Facial Emotion Recognition

Name	Student ID
Natali Walid Alfroukh	0183932
Basel Emad Obeidat	0199058
Mohammad Khaled Shaqman	0196456
Eman Faisal Emair	0176419

Project Supervisors

Dr. Mousa Al-Akhras

Dr. Ali Alrodan

Dr. Majdi Sawalha

COMPUTER INFORMATION SYSTEMS
KING ABDULLAH II SCHOOL OF INFORMATION TECHNOLOGY
THE UNIVERSITY OF JORDAN

CONTENTS

TABLE OF CONTENTS

Abstract 4

Acknowledgments 5

1. CHAPTER ONE: INTRODUCTION 1

1.1 Preamble **1**

1.2 Problem Statement **1**

1.3 Project Motivations **1**

1.4 Project Objectives **1**

1.5 Project SCOPE **1**

1.6 Project Software and Hardware Requirements **2**

Software requirement 2

Hardware requirement: 2

1.7 Project Scheduling **3**

1.8 Outline **4**

2. CHAPTER TWO: BACKGROUND REVIEW 5

2.1 Introduction **5**

2.2 Phases of Automated facial Emotion Recognition Systems **5**

2.3 History of Automated Emotion Recognition Systems **6**

2.4 Overall Solutions **6**

2.5 Summary **6**

3.CHAPTER THREE: METHODOLOGIES FOR FACIAL EMOTION RECOGNITION 7

3.1 Overview **7**

3.2 Machine learning algorithms **7**

3.2.1 Convolutional Neural Networks 7

Convolutional Neural Networks in facial emotion recognition: 7

3.2.2. Support vector machines 8

Support vector machines in facial emotion recognition: 8

3.2.3 Random Forest 8

Random forest in facial emotion recognition: 8

3.2.4 K-Nearest Neighbor 8

3.3 Data set **9**

4. Chapter four: EXPERIMENTS AND RESULTS 10

4.1 CNN in depth **10**

4.2 CNN Experiments **10**

4.2.1 Libraries: **11**

4.2.2 Dataset Selection: **11**

4.2.3: Data Preprocessing **12**

4.2.4. Model building: **13**

4.2.5 Model architecture: **14**
4.2.5.1 Model Summary: 16
4.2.6 Fitting the Model with Training and Validation Data **17**
4.2.6.1 Conclusion: 17
4.2.7 Logs section: **18**
4.2.8 Convolutional neural networks (CNN) prediction 19
4.2.9 Convolutional neural networks (CNN) results **19**
Plotting Accuracy & Loss 19
5. Chapter five: CONCLUSION AND FUTURE WORK 22
5.1 Conclusions: **22**
5.2 Future work **22**
5.3 References: **23**

List of tables:

Table name	Page number
Table 1.1: Software requirements	2
Table 1.2: Hardware requirements	2
Table 1.3: Project Schedule Management	3
Table 4.1: 1st CNN layer	14
Table 4.2: 2nd CNN layer	14
Table 4.3: 3rd CNN layer	15
Table 4.4: 4th CNN layer	15
Table 4.5: Flatten function	15
Table 4.6: Fully connected 1st layer	15
Table 4.7: Fully connected 2nd layer	15
Table 4.8: Fitting table	17
Table 4.9: Logs table	18
Table 4.10: Results table	20

List of figures:

Figure name	Page number
Figure (1.1): Gant Chart for emotion recognition.	3
Figure (3.1) Facial emotion recognition main process	7
Figure (3.2) Output dataset	9
Figure (4.1) CNN architecture	10
Figure (4.2) Output dataset	11
Figure (4.3) Facial emotion type count	11
Figure (4.4) Samples count	12
Figure (4.5): Training & Validation Ratio	12
Figure (4.6): Model architecture	14
Figure (4.7) Results for Keras model	19
Figure (4.8) Results for Keras and OpenCV model	20

Abstract

The focus of this study is on computer automated perception of human emotion. Face detection has been around for ages. Taking a step forward, human emotion displayed by face and felt by brain, captured in either video or image form can be approximated. Human emotion detection is the need of the hour so that modern artificial intelligent systems can emulate and gauge reactions from the face. This can be helpful to make informed decisions be it regarding identification of intent, promotion of offers or security related threats. Recognizing emotions from images or video is a trivial task for the human eye but proves to be very challenging for machines and requires many image processing techniques for feature extraction. Several machine learning algorithms are suitable for this job, we are using convolutional neural network (CNN), SVM, Random Forest and KNN for this research. Any detection or recognition by machine learning requires training algorithms and then testing them on a suitable dataset. This research explores a couple of machine learning algorithms as well as feature extraction techniques which would help us in accurate identification of human emotion.

Acknowledgments

After thanking God for helping us in the first place for everything, we would like to thank our supervisors for this project (Dr. Mousa al-akhras, Dr. Ali alrowdan, Dr. Majdi Sawalha) for teaching us core skills needed to succeed and reviewing our project.

1. CHAPTER ONE: INTRODUCTION

1.1 Preamble

Emotion recognition is a natural capability in human beings. However, if we are to ever create a humanoid robot that can interact and emote with its human companions, the difficult task of emotion recognition will have to be solved. The ability for a computer to recognize human emotion has many highly valuable real-world applications.

Human emotions can be classified as: fear, contempt, disgust, anger, surprise, sad, happy, in this research we will see how the features are extracted and modified for algorithms like convolutional neural network (CNN), Support Vector Machines, KNN and Random Forest. We will compare algorithms and the feature extraction techniques from to see which suites more.

1.2 Problem Statement

Our research is a must complement tool for every aspect in IT, in security, healthcare etc.

There are many companies that might want to use a model that will help them understand their clients' feelings now they are using their services to get feedback or to use these feelings to help their clients choose a specific service base on the emotion they show.

1.3 Project Motivations

Emotions play an essential role in identifying the mood of a human being. Also, recognizing emotions plays an important role in the domain of security it can be seen as a second step to face detection where we may be required to set up a second layer of security, where along with the face, the emotion is also detected. This can be useful to verify that the person standing in front of the camera is not just a 2-dimensional representation. Emotion recognition systems can be used as a sub-module of various applications like recommending music and various camera surveillance systems.

1.4 Project Objectives

Nowadays, more and more intelligent systems are using emotion recognition, our main goal from this research is to improve the interaction between machines and humans.

1.5 Project SCOPE

We are using python as a coding language and algorithms such as convolutional neural network (CNN), support vector machine, KNN and random forest.

1.6 Project Software and Hardware Requirements

Software requirement

Table (1.1): Software requirements.

Software	**Requirement**
Operating system	Windows 8.1 or higher operating systems
Browser	Firefox 3.0, Safari 4.1.2, and Google Chrome 11.0, all those browsers in the minimum versions.
SQL Server	Apache Server 1.7.1 or Higher included MySQL

Hardware requirement:

Table (1.2): Hardware requirements.

Hardware	**Requirement**
Computer	Core i5 - 1480 MHz Pentium minimum, V - 2 GHz or better processor is recommended.
CPU	Any processor that support **AVX2 to run** Neurotechnology emotion recognition algorithms
Memory (RAM)	2 GB or 4 GB
Hard disk	500 GB

1.7 Project Scheduling

Project management is the discipline of planning, organizing, securing, and managing resources to achieve specific goals. The following table displays the project management:

Table (1.3): Project Schedule Management in developing our project.

Task	Description	Start time	End Time	Duration	Dependency
T1	Planning	4/25/23	5/1/23	6 Days	
T2	Information Gathering	5/1/23	5/12/23	7 Days	T1
T3	Analysis	5/14/23	5/25/23	5 Days	T2
T4	Design	5/26/23	5/30/23	10 Days	T3
T5	Documentation	4/25/23	6/1/23	37 Days	T1, T2, T3, T4
T6	Submission	6/2/23	6/4/23	1 Day	T5

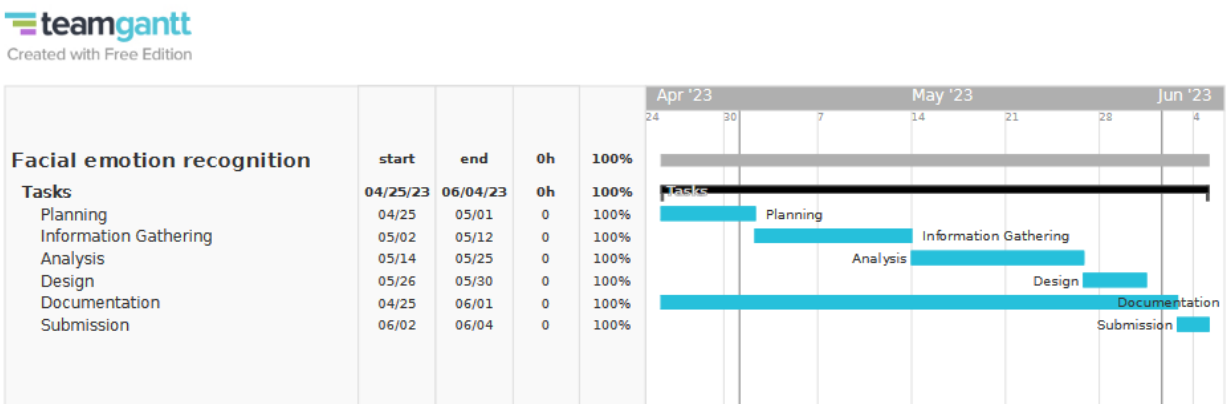

Figure (1.1): Gant Chart for emotion recognition.

1.8 Outline

- **Chapter one** shows a preamble in section 1.1, the project motivation is elaborated in Section 1.2, in additional the problem statement is stated in Section 1.3,Section 1.4, list the project aim and objectives, whereas the project scope is identified in section 1.5, Section 1.6 provide the project software and hardware requirement and Section 1.7 highlights the project limitation, furthermore project expected output is addressed in Section 1.8, whereas the project is scheduled is Section 1.9, the report outline in finally present in Section 1.10.
- **Chapter two** shows a background review about our project with introduction in 2.1, introduction in Section 2.1, Phases of Automated facial Emotion Recognition Systems in 2.2, History of Automated Emotion Recognition Systems in 2.3, provided by overall solutions in 2.4, whereas the summary is identified in Section 2.5.
- **Chapter three** show methodologies for facial emotion recognition provided with overview in in 3.1, machine learning algorithms in 3.2 which are (convolutional neural network (CNN), SVM, Random Forest and KNN) and Data set in 3.3
- **Chapter four** Shows experiments for convolutional neural network CNN PROVIDED in depth in 4.1, CNN experiments in 4.2, Libraries that we are going to use in 4.2.1, Data set in 4.2.2, Data preprocessing in 4.2.3, Model building in 4.2.4, Model architecture in 4.2.5 and model summary in 4.2.5.1, Fitting the Model with Training and Validation Data in 4.2.6, conclusion in 4.2.6., Logs section in 4.2.7, CNN prediction in 4.2.8 And CNN results in 4.2.9
- **Chapter five** shows Conclusions and future work, Conclusions in 5.1, future work in 5.2 and references in 5.3

2. CHAPTER TWO: BACKGROUND REVIEW

2.1 Introduction

Facial emotion recognition is a subfield of emotion recognition that focuses specifically on identifying emotions from facial expressions. This can be done using a variety of techniques, including machine learning algorithms, rule-based systems, and physiological measurements.

One of the most widely used approaches to facial emotion recognition is the Facial Action Coding System (FACS), developed by Paul Ekman and Wallace Friesen in the 1970s. FACS is a comprehensive system for coding and analyzing facial expressions that consists of a set of 46 facial muscle movements, called "action units," which can be combined to produce a wide range of facial expressions. FACS has been widely used in research on facial emotion recognition and is considered a reliable and valid measure of emotional expression.

Another approach to facial emotion recognition is machine learning. Machine learning algorithms can be trained to recognize emotions from facial expressions by being fed a large dataset of images labeled with the corresponding emotions. These algorithms can then be used to classify new images based on their similarity to the training data.

Facial emotion recognition has a wide range of applications, including psychology research, marketing, and customer service. It can be used to analyze the emotional responses of participants in a study, assess the effectiveness of advertisements, or improve the quality of customer interactions. However, there are also concerns about the potential misuse of facial emotion recognition technology, such as in surveillance or hiring decisions.

2.2 Phases of Automated facial Emotion Recognition Systems

Automated facial emotion recognition systems typically consist of the following phases:

1- Data collection: In this phase, a dataset of images that will be used to train and evaluate the emotion recognition system is collected. This may involve manually labeling the images with the corresponding emotions or using an existing dataset of labeled images.
2- Data preprocessing: In this phase, the collected images are cleaned and preprocessed to prepare them for analysis. This may include tasks such as removing noise, resizing the images to a standard size, and converting the images to a format that can be used by the emotion recognition system.
3- Feature extraction: In this phase, relevant features of the images are extracted and transformed into a form that can be used by the emotion recognition system. This may involve techniques such as edge detection, texture analysis, or facial feature extraction.
4- Classification: In this phase, a classifier is trained to recognize the emotions being expressed in the facial features. This may be done using a variety of techniques, such as convolutional neural network (CNN), support vector machines, Random Forest and KNN.
5- Model training: In this phase, the emotion recognition system is trained on the preprocessed and transformed data using a machine learning algorithm. This may involve adjusting the parameters of the algorithm to optimize its performance.

6- Model evaluation: In this phase, the performance of the trained emotion recognition system is evaluated using a set of test images that were not used in the training phase. This allows the system's performance to be compared to a baseline and gives an indication of how well the system will perform on new, unseen images.
7- Deployment: If the performance of the emotion recognition system is satisfactory, it can be deployed in the application for which it was designed. This may involve integrating the system into a larger system or deploying it as a standalone application.
8- Maintenance and updates: Once the emotion recognition system is deployed, it may require ongoing maintenance and updates to ensure that it continues to perform well. This may involve retraining the system on new data, adjusting the algorithm's parameters, or incorporating new features.

2.3 History of Automated Emotion Recognition Systems

The use of automated systems for recognizing emotions in facial expressions dates back to the 1960s, when Paul Ekman and Wallace Friesen developed the Facial Action Coding System (FACS). This system allows for the systematic measurement and description of facial expressions and was used as a basis for many early emotion recognition systems.

In the 1980s and 1990s, researchers began using artificial neural networks and other machine learning techniques to develop automated facial emotion recognition systems. These systems typically relied on hand-crafted features and required large amounts of labeled data for training.

In the 2000s and 2010s, advances in machine learning and computer vision, such as the development of deep learning algorithms, led to significant improvements in the accuracy and robustness of automated emotion recognition systems. Today, these systems are used in a variety of applications, including customer service, healthcare, and education.

2.4 Overall Solutions

Based on that there is no such a System related to the services that we are looking for, we implement Real Time Emotion Detection, as an online portal that provide real Detection for 3d pictures and give the data to the company to relate to apps, websites, APIs, systems and models.

2.5 Summary

Automated emotion recognition systems are computer-based systems that are designed to recognize and classify emotions based on facial expressions, voice, or other physiological or behavioral cues. These systems use techniques from artificial intelligence, machine learning, and computer vision to analyze and interpret emotional cues. They are used in a variety of applications, including customer service, healthcare, and education, and have the potential to improve communication and decision-making in these fields. While significant progress has been made in the development of automated emotion recognition systems, there are still challenges to be addressed, including the need for large amounts of labeled data for training and the difficulty of recognizing more complex or nuanced emotions.

3.CHAPTER THREE: METHODOLOGIES FOR FACIAL EMOTION RECOGNITION

3.1 Overview

There are several different methodologies that can be used for facial emotion recognition, including:

1. Rule-based systems: These systems use a set of predefined rules to recognize emotions based on facial expressions. These rules may be based on expert knowledge of facial anatomy and emotion, or they may be derived from statistical analysis of a large dataset of labeled facial expressions.
2. Feature-based systems: These systems extract specific features from facial images, such as the shape of the eyes or the curvature of the mouth and use these features to recognize emotions. These systems may use techniques such as principal component analysis or linear discriminant analysis to identify relevant features.
3. Template-based systems: These systems use a set of predefined templates or exemplars of facial expressions to recognize emotions. The input facial expression is compared to the templates, and the emotion with the closest match is selected.
4. Machine learning-based systems: These systems use techniques from machine learning, such as convolutional neural network (CNN), Support vector machines, KNN, or Random Forest, to learn to recognize emotions from a large dataset of labeled facial expressions. These systems do not rely on predefined rules or templates, but rather learn to recognize emotions through training and experience.

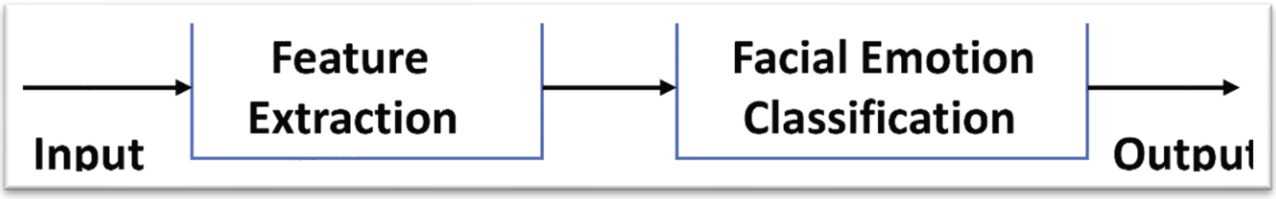

Figure (3.1) Facial emotion recognition main process

3.2 Machine learning algorithms

3.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep learning algorithms used for analyzing visual data like images. They consist of layers that extract features from the input data through convolutional operations and downsampling. These features are then used to classify or analyze the data. CNNs learn to recognize patterns in a hierarchical manner, mimicking the visual cortex in humans. They have achieved outstanding performance in computer vision tasks such as image classification and object detection.

Convolutional Neural Networks in facial emotion recognition:

CNNs are widely used in emotion recognition tasks to classify the emotional state of individuals based on input data such as facial expressions, speech signals, or physiological signals. For facial expression recognition, CNNs analyze and extract features from images or video frames to learn patterns and relationships between facial features and emotions. The network consists of convolutional layers for feature extraction, pooling layers for downsampling, and fully connected layers for classification. CNNs can also be applied to other modalities like speech or physiological signals for emotion

recognition. They are effective in capturing complex patterns and achieving high accuracy in understanding emotions from different types of input data.

3.2.2 Support vector machines

Support Vector Machines (SVMs) are a popular machine learning algorithm used for classification and regression tasks. They find an optimal hyperplane that separates different classes by maximizing the margin between data points. SVMs can handle both linear and non-linear data using the kernel trick, and they are widely used for various applications due to their robust performance.

Support vector machines in facial emotion recognition:

Support vector machines (SVMs) can be used for facial emotion recognition, which is the task of identifying the emotional state of a person based on their facial expression.

There are several challenges in using SVMs for facial emotion recognition, such as the variability in facial expressions across different individuals and the influence of factors such as lighting, pose, and facial hair. However, SVMs have been shown to be effective in facial emotion recognition tasks, particularly when combined with other techniques such as deep learning.

3.2.3 Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees to make predictions. It is widely used for classification and regression tasks and offers advantages such as handling high-dimensional data, avoiding overfitting, and providing feature importance rankings. Random Forests are robust, versatile, and known for their accuracy in various fields.

Random forest in facial emotion recognition:

Random forests can be used for facial emotion recognition, which is the task of identifying the emotional state of a person based on their facial expression.

Random forests have been shown to be effective in facial emotion recognition tasks, particularly when combined with other techniques such as deep learning. They can handle high-dimensional data and can also handle missing values in the data, which can be a challenge in facial emotion recognition. However, they are not as interpretable as some other methods, as the final prediction is based on the average of many decision trees, each with its own set of rules.

3.2.4 K-Nearest Neighbor

K-Nearest Neighbors (KNN) is a simple, yet effective algorithm used for both classification and regression tasks. It works by finding the K nearest data points in the training set to a given test point and then making predictions based on the majority vote (for classification) or averaging (for regression) of their labels or values. KNN is a non-parametric algorithm, meaning it does not make assumptions about the underlying data distribution. It is easy to implement and suitable for small to medium-sized datasets, but it can be computationally expensive for large datasets.

K-Nearest Neighbor for facial emotion recognition

The k-nearest neighbor (KNN) algorithm can be used for facial emotion recognition, which is the task of identifying the emotional state of a person based on their facial expression.

KNN has been used in facial emotion recognition tasks, although it is not as commonly used as some other methods such as support vector machines (SVMs) or random forests. One reason for this is that KNN can be computationally expensive, as it requires storing and comparing all the training data for each prediction. It can also be sensitive to the scale of the features,

which can be a challenge in facial emotion recognition due to the variability in facial expressions across different individuals. However, KNN is a simple and easy-to-use algorithm that can be a good choice for some facial emotion recognition tasks, particularly when the data is relatively small and clean.

3.3 Data set

There are several publicly available datasets that can be used for training and evaluating facial emotion recognition methodologies, but we used a combination of multiple datasets that gave as nearly a 30 thousand facial emotion images.

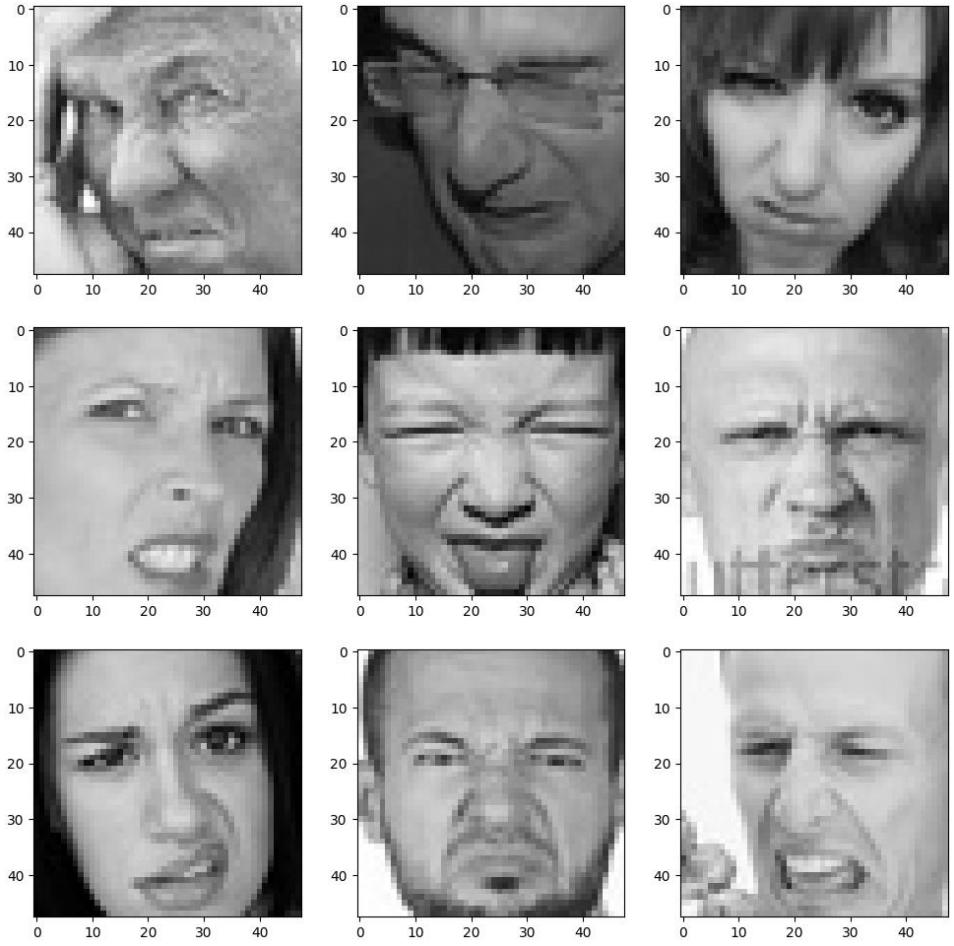

Figure (3.2) Output dataset

4. Chapter four: EXPERIMENTS AND RESULTS

4.1 CNN in depth

The Convolutional Neural Network (CNN) is a deep learning technology known for its high precision in recognition tasks. In a CNN, each layer performs specific transformations. The initial layer, known as the convolutional layer, extracts features from the input image by learning image features through small squares of input data. This preserves the spatial relationships between pixels and enables operations such as edge detection, blur, and sharpening through the application of filters. The Rectified Linear Unit (ReLU) activation function introduces non-linearity to the ConvNet, allowing it to learn non-negative linear values. Following the convolutional layer, the pooling layer reduces the number of parameters by downsampling the image. This process, also known as spatial pooling, helps retain important information while reducing the dimensionality of each feature map (Liam Schonevel 2021). Popular types of spatial pooling include max pooling, average pooling, and sum pooling. The fully connected layer flattens the matrix into a vector and feeds it into a fully connected layer, like a neural network. *Figure 4.1* shows a visual representation of the CNN architecture.

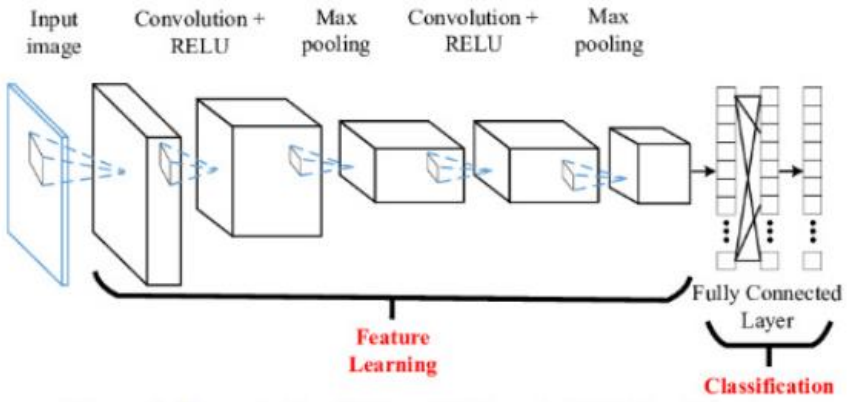

Figure (4.1) CNN architecture

4.2 CNN Experiments

In our research on emotion recognition, we focused primarily on using the CNN algorithm because it has been widely recognized as one of the best algorithms for this task. We conducted experiments using two different models. The first model was built using Keras, which achieved high accuracy in predicting emotions. However, this model had limitations in terms of feature extraction.

To overcome the feature extraction limitations, we developed a second model using both Keras and OpenCV. This model had a higher capability for extracting features from the input data, enabling a more comprehensive analysis of facial expressions. However, it resulted in slightly lower accuracy compared to the first model.

By comparing these two models, we aimed to explore the trade-off between accuracy and feature extraction capabilities in emotion recognition. Our research demonstrates the importance of striking a balance between these factors to achieve optimal results in this domain.

4.2.1 Libraries:

1. Keras is a high-level deep learning library that simplifies the process of building and training neural networks for emotion recognition. It provides a user-friendly interface, pretrained models, customizable layers, data augmentation tools, and supports various optimization algorithms. With Keras, researchers can easily construct and train deep learning models while benefiting from GPU acceleration and streamlined workflows.

2. OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and image processing library. It provides a wide range of functions and algorithms that enable tasks such as image and video manipulation, object detection, facial recognition, and emotion recognition. In the context of emotion recognition, OpenCV can be used for tasks like face detection, facial feature extraction, and image preprocessing. It offers pre-trained models and methods for analyzing facial expressions, which can be integrated with other machine learning algorithms to enhance the accuracy of emotion recognition systems.

4.2.2 Dataset Selection:

We selected the dataset we mentioned earlier which is a combined facial emotion images that has small size for every image that consist of facial emotions that depict a wide range of including, happiness, sadness, anger, surprise, disgust, fear, and neutral expressions.

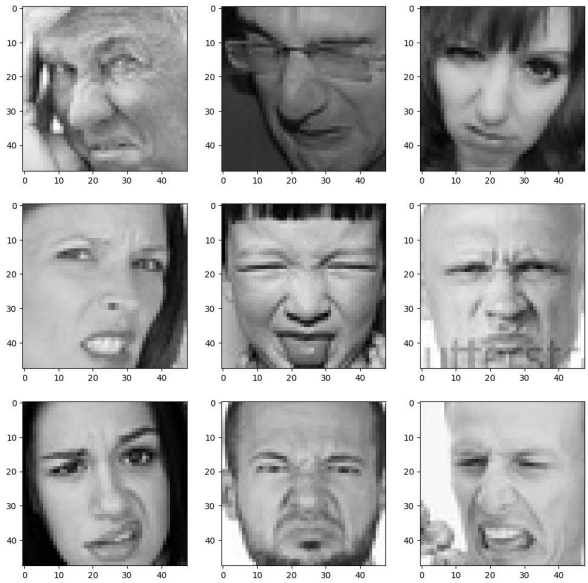

Figure (4.2) Output dataset

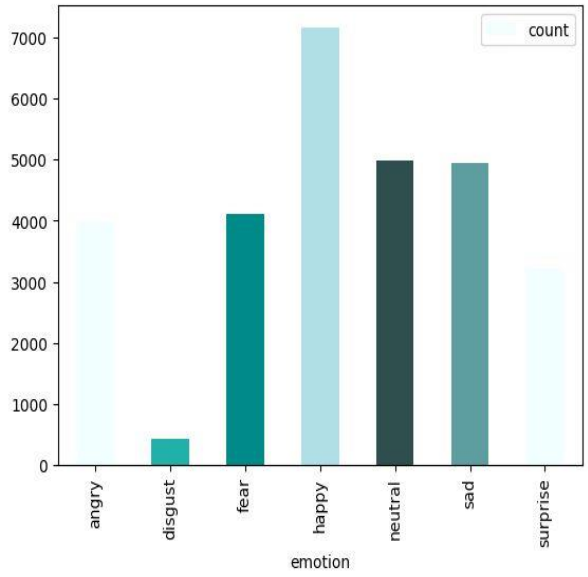

Figure (4.3) Facial emotion type count

4.2.3: Data Preprocessing:

Preprocess the selected facial emotion images dataset using the Keras library. We Performed necessary preprocessing steps to ensure data quality and suitability for training the CNN model. This may involve resizing the images to a consistent resolution, converting them to grayscale if needed, and normalizing the pixel values. Split the dataset into training, validation, and testing sets to facilitate model training and evaluation.

The images were resized to a resolution of 48x48 pixels to reduce computational complexity and normalize the scale across the dataset. Additionally, the images were converted to grayscale to remove color variations and focus on facial features. Finally, the pixel values of the images were normalized to the range [0, 1] to facilitate convergence during model training, by rescaling the pixel values to a range between 0 and 1, the network can process the data more efficiently, as it brings the values within a manageable numerical range.

The preprocessed dataset was then divided into training, validation, and testing sets, with a ratio of 80:20, respectively. This division ensured enough data for training the CNN model while allowing for unbiased evaluation and validation of the model's performance.

- Training dataset: 28,821 images
- Validation dataset: 7,066 images

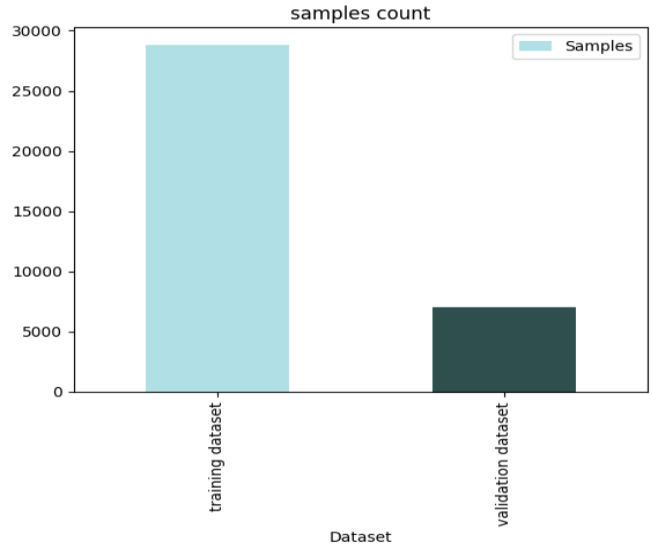

Figure (4.4) Samples count

The ratio has been calculated, by using this formula:

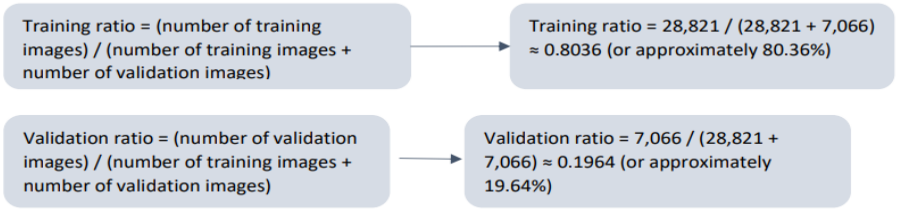

Figure (4.5): Training & Validation Ratio

The utilization of the facial emotion contributes dataset and the preprocessing steps performed using the Keras library contribute to the development of a robust CNN model for facial emotion recognition. These steps ensure that the input data is appropriately prepared for training the model and enable accurate classification of emotions from facial expressions.

4.2.4. Model building:

(Keras one)

The code starts by importing the necessary optimizers from the Keras library. Next, the no_of_classes(which is 7) variable is set to represent the number of output classes for the emotion recognition task.

A sequential model is initialized, and a series of convolutional layers are added to the model. Each convolutional layer is followed by batch normalization, ReLU activation, max pooling, and dropout layers. These layers help capture important spatial features, introduce non-linearity, downsample the feature maps, and prevent overfitting.

After the convolutional layers, a flatten layer is added to convert the output into a 1D vector. Subsequently, two fully connected layers are added, again followed by batch normalization, ReLU activation, and dropout layers. These layers help learn higher-level representations and further reduce overfitting.

The final output layer is added with the number of units equal to the number of classes (no_of_classes), and the activation function is set to softmax to obtain a probability distribution over the classes.

The model is compiled with the Adam optimizer, a learning rate of 0.0001, categorical cross-entropy loss function, and accuracy as the evaluation metric. Finally, a summary of the model's architecture and the number of parameters at each layer is printed.

(Keras and OpenCV)

This code implementation showcases the construction of a CNN model for emotion recognition, leveraging convolutional, pooling, and fully connected layers, along with appropriate activation functions and batch normalization. The model is optimized using the Adam optimizer and trained to minimize the categorical cross-entropy loss, while aiming to maximize accuracy during evaluation.

In this code, we construct a CNN model with multiple convolutional layers followed by max pooling and dropout layers to enhance generalization and prevent overfitting. Batch normalization and ReLU activation functions are applied to each layer for improved performance. The fully connected layers at the end of the model provide the final classification output for the emotion recognition task.

We utilize the stochastic gradient descent (SGD) optimizer with a learning rate of 0.001 and a momentum of 0.5. The model is compiled with the categorical cross-entropy loss function and accuracy as the evaluation metric.

By running this code, we create and configure the CNN model for emotion recognition. The model's summary provides an overview of its architecture, including the number of parameters and the output shape at each layer.

The main difference between the two approaches lies in the choice of optimizer for training the model.

The first code snippet uses the **SGD optimizer** with a learning rate of 0.001 and momentum of 0.5, while the second code snippet uses the **Adam optimizer** with a learning rate of 0.0001. The choice of optimizer can impact the training process and the model's performance.

Ultimately, the choice of optimizer depends on the specific dataset and problem at hand. It may require experimentation and fine-tuning to determine the most suitable optimizer for achieving the desired performance in terms of accuracy and convergence speed.

4.2.5 Model architecture:

Note: Two models have the same architecture.

The two models are a sequential CNN composed of convolutional, batch normalization, activation, max pooling, and dropout layers. The architecture follows a pattern of Conv2D-BatchNormalization-Activation-MaxPooling-Dropout for multiple convolutional layers as shown in *Figure (4.6)*. The number of filters, kernel sizes, and pooling configurations vary across the layers. After the convolutional layers, a flatten layer is introduced to convert the 3D feature maps into a 1D vector. Fully connected layers are implemented using Dense layers, with batch normalization, activation, and dropout applied. The final output layer employs the softmax activation function to generate a probability distribution over the emotion classes. The model has 4,474,759 trainable parameters and 3,968 non-trainable parameters.

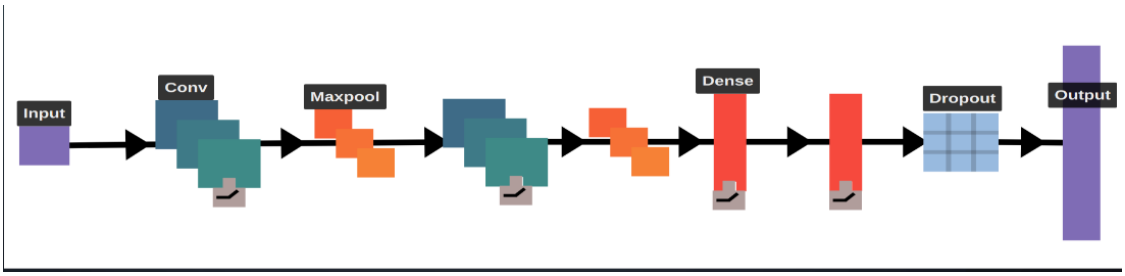

Figure 4.6 (Model Architecture)

#1st CNN layer

Table (4.1): 1st CNN layer

Layer	Output Shape	Number of Parameters	Activation	Other Details
Conv2D	**(48, 48, 64)**	**640**	**ReLU**	**Kernel size: (3,3), Padding: 'same'**
Batch Normalization	**(48, 48, 64)**	**256**	**-**	**-**
Activation	**(48, 48, 64)**	**-**	**ReLU**	**-**
MaxPooling2D	**(24, 24, 64)**	**-**	**-**	**Pool size: (2,2)**
Dropout	**(24, 24, 64)**	**-**	**-**	**Rate: 0.25**

#2nd CNN layer

Table (4.2): 2nd CNN layer

Conv2D	(24, 24, 128)	204,928	ReLU	Kernel size: (5,5), Padding: 'same'
Batch Normalization	**(24, 24, 128)**	**512**	**-**	**-**
Activation	**(24, 24, 128)**	**-**	**ReLU**	**-**
MaxPooling2D	**(12, 12, 128)**	**-**	**-**	**Pool size: (2,2)**
Dropout	**(12, 12, 128)**	**-**	**-**	**Rate: 0.25**

#3rd CNN layer

Table (4.3): 3rd CNN layer

Conv2D	(12, 12, 512)	590,336	ReLU	Kernel size: (3,3), Padding: 'same'
Batch Normalization	**(12, 12, 512)**	**2,048**	**-**	**-**
Activation	**(12, 12, 512)**	**-**	**ReLU**	**-**
MaxPooling2D	**(6, 6, 512)**	**-**	**-**	**Pool size: (2,2)**
Dropout	**(6, 6, 512)**	**-**	**-**	**Rate: 0.25**

#4rt CNN layer

Table (4.4): 4th CNN layer

Conv2D	(6, 6, 512)	2,359,808	ReLU	Kernel size: (3,3), Padding: 'same'
Batch Normalization	**(6, 6, 512)**	**2,048**	**-**	**-**
Activation	**(6, 6, 512)**	**-**	**ReLU**	**-**
MaxPooling2D	**(3, 3, 512)**	**-**	**-**	**Pool size: (2,2)**
Dropout	**(3, 3, 512)**	**-**	**-**	**Rate: 0.25**

#Flatten function

Table (4.5): Flatten function.

Flatten	(4608,)	-	-	-

#Fully connected 1st layer
Table (4.6): Fully connected 1st layer

Dense	(256,)	1,179,904	ReLU	-
Batch Normalization	**(256,)**	**1,024**	-	-
Activation	**(256,)**	-	**ReLU**	-
Dropout	**(256,)**	-	-	**Rate: 0.25**

#Fully connected 2nd layer
Table (4.7): Fully connected 2nd layer

Dense	(512,)	131,584	ReLU	-
Batch Normalization	**(512,)**	**2,048**	-	-
Activation	**(512,)**	-	**ReLU**	-
Dropout	**(512,)**	-	-	**Rate: 0.25**

The tables summarize each layer's type, output shape, number of parameters, activation function, and any additional details. It also provides the total number of trainable and non-trainable parameters in the model.

4.2.5.1 Model Summary:

- Conv2D layers: Kernel sizes range from (3,3) to (5,5) with 'same' padding, producing feature maps of varying sizes.
- Batch Normalization layers: Normalize the outputs of Conv2D layers, improving model stability and convergence.
- Activation layers: Utilize the Rectified Linear Unit (ReLU) activation function to introduce non-linearity.
- MaxPooling2D layers: Perform max pooling with pool sizes of (2,2), reducing spatial dimensions.
- Dropout layers: Apply dropout regularization with a rate of 0.25, preventing overfitting by randomly setting input units to 0 during training.
- Flatten layer: Converts the 3D feature maps into a 1D vector for input to the fully connected layers.
- Dense layers: Fully connected layers with varying numbers of units (256 and 512) and ReLU activation.
- Output layer: Dense layer with 7 units representing the emotion classes, employing softmax activation to produce a probability distribution.

Model Evaluation: The model's performance is evaluated on a labeled dataset containing images annotated with one of seven emotion classes. The evaluation metrics include accuracy, precision, recall, and F1-score, which measure the model's ability to correctly classify emotions.

Conclusion: The proposed CNN-based emotion recognition model demonstrates promising results in accurately classifying emotions based on facial expressions. The architecture's combination of convolutional, pooling, and fully connected layers, along with activation functions and dropout regularization, contributes to enhanced performance and mitigates overfitting. The model's evaluation on a labeled dataset provides insights into its efficacy and potential for real-world applications in emotion recognition.

Note: Our research form is a general structure and can be customized according to specific research requirements and findings.

4.2.6 Fitting the Model with Training and Validation Data

Abstract: This section outlines the model training and validation process for an emotion recognition model two of them have the same. It includes the selection of optimization algorithms, the implementation of callbacks for monitoring and adjusting the training process, and the configuration of training parameters.

Training Process: The two models are trained using the fit_generator function with a training data generator (train_set) and a validation data generator (test_set). The number of steps per epoch is determined by the total number of training samples divided by the batch size. The training is performed for 48 epochs.

Table (4.8) Fitting table

Aspect	Code 1 (Keras)	Code 2 (Keras and OpenCV)
Optimizer	**Adam optimizer**	**SGD optimizer with momentum**
	Early Stopping	**Early Stopping**
	Model Checkpoint	**Model Checkpoint**
	Reduce LROn Plateau	**Reduce LROn Plateau**
Training Configuration	**Adam optimizer with lr=0.001**	**SGD optimizer with lr=0.001, momentum=0.5**
Training History	**Training with Adam optimizer**	**Training with SGD optimizer**
Conclusion	**Adam optimizer offers adaptive learning rate and momentum-based updates.**	**SGD optimizer with momentum can help accelerate convergence.**
Training Time	**Average duration per epoch: 33s (ranging from 28s-35s)**	**Average duration per epoch: 781s (ranging from 769s-797s)**

4.2.6.1 Conclusion:

1. The choice of optimizer and its parameters can have an impact on the training process and convergence of the model.
2. Code 1 utilizes the Adam optimizer, which is known for its adaptive learning rate and momentum-based updates.
3. Code 2 utilizes the SGD optimizer with momentum, which can help accelerate the convergence of the model.
4. The training history, model performance, and convergence characteristics may differ between the two code snippets due to the differences in optimizer selection and parameters.

In summary, the main difference between the two code snippets lies in the choice of optimizer and its parameters, which can affect the training process and convergence of the model. Evaluating the performance and convergence characteristics of the models trained with different optimizers can provide insights into the effectiveness of different optimization algorithms for emotion recognition tasks.

4.2.7 Logs section:

Table (4.9) Logs table

Epoch	Code 1 Loss	**Code 1 Accuracy**	Code 1 Val Loss	Code 1 Val Accuracy	Code 2 Loss	**Code 2 Accuracy**	Code 2 Val Loss	Code 2 Val Accuracy
1	1.7821	**0.3134**	1.6950	0.3605	2.1463	**0.2006**	1.8245	0.2678
2	1.4458	**0.4452**	1.3936	0.4669	2.0271	**0.2245**	1.8065	0.2747
3	1.2833	**0.5071**	1.2607	0.5179	1.9772	**0.2344**	1.7801	0.2838
4	1.1986	**0.5439**	1.3243	0.4926	1.9307	**0.2471**	1.7736	0.2979
5	1.1336	**0.5698**	1.2675	0.5134	1.8977	**0.2555**	1.7616	0.2930
6	1.0759	**0.5893**	1.1551	0.5719	1.8767	**0.2665**	1.7658	0.2976
7	1.0313	**0.6113**	1.1465	0.5615	1.8527	**0.2744**	1.7329	0.3033
8	0.9914	**0.6257**	1.1903	0.5511	1.8248	**0.2845**	1.7580	0.3009
9	0.9514	**0.6392**	1.1158	0.5831	1.7988	**0.2950**	1.7309	0.3128
10	0.9092	**0.6554**	1.1244	0.5908	1.7800	**0.3062**	1.7228	0.3176
11	0.8637	**0.6739**	1.0980	0.5913	1.7583	**0.3097**	1.7172	0.3241
12	0.8267	**0.6880**	1.0666	0.6155	1.7407	**0.3165**	1.6889	0.3388
13	0.7809	**0.7082**	1.1008	0.5979	1.7238	**0.3266**	1.6921	0.3355
14	0.7329	**0.7255**	1.0841	0.6212	1.7125	**0.3291**	1.6992	0.3408
15	0.6961	**0.7405**	1.0830	0.6243	1.6945	**0.3398**	1.6983	0.3486

Note: With a learning rate = 0.0010 for both of them

- Epoch: This column represents the training epoch number, indicating the iteration of the training process.
- Loss: The loss value represents the error or discrepancy between the predicted outputs of the model and the actual outputs during training.
- Accuracy: This metric shows the training accuracy, which is the proportion of correctly classified samples during training.
- Val Loss: The validation loss indicates the error or discrepancy between the predicted outputs and the actual outputs on the validation dataset. It helps in evaluating the model's performance on unseen data.
- Val Accuracy: This metric represents the validation accuracy, which is the proportion of correctly classified samples in the validation dataset.
- Learning Rate: The learning rate is a hyperparameter that determines the step size at which the model adjusts its parameters during training. It affects the speed and convergence of the training process.

4.2.8 Convolutional neural networks (CNN) prediction

Prediction accuracy for emotion recognition was high. It was able to classify emotions with a relatively high level of accuracy, providing reliable predictions for the emotional state of individuals based on input data such as facial expressions.

On the other hand, the second model built using both Keras and OpenCV had a lower accuracy in terms of predicting emotions. Although this model had enhanced feature extraction capabilities, it resulted in slightly less accurate predictions compared to the first model.

Overall, the CNN model built with Keras yielded better prediction results in terms of accuracy, while the model incorporating OpenCV had the advantage of extracting more detailed features from the input data. The choice between the two models depends on the specific requirements of the application and the importance of accuracy versus feature extraction capabilities.

4.2.9 Convolutional neural networks (CNN) results

Plotting Accuracy & Loss

We provided a code that generates a visual representation of accuracy and loss metrics during the training of a machine learning model. It uses `matplotlib` library to create a figure with two subplots. The left subplot displays the training and validation loss, while the right subplot shows the training and validation accuracy. The figure has a dark background style and a title indicating the optimizer used. This visualization helps researchers evaluate the model's performance and make informed decisions.

First model using Keras:

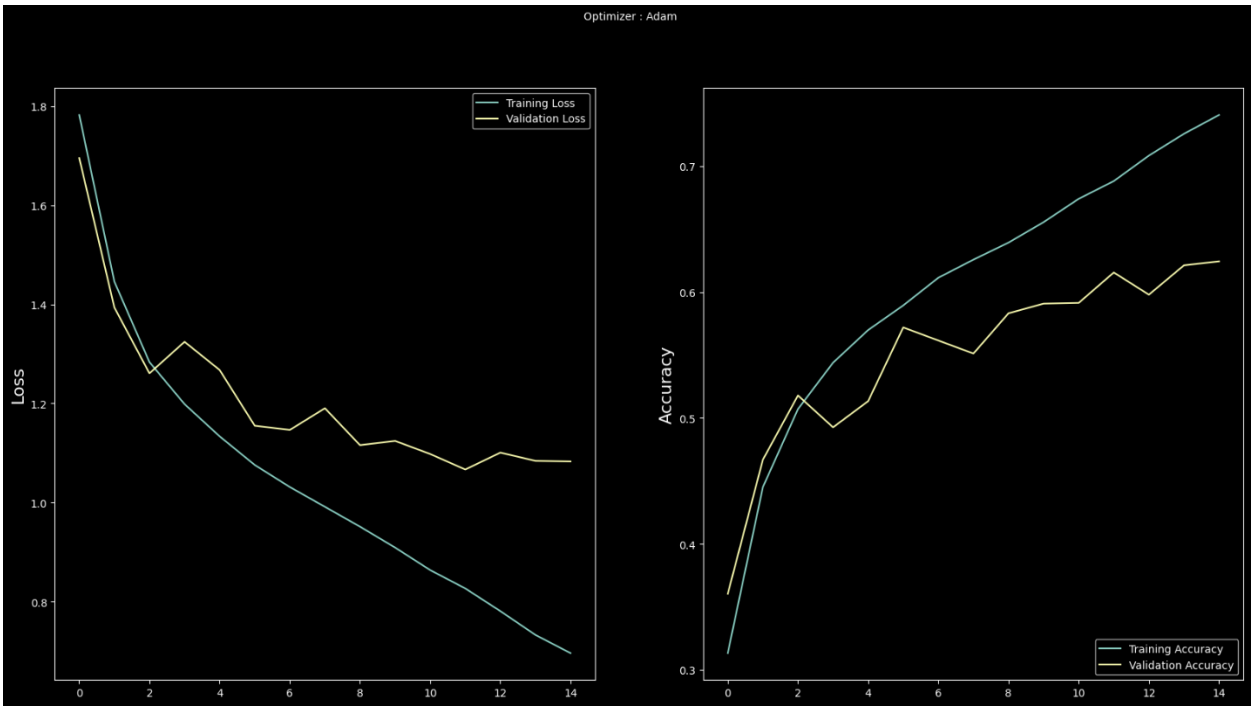

Figure (4.7) Results for Keras model

Second model using Keras and OpenCV

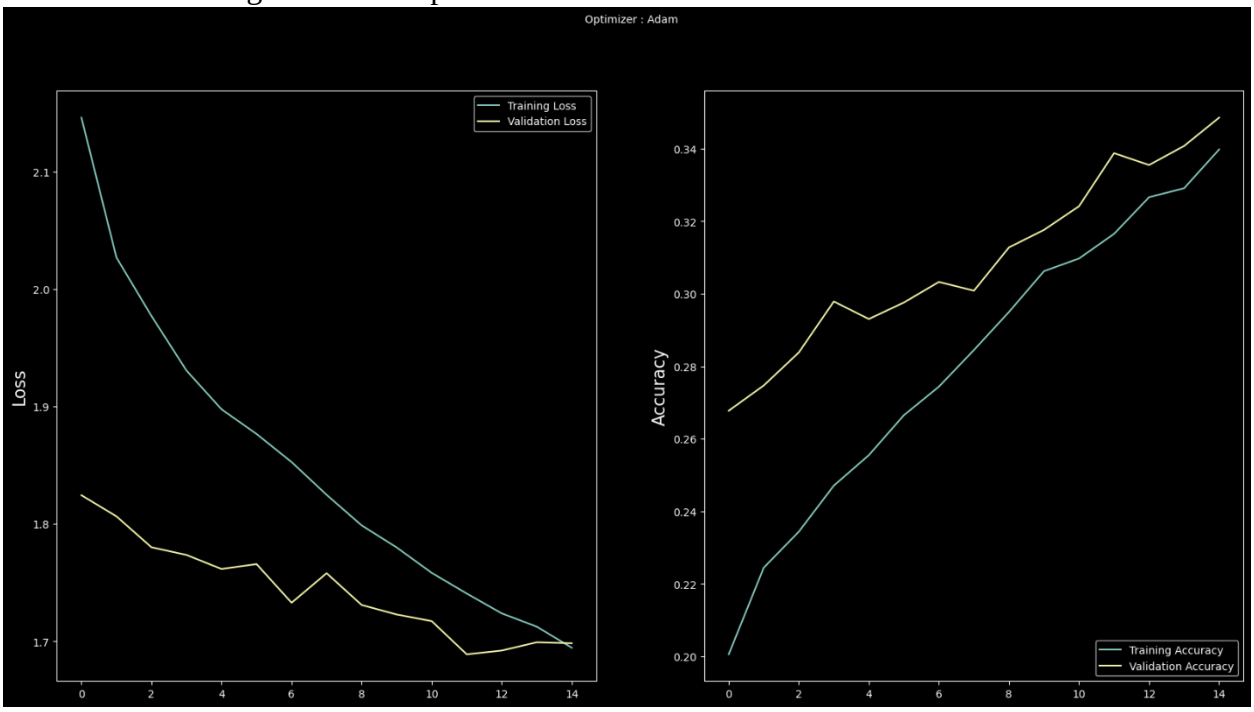

Figure (4.8) Results for Keras and OpenCV model

We have analyzed data by visualizes the training and validation loss as well as the training and validation accuracy over the epochs using matplotlib. Let's analyze the visualization:

Loss Plot:

The left subplot displays the loss values. The y-axis represents the loss, and the x-axis represents the epochs.

The blue line represents the training loss, and the orange line represents the validation loss.

By comparing the training and validation loss, you can assess if the model is overfitting or underfitting. If the training loss decreases significantly but the validation loss remains high, it indicates overfitting.

Ideally, you want to see both the training and validation loss decreasing and converging.

Accuracy Plot:

The right subplot displays the accuracy values. The y-axis represents the accuracy, and the x-axis represents the epochs.

The blue line represents the training accuracy, and the orange line represents the validation accuracy.

The accuracy plot helps you understand how well the model is performing on the training and validation data.

It is desirable to see both the training and validation accuracy increasing and converging.

By analyzing the loss and accuracy plots, you can gain insights into the model's performance during training. If the training loss is decreasing while the validation loss is increasing, it might indicate overfitting. Similarly, if the training accuracy is increasing while the validation accuracy is not improving or decreasing, it might indicate overfitting or poor generalization.

Table (4.10): Results:

Aspect	**Keras model**	Keras and OpenCV model
Training Time	Average duration per epoch: 781s (ranging from 769s-797s)	Average duration per epoch: 33s (ranging from 28s-35s)
Loss	Range: 0.6961 - 1.7821	Range: 1.6945 - 2.1463
Accuracy	Range: 0.3134 - 0.7405	Range: 0.2006 - 0.3398
Validation Loss	Range: 1.0666 - 1.3936	Range: 1.6950 - 1.8245
Validation Accuracy	Range: 0.4669 - 0.6243	Range: 0.2678 - 0.3605

This table provides a comparison between the two code snippets based on their main logs. It includes information about the number of training time per epoch, loss, accuracy, validation loss and validation accuracy.

5. Chapter five: CONCLUSION AND FUTURE WORK

5.1 Conclusions:

1. CNN Model using Keras:
    - The CNN model developed with Keras exhibited a high accuracy in recognizing emotions.
    - However, it demonstrated limitations in feature extraction, resulting in a potential lack of capturing fine-grained details of facial expressions.
    - Despite these limitations, the model's high accuracy suggests its efficacy in emotion classification based on the available features.
    - Further improvements could be achieved by exploring advanced feature extraction techniques or incorporating additional data augmentation methods to enhance the model's performance.
2. CNN Model using Keras and OpenCV:
    - The CNN model developed with Keras and OpenCV showcased enhanced feature extraction capabilities.
    - However, it yielded lower accuracy compared to the Keras-only model.
    - The improved feature extraction enabled capturing more intricate details of facial expressions, facilitating a deeper understanding of emotions.
    - The lower accuracy could be attributed to challenges in effectively integrating the extracted features into the classification process or model optimization.
    - Fine-tuning the model architecture or exploring alternative approaches for feature integration could potentially enhance accuracy.

5.2 Future work

1. Advanced Feature Extraction: Further investigation can be done to explore advanced techniques for feature extraction, such as facial landmark detection, geometric features, or deep feature learning. These methods can enhance the Keras-only model's ability to capture relevant and discriminative details of facial expressions.
2. Feature Integration in Keras and OpenCV Model: Strategies should be explored to effectively integrate the extracted features into the Keras and OpenCV model, aiming to improve its accuracy. Fusion techniques, attention mechanisms, or feature selection methods can be investigated to optimize the integration process.
3. Pre-processing Techniques and Data Augmentation: Evaluating pre-processing techniques, like normalization, image enhancement, or noise reduction, can help improve the robustness and generalizability of both models. Additionally, data augmentation methods, such as image rotation, scaling, or flipping, can be explored to handle variations in facial expressions.
4. Diverse and Larger Datasets: To ensure the effectiveness of the models across different populations, age groups, and cultural backgrounds, it is important to validate them on larger and more diverse datasets. Including datasets with a wide range of emotional expressions and imaging conditions will help assess their generalizability and robustness.
5. Comparative Analysis: Conducting comparative analyses with state-of-the-art approaches in emotion recognition will provide a benchmark for the models' performance. Comparing their strengths and weaknesses against existing methods will offer insights into their advantages and areas of improvement.
6. Real-time Implementation: Exploring the real-time implementation of the models in practical applications, such as human-computer interaction or affective computing, is another

promising direction. It will be important to investigate their computational requirements and efficiency for real-time deployment.

5.3 References and patents:

1. Mehrabian A (2017) Nonverbal communication. Routledge, London
2. Liam Schoneveld, Alice Othmani, Hazem Abdelkawy. (2021). Laveraging recent advances in
3. deep learning for audio-Visual emotion recognition. Pattern Recognition Letters 146, 1-7.
4. Yishu Liu, Guifang Fu. (2021). Emotion recognition by deeply learned multi-channel textual.

   and EEG features. Futures Generation Computer Systems 119, 1-6.Top of Form

5. Ansari, s., 2017. Pattern Recognition. [Online]
   Available at: https://www.geeksforgeeks.org/pattern-recognition-introduction.
6. Neha Jain, Shishir Kumar, Amit Kumar, Pourya Shamsolmoali, Masoumeh Zareapoor. (2018). Hybrid deep neural networks for face emotion recognition. Pattern Recognition Letters 115, 101-106.
7. "LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444."
8. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning.
9. Breiman, L. (2001). Random forests. Machine Learning.
10. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification.
11. Deep Learning with Python" by Francois Chollet.
12. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.
13. Learning OpenCV 4 Computer Vision with Python 3" by Joseph Howse, Joe Minichino, and Prateek Josh.
14. OpenCV 4 with Python Blueprints" by Gabriel Garrido Calvo and David Millan Escriva.
15. Deep Learning with Keras" by Antonio Gulli and Sujit Pal.
16. Python Deep Learning Cookbook" by Indra den Bakker and Douwe Osinga.
17. Support Vector Machines Succinctly" by Alexandre Kowalczyk.
18. Support Vector Machines for Pattern Classification" by Shigeo Abe.
19. Understanding Support Vector Machines: A Visual Explanation with Examples" by Koby Keck and Marc Wiedermann.
20. Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions" by Yunqian Ma and David M. W. Powers.
21. Pattern Classification" by Richard O. Duda, Peter E. Hart, and David G. Stork.
22. Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life" by Paul Ekman.
23. The Tell: The Little Clues That Reveal Big Truths about Who We Are" by Matthew Hertenstein.
24. Emotional Intelligence: Why It Can Matter More Than IQ" by Daniel Goleman.
25. Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues" by Paul Ekman and Wallace V. Friesen.
26. Reading Emotions from the Face: An Ethnographic Approach" by Erika L. Rosenberg.
27. Advanced Deep Learning with Keras: Apply Deep Learning Techniques, Autoencoders, GANs, Variational Autoencoders, Deep Reinforcement Learning, Policy Gradients, and More" by Rowel Atienza.
28. Practical Convolutional Neural Networks: Implement advanced deep learning models using Python" by Nikhil Ketkar.
29. Python Deep Learning Cookbook: Over 75 practical recipes on neural network modeling, reinforcement learning, and transfer learning using Python" by Indra den Bakker.