversity and amount of data available for training deep learning models. One way to augment images is by performing image translations. This involves randomly shifting an image in the horizontal and/or vertical direction by a certain number of pixels. In this project, we have implemented image augmentation by image translations using the ImageDataGenerator class from the Keras library [39]. This class allows us to generate new images by applying various transformations to the original images. The

transformations applied in this project include rotation, zoom, width and height shift, shear, and horizontal flip. The datagen object created by the ImageDataGenerator class is then used to generate augmented images during the training process. We also calculated class weights based on the integer labels to balance the distribution of samples across different classes. The class_weights parameter in the model.fit() function is used to assign higher weights to underrepresented classes, helping to address class imbalance issues. The use of image augmentation techniques such as translations can help to improve the performance of deep neural networks, especially when working with limited amounts of training data.

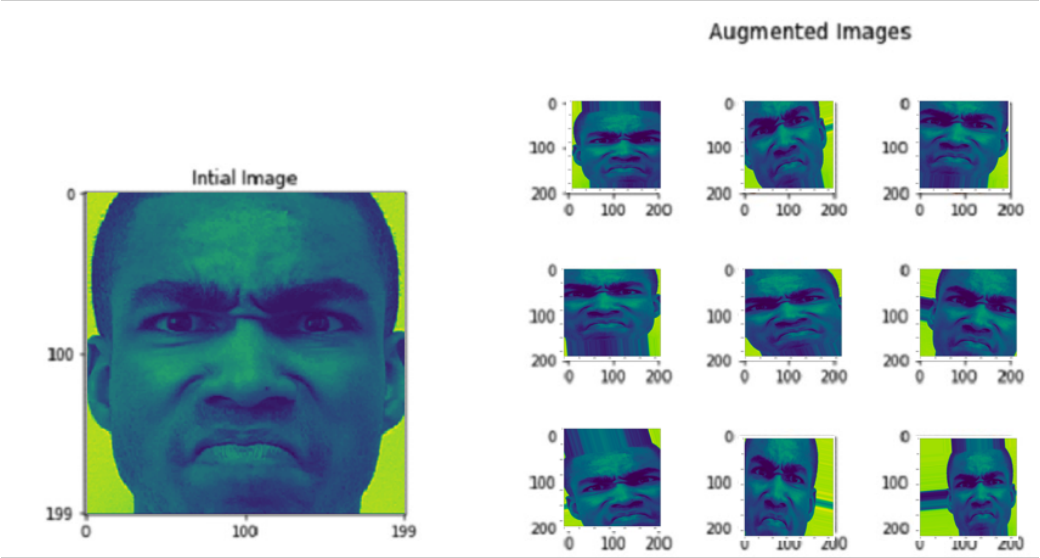


Figure 10: Data augmentation

### 5.5.2 Image Augmentation by adding noise

Another way to achieve data augmentation is to generate synthetic images by adding noise to the original dataset. In our study, we have implemented this technique by adding Gaussian noise to the facial images in the dataset. The Gaussian noise added to the images is random and helps to increase the variability of the data, making the neural network more robust to variations in the input. By adding these

synthetic images to the original dataset during training, we were able to significantly increase the number of samples of our facial emotion recognition model. In our implementation, we generate random Gaussian noise with a normal distribution using 'numpy.random.normal(mean, std, img.shape)'. The noise generated will have the same dimensions as the input image, and its values will be randomly distributed around the mean with a standard deviation of 0.5. We then calculated class weights based on the training data to ensure that the synthetic images were added in a balanced way. We then selected a subset of the images and labels for each expression based on the class weight and generated the required number of images with noise. We randomly selected images from the dataset and added Gaussian noise to generate synthetic images. However, we have taken a different approach from [23] by only utilizing the generated images for training the model, rather than including them in both the training and testing datasets. We believe this will prevent data overlap between the two sets and provide a more accurate evaluation of the FER model.

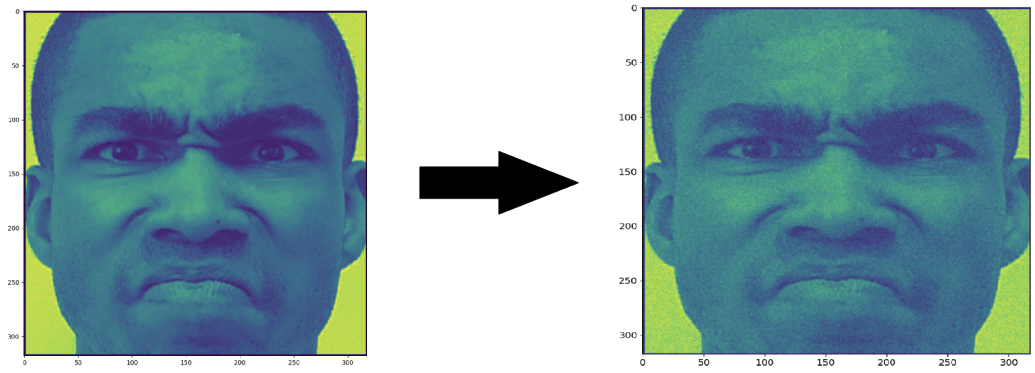

Figure 11: Applying gaussian noise on a sample image

# CHAPTER 6

# Facial Emotion Recognition

## 6.1 FER using Machine Learning

In the conventional Machine Learning process for FER, region of interest selection is done first by detecting the face in the input image. Haar Cascade Classifier is one of the commonly used techniques for face detection. Following face recognition, it's necessary to extract the facial characteristics brought on by facial expressions. Traditional feature extraction approaches most frequently used include local binary patterns, clustering algorithms, and face landmarks. Face landmarks are discovered within the bounding box using feature detection techniques, as shown in the figure below.


Figure 12: Face landmarks [2]

The emotion class is then identified using these features as input to classification

models like KNN, HMM, and SVM, as illustrated in Figure 13.

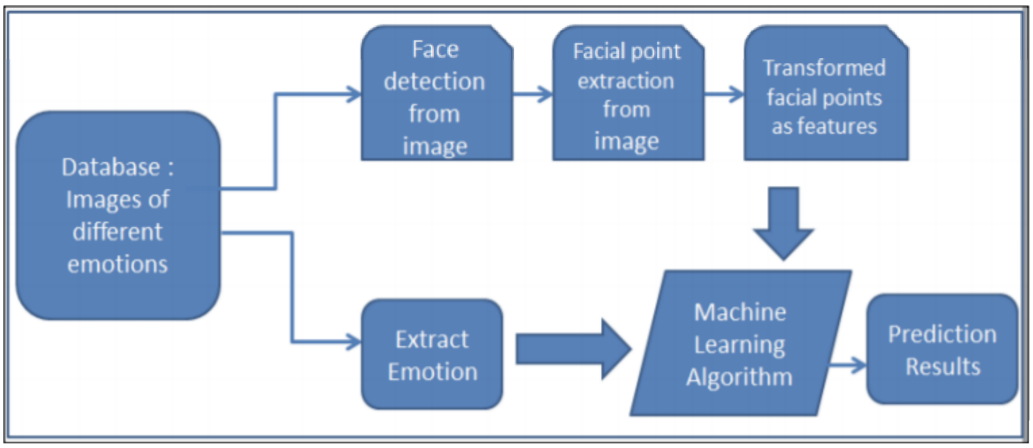


Figure 13: Machine Learning based FER approach [2]

### 6.2 FER using Deep Learning

#### 6.2.1 FER using CNN

Facial features are automatically extracted by CNN, as opposed to traditional machine learning, where facial features need to be manually extracted from an input image to complete emotion categorization. End-to-end learning, used by deep learning networks, condenses the feature extraction and classification processes into a single stage. CNN is one such variant of neural networks that can collect spatial data and gather valuable characteristics from an image. A typical CNN architecture for FER consists of several layers including convolutional layers, dropout layers, pooling layers, and fully connected layers.

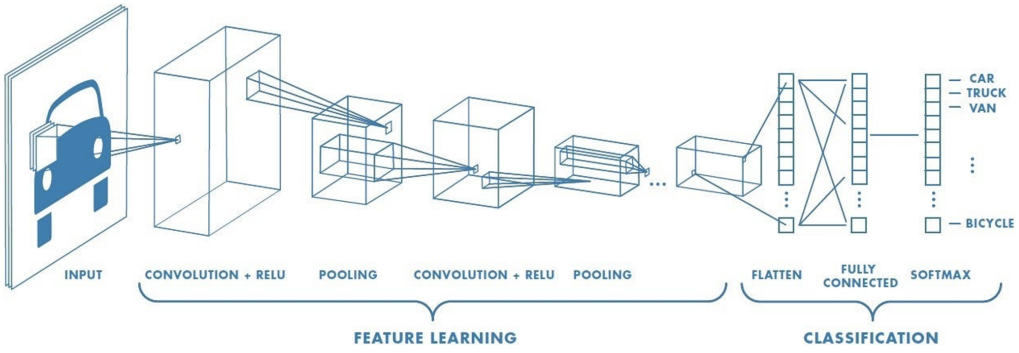


Figure 14: CNN architecture [3]

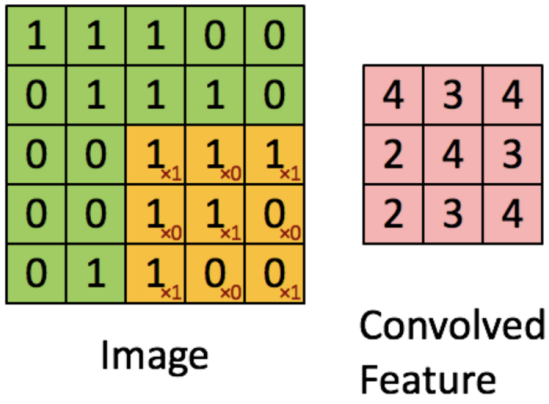


Figure 15: 3x3 kernel operation on 5x5 image [3]

#### 6.2.1.1 Convolutional Layer

A kernel or filter is applied to the input picture with some stride in a convolutional layer to produce the convolved feature. A kernel is essentially a collection of weights that the network backpropagates using the loss function after they are originally allocated randomly. A dot product between the kernel weights and the region being operated on is computed when the kernel is applied to an input image. The output is a feature map that results from a dot product operation applied to the input image as the kernel moves across it.

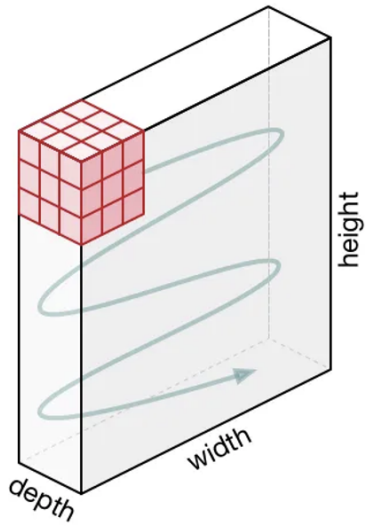


Figure 16: Kernel movement [3]

The kernel is applied at the same depth as the input. For instance, the depth of the kernel for a grayscale picture is 1, but the depth of the kernel for an RGB (Red, Green, and Blue) color image has three channels. As a result, three distinct channels would each get a different kernel. To get the feature output, the dot products of the three channels are added together.

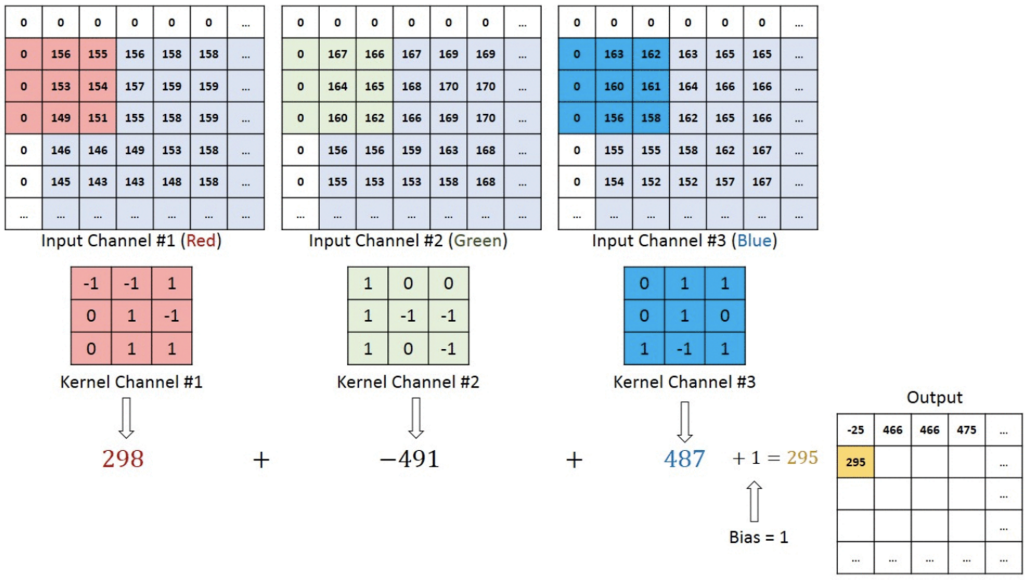


Figure 17: 3 kernels applied on RGB channels [3]

A CNN may have many convolutional layers. Color, edge, and gradient direction are just a few examples of the low-level properties that the first convolutional layers of an image extract. The more complicated high-level information, such as objects, and shapes, is extracted by the deeper convolutional layers. Convolutions also allow us to select the kind of padding to use. Padding is a technique used to maintain the spatial dimensions of the input image after convolution. By adding zeros around the image, the spatial dimensions can be preserved. Padding also helps to increase the receptive field of the convolutional layer. The "same padding" operation, which pads the image's edges with zeroes, might be used if we do not want to diminish the input's size. On the other hand, "valid padding" might be used if we do not want to decrease the input's dimension.

#### 6.2.1.2 Pooling layer

After the convolutional layer, a pooling layer is used to minimize dimensionality and collect the most important data. Max Pooling and Average Pooling are the two

most often employed forms of pooling. Max pooling only chooses the region's highest value when the kernel is applied to it. The average of all the data in the area where the kernel is applied is calculated using average pooling. Max pooling is widely used because it reduces dimensions while simultaneously removing noise and discarding less important data. Figure 18 displays the outcome of applying a 2x2 average pooling and 2x2 max pooling with a 2 stride to an input feature map.

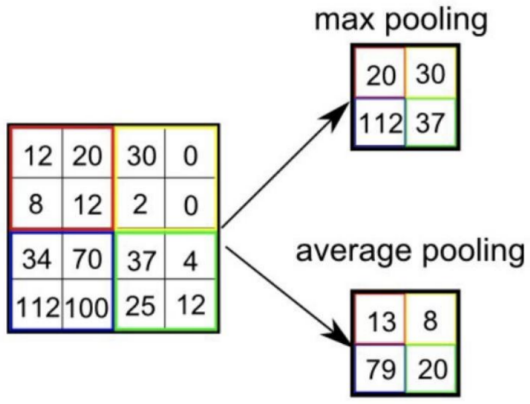


Figure 18: Pooling types [3]

#### 6.2.1.3 Dropout layer

Dropout is a regularization technique that helps to prevent overfitting. It randomly drops out units (neurons) in the layer during training. As a result, the network is forced to learn redundant representations and becomes more noise-resistant. Each convolutional or fully connected layer in a CNN can have dropout added after it, and the dropout rate controls the proportion of neurons that will be randomly removed during training. Dropout is useful in enhancing the generalization performance of the network, particularly in tasks with little training data, even though it can lengthen the training time of a CNN.

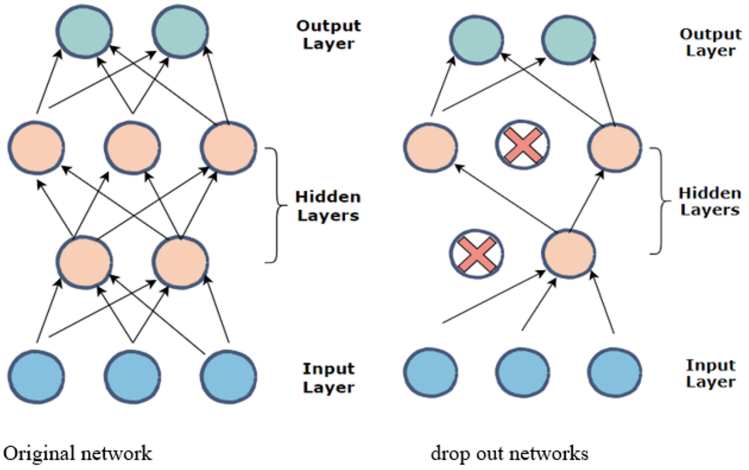


Figure 19: Dropout layers [4]

#### 6.2.1.4 Fully Connected layer

These layers are in charge of applying a weight matrix to the output of the preceding layer to create a fresh set of activations. Each neuron in a fully connected layer is linked to every neuron in the layer below, enabling them to learn intricate non-linear connections. To enable the network to learn more complex information and make predictions, these layers are often added near the conclusion of a CNN. Fully connected layers may significantly increase the amount of trainable parameters in a network, which makes them vulnerable to overfitting. Techniques like dropout, weight decay, and early halting are frequently used on fully linked layers to avoid overfitting. We commonly utilize a Softmax layer immediately following the fully connected layers to conduct classification. In this study, we implemented a custom CNN model to classify facial images into six different expressions. Further details on the architecture and the implementation of our model are provided in next chapter.

### 6.2.2 FER using Transfer Learning

A pre-trained model is utilized as the foundation for another task when using the deep learning approach known as transfer learning. Usually trained on a big

dataset, the pre-trained model has acquired specific features that may be used for a new task. The idea behind transfer learning is to use the knowledge learned by the pre-trained model to improve the performance of the new task, especially when the new dataset is small or lacks diversity. Transfer learning has been widely used in FER, and it has shown promising results in improving the performance of FER models. Feature extraction and fine-tuning are the two basic methods of applying transfer learning in FER. In the feature extraction approach, the pre-trained model is used to extract features from the images, and then a classifier is trained on these features to predict the emotions. Typically, deep CNNs that have been pre-trained on huge image datasets like ImageNet are the pre-trained models employed in FER. The features extracted by these pre-trained models are effective in FER tasks, even when the FER dataset is small. In the fine-tuning approach, the weights of the model are adjusted on the fresh dataset using the pre-trained model as a starting point. This approach is especially useful when the pre-trained model is similar to the new task. For example, a pre-trained model on a large face recognition dataset can be fine-tuned on a smaller FER dataset to improve the performance of the FER model. Several commonly used transfer learning models are effective in a variety of computer vision tasks, including FER [26]. Some of the most popular transfer learning models include VGG16, ResNet50, FaceNet, and SeNet50.

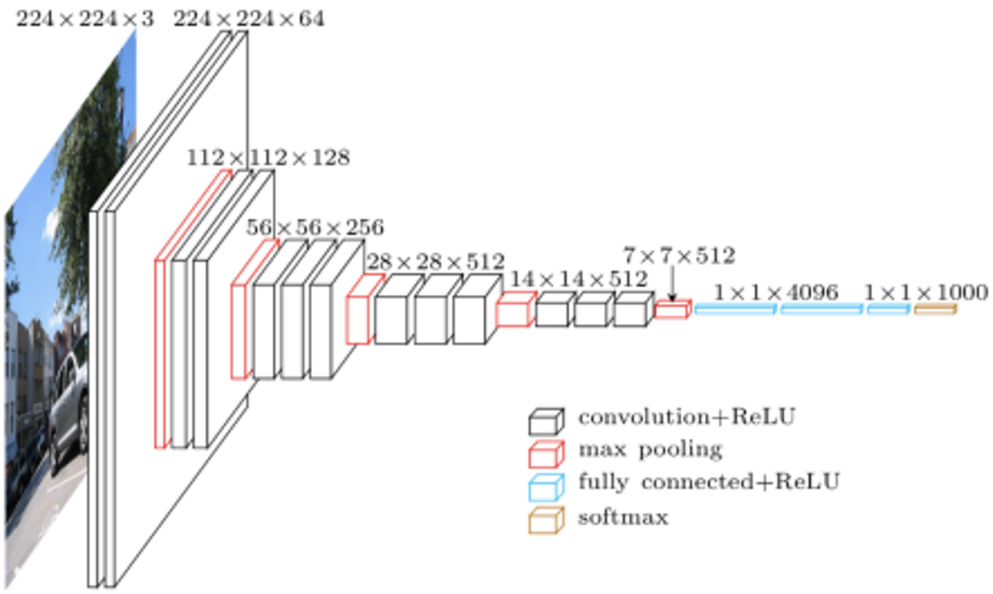


Figure 20: VGG architecture[5]

# CHAPTER 7

## Proposed Models

We implemented a 3-layer CNN, details of which are explained in the following section. Our CNN architecture was employed to create 15 different facial emotion recognition (FER) models with diverse image pre-processing techniques. Similar to [23], we initially trained and validated our models on 6 classes (surprise, disgust, fear, anger, sadness, and happiness) comprising 767 samples. Prior to the application of any image processing or training, the images were first converted to grayscale and resized to 64x64 pixels. A test train split of 90:10 was utilized for all models, and to ensure a fair comparison, we maintained the same random_state while performing the split. Furthermore, to obtain more reliable outcomes, we employed a 5-fold cross-validation for all the models. After identifying the pre-processing pipeline that yielded the best FER model, we trained and tested it using 868 samples from 7 classes, namely surprise, disgust, fear, anger, sadness, happiness, and contempt.

### 7.1 Custom 3-layer CNN

This is a simple 3-layer CNN model built using the Keras framework in Python. The model consists of three convolutional layers, two fully connected layers and one dropout layer. The input to this model is a grayscale image with dimensions of 64x64 pixels. The model consists of several layers that extract features from the input image.

The first layer is a convolutional layer with 6 filters of size 5x5, which means that each filter scans a patch of 5x5 pixels across the input image and produces a new feature map. The 'padding' argument is set to 'same', which means that the output feature map will have the same dimensions as the input image. The activation function used in this layer is ReLU, which introduces nonlinearity to the model and helps to improve its performance. After that, a max pooling layer with a pool size of 2x2 is applied to the convolutional layer's output. By taking the largest value within

each 2x2 block, this layer shrinks the feature map's spatial dimensions, reducing overfitting and speeding up computation.

The second and third layers are similar to the first layer, but with more filters and smaller kernel sizes. The second convolutional layer has 16 filters of size 5x5, followed by a ReLU activation function and a max pooling layer. The third convolutional layer has 64 filters of size 3x3, again followed by a ReLU activation function and a max pooling layer. The output of the third max pooling layer is a feature vector that is flattened into a one-dimensional array using the 'Flatten' layer. This vector is then passed through two fully connected layers. The first fully connected layer has 128 units and uses the ReLU activation function, which helps to introduce nonlinearity and capture complex relationships between the features. The dropout layer is added to prevent overfitting by randomly dropping out 50% of the neurons in the layer during training. The second fully connected layer utilizes the softmax activation function, which generates a probability distribution over the 6 or 7 classes, and has 6 or 7 neurons, depending on the number of classes in the dataset.

The categorical cross-entropy loss function, which calculates the discrepancy between the predicted and actual class probabilities, is used to train the model. This loss function is minimized using the Adam optimizer, with a learning rate of 0.001. The accuracy metric, which measures the percentage of correctly categorized images, is used to assess the model's performance during training and validation. In total, the model has 414,291 trainable parameters.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 64, 64, 6)         156

max_pooling2d_3 (MaxPooling  (None, 32, 32, 6)         0
2D)

conv2d_4 (Conv2D)            (None, 32, 32, 16)        2416

activation_1 (Activation)    (None, 32, 32, 16)        0

max_pooling2d_4 (MaxPooling  (None, 16, 16, 16)        0
2D)

conv2d_5 (Conv2D)            (None, 14, 14, 64)        9280

max_pooling2d_5 (MaxPooling  (None, 7, 7, 64)          0
2D)

flatten_1 (Flatten)          (None, 3136)              0

dense_2 (Dense)              (None, 128)               401536

dropout_1 (Dropout)          (None, 128)               0

dense_3 (Dense)              (None, 7)                 903

=================================================================
Total params: 414,291
Trainable params: 414,291
Non-trainable params: 0
```

Figure 21: CNN architecture

### 7.2 K-fold Cross Validation

We implemented 5-fold cross-validation for evaluating the performance of all FER models. The KFold function from the scikit-learn library is defined with 'n_splits' set to 5, indicating that the dataset is divided into 5 equally sized folds. During each iteration, one fold is used for testing, and the remaining four folds are used for training the CNN model. The history of the training is recorded, and the maximum validation accuracy and corresponding loss are recorded for each fold. Finally, the mean of the losses and accuracies are calculated as the final evaluation metric for the model. This technique is useful for obtaining a more reliable estimate of the CNN's performance

on unseen data by evaluating the model on multiple, non-overlapping subsets of the data.

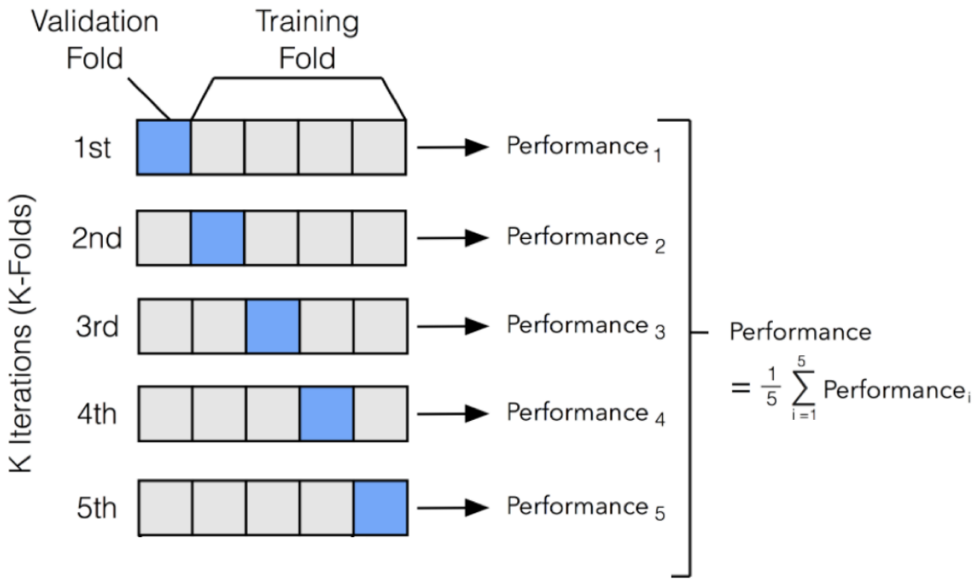


Figure 22: K-fold cross validation [6]

### 7.3 CNN without Image Pre-processing

Initially, we trained the CNN on the grayscale 64x64 input images without applying any image pre-processing. The model was trained for 50 epochs and this resulted in a maximum accuracy of 68.8% and a corresponding validation loss of 1.62. Figure 23 depicts the validation/training accuracy and loss of this CNN. Average precision and recall were 0.68 and 0.65 respectively. On performing 5-fold cross-validation the mean accuracy observed was 66.2%.

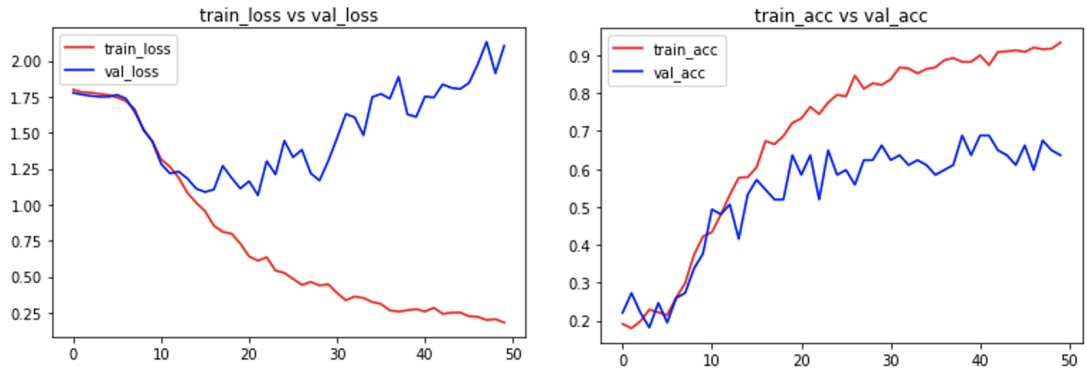


Figure 23: Training/validation loss and accuracy for FER model without pre-processing

From the above graphs, we can see that the gap between the validation loss and training loss keeps on increasing which indicates that the model is performing well on the training data but not on the validation data. This implies that the model is memorizing the training data instead of learning its underlying patterns, leading to overfitting. This is mainly due to the limited dataset. To address this issue, we implemented several data preprocessing techniques discussed below.

By applying these techniques, we improved the generalization of the model, reduced overfitting, and obtained better performance on unseen data.

Table 2: Precision and recall values for FER model without pre-processing

| **Emotion** | **Precision** | **Recall** |
|---|---|---|
| **surprise** | 0.777778 | 0.823529 |
| **happy** | 0.722222 | 0.764706 |
| **sadness** | 0.8 | 0.5 |
| **anger** | 0.555556 | 0.555556 |
| **disgust** | 0.666667 | 0.705882 |
| **fear** | 0.555556 | 0.555556 |
| **Average** | 0.68 | 0.65 |

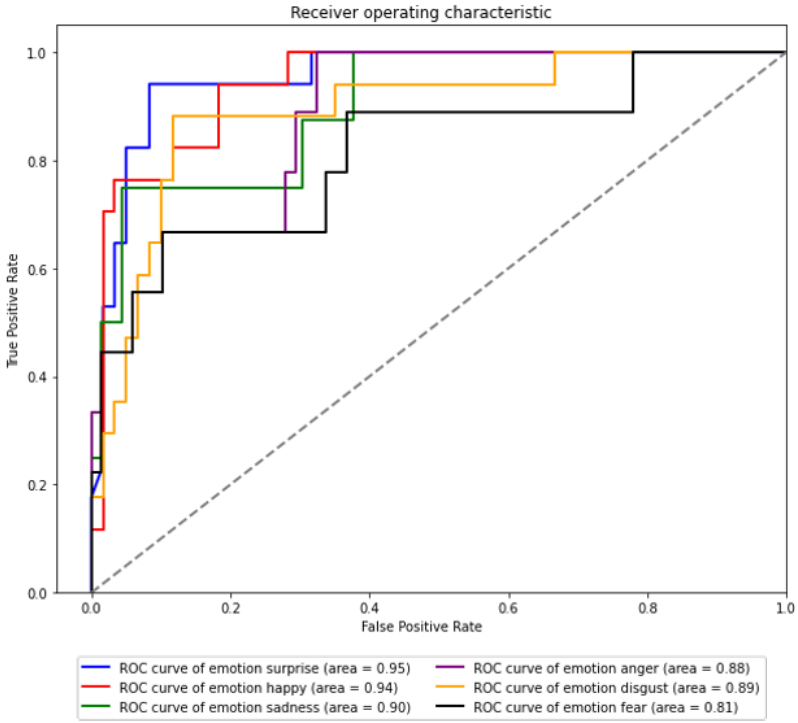


Figure 24: ROC curve for FER model without pre-processing

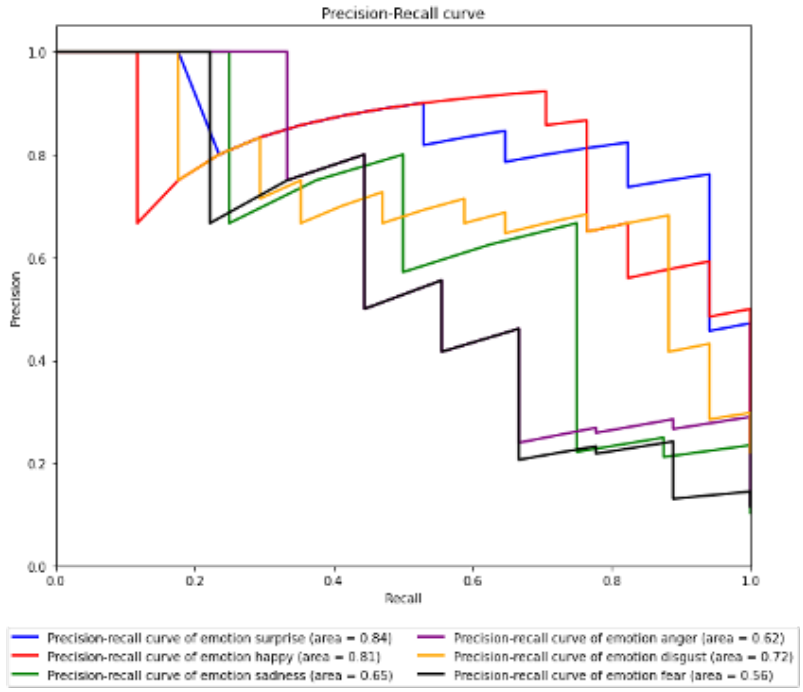


Figure 25: PR curve for FER model without pre-processing

### 7.4 CNN with Image Pre-processing

#### 7.4.1 CNN with Face Detection

The first image pre-processing technique that we experimented with was face detection and region of interest selection. After converting the input images to grayscale and resizing them, Haar Cascade Classifier was used to detect and crop the region consisting only of the face in the images. Using these images as input to our CNN model resulted in an accuracy of 87% without cross-validation and a mean accuracy of 82% after cross-validation. This is a 15.8% increase in mean accuracy from the FER model without any image processing. The validation loss also reduced from 1.628 to 0.475. From the below figures we can see the gap between training loss and validation loss is significantly reduced, decreasing the amount of overfitting.

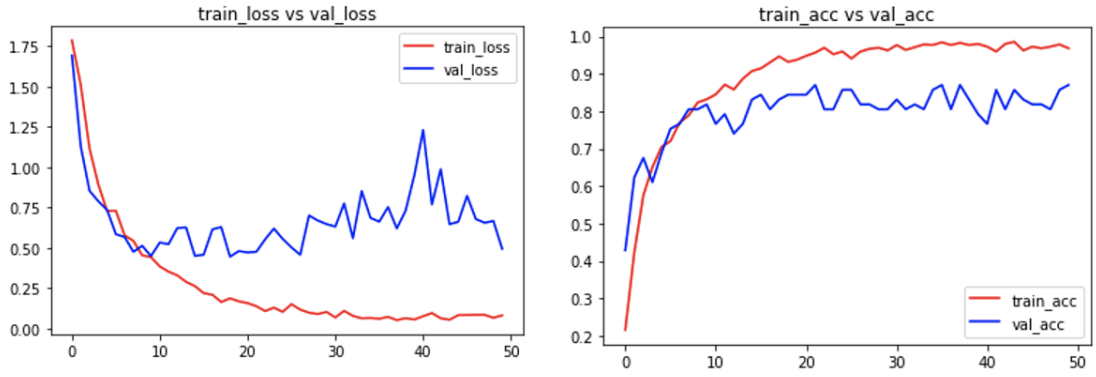


Figure 26: Training/validation loss and accuracy for FER model with face detection

The mean precision and mean recall, both, for this model were 0.83. Face detection is used as the base image pre-processing technique in all the further models.

Table 3: Precision and recall values for FER model with face detection

| Emotion | Precision | Recall |
|---|---|---|
| **surprise** | 0.888889 | 0.941176 |
| **happy** | 1 | 1 |
| **sadness** | 0.625 | 0.625 |
| **anger** | 0.666667 | 0.666667 |
| **disgust** | 0.9375 | 0.882353 |
| **fear** | 0.888889 | 0.888889 |
| **Average** | 0.83 | 0.83 |

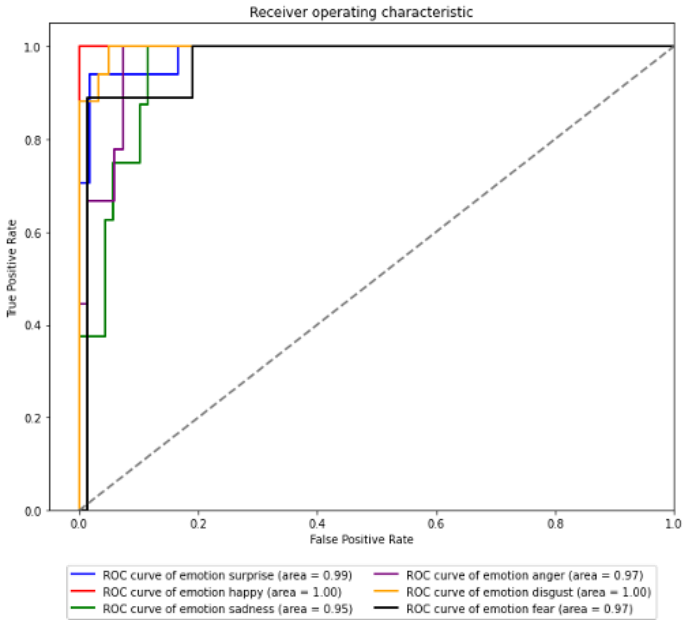


Figure 27: ROC curve for FER model with face detection

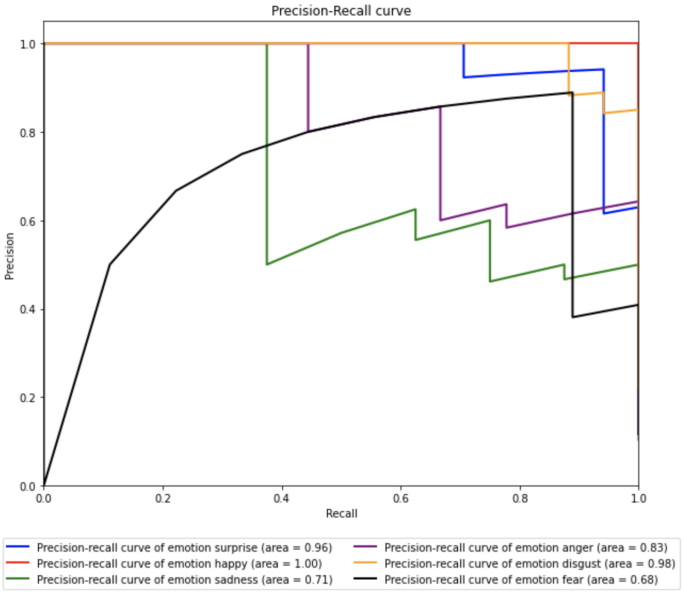


Figure 28: PR curve for FER model with face detection

### 7.4.2 CNN with Image Sharpening

For image sharpening, we implemented unsharp mask technique in which, a blurry image is subtracted from the original image to obtain just its edges, and the resulting addition to the original image creates an improved version. Unsharp mask is applied along with face detection and cropping. This CNN resulted in an accuracy of 88.3% without any cross-validation and a mean accuracy of 87.5% after cross-validation. This is a 5.5% increase in the mean accuracy from that of the CNN model with just face detection. The mean precision and mean recall observed were 0.86 and 0.88 respectively.

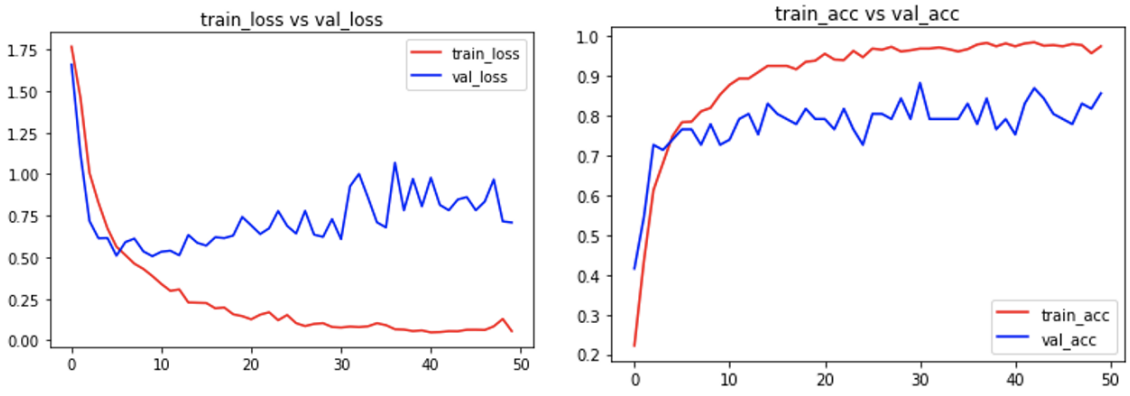


Figure 29: Training/validation loss and accuracy for FER model with face detection + unsharp mask

Table 4: Precision and recall values for FER model with face detection + unsharp mask

| **Emotion** | **Precision** | **Recall** |
|---|---|---|
| **surprise** | 1 | 0.823529 |
| **happy** | 0.941176 | 0.941176 |
| **sadness** | 0.75 | 0.75 |
| **anger** | 0.75 | 1 |
| **disgust** | 1 | 0.882353 |
| **fear** | 0.727273 | 0.888889 |
| **Average** | 0.86 | 0.88 |

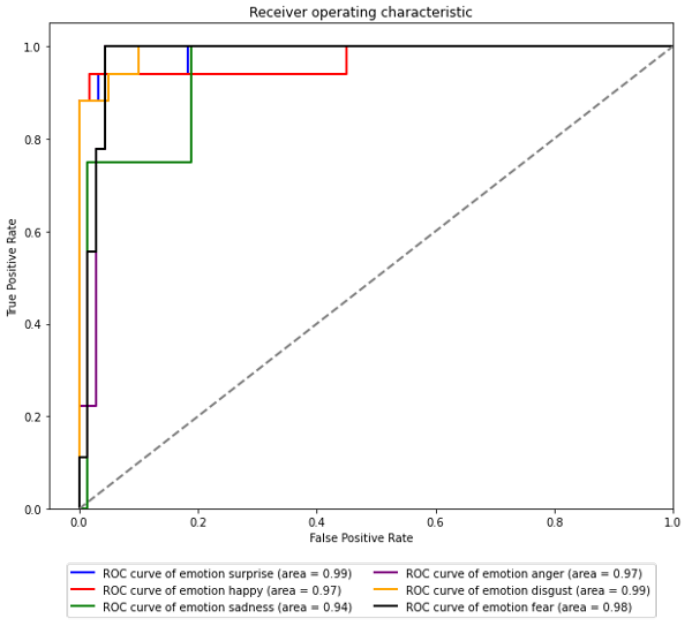


Figure 30: ROC curve for FER model with face detection + unsharp mask

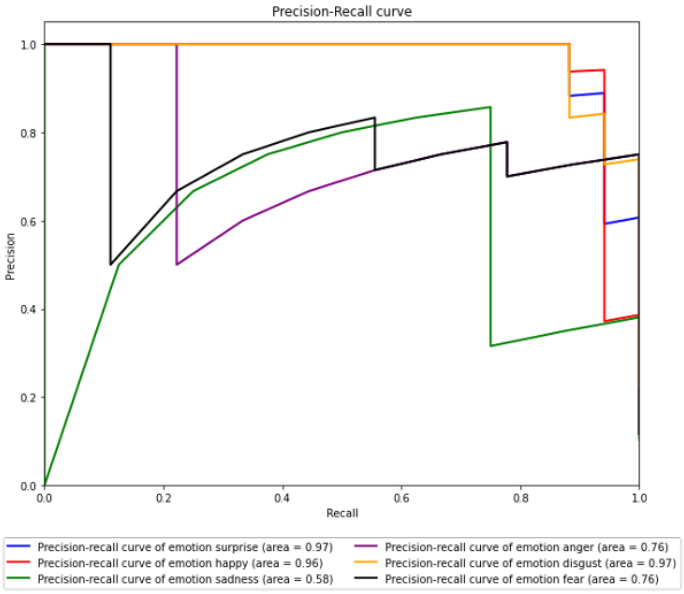


Figure 31: PR curve for FER model with face detection + unsharp mask

### 7.4.3 CNN with Image Normalization

We experimented with 3 different normalization techniques: Global contrast normalization (GCN), Histogram Equalization, and Adaptive Histogram Equalization

(AHE). Image normalization is nothing but the adjustment of the pixel values of an image so that they fall within a specific range or distribution. It is mainly done to correct variations in lighting conditions across different images. When images are captured under different lighting conditions, such as different times of the day or in different weather conditions, the resulting images can have very different brightness and contrast levels. Normalizing the images can help to correct these variations, making it easier to compare and analyze them. Another reason for image normalization is to improve the performance of CNN as machine learning algorithms including CNN are sensitive to the scale and distribution of the input data, and normalizing the images can help to ensure that the algorithm performs optimally.

We evaluated the performance of the models without any cross-validation and then again after performing 5-fold cross-validation. The results without cross-validation showed that the CNN with AHE technique gave the best accuracy of 90.9%, with a loss of 0.476, mean precision of 0.90, and mean recall of 0.91. The Histogram Equalization technique gave an accuracy of 88.3%, with a loss of 0.389, mean precision of 0.85, and mean recall of 0.86. Lastly, the GCN pre-processing technique gave an accuracy of 89.6%, with a loss of 0.417, mean precision of 0.89, and mean recall of 0.89. Also, when we performed 5-fold cross-validation, we observed similar results. The CNN with GCN pre-processing technique had a mean accuracy of 87.4%. The Histogram Equalization technique had a mean accuracy of 86.7%. Lastly, the AHE had the highest mean accuracy of 87.9%.

The results suggest that CNN with AHE pre-processing technique is the most effective normalization method, as it provided a higher accuracy with better precision and recall scores. We believe that this could be because AHE provides more local adaptivity and better preservation of image features while avoiding over-enhancement of noise and artifacts.

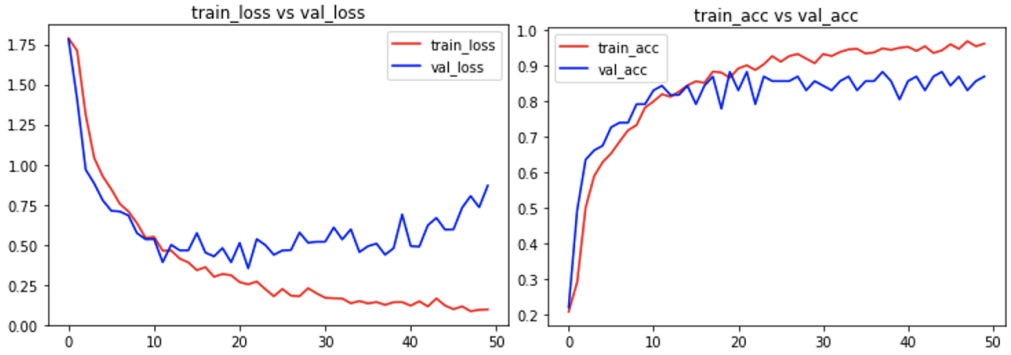


Figure 32: Training/validation loss and accuracy for FER model with face detection + histogram equalization

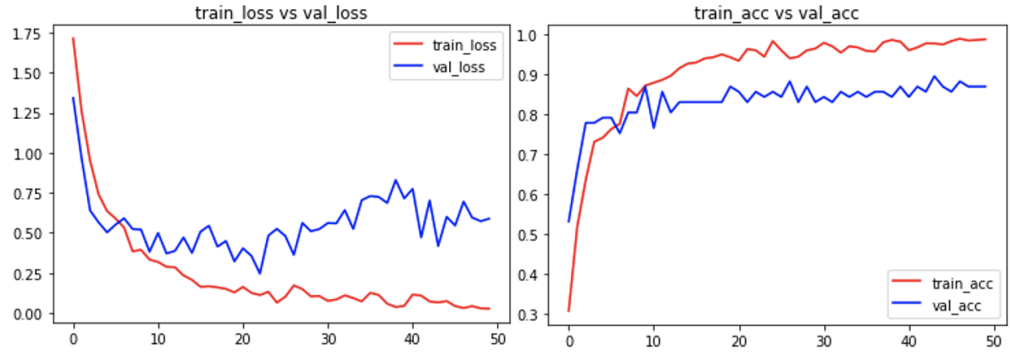


Figure 33: Training/validation loss and accuracy for FER model with face detection + GCN

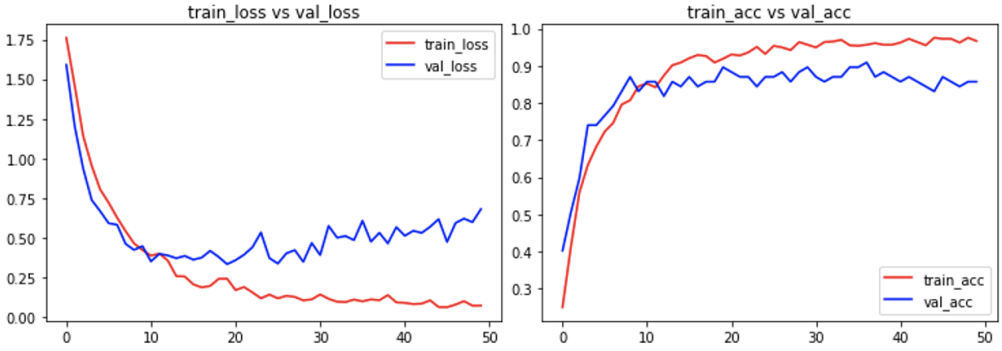


Figure 34: Training/validation loss and accuracy for FER model with face detection + AHE

Table 5: Precision and recall scores. face detection + normalizations

| | **Face detection + Histogram Equalization** | | **Face detection + GCN** | | **Face detection + AHE** | |
|---|---|---|---|---|---|---|
| **Emotion** | **Precision** | **Recall** | **Precision** | **Recall** | **Precision** | **Recall** |
| **surprise** | 0.941176 | 0.941176 | **0.944444** | **1** | 0.842105 | 0.941176 |
| **happy** | **1** | **1** | 0.9375 | 0.882353 | 0.941176 | 0.941176 |
| **sadness** | 0.6 | 0.75 | 0.666667 | **1** | **0.888889** | **1** |
| **anger** | 0.777778 | 0.777778 | 0.8 | 0.888889 | **0.9** | **1** |
| **disgust** | **1** | **0.823529** | **1** | **0.882353** | **1** | **0.882353** |
| **fear** | 0.8 | **0.888889** | **1** | 0.666667 | 0.857143 | 0.666667 |
| **Average** | 0.85 | 0.86 | 0.89 | 0.89 | **0.9** | **0.91** |

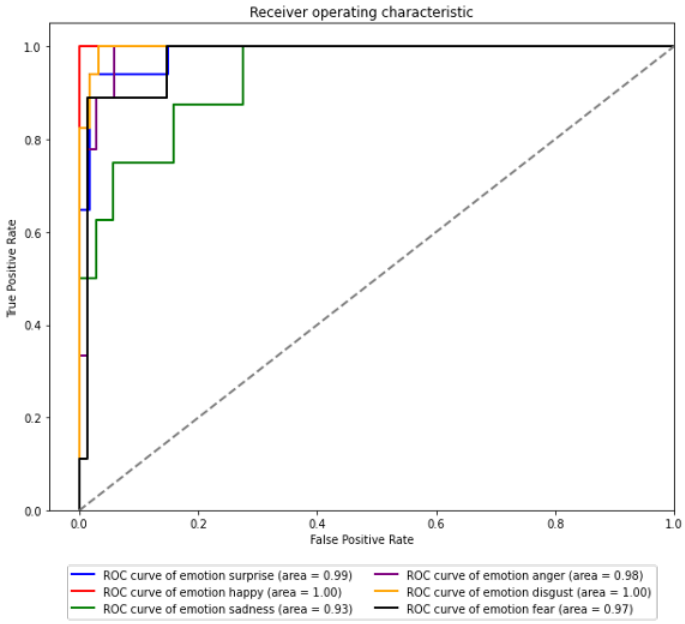


Figure 35: ROC curve for FER model with face detection + histogram equalization

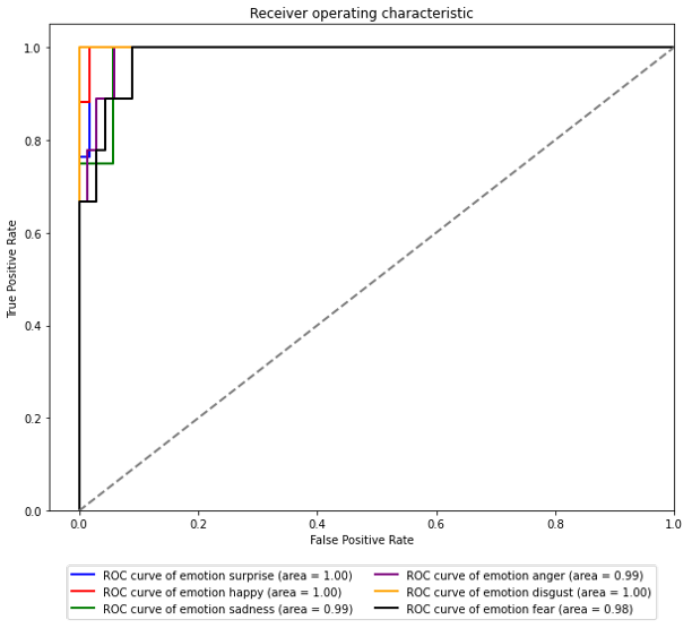


Figure 36: ROC curve for FER model with face detection + GCN

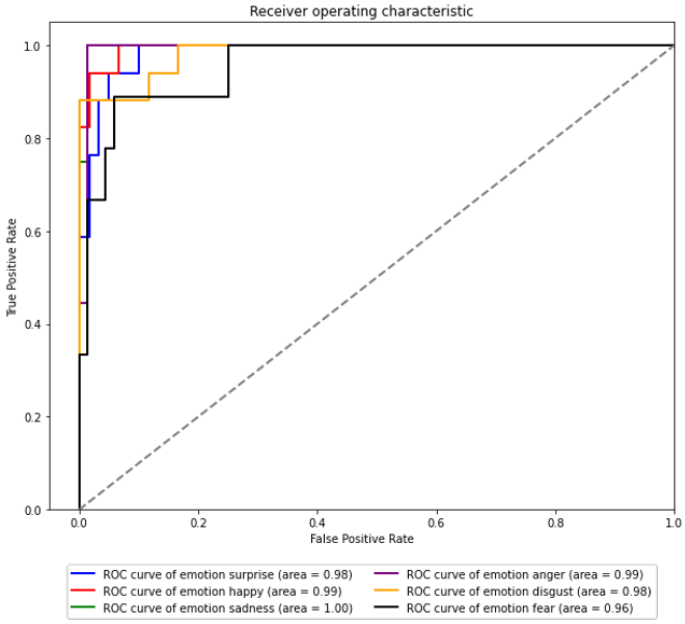


Figure 37: ROC curve for FER model with face detection + AHE

### 7.4.4 CNN with Image Sharpening and Image Normalization

We further experimented by combining unsharp mask with the image normalization techniques. By combining unsharp masking with image normalization, it may be possible to enhance the fine-grained details and edges in the images while also reducing the effects of illumination and contrast variations. This could improve the accuracy of the CNN in FER image classification task. We saw an increase in mean accuracy (with cross-validation) from 87.4% to 89.4% when we combined unsharp mask with GCN. Similar to this, after combining with unsharp mask, the mean accuracies for histogram equalization and AHE rose by 0.9% (86.7% to 87.6%) and 0.6% (87.9% to 88.5%), respectively. Without cross validation the accuracies were 88.3%, 90.9%, 92.2% for unsharp mask with histogram equalization, unsharp mask with GCN, and unsharp mask with AHE respectively.

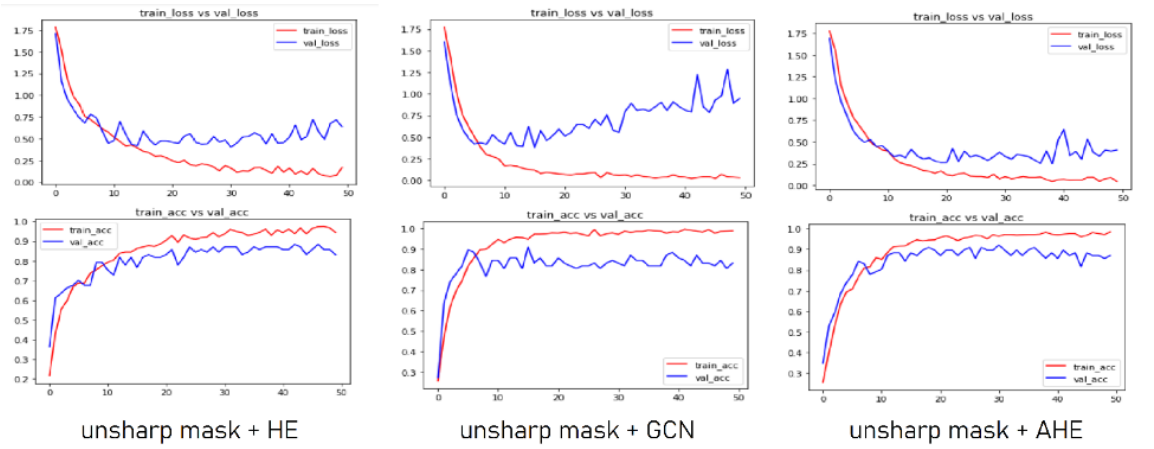


Figure 38: Training/validation loss and accuracy graphs of FER models with image sharpening + image normalization

Table 6: Precision and recall scores. face detection + unsharp mask + normalizations

| | **Face detection + Unsharp Mask + Histogram Equalization** | | **Face detection +Unsharp Mask + GCN** | | **Face detection + Unsharp Mask +AHE** | |
|---|---|---|---|---|---|---|
| **Emotion** | **Precision** | **Recall** | **Precision** | **Recall** | **Precision** | **Recall** |
| **surprise** | **0.941176** | 0.941176 | 0.85 | **1** | 0.894737 | **1** |
| **happy** | 0.941176 | 0.941176 | 0.944444 | **1** | **1** | 0.941176 |
| **sadness** | 0.714286 | 0.625 | **0.857143** | 0.75 | 0.777778 | **0.875** |
| **anger** | 0.75 | **1** | **0.888889** | 0.88889 | 0.818182 | **1** |
| **disgust** | **1** | 0.882353 | **1** | 0.94118 | **1** | **0.941176** |
| **fear** | 0.777778 | **0.777778** | 0.857143 | 0.66667 | **1** | 0.666667 |
| **Average** | 0.85 | 0.86 | 0.9 | 0.87 | **0.92** | **0.9** |

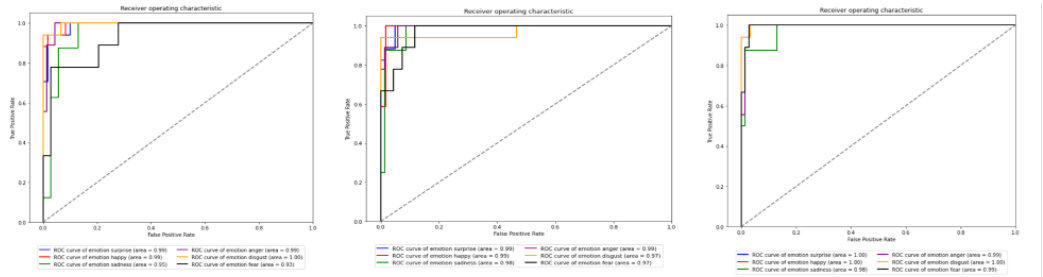


Figure 39: ROC curves. face detection + unsharp mask + HE vs face detection + unsharp mask + GCN vs face detection + unsharp mask + AHE

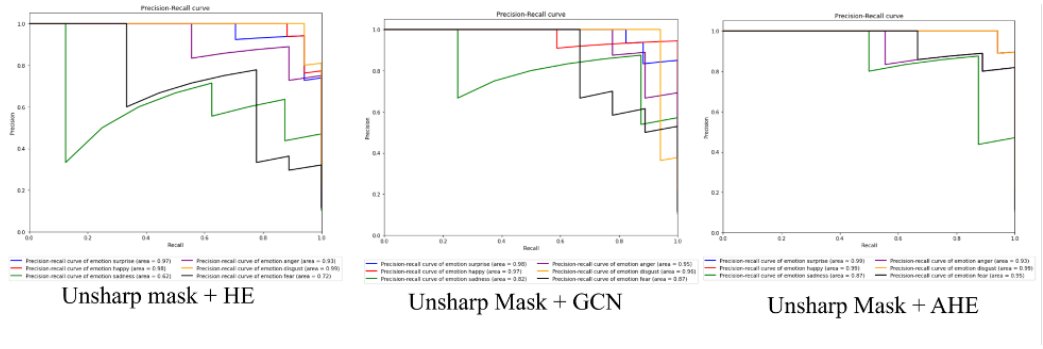


Figure 40: PR curves. face detection + unsharp mask + HE vs face detection + unsharp mask + GCN vs face detection + unsharp mask + AHE

### 7.4.5 CNN with Image Pre-processing and Data Augmentation

To experiment with Data Augmentation (DA) techniques, we selected the best 2 performing models: Face detection + Unsharp Mask + AHE with CNN and Face

detection + Unsharp Mask + GCN with CNN. Data augmentation using image translations and data augmentation by adding noise were performed on these models and compared.

#### 7.4.5.1 Data Augmentation by image translation

Image translations were achieved using the ImageDataGenerator function of the Keras library. We applied various image transformations, including rotation, width and height shift, zoom, shear, and horizontal flip, to increase the variability of the training data and improve the robustness of the models. The results showed that data augmentation improved the performance of the models significantly. Without using cross-validation, the accuracy of the CNN with face detection and cropping + unsharp mask + AHE pre-processing pipeline increased by 1.3% after applying image translations. The accuracy of CNN with face detection and cropping + unsharp mask + GCN pre-processing pipeline increased by 5.2%. With cross-validation, face detection and cropping + unsharp mask + GCN pre-processing pipeline CNN had an increase in mean accuracy from 89.4% to 92.3%, and a reduction in mean loss from 0.572 to 0.441 after data augmentation. In a similar vein, when we used data augmentation, the model with face recognition and cropping + unsharp mask + AHE pre-processing pipeline saw an increase in mean accuracy from 88.5% to 93.3%, and a decrease in mean loss from 0.434 to 0.313. This is our best-performing model with the highest mean accuracy. We also trained this model on 7 classes (including contempt) and achieved a mean accuracy of 91%.

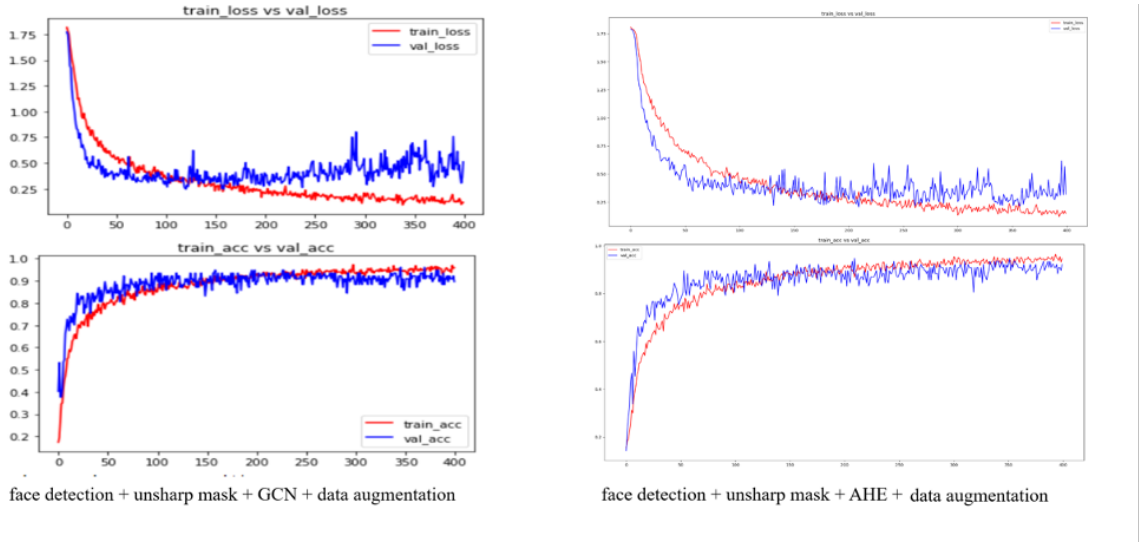


Figure 41: Training/validation loss and accuracy graphs. Face detection + unsharp mask + GCN + image augmentation vs Face detection + unsharp mask + AHE + image augmentation

Table 7: Precision and recall scores. face detection + unsharp mask + normalization + image augmentation

| | **Face detection + Unsharp Mask + GCN + Image Normalization** | | **Face detection + Unsharp Mask + AHE + Image Normalization** | |
|---|---|---|---|---|
| **Emotion** | **Precision** | **Recall** | **Precision** | **Recall** |
| **surprise** | **0.941176** | **0.941176** | **0.941176** | **0.941176** |
| **happy** | **1** | **1** | **1** | **1** |
| **sadness** | **1** | **0.875** | 0.875 | **0.875** |
| **anger** | 0.9 | **1** | **1** | 0.888889 |
| **disgust** | **1** | **1** | **1** | **1** |
| **fear** | **0.888889** | **0.888889** | 0.8 | **0.888889** |
| **Average** | **0.96** | **0.95** | 0.94 | 0.93 |

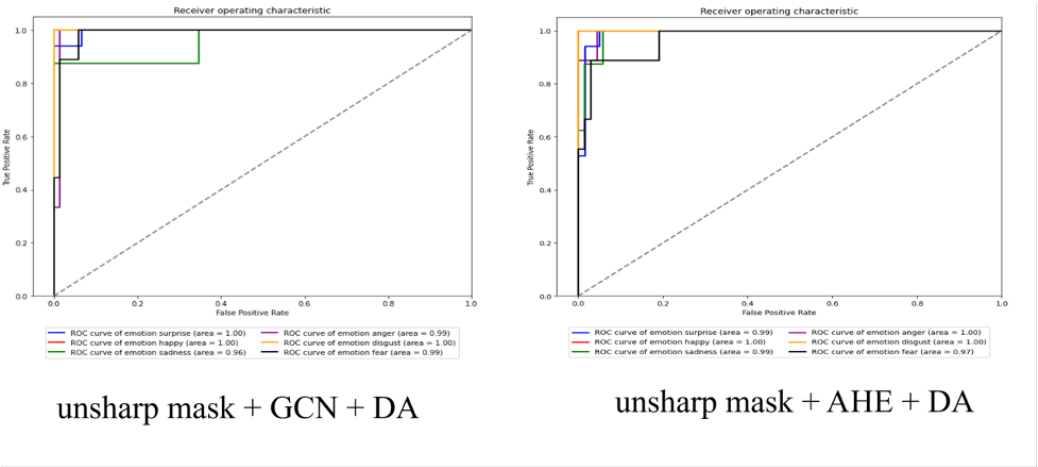


Figure 42: ROC curves. face detection + unsharp mask + HE vs face detection + unsharp mask + GCN + image augmentation vs face detection + unsharp mask + AHE + image augmentation

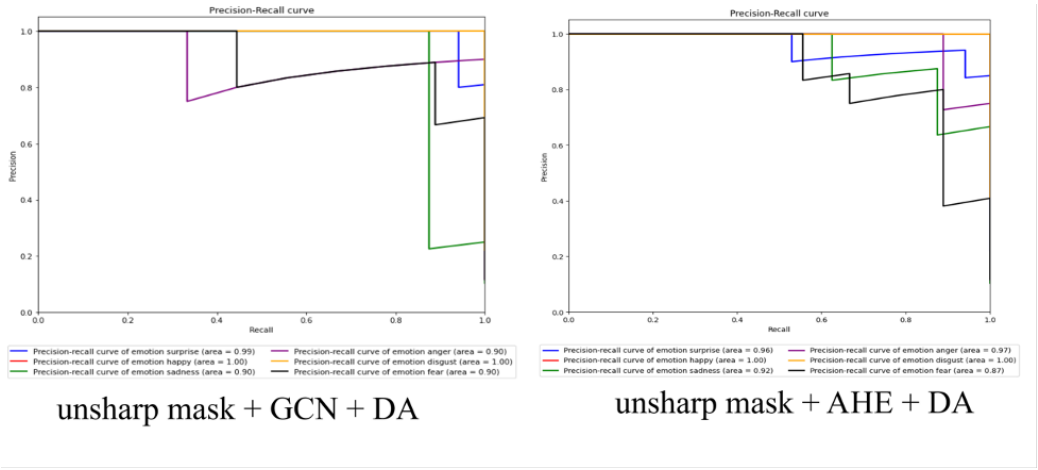


Figure 43: PR curves. face detection + unsharp mask + HE vs face detection + unsharp mask + GCN + image augmentation vs face detection + unsharp mask + AHE + image augmentation

The results suggest that data augmentation is a powerful technique to improve the performance of CNN based FER. It helps to increase the variability of the training data and improve the robustness of the models to changes in lighting, contrast, and other factors that may affect the appearance of the faces in the images.

#### 7.4.5.2 Data Augmentation by adding noise

To investigate the impact of data augmentation by adding noise we followed 2 approaches. First approach was similar to the one followed by [19], where the synthetic images with added Gaussian noise were included in both the training and testing sets. However, this could result in data overlap where there could be a clean version of an image in the test set and a noisy version of the same image in the training set or vice versa. This can lead to an overly optimistic evaluation of the model's performance and may not generalize well to real-world scenarios where noise is present. For a fair and reasonable evaluation, in second approach, we implemented the CNN models in which the generated noisy images were only present during training, without including them in the testing datasets. This is done to prevent data overlap between the two sets and provide a more accurate evaluation of the model's performance.

The results (with cross-validation) showed that the accuracy of the CNN with face detection and cropping + unsharp mask + GCN pre-processing pipeline was 96.4% (7% increase) and the accuracy of the CNN with face detection and cropping + unsharp mask + AHE pre-processing pipeline was 98% (9.5% increase) when the synthetic images with added Gaussian noise were included in both the training and testing sets. However, when we evaluated the impact of adding synthetic images with noise only to the training set, we observed a slight decrease in the mean accuracy of both models. Specifically, for CNN with face detection and cropping + unsharp mask + GCN pre-processing pipeline, the mean accuracy decreased from 89.4% to 88.9%, while for CNN with face detection and cropping + unsharp mask + AHE pre-processing pipeline, the mean accuracy decreased from 88.5% to 86.7%. These results suggest that data augmentation by adding synthetic images with noise did not improve the model's performance. By looking at the below graphs we can interpret that the added noise is making the model too specific to the training data which is

leading to overfitting.

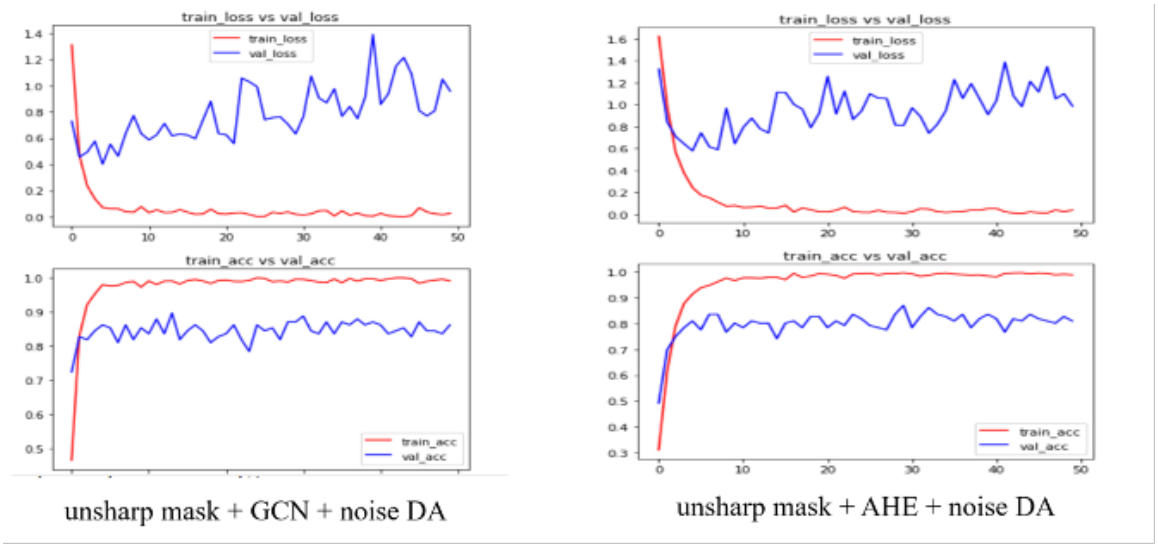


Figure 44: Training/validation loss and accuracy graphs. Face detection + unsharp mask + GCN + noise image augmentation vs Face detection + unsharp mask + AHE + noise image augmentation

Table 8: Comparison between accuracy and loss. Noise DA on both train and test set vs Noise DA on just train set

| **Pipeline** | **Mean accuracy** | **Mean validation loss** |
|---|---|---|
| Face detection + unsharp mask + GCN + noise DA (both training and test) | 96.4% | 0.178 |
| Face detection + unsharp mask + AHE + noise DA (both training and test) | 98% | 0.117 |
| Face detection + unsharp mask + GCN + noise DA (only training) | 88.9% | 0.764 |
| Face detection + unsharp mask + AHE + noise DA (only training) | 86.7% | 0.546 |

# CHAPTER 8

## Results

The preprocessing pipeline that gave the highest mean accuracy of 93.3% was face detection and cropping + unsharp mask + Adaptive Histogram Equalization + data augmentation by image translations. The cross validation mean accuracy of each pipeline is shown in the table below.

Table 9: Comparison of different pre-processing pipelines

| **Pre-processing pipeline** | **Mean Accuracy** | **Mean Loss** |
|---|---|---|
| no pre-processing | 66.2% | 1.352 |
| face detection and cropping | 82.0% | 0.544 |
| face detection and cropping + unsharp mask | 87.5% | 0.518 |
| face detection and cropping + GCN | 87.4% | 0.540 |
| face detection and cropping + Histogram Equalization | 86.7% | 0.567 |
| face detection and cropping + AHE | 87.9% | 0.647 |
| face detection and cropping + unsharp mask + GCN | 89.4% | 0.572 |
| face detection and cropping + unsharp mask + Histogram Equalization | 87.6% | 0.600 |
| face detection and cropping + unsharp mask + AHE | 88.5% | 0.434 |
| **face detection and cropping + unsharp mask + AHE + data augmentation by image transformations** | **93.3%** | **0.313** |
| face detection and cropping + unsharp mask + AHE + data augmentation by image transformations (7 classes) | 91% | 0.363 |
| face detection and cropping + unsharp mask +GCN + data augmentation by image transformations | 92.3% | 0.441 |
| face detection and cropping + unsharp mask + AHE + noise data augmentation (both train and test set) | 98.0% | 0.117 |
| face detection and cropping + unsharp mask + GCN + noise data augmentation (both train and test set) | 96.4% | 0.178 |
| face detection and cropping + unsharp mask + AHE + noise data augmentation (only training) | 86.7% | 0.546 |
| face detection and cropping + unsharp mask + GCN + noise data augmentation (only training) | 88.9% | 0.764 |

Our results show that applying face detection and ROI selection alone increased

the accuracy of the model by 15.8% compared to the baseline model without any preprocessing. Adding unsharp mask to the face detection and ROI selection pipeline further improved the accuracy by 5.5%. Adaptive Histogram Equalization performed the best with an accuracy of 87.9% among all image normalization techniques. When Adaptive Histogram Equalization was combined with unsharp mask, accuracy increased to 88.5%. Global contrast normalization performed slightly better than Adaptive Histogram Equalization when combined with unsharp mask, giving an accuracy of 89.4%. However, when we incorporated data augmentation by image translations while training the model, the face detection + unsharp mask + Adaptive Histogram Equalization pipeline outperformed face detection + unsharp mask + Global Contrast Normalization by 1% with an accuracy of 93.3%. Furthermore, the model with Adaptive Histogram Equalization had a lower mean loss than the model with Global Contrast Normalization. We believe that Adaptive Histogram Equalization performed better than other normalization techniques because it can enhance the contrast of images while preserving local details and avoiding the over-enhancement of noise and artifacts. AHE achieves this by dividing the image into small regions and applying histogram equalization independently to each region. Another thing to note is that combining image normalization techniques with unsharp mask further increased accuracy. This could be because combining both methods could enhance the fine-grained details and edges in the images while also reducing the effects of illumination and contrast variations. We trained and tested our best model on 7 classes by adding images of the contempt class and got an accuracy of 91%. The PR curve area was computed for each class, and the results were compared between the 6-class and 7-class models.

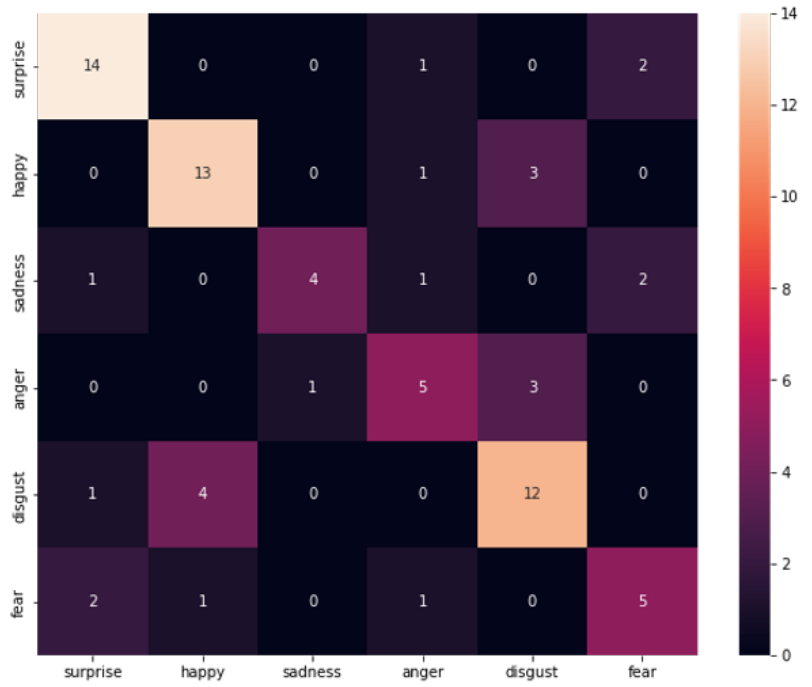


Figure 45: Confusion matrix of FER with no pre-processing

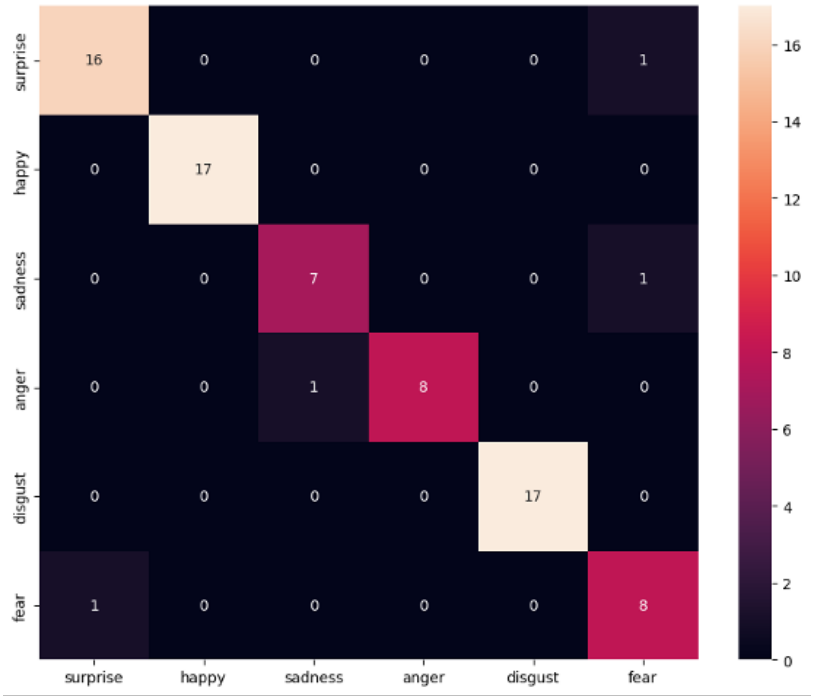


Figure 46: Confusion matrix of FER with face detection and cropping + unsharp mask + AHE + data augmentation by image transformations (6 classes)

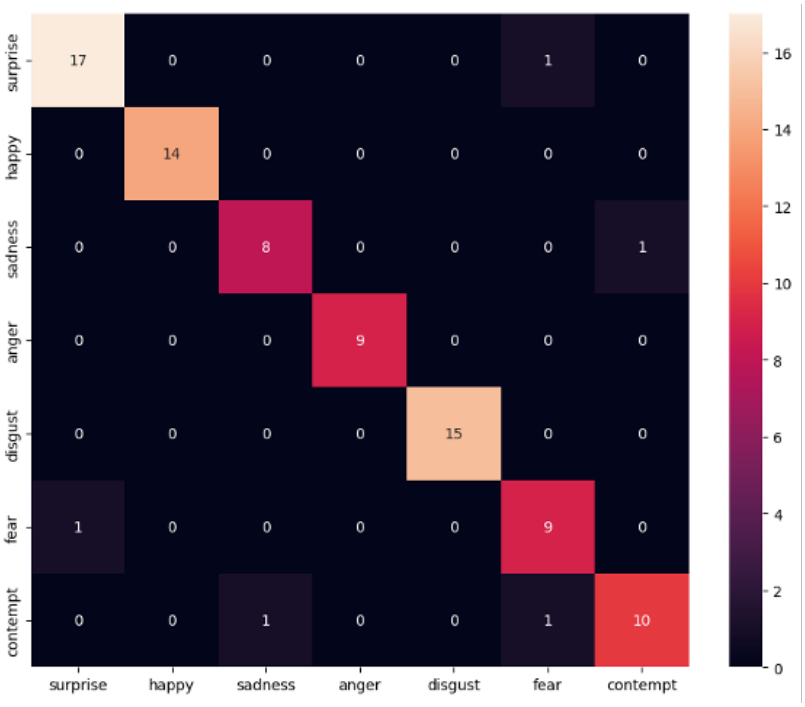


Figure 47: Confusion matrix of FER with face detection and cropping + unsharp mask + AHE + data augmentation by image transformations (7 classes)

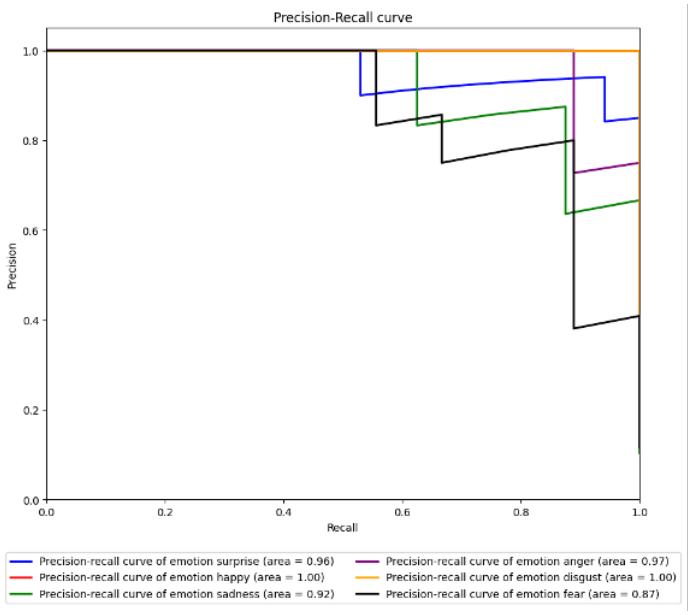


Figure 48: PR curve of FER with face detection and cropping + unsharp mask + AHE + data augmentation by image transformations (6 classes)

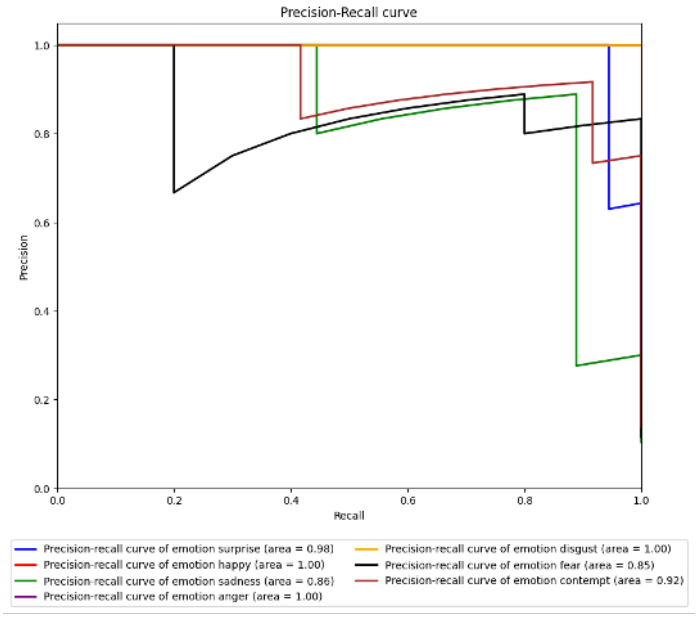


Figure 49: PR curve of FER with face detection and cropping + unsharp mask + AHE + data augmentation by image transformations (7 classes)

The 6-class model had the lowest PR curve area for fear (0.87), followed by sadness (0.92) and surprise (0.96). The 7-class model also had the lowest PR curve area for fear (0.85), followed by sadness (0.96) and contempt (0.92). These results indicate that the model had difficulty differentiating between fear and other emotions, especially sadness and surprise. It is important to note that some facial expressions can have multiple interpretations, which makes it even harder for models to identify differentiating patterns.

Data augmentation by adding Gaussian noise to training images did not increase the performance of our models. In fact, it resulted in lower accuracies and higher validation loss than models without data augmentation by adding noise. When compared to another multi-database FER model by [23], our model outperformed, as shown in the table below.

Table 10: Comparison with Pitloka et al. FER models between accuracy and loss.

| **Pitloka et al. Pre-processing pipeline** | **Pitloka et al. accuracies (no cross-validation)** | **Our Pre-processing pipeline** | **Our accuracies (5 fold cross-validation)** |
|---|---|---|---|
| No pre-processing | 62.35% | No pre-processing | 61.6%, 67.5%, 69.2%, 69.3%, 63.4% Mean: **66.2%** |
| Face detection | **87.06%** | Face detection | 84.4%, 87%, 83%, 79.7%, 75.8% Mean: 82% |
| Face detection + GCN | 89.41% | Face detection + unsharp mask+ GCN + DA | 88.9%, 92.8%, 94.1%, 96.1%, 89.5% Mean: **92.3%** |
| Face detection + Histogram Equalization | 90.56% | Face detection + unsharp mask+ AHE + DA | 93.5%, 94.8%, 92.1%, 92.1%, 94.1% Mean: **93.3%** |

In addition, a comparison with other studies that did not use multi-database validation was also performed [24] [25] [26]. The results showed that our model achieved higher or comparable accuracy to most of the compared models, as shown in the table below. This is significant considering that our model is trained on multiple

datasets, exposing our model to a wider range of facial expressions, poses, lighting conditions, and image quality compared to models trained on a single dataset. By training on multiple datasets, our model is better able to generalize to new, unseen data, and is less likely to overfit to the specific characteristics of a single dataset. In contrast, models trained on a single dataset may perform well on that dataset but may not generalize well to other datasets or real-world scenarios. This is because the model has learned to recognize the specific characteristics of that dataset, and may not be able to adapt to variations in facial expressions and image quality that are present in other datasets or real-world scenarios. Therefore, our approach has the potential to be more widely applicable and effective in real-world scenarios where facial expression images may come from different sources with varying characteristics.

Table 11: Performance of previous recent studies that did not use multi-database validation

| **Previous research** | **Year** | **Model** | **Pre-processing techniques** | **Dataset** | **Accuracy** | **Cross-validation** |
|---|---|---|---|---|---|---|
| Vepuri | 2021 | Ensemble (5-layer CNN, Resnet-50, Senet-50, FaceNet) | Data augmentation, unsharp mask, histogram equalization | FER13 | 76.3% (6 classes) | no |
| Chowdary et al. | 2021 | VGG 19 | Image resizing | CK+ | 96% (7 classes) | no |
| Borgalli et al. | 2022 | 5-layer CNN | Face detection, data augmentation | FER13 | 86.71% (7 classes) | 10 fold |
| Borgalli et al. | 2022 | 5-layer CNN | Face detection, data augmentation | CK+ | 92.27% (7 classes) | 10 fold |
| Borgalli et al. | 2022 | 5-layer CNN | Face detection, data augmentation | JAFFE | 91.58% (7 classes) | 10 fold |

# CHAPTER 9

## Conclusion and Future Work

Based on the results presented in the project report, we can conclude that preprocessing techniques such as face detection and ROI selection, as well as the addition of unsharp mask, image normalization and data augmentation by image translations, can significantly improve the accuracy of facial expression recognition models. The results indicate that Adaptive Histogram Equalization is the most effective normalization technique for enhancing the contrast of facial expression images while preserving local details, and combining it with unsharp mask can further improve accuracy. Furthermore, our model achieved higher or comparable overall accuracy than models trained on a single dataset and outperformed previous multi-database FER model, indicating that our approach of training on multiple datasets has the potential to be more widely applicable and effective in real-world scenarios.

In terms of future work, there are several areas that could be explored to further improve the performance of the facial expression recognition model. Incorporating more advanced deep learning architectures, such as attention mechanisms or multi-modal fusion, could potentially improve the accuracy of the model. Enhancing human emotion recognition can have a big impact on a lot of different areas, like helping autistic children, making it easier for people who are blind to read facial expressions, making it possible for robots to communicate with people more effectively, and improving driver safety by keeping an eye on attention while driving. A better customer experience may be achieved by applying emotion detection technology, which can also increase the emotional intelligence of numerous apps. Overall, there are several exciting avenues for future research in the field of facial expression recognition, and we believe that our work has laid a strong foundation for further exploration in this area.

## LIST OF REFERENCES


[1] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, ''The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression,'' in *2010 ieee computer society conference on computer vision and pattern recognition-workshops*. IEEE, 2010, pp. 94--101.

[2] N. Raut, ''Facial emotion recognition using machine learning,'' 2018.

[3] S. Saha, ''A comprehensive guide to convolutional neural networks-the eli5 way,'' Nov 2022. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[4] T. Koffi, ''Introduction of dropout and ensemble model in the history of deep learning,'' Feb 2021. [Online]. Available: https://medium.com/unpackai/introduction-of-dropout-and-ensemble-model-in-the-history-of-deep-learning-a4c2a512dcca

[5] D. Frossard, ''Vgg in tensorflow,'' Jun 2016. [Online]. Available: https://www.cs.toronto.edu/~frossard/post/vgg16/

[6] A. D. Nishad, ''K-fold cross validation with simple example,'' Jul 2021. [Online]. Available: https://medium.com/@nishad009adi/k-fold-cross-validation-with-simple-example-e023bb2e2d43

[7] P. Ekman, ''An argument for basic emotions,'' *Cognition & emotion*, vol. 6, no. 3-4, pp. 169--200, 1992.

[8] A. K. Jain and S. Z. Li, *Handbook of face recognition*. Springer, 2011, vol. 1.

[9] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. R. Wrobel, ''Emotion recognition and its applications,'' *Human-Computer Systems Interaction: Backgrounds and Applications 3*, pp. 51--62, 2014.

[10] M. Dubey and L. Singh, ''Automatic emotion recognition using facial expression: a review,'' *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 2, pp. 488--492, 2016.

[11] Y. Tian, T. Kanade, and J. F. Cohn, ''Facial expression recognition,'' *Handbook of face recognition*, pp. 487--519, 2011.

[12] A. H. Mansour, G. Z. A. Salh, and A. S. Alhalemi, ''Facial expressions recognition based on principal component analysis (pca),'' *arXiv preprint arXiv:1506.01939*, 2014.

[13] S. Bansal and P. Nagar, ''Emotion recognition from facial expression based on bezier curve,'' *Int J Adv Inf Technol*, vol. 5, no. 4, p. 5, 2015.

[14] X. Guo, X. Zhang, C. Deng, and J. Wei, ''Facial expression recognition based on independent component analysis.'' *Journal of Multimedia*, vol. 8, no. 4, 2013.

[15] C. Shan, S. Gong, and P. W. McOwan, ''Facial expression recognition based on local binary patterns: A comprehensive study,'' *Image and vision Computing*, vol. 27, no. 6, pp. 803--816, 2009.

[16] N. Wang, Q. Li, A. A. A. El-Latif, J. Peng, and X. Niu, ''Two-directional two-dimensional modified fisher principal component analysis: an efficient approach for thermal face verification,'' *Journal of Electronic Imaging*, vol. 22, no. 2, pp. 023 013--023 013, 2013.

[17] T. Senthilkumar, S. Rajalingam, S. Manimegalai, and V. G. Srinivasan, ''Human facial emotion recognition through automatic clustering based morphological segmentation and shape/orientation feature analysis,'' in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2016, pp. 1--5.

[18] S. Rifai, Y. Bengio, A. Courville, P. Vincent, and M. Mirza, ''Disentangling factors of variation for facial expression recognition,'' in *Computer Vision--ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*. Springer, 2012, pp. 808--822.

[19] F. Z. Salmam, A. Madani, and M. Kissi, ''Facial expression recognition using decision trees,'' in *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*. IEEE, 2016, pp. 125--130.

[20] P. P. Thakare and P. S. Patil, ''Facial expression recognition algorithm based on knn classifier,'' *International Journal of Computer Science and Network*, vol. 5, no. 6, p. 941, 2016.

[21] X. Wang, J. Huang, J. Zhu, M. Yang, and F. Yang, ''Facial expression recognition with deep learning,'' in *Proceedings of the 10th international conference on internet multimedia computing and service*, 2018, pp. 1--4.

[22] A. T. Lopes, E. De Aguiar, A. F. De Souza, and T. Oliveira-Santos, ''Facial expression recognition with convolutional neural networks: coping with few data and the training sample order,'' *Pattern recognition*, vol. 61, pp. 610--628, 2017.

[23] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, ''Enhancing cnn with preprocessing stage in automatic emotion recognition,'' *Procedia computer science*, vol. 116, pp. 523--529, 2017.

[24] K. S. Vepuri, ‘‘Improving facial emotion recognition with image processing and deep learning,’’ 2021.

[25] M. K. Chowdary, T. N. Nguyen, and D. J. Hemanth, ‘‘Deep learning-based facial emotion recognition for human--computer interaction applications,’’ *Neural Computing and Applications*, pp. 1--18, 2021.

[26] M. R. A. Borgalli and S. Surve, ‘‘Deep learning for facial emotion recognition using custom cnn architecture,’’ in *Journal of Physics: Conference Series*, vol. 2236, no. 1. IOP Publishing, 2022, p. 012004.

[27] M. Lyons, M. Kamachi, and J. Gyoba, ‘‘The japanese female facial expression (jaffe) dataset,’’ *The Images Are Provided at No Cost for Non-Commercial Scientific Research Only. If You Agree to the Conditions Listed Below, You May Request Access to Download*, 1998.

[28] N. Aifanti, C. Papachristou, and A. Delopoulos, ‘‘The mug facial expression database,’’ in *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*. IEEE, 2010, pp. 1--4.

[29] N. Aifanti, C. Papachristou, and A. Delopoulos, ‘‘The mug facial expression database,’’ in *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*. IEEE, 2010, pp. 1--4.

[30] J. Wang, S. Wang, and Q. Ji, ‘‘Early facial expression recognition using hidden markov models,’’ in *2014 22nd International conference on pattern recognition*. IEEE, 2014, pp. 4594--4599.

[31] S. Zhou, Y. Liang, J. Wan, and S. Li, ‘‘Facial expression recognition based on multi-scale cnns,’’ 09 2016, pp. 503--510.

[32] M. Shin, M. Kim, and D.-S. Kwon, ‘‘Baseline cnn structure analysis for facial expression recognition,’’ 2016.

[33] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, *et al.*, ‘‘Challenges in representation learning: A report on three machine learning contests,’’ in *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*. Springer, 2013, pp. 117--124.

[34] K. Ricanek and T. Tesafaye, ‘‘Morph: A longitudinal image database of normal adult age-progression,’’ in *7th international conference on automatic face and gesture recognition (FGR06)*. IEEE, 2006, pp. 341--345.

[35] ‘‘Face detection using haar cascades.’’ [Online]. Available: https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html

[36] ‘‘Smoothing images.’’ [Online]. Available: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

[37] ‘‘Histogram equalization.’’ [Online]. Available: https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html

[38] ‘‘Histograms.’’ [Online]. Available: https://docs.opencv.org/4.x/d6/dc7/group__imgproc__hist.html

[39] ‘‘Tf.keras.preprocessing.image.imagedatagenerator nbsp;: nbsp; tensorflow v2.12.0.’’ [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator