

DESIGN OF MUSIC RECOMMENDATION SYSTEM BASED ON FACIAL RECOGNITION USING MINI-XCEPTION CNN

A PROJECT REPORT

Submitted by

**CHANDAN SINGH [RA1911031010076]
V. HIMAYANTH [RA1911031010088]**

Under the Guidance of

Dr. B. BALAKIRUTHIGA

(Assistant Professor, Department of Networking and Communications)

in partial fulfillment of the requirements for the degree of

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in INFORMATION
TECHNOLOGY**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603 203**

MAY 2023

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY****KATTANKULATHUR – 603 203****BONAFIDE CERTIFICATE**

Certified that this B.Tech project report titled “**Design of Music Recommendation System Based on Facial Recognition using Mini-Xception CNN**” is the bonafide work of Mr. CHANDAN SINGH [Reg. No.: RA1911031010076] and Mr. V. HIMAYANTH [Reg. No. RA1911031010088] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. B.Balakiruthiga**SUPERVISOR**

Associate Professor

Department of Networking and
Communications**Dr. ANNAPURANI K****HEAD OF THE DEPARTMENT**Department of Networking and
Communications**SIGNATURE OF INTERNAL
EXAMINER****SIGNATURE OF EXTERNAL
EXAMINER**



Department of Networking and Communications
SRM Institute of Science & Technology
Own Work* Declaration Form

Degree/ Course : B.Tech in Computer Science and Engineering with specialization in Information Technology

Student Names : CHANDAN SINGH, V. HIMAYANTH

Registration Number : RA1911031010076, RA1911031010088

Title of Work : Design of Music Recommendation System Based on Facial Recognition using Mini-Xception CNN

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website
-

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my own work, except where indicated by referring, and that I have followed the good academic practices noted above.

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his valuable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work. We are incredibly grateful to our Head of the Department, **Dr. Annapurani Panaiyappan.K** Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators **Dr. Thenmozhi M**, Professor, and Panel Head, **Dr. Balachander T**, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support. We register our immeasurable thanks to our Faculty Advisor, **Dr. Supraja P**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. B. Balakiruthiga**, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under his/her/their mentorship. She provided me with the freedom and support to explore the research topics of my interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

CHANDAN SINGH [Reg No:RA1911031010076]

V. HIMAYANTH [Reg No:RA1911031010088]

ABSTRACT

Due to the emerging developments in Artificial Intelligence and Machine Learning Technologies, various prediction systems are being developed based on human emotions and real time aspects of human psychology as well. Facial recognition system is one such mechanism which is the most vibrant strategy used for predicting human emotions. It is extensively applied in surveillance systems, fault identification and other security related aspects. Based on the human emotions researchers have already proposed several music recommendation systems. This paper aims to propose a Facial recognition-based music recommendation system to treat the psychology patients. This helps to recover the patients from mental stress, anxiety, and depression. The suggested method aims to take into account the limitations of the face recognition system in current frameworks, such as the requirement to lower the processing delay for deep feature extraction and the necessity to design a Mini exception technique based on Deep Convolutional Neural Network (DCNN) architecture. The FER- 2013 image dataset, which consists of 35000 face photos with automated labelling is considered. It is used to determine how well the proposed approach would detect the various emotion classes. In comparison to other states of methods, the Mini exception technique utilised in CNN layers acts as a lightweight system. The proposed solution has a 92% accuracy rate and removes the barrier between the current frameworks. The suggested music is taken from a music database and then further mapped in accordance with the algorithm's output.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ACKNOWLEDGEMENTS	iv
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Motivation of the project	3
	1.2 Proposed Work	4
2	LITERATURE SURVEY	6
3	PROPOSED ALGORITHM	11
	3.1 Image Classification	12
	3.2 CNN Architecture	13
	3.3 Workflow of music recommendation system	16
	3.4 System Architecture	19
4	IMPLEMENTATION	20
	4.1 Face Detection	20
	4.2 Music Mapping	21
5	SYSTEM REQUIREMENT	24
	5.1 Hardware Requirement	24
	5.2 Software Requirement	24
	5.3 Software Environment	24
6	SYSTEM STUDY	28
	6.1 Feasibility Study	28
	6.1.1 Economical Feasibility	28
	6.1.2 Technical Feasibility	29
	6.1.3 Social Feasibility	30

7	SYSTEM TESTING	31
	7.1 Types of Tests	31
	7.1.1 Unit Testing	31
	7.1.2 Integration Testing	31
	7.1.3 Functional Testing	32
	7.1.4 System Testing	32
	7.1.5 White Box Testing	32
	7.1.6 Black Box Testing	32
	7.2 Unit Testing	33
	7.3 Integration Testing	33
	7.4 Acceptance Testing	34
8	RESULTS AND DISCUSSION	35
9	CONCLUSION	37
	REFERENCE	38
	APPENDIX	41
	PLAGIARISM REPORT	62
	PROOF OF PUBLICATION	66

LIST OF FIGURES

1.1 Emotion Wheel	1
1.2 Different types of emotions	2
1.3 Basic types of emotions	2
1.4 Neural networks	3
1.5 Existing applications for songs	4
3.1 Haar Cascade method of image detection	11
3.2 Mini-Xception Architecture	12
3.3 Traditional Convolutional layer	13
3.4 Depth wise Separable Convolution layers	14
3.5 Max Pooling Function	15
3.6 Global Pooling Function	16
3.7 Sequence Diagram of music recommendation System	17
3.8 Happy Faces	18
3.9 Neutral Faces	18
3.10 Angry Faces	18
3.11 System Architecture	19
4.1 Implementation and Running of our model	21
4.2 Image Classification with Accuracy Percentage	21
4.3 Image Classification with Accuracy Percentage	22
4.4 Display of phrase Playing Happy Songs	22
4.5 Emotion Classification from recorded Video	23
4.6 Song link from YouTube	23
6.1 Types of Feasibility Study	28
6.2 Economic Feasibility	29
6.3 Technical Feasibility	29
8.1 Loss and Accuracy graphs for every epoch	35
A1.1-1.2 Training the model	44

A2.1 Loading the dataset	45
A3.1 – 3.4 Facial recognition and music mapping algorithm	45-47

CHAPTER 1

INTRODUCTION

Emotions are defined by many, but not easily defined. Emotions can be defined as anything that a person feels and reflects on and are sometimes referred to as the "internal state" of an organism. Emotions are the result of cognitive changes and are related to emotions, communicative responses, and some degree of gratification and irritability. Emotions can be triggered by positive or negative cognitive changes. Emotions can also be defined in a more technical sense as "a good or bad experience associated with a particular pattern of physical activity." Emotions are responsible for many changes in the human body, as well as physiological and psychological conditions. Initially, emotion played a role in facilitating dynamic behavior, which contributed to genetic transmission, survival, and relative selection. Emotions are very important in the real world, be it health, psyche, or science. Because this area of research requires a lot of scrutiny, people all over the world have studied different aspects of emotions, including how they work and the roles they play. Research into emotions has increased significantly in recent years, with contributions from various fields such as psychology, medicine, history, sociology, and artificial intelligence. Extensive research on this subject has been fueled by a variety of theories that attempt to explain all aspects of emotions, including their origin and function. Recent research on the concept of emotion has focused on issues such as the development of stimulants and stimulants. There are a lot of emotions that are imparted on the face as it can be seen in fig 1.1- the emotion wheel and fig 1.2.



Fig. 1.1. Emotion Wheel

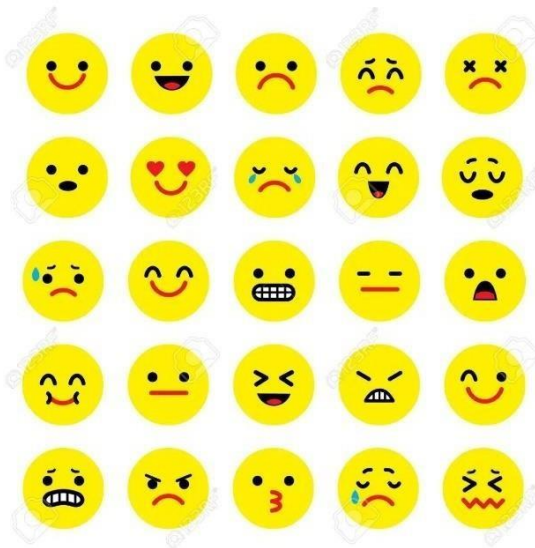


Fig. 1.2. Different types of emotions

However, feelings are categorised into the following groups: joy (Happy), excited, anger, sadness, surprise, fear, and tender as in fig 1.3.

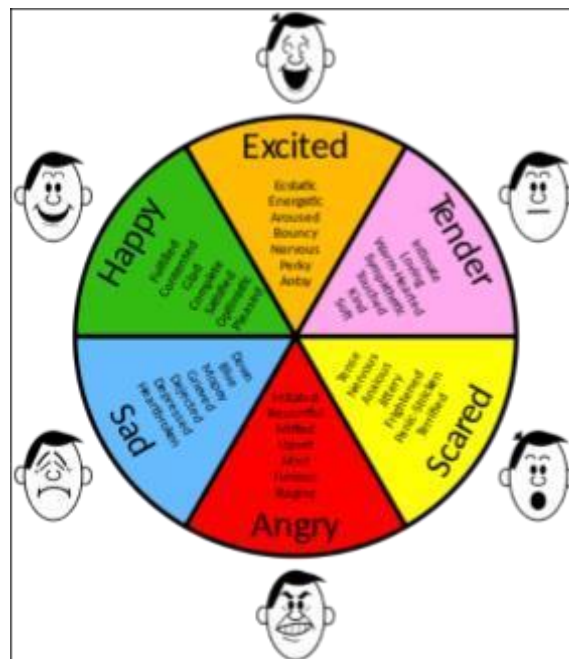


Fig. 1.3. Basic types of emotions

1.1. Motivation of the Project

Various projects are being carried out in this research area, such as recognition of facial expressions and actions, and visual scanning. However, implementations of deep learning have made great strides in the field of emotion recognition. Machine learning allows computers to exhibit eerily human-like behaviors and patterns. Examples of machine learning include learning. Deep learning technology is used in non-motorized vehicles to recognize stop signs and distinguish between pedestrians and light poles. It allows you to control your electronic devices such as phones, tablets, TVs and speakerphones with your voice. The fact that it has produced unprecedented results has attracted a lot of interest recently, and it is fully justified. You can learn to perform tasks that distinguish. The accuracy achievable by deep learning models can match or even exceed human accuracy in some cases. Model training uses neural network topologies with large amounts of tagged data and many layers. Deep learning has many potential applications including, but not limited to, automotive driving, aerospace, military, medical research, industrial automation, and electronics.

Deep neural networks represent a popular term for deep learning models because they utilize neural network topologies in many of their methodologies. The term "deep" often refers to the number of layers that make up a neural network. In contrast to traditional neural networks, which only have a couple of layers, deep neural networks may consist of as many as 150 layers that are hidden. A deep neural network is made up of the insertion layer, hidden layers, and output layers, as depicted in Figure 1.4. Deep learning models are taught with massive, labelled data sets and neural network structures that learn features directly from the data rather than having to extract them manually.

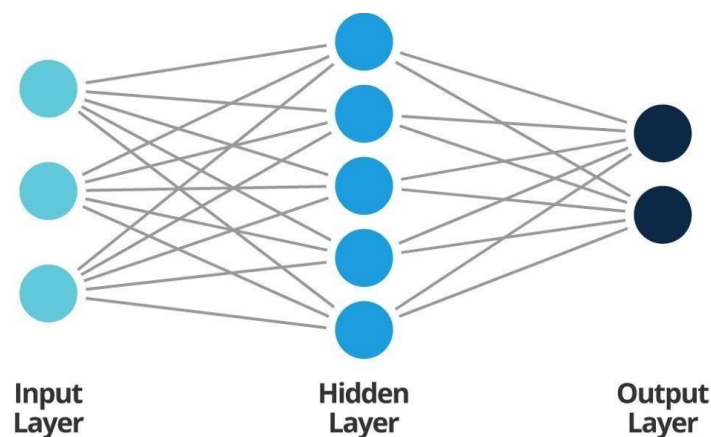


Fig. 1.4. Neural Networks

1.2. Proposed Work

Convolutional neural networks are considered as among the most well-liked and prevalent sorts of deep neural networks (CNNs or ConvNets). CNNs not only integrate learned features into the input data, but also use a 2D transformation layer, making this structure ideal for processing images that have only two dimensions. CNN no longer renders features manually, eliminating the need to use separate images. CNN spends a lot of time extracting features from photos. Not all relevant functions are pre-trained. Rather, they are read while the network is being trained on the image collection. This automatic sharing feature allows you to train detailed models that provide more accurate results on computer vision tasks such as object classification. One application of human face technology is facing classification.

It was very widely used for the purpose of emotion recognition, using the concept of convolutional neural networks as the main tool.

In today's society, we should listen to music based on our emotions. There are many applications with songs categorized by mood, but you should choose one based on your current mood. However, humans tend to be lazy, so automation has become necessary. As shown in Figure 1.5, applications such as Wynk music, Jio Saavn, and Hungama music show only three or four characteristics of music selection: motivation, dedication, joy, and party. However, there is currently no software that can automatically reproduce emotions according to people's feelings.

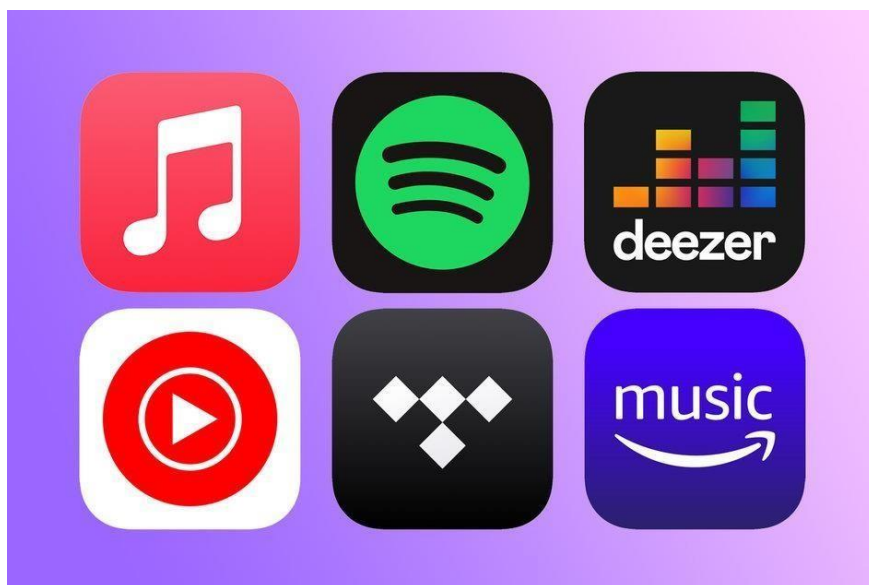


Fig. 1.5. Existing applications for songs based on emotions

We suggested the use of a sentimentality-based music suggestion system to address this issue. Multiple components make up a system for recommending music based on how people feel. Those who can identify face traits, extract features, identify emotions, and make suggestions. The suggested method was developed using the 35,000 emotion-labeled photos from the FER 2013 dataset as its foundation. The Haar Cascade facial classifier uses real-time face detection and processing. Convolutional neural network layers receive facial feature information for labelling. Here, a deep learning-based mini-xception approach is used to recognise emotions. For labelling, facial characteristics are collected and delivered to the convolutional neural network's layers. Following this, the recommender provides the user suggestions for music depending on the feeling it has detected. The framework of currently underway research related to a system of recommendations built on the13raininn is complex and it consumes an enormous amount of resources. Since time is of the simple terms, we have created an algorithm built around an efficient architecture that can quickly identify faces, extract characteristics, and classify feelings without wasting any time.

The basic structure of the framework we have suggested the algorithms applied for face identification and emotion categorization, and the system of suggestions used to suggest music to the individual are all described in this area of our study. The first section serves as an introduction, the second section gives a brief overview of the existing works, the third section gives an overview of the architecture needed to run the system we have proposed, the fourth section discusses how the system would be implemented in real-time, and the final part concludes by discussing the outcomes of our proposed system with various applications and its benefits. These sections are arranged in the following order: The first section serves as an introduction, the second section gives a brief overview of the existing works, the third chapter gives an introduction of the design elements needed to run the system we have proposed, the fourth section discusses how the system is implemented in real-time, and the fifth section concludes by discussing the outcomes of our model with various applications and its benefits.

CHAPTER 2

LITERATURE SURVEY

S In her article, S. Gilda [1] explained how she discovered suggested music utilising three separate modules. The sense of emotion component isolates moods, while the music isolation and recommendation modules provide approved songs based on song kind and psychological similarity. Recommended. These modules were used to determine which songs to recommend to users. A user's image serves as input, and different kinds of emotions are produced as output. We observed that p Gilda [1] described how to locate suggested songs by utilising three separate components in her article. The modules that segregate emotions, isolate music, and suggest associated music according to emotion similarities and song kind are all advised. These modules were used to determine which songs to recommend to users. A user's image serves as input, and different kinds of emotions are produced as output. They distinguish only the most basic emotions. Filter out all but the most basic emotions.

Recording a user's facial expressions using a webcam was identified as the most successful technique by S. Metilda [2]. A fish-face algorithm is used to classify the user's emotions. You can get recommended songs through the website. There were only two facials done, a happy facial and a sad facial. Therefore, we found that using a live webcam to recognize facial expressions was a better choice. However, this method can significantly increase the computational cost and also leads to a loss of accuracy.

Rajp p [3] eliminated geometry- and appearance-based input elements. These characteristics aided in maintaining strong facial intensity of pixels and contours. Using the Vector Support Machine (SVM) algorithm, we were able to achieve 98-100% accuracy for still images and 85-90% accuracy for real-time images. Numerous studies have been conducted to understand human emotions and the expression of those emotions in order to better define human characteristics. Therefore, finding the most accurate model is key, and using a built-in library of models is one of his ways to make the model even more accurate.

OpenCV and the use of support vector machines were both employed by Deny John Samuel [4] to create the model (SVM). Both OpenCV and SVM have helped me extract features from images. OpenCV was especially useful for emotion prediction. His four emotional categories are identified in this work: happy, surprise, anger and neutrality.

In order to create a mechanism for automatically identifying moods and sound patterns, AS Bhat [5] presented a system that could examine the visual and harmonic aspects of the notes in music as well as human emotions. This method divided music into different categories based on the listener's understanding of the music, in accordance with the Thayer model, which explains numerous components of music like beat, spectrum, and twilight.

An article on disentangling sentiments and facial expressions was written by Renuka R. Londhe [6]. She conducted studies on how variations in facial curves and intensity across photos taken at various pixel counts affect the expressions of the face. Different emotions were recognised using an artificial neural network (ANN), and playlist creation suggestions were produced based on the discovered emotions.

In his work, Zeng [7] proposed a new extension of facial features, including 1) vision-based feature extraction and 2) geometry-based rendering. These were the two main components of his proposal. Through the use of integrated geometrical segmentation utilising geometric aspect elements, visual extraction and configuration-based rendering have been used to eliminate crucial facial features including the eyes, lips, and brows.

An altered version of OpenCV based on the AdaBoost algorithm was proposed by Zhang [8]. He also employed two distinct methods for facial recognition. The outcomes demonstrated how straightforward and effective the two-wire technique was used.

Parul Tambe [9] suggested an approach in which the player's user interface communicates with the music player seamlessly. Its model initiates and learns user preferences and various user features, and music recommendations are based on the user's mood. They observed the user's face and recorded different facial expressions to infer how the user was feeling.

Kanwal Garg [10] developed an algorithm that takes into account the user's experience in selecting music from the user's playlist. The algorithm designed required only a short computational time, which greatly reduced the effort for using hardware. The idea classified feelings into five categories: joy, sorrow, surprise, wrath, and fear, offering the most appropriate and most effective approach to match music. These are the emotions of joy, sorrow, surprise, rage, and fear.

Aastha Joshi [11] was able to determine a user's disposition by analyzing facial features. This is not surprising as we often use expressions to convey emotions. Reduce the amount of

time consumers listen to music based on emotion. Since most people have a wide range of songs in their music playlists, playing different music at random was not enough to satisfy the customer's mood. This technology allowed users to play music automatically based on their mood at the time. Internet-connected digital cameras are used to capture and store images of people. After the image has been captured, its RGB format is converted to binary format so that it can be saved.

Detecting feature points is a method that can be implemented in a variety of settings, and it is known as a feature point detection approach to representing data, was accomplished by employing the Haar Cascade algorithm that was made available by the author's OpenCV in order to process the image data. The tracks that were created by the contestants are converted into Java software, which is then used to manage the database and play music based on how the user is feeling. Aditya and the rest of the team worked on developing his application for Android. [12] This can display auto-curated songs just for you. The model applied photo processing techniques to analyze the user's emotions and suggested songs for the user to listen to base on how those emotions were interpreted. Eclipse and OpenCV were used to implement the face reputation algorithm.

This has allowed us to improve and develop this method. This paper reviewed the evaluation of various face recognition algorithms. In this study, the authors asked participants to take pictures of themselves with a mobile phone camera and analyzed how they felt. A method to identify emotional facial expressions proposed by A. Habibzad [13] included three stages: 1) preprocessing, 2) analysis, and 3) output. 2) feature extraction; 3) a classification system outlining the various stages of the photo development process;

Zahara [14] came up with the idea of a Face Image Thresher (FIT), which effectively combines the complexity of pre-trained face recognition with the training of his Xception algorithm. Besides augmenting existing data, the FIT machine required removing facial images, collecting facial images, correcting distorted facial data, and collecting real-time data at scale. The accuracy of the final FER results obtained with the proposed approach improved by 16.95% compared to those obtained with the conventional method using the FER2013 database. Hernández-Pérez [15] proposed an approach that integrates facial features based on local-her binary-her patterns with FAST-focused, SHORT-oriented (ORB) features. (LBP). First, a facial recognition algorithm is applied to each image to extract as much information as possible.

Second, the ORB and LBP functionality was removed from the interface to speed up computation. More specifically, regional subdivisions were used in previous ORB novel techniques to prevent functional emphasis. Changing size, graying, or rotation will not affect the feature. Finally, split the compressed section (S.V.M) using the Vector Support Machine. The proposed method is evaluated against a number of sophisticated databases, including those found on websites such as CK+, JAFFE, and MMI.

YK Bhatti [16] developed a method to identify a teacher's face while in the classroom. To successfully extract high-quality features, we first need to identify faces in recorded videos. Then select the keyframes and delete all unwanted frames at the same time. He then used some Convolutional Neural Tweaking Networks to obtain more in-depth features and to assign phase differences to those features. This requirement is more important than classroom instruction.

The classroom is divided into five distinct sections using machines, which are referred to as RELM for short. This facilitates more effective learning and practice. A self-reflective paper proposed by Zhang Qinhu [17] builds on the idea of residual networks. Relative values are calculated by taking the Weights of all local pixels. This allows us to train channel attention to multiple interaction possibilities and draw attention to completely different channel domains to train channel attention. In the CK+ database, this study achieved an accuracy rate of 97.

89%, reaching 74.15% in the FER2013 database. The Convolutional Spatial Neural Network (DCSNN) was proposed by Subarna B. and Daleesha M. Viswanathan [18] for real-time facial recognition. The network was trained using JAFFE at FER-2013 and tested with live webcams. The Viola Jones algorithm, which uses hair features, is responsible for the first step in facial recognition. This DCSNN consists of 3 modes, 2 combinations and a fully integrated SoftMax layer. SoftMax layers have the ability to activate the rectified linear units (Re Lu) that are used to separate the equations with functions.

The author of this article mentions DCSNN as an alternative approach to her traditional FER method. CNN's recommendations for his FER work were endorsed by Manjun Wang et al. [19] Based on bulk depth of CNN layers. Using a CNN with 5 continuous convolutional layers, 3 high performance layers, 1 fully integrated layer and 1 output layer, run experiments on CK+ and JAFFE databases and recognize expressions using SoftMax separator Did.

Compared to methods such as HOG, LBP and CNN standard, the accuracy of the algorithms used in this paper provides an overall accuracy of 87.2% on both high databases.

This is based on the accuracy table provided. Deep learning and Grassmann iterations were used by Bindu Verma and Aisha Chowdhury [20] to develop his real-time framework for displaying the driver's emotions. Regions of interest (RoI) were found using hair cascades and deep neural networks (DNN). We carefully examined Alex Net and his VGG16 to extract the latest Grassmann Graph Embedding Discriminant Analysis (GGDA) features. This enables 98.8% accuracy on JAFFE, 98.8% accuracy on MMI, 97.4% accuracy on CK+ result view, all state-of-the-art programs.

Chieh-Ming Kuo and colleagues [21] presented a number of methods optimized for use in wearable devices. Some of them took into account temporal data obtained from digital images of faces. A CNN was used and included 4 folds, 2 combinations, 1 fully integrated layer and a SoftMax layer. This methodology was implemented and applied using his ReLu function and the result was shown to yield a high level of accuracy on his standard website. As a result, we found that the division accuracy reached 95.33%.

Distributional-Semantics Prefiltering (DSPF) is a technique used to build models that predict the most accurate, exploitative, and novel distributions of contextual semantics. This method was used by Victor Codina, Francesco Ricci, and Luigi Ceccaroni [22]. In addition, we show how integration strategies can be used in the development of DSPF.

Norma Saiph [23] used the latest version of the Neighborhood Consultation Program, which includes Decision Trees (DT) and Discrete Hidden Markov Models (H.M.M.). The combination of HMM and DT allowed us to distinguish different transport modes and reduce noise.

Sarabjot Singh Anand and Bamshad Mobasher [24] propose a recommendation system that takes into account both the data that is stored in short-term recall and the information in permanent memory.

CHAPTER 3

MUSIC MAPPING RECOMMENDATION ALGORITHM BASED ON FACIAL EXPRESSION

In this chapter, we discuss suggestion systems in detail, from recognizing the user's face to suggesting favorite music. The first step in the operation of our system is the identification of facial expressions for the purposes of emotion assessment. There are various algorithms that can do this. B. Histogram of oriented gradients (HoG) method. This method is also commonly used for face detection, in addition to detecting other types of objects. This algorithm leverages the SVM linear machine learning algorithm in addition to being a powerful feature descriptor. The Haar Cascade classifier is the algorithm that comes next. In 2001, Paul Viola and Michael Jones published a study named "Rapid Object Detection using a Boosted Cascade of Simple Features" that served as the foundation for the technique. Using a huge number of monochrome images, this machine learning method trains a cascade activity.

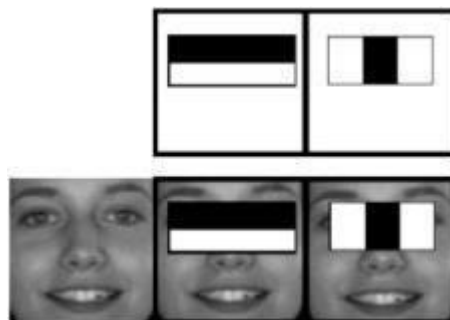


Fig. 3.1. Haar Cascade method of image detection

In our research, we implemented a very similar technique for real-time face detection as well as faces viewed from different angles. There are other facial recognition algorithms, but the method described here is easy to implement. During this process, images are captured frame by frame and these frames are sent to an algorithm called Haar Cascade classifier. There it is converted to grayscale, cloned for drawing, and finally placed in a region of interest (ROI). A face like a captured figure.3.1, the images were then passed to a CNN model for classification. You can also use videos or images processed in the same way to detect faces in videos or images.

3.1. Image classification

The next step is image classification. This means that after locating faces, the image must pass through several CNN layers to determine the type of emotion being conveyed. The Mini-Xception architecture is the most efficient way to achieve this. This architecture represents a very deep network model for convolutional neural networks. We trained the model on a dataset containing images representing neutral, scared, angry, and happy, sad, surprised disgusted facial expressions. CNN is capable of automatic learning. In other words, it learns functions independently based on the architecture developed for mini-xception. This allows CNNs to associate images with specific emotions. A CNN consists of multiple layers, each responsible for training and testing an image dataset, and then using that information to generate sentiment. The architecture of the mini-primary xception is shown in Figure 1. Shows how a captured image of a user is sent through the first layer of a convolutional neural network.

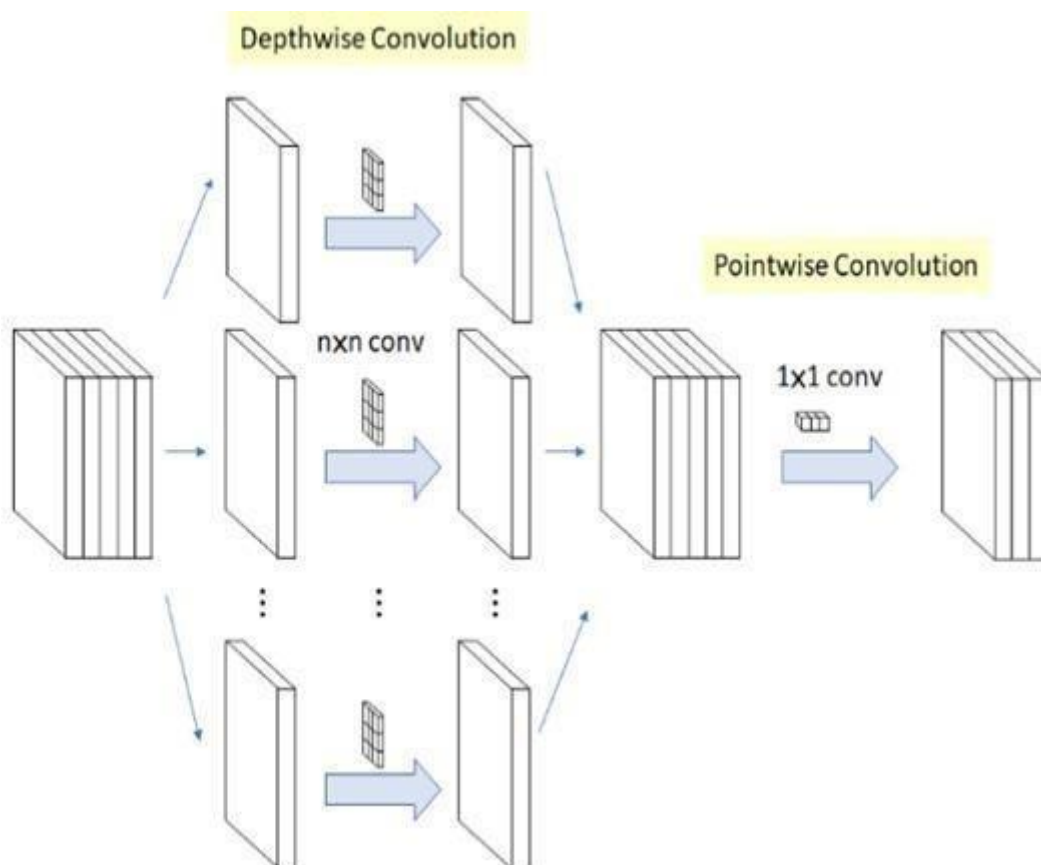


Fig 3.2. Mini Xception Architecture

3.2. CNN Architecture

The first layer consists of many different conv2d layers and batch norm layers. A Conv2d layer is a traditional convolution kernel layer that helps generate an output tensor by having an input stage. The conv2D slide layer filters or kernels that are run on the 2D input data are intelligent.

Duplicate. The end result is that the results are combined into a single output pixel. The output of the first hidden layer is sampled using the batch norm and normalized before being forwarded as the input of subsequent hidden layers. The problem of internal covariate shift is primarily addressed by this solution. This most commonly occurs when the CNN layer deep learning model fails. Tries to adapt to new changes in the input distribution in the network. This slows down the architecture training process. However, using batch normalization improves the learning speed. [Cause and Effect] The CNN architecture of the mini-Xception model consists of multiple layers: Sep-Conv2D, Batch Normand MaxPool 2D, where the second layer is the deeper hidden layer. Speaking of Sep-Conv2D, this is a significantly more advanced version of the Conv2D algorithm. Reducing the number of convolutions compared to the Conv2D algorithm while maintaining the same accuracy makes the architecture easier to learn. This is achieved by dividing the core into two distinct parts: point creases and depth creases. This can be understood using the attached diagram.

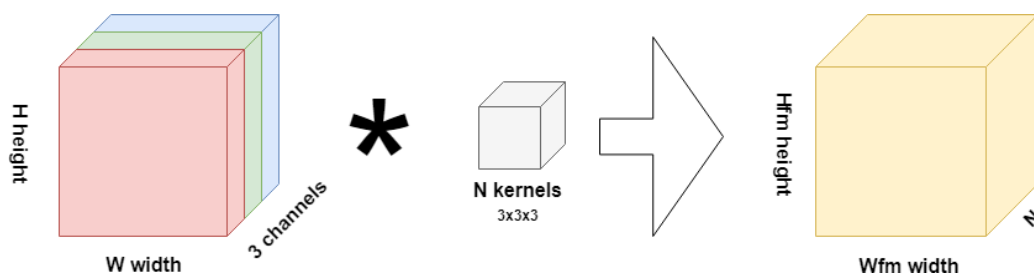


Fig.3.3. Traditional Convolutional layer

Here, as seen in Figure 3.3, to completely simplify one image, $(15-3) + 1$ multiplied (horizontal motion) $\times (15-3) + 1$ multiplied (direct movement) is required. $\times (3 \times 3 \times 3 \times N) = 4563N$ multiplications (N cores). If your image is (15×15) pixels in RGB and therefore has 3 channels, you need to perform this operation in multiples of 45000. We also know that the project will use 20 or 50 kernels to process the images. This is because the data set contains thousands of images and the neural network design involves many flexible layers extracting all the resources, which can make the process very cumbersome. Because there is We also know that a project that uses 20 or 50 kernels to process an image is called an image processing

project. Now there are depth-separable convolutional layers, also known as Sep-Conv2D layers, to help streamline the whole process. The M filter is used to convolve with layer-based depth wise convolution, and the N (number of original kernels) filter is used to convolve with volume-based pointwise convolution. This is shown in Figure 3.4. Traditional N-node kernels can be divided into depth-wise convolutions and point-wise convolutions. Since only one layer needs to be convolved for the intermediate result and the size of the M filter is 3×3 , the M filter requires $3 \times 3 \times 1$ multiplication factors for each individual turn. hold.

N filters are therefore $1 \times 1 \times 3$ multiplications for a single convolution to cover the entire volume, i.e., all channels of the intermediate result, consisting of 1×1 pixels. Since there are 3 channels, the sum of the intermediate results is 1. pixels. Since there are 3 channels, the sum of the intermediate results is 1.

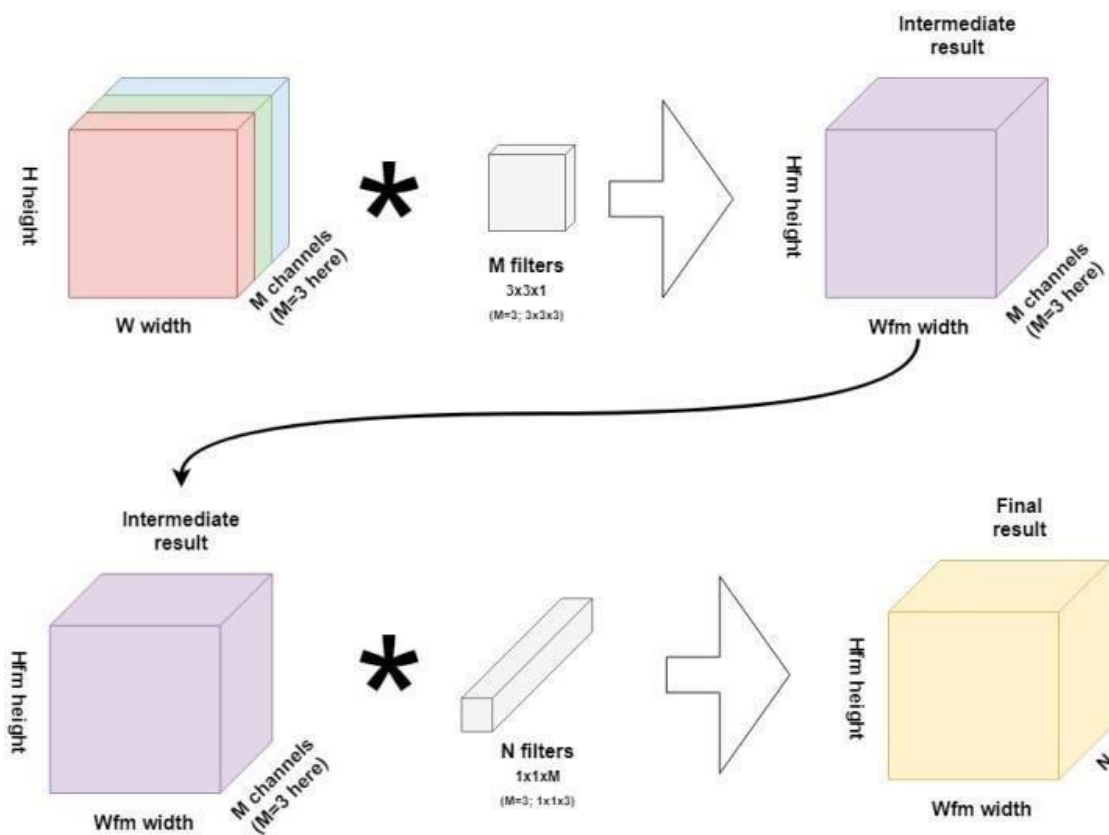


Fig. 3.4. Depth wise Separable Convolution layers

As you can see from the table below, we can achieve the same result as the traditional layering method with only 9633 iterations of the depth wise separable convolution (Sep-Conv2D). The formula for calculating the depth convolution is: $(13 \times 13 \times 3)$ (for 3 channels with horizontal and vertical motion) $\times 3 \times 3 \times 1$ (M filter) = 4563. to calculate $(13 \times 13 \times 10)$ (were

10 is the number of cores traditionally used for both horizontal and vertical movement) * 1 * 1 * 3 = 5070 So this is one less than the original 45630 we had to do. Compared to using a traditional Conv2D, this represents a significant reduction in the number of multiplications performed to get the same result. As a result, the term "Depth Wise Separable Convolutions" was coined. (Sep-conv2D) is used to efficiently optimize the 25regen process without sacrificing accuracy because the operations performed are equivalent. The CNN architecture uses the MaxPool 2D function whenever max pooling operation needs to be performed. The CNN architecture tool MaxPool 2D is responsible for performing the max pooling operation. As part of the merging process, a feature called Max Pooling selects a significant portion of the feature map that is affected by the filter. As a result, the feature map produced by the max pooling layer contains all significant features that were present in the previous feature map. It's exactly like a convolutional layer, except it uses the region size from the input encoded by the kernel instead of the dot product of the input and the kernel.

Figure 3.5 shows that the maximum possible value of four numbers was used.

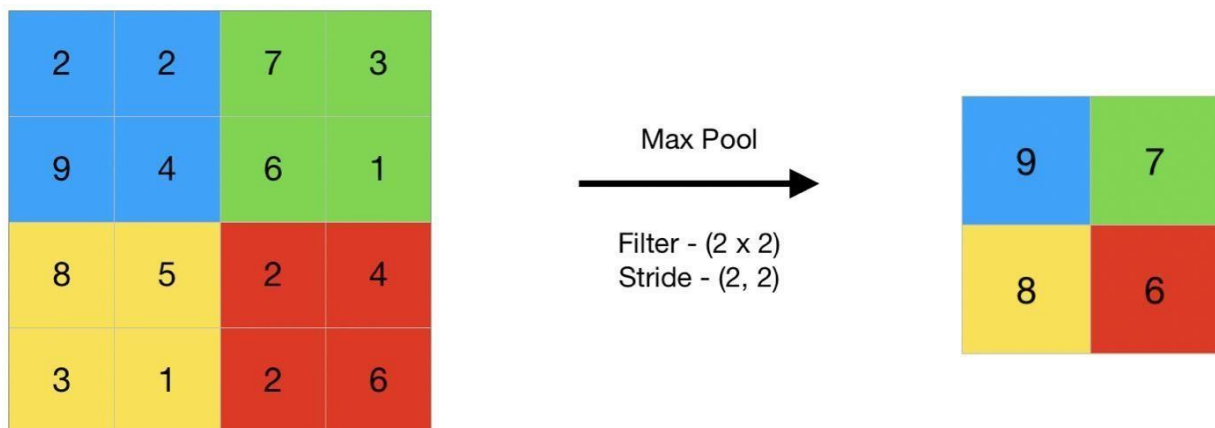


Fig. 3.5. Max Pooling

Pooling layers are used to reduce the overall size of map elements by 25%. A direct result of this is a reduction in both the number of parameters read and the number of network computations performed. The pooling layer makes it look like it is raining on the feature map regions generated by the convolutional layers. As a direct result, certain operations are performed on truncated features rather than well-structured features in convolutional layers. Because of this, the contrast between the model and the feature regions of the input image is very pronounced. Conv2D, Global Average Pooling and SoftMax features are in the following layers: Through the process of global pooling, each channel on the feature map is reduced to

his one number. The central 2D global pooling block uses tensor size (input width) x (input length) x for each matrix (channel) input (input channel). The output is a one-dimensional tensor, as shown in Figure 3.6, accessible here (input channel).

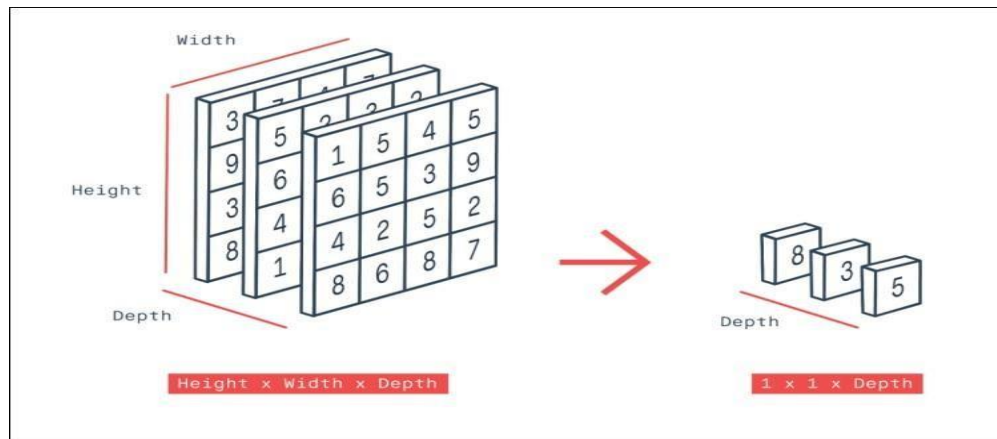


Fig. 3.6. Global Average Pooling 2D operation where 'Depth' = 'Filters'

Now let's talk about the SoftMax feature. The SoftMax function can be thought of as a kind of "rain" function. The output of a function can only be limited to the range 0 to 1 by an activity called "squashing". So, the output can be translated as literally as possible. This is necessary to compute the detected emotions and associated probabilities. For example, when an image is provided as input and a face is identified, the network outputs a vector of specific emotions. However, using the SoftMax function transforms this vector into relatively easy-to-understand probabilities. The CNN's layers work this way, training an architecture to classify emotions. Before entering

He separated the dataset used for CNN into two categories. 20% of the data sample is used for testing, while the remaining 80% are used for training. The model's accuracy is tested before being finished once the pictures have been trained on CNN. During the testing phase, accuracy is determined by determining whether face images are classified correctly. Image classification accuracy can be improved in two ways, he said: either lengthening the testing period or expanding the dataset's image count. In order to increase the amount of photos in the dataset, augmentation techniques are applied.

3.3. Workflow of music recommendation system

In our study, this technique is used, and the model that is created undergoes training using the FER 2013 dataset, which consists of 35,000 pictures reflecting various moods.

It takes a long time to learn a model. The CNN layers are subsequently utilised to process the faces that are picked up from the streaming camera footage and classify the emotions. The process of making music recommendations using the suggested mapping algorithm follows sentiment categorization. Users are directed to a list of music according to what emotion they experience because of this. This method uses a basic line of code to connect an emotion detection model to a link of a video found on YouTube.

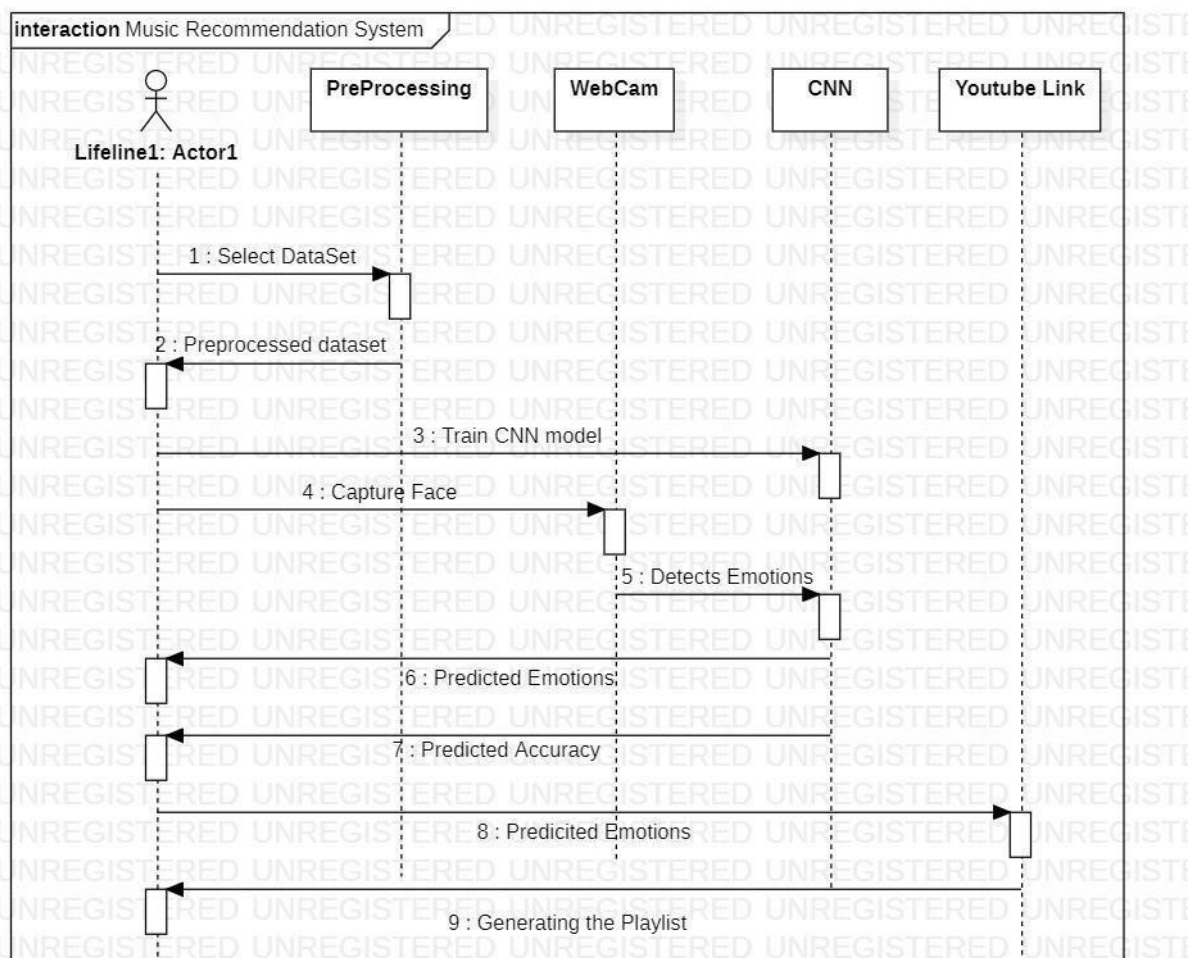


Fig. 3.7. Sequence Diagram of music recommendation System

Figure 3.7 displays a sequence diagram for the full model. This picture shows how the user's dataset is received, sent to a preprocessor, and then given back to the user so they may train a model using CNN to recognise emotions. The user's recognition of facial features using the webcam comes next. The CNN model processes this facial recognition, accurately expresses the emotion, and displays it to the user. Predicted emotions are counted with a mapping algorithm that has the ability to open YouTube playlists based on detected emotions. 50 specific numbers are set to detect emotions.

This means that when the emotion is detected 50 times in a row, open the provided YouTube video link. The mapping algorithm otherwise used the nested case technique to determine the user's sentiment, and a specific count of 50 was set to detect sentiment. Emotions will continue to appear on the screen unless a constant emotion is continuously detected, and the same emotion appears on the screen at the same time.

The Haar Cascade technique is used to process the facial image after it has been taken, and it is then given to the mini-Xception model. Following that, the model categorises the feelings, illuminating both the feelings themselves and their correctness. Figure 3.8 displays the emotion as neutral with an accuracy of 76.06%; Figure 3.9 displays pleased faces with a performance of 40.15%; and Figure 3.10 displays an angry face with a performance of 37.45%.

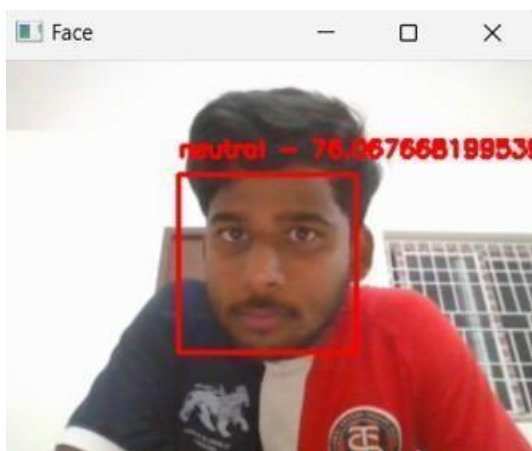


Fig 3.8. Neutral face

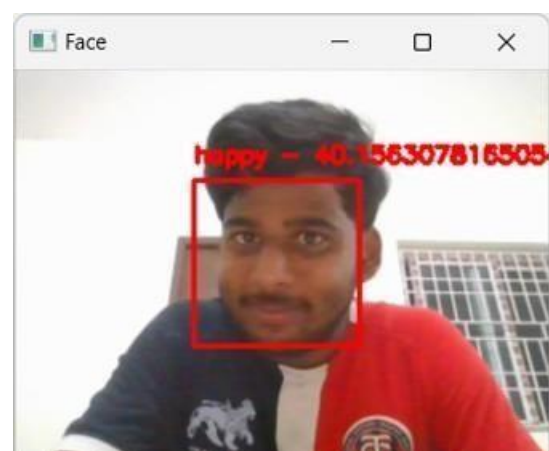


Fig 3.9. Happy face

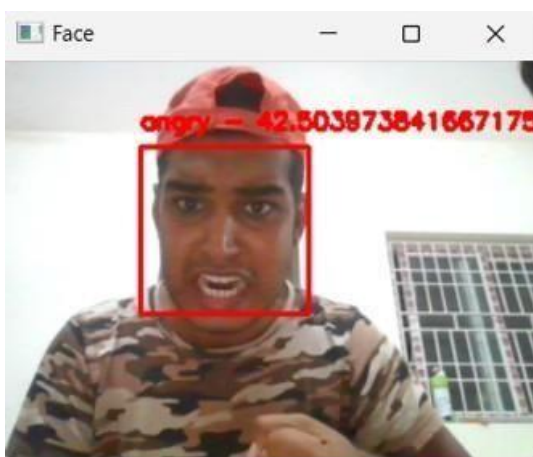


Fig 3.10. Angry Face



Fig 3.11. Scared Face

3.4. System Architecture

The FER-2013 dataset is used to generate the images, which are then divided into two portions (about eighty percent to be used for training and 20 percent for evaluation) and preprocessed. Next, two portions of the training set are separated. You can easily step through multiple layers of the CNN with x-training (feature extraction) and y- training (tagging). The CNN mini-xception algorithm is then applied to the partitioned dataset, as shown in Figure 3.11 test or real-time inputs are preprocessed and then sent to the trained model to be classified according to facial expressions. It is then sent to the created music mapping algorithm and served to the user on YouTube. with matching songs.

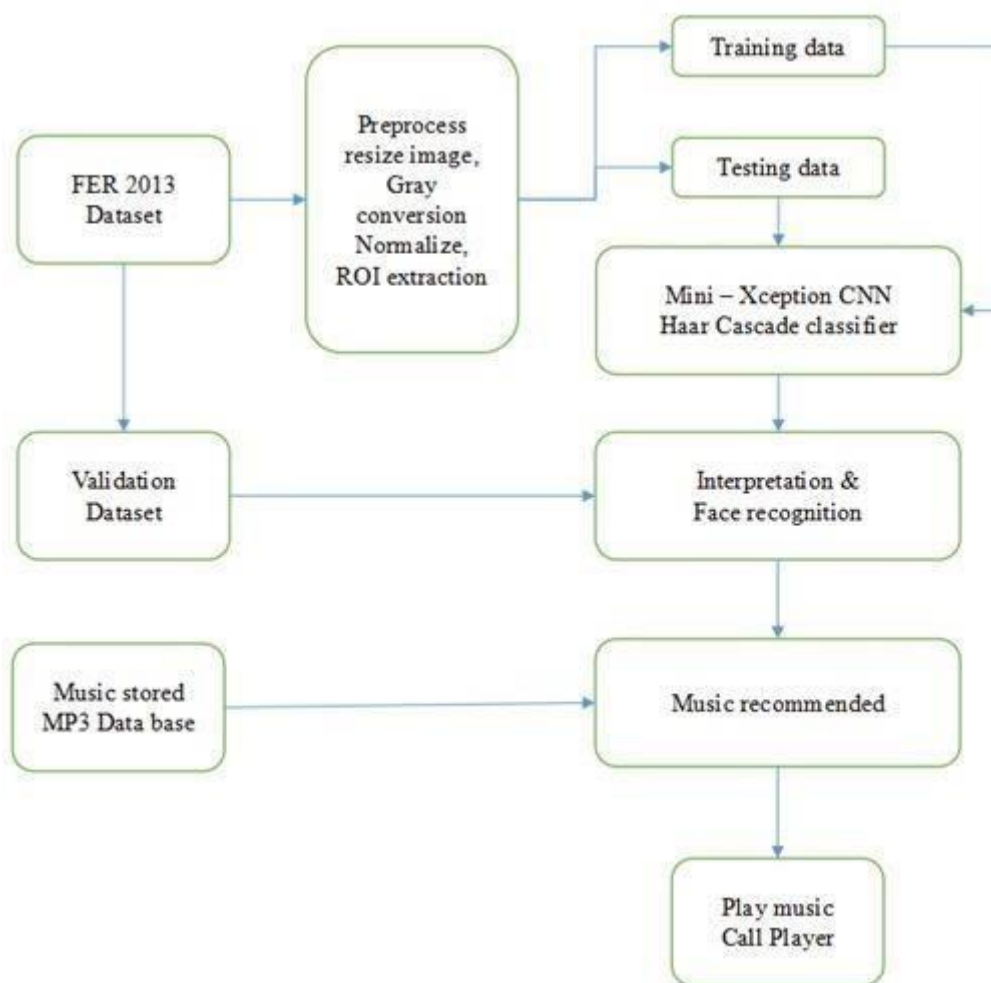


Fig. 3.11. System Architecture

CHAPTER 4

IMPLEMENTATION OF MUSIC RECOMMENDATION SYSTEM

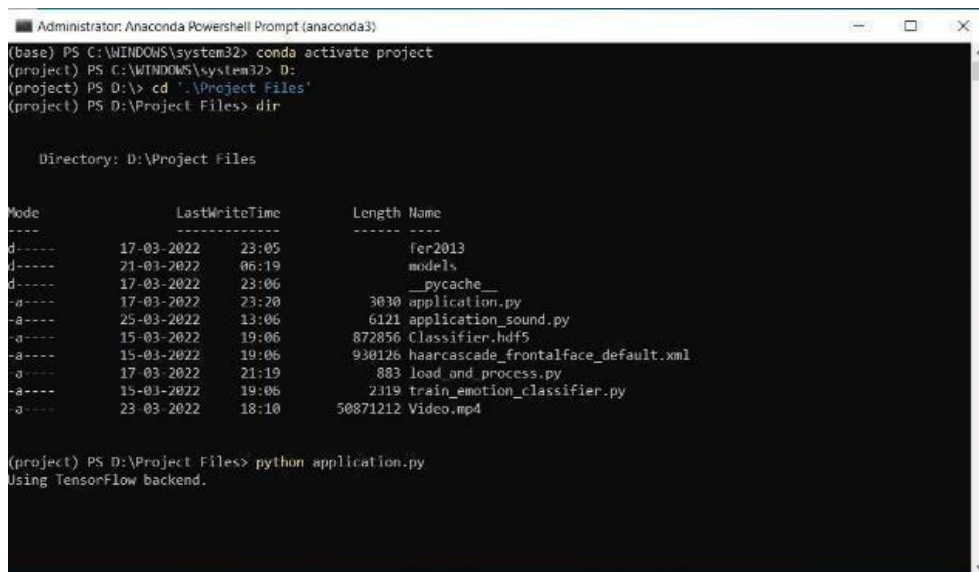
The FER-2013 dataset's visuals have been extracted, preprocessed, and split into two groups. 20% is utilised for testing, while 80% is utilised for training. After then, the workout set is split into two parts. Using x-training (aka feature extraction) and y-training, we can easily traverse multiple layers (tagging) of a CNN. The CNN mini-exception algorithm is then applied to the partitioned dataset, as shown in Figure 3.11. preprocessed test or real-time inputs are passed to the trained model and classified by facial expressions. It is then passed through a developed music mapping algorithm to show the appropriate songs to the user on YouTube.

4.1. Face Detection

Once the model is trained, the first step is to recognize faces and classify the subject's emotions. To achieve this, I used a Haar cascade frontal face default.xml file that detects a real-time image of the user standing in front of the camera. On the other hand, with a classifier, just find the face and cut off all the excess. An HDF5 file that separates and captures the subject's face from the background of the image. She classified the spectrum of human emotions into seven categories: happy, neutral, angry, sad, surprised, fearful, and disgusted. I've also implemented code that can face uploaded video files. This step attempts to capture an image of the exact moment the emotion was expressed. The image is then resized to the necessary format and transformed to grayscale since the model can just recognise black and white pixels. After that, the picture is cloned so it may be drawn. The picture is then preprocessed for network analysis after the area of interest (ROI) of the human face is retrieved from it. Regions of interest (ROI) are used to make predictions, displaying both sentiment labels and associated probabilities. In this method, the emotion in question is displayed on the screen above a rectangle obscuring the face, with readings of high probability ranging from 0% to 100%. Real-time facial recognition uses cameras that take pictures pixel by pixel and send them to a classification system called the Haar Cascade. To categorise the sentiment, this classifier merely collects the ROI and feeds it to the CNN model.

4.2. Music Mapping

Emotions that last for the specified number of frames open a dialog box with the text "Playing the song [insert emotion here]" and enable links to related YouTube videos. Figure 4.1 shows an implementation that launches an Anaconda command prompt with administrator privileges, follows the steps below to open his webcam, and takes a picture while capturing the user's face in real time.



```
Administrator: Anaconda PowerShell Prompt (anaconda3)
(base) PS C:\WINDOWS\system32> conda activate project
(project) PS C:\WINDOWS\system32> D:
(project) PS D:\> cd '.\Project Files'
(project) PS D:\Project Files> dir

Directory: D:\Project Files

Mode                LastWriteTime         Length Name
----                -
d-----         17-03-2022    23:05             fer2013
d-----         21-03-2022    06:19             models
d-----         17-03-2022    23:06             __pycache__
-a-----         17-03-2022    23:20             3030 application.py
-a-----         25-03-2022    13:06             6121 application_sound.py
-a-----         15-03-2022    19:06             872856 Classifier.hdf5
-a-----         15-03-2022    19:06             930126 haarcascade_frontalface_default.xml
-a-----         17-03-2022    21:19             883 load_and_process.py
-a-----         15-03-2022    19:06             2319 train_emotion_classifier.py
-a-----         23-03-2022    18:10             50871212 Video.mp4

(project) PS D:\Project Files> python application.py
Using TensorFlow backend.
```

Fig. 4.1. Deployment and Operation of our System

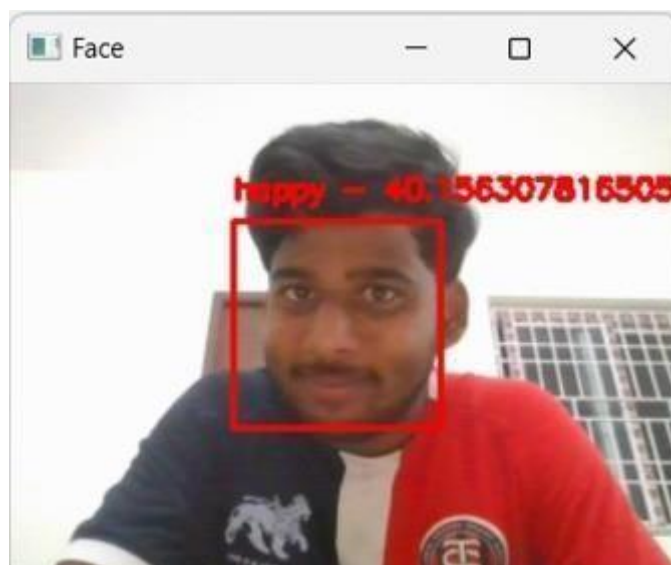


Fig.4.2. Image Classification with Accuracy Percentage

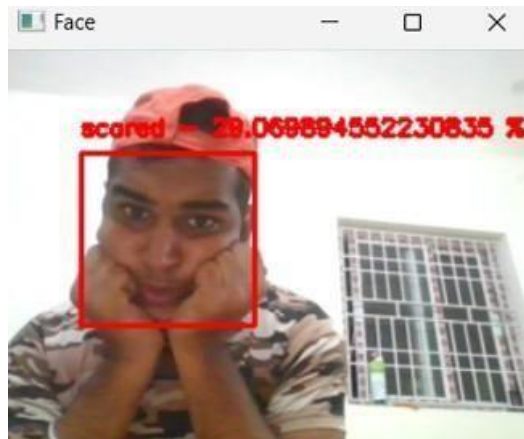


Fig. 4.3. Image Classification with Accuracy Percentage

Figures 4.2 and 4.3 depict the image categorization; in Fig. 4.2, the emotion in question is "happy," and in Fig. 4.3, the emotion in question is "scared," with an accuracy rate of 29.06%. The user is then presented the identified emotion with a rectangle representing it on their face along with the accuracy of the predicted feeling.

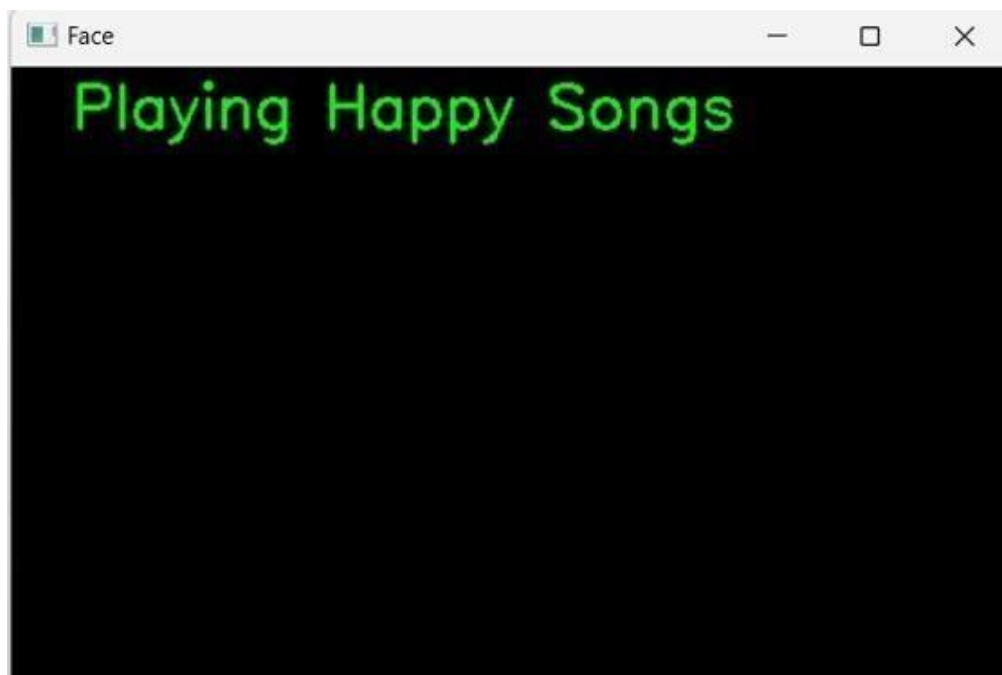


Fig. 4.4. Display of phrase Playing Happy Songs

After 15 repetitions of the same feeling are recognised by the system, the screen shows a phrase as illustrated in fig. 4.4, and the song is then suggested by the recommender. The expression's name is determined by the sort of emotion detected; it may print "Detecting Emotion" songs and be joyful, sad, furious, surprised, neutral, disgusted, or afraid.

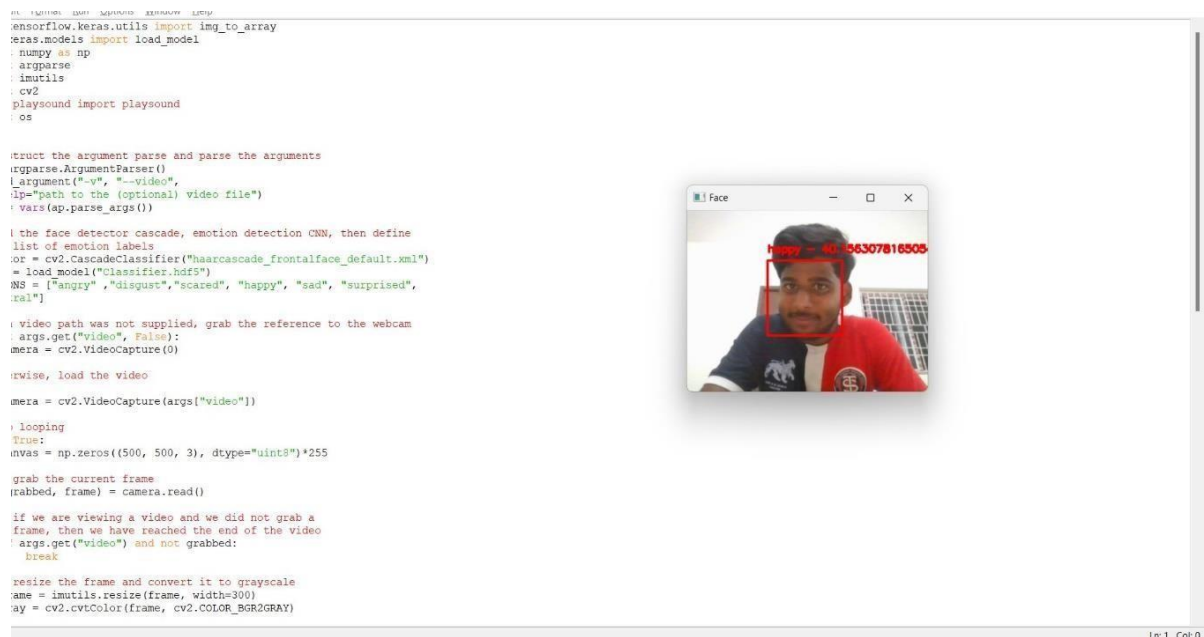


Fig. 4.5. Emotion Classification from recorded Video

The emotion label continues to display until the same emotion is constantly detected for 50 counts, at which point it stops, and the expression Playing "Emotion detected" Songs appears. The YouTube link then appears, that corresponds to the detected emotion, and plays the appropriate song. The process is shown in Fig. 4.5, along with the predicted emotion and its corresponding accuracy.

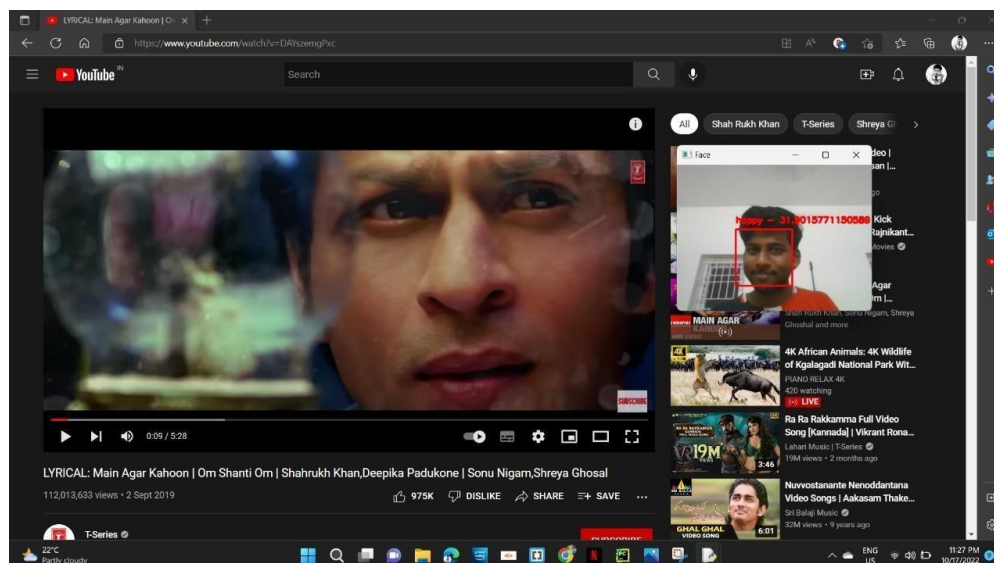


Fig. 4.6. Song link from YouTube

CHAPTER 5

SYSTEM REQUIREMENTS

5.1. Hardware Requirements

- System: Pentium i3 Processor.
- Hard Disk: 500 GB.
- Monitor: 15'' LED
- Input Devices: Keyboard, Mouse
- Ram: 4 GB

5.2. Software Requirements

- Operating system: Windows 10.
- Coding Language: Python 3.11.

5.3. Software Environment

A common high-level programming language is Python. Code readability is important in its design and uses indentation heavily. Python uses garbage collection and has dynamic typing. It supports a variety of operations and operating procedures, including methods, object orientations, and operation functions (which is unique). It is often referred to as a "battery built-in" language because of its extensive library.

Install Python on your computer system

1. You can use ImageAI and TensorFlow, Numpy, OpenCV etc. Install necessary third-party software
2. Save the sample search file to your computer. Instructions to follow:
 - 1) Go to the official Python website <https://python.org> and download and install Python version 3.
 - 2) Use pip to install the above dependencies:

i. Tensorflow:

An open source package called Tensorflow is used for data flow and different programming for various tasks. It is a code library that can be used in machine learning like neural networks. Google uses it for research and development. The Google Brain team developed Tensorflow for use at Google. It was made available under the Apache License 2 on November 9, 2015. The second version of the 0. Google Brain system is called Tensorflow. The first version of Tensorflow was released on February 11, 2017. While Tensorflow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for computational graphics processing), app usage only supports one device. Tensorflow is compatible with many operating systems including 64-bit Linux, macOS, Windows, and mobile operating systems such as Android and iOS. Tensorflow's architecture makes it easy to deploy computations across multiple platforms (CPUs, GPUs and TPUs), from PCs to servers to mobile and edge devices. Stateful dataflow graphs are used to represent calculations in Tensorflow. These neural networks work on different datasets called tensors, from which Tensorflow is named.

ii. Numpy:

The Numpy package for the Python programming language adds support for many different numbers and numbers, as well as many advanced arithmetic operations that can be performed on these arrays. NumPy's predecessor, Numeric, was originally developed by Jim Hugunin with the help of many other developers. Travis Olphant created NumPy in 2005, continuing to combine and replace the Numarray functionality with Numeric. Many people have contributed to the open source NumPy program.

iii. SciPy:

SciPy modules cover multiple optimizations, linear equations, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other function engineering. SciPy is part of the NumPy family, which also includes many scientific libraries and tools such as Matplotlib, pandas, and SymPy. SciPy mostly abstracts NumPy array objects.

This NumPy stack can be used in parallel with programs such as MATLAB, Octave, and Scilab. SciPy cluster is another name for NumPy cluster.

The SciPy library is currently available under the BSD license and supports and supports open source community development. Additionally, NumFOCUS is a community-based organization that supports both reversible and viable research.

iv. OpenCV:

A set of programming tools called OpenCV mainly focuses on real-time computation. It was originally developed by Intel and later supported by Willow Garage and Itseez. The cross-platform and open-source BSD license makes the library free to use.

v. Pillow:

Python Imaging Library is a free library for the Python programming language that supports opening, editing, and saving various types of image files. This feature is supported on Linux, Mac OS X and Windows.

vi. Matplotlib:

Python's NumPy numeric extension and Matplotlib plotting tool are written in Python. It provides an object-oriented API for GUI tools such as Tkinter, wxPython, Qt, or GTK+, which can be used to pull images into projects.

vii. H5py:

Python's NumPy numeric extension and Matplotlib plotting tool are written in Python. It provides an object-oriented API for GUI tools like Tkinter, and the h5py program provides high-level and low-level APIs for Python's HDF5 library. High-level components use well-designed Python and NumPy to support access to HDF5 data, files, and groups, while the low-level interface should be a full version of the HDF5 API.

The simplifies the process of reading and writing data from Python by providing priority automatic conversion between Python (Numpy) data types and data structures and their equivalent HDF5. wxPython, Qt or GTK+ can be used to embed projects in projects.

viii. Keras:

Python-based Keras is an open source neural network library. It can run on Microsoft Cognitive Toolkit, Theano, PlaidML or TensorFlow. User friendliness, modularity and scalability are design goals as it aims to facilitate rapid experiments with deep neural networks.

ix. ImageAI:

ImageAI provides an API that identifies 1000 different objects in an image using pre-trained models built on the ImageNet-1000 dataset. Examples of SqueezeNet, ResNet, InceptionV3, and DenseNet are given.

3) Download the YOLOv8 model file that will be used for object detection.

CHAPTER 6

SYSTEM STUDY

6.1. Feasibility Study

At this stage the feasibility of the project is evaluated and a business proposal is developed with a fairly simple plan and some cost estimates. As part of the systems analysis, the feasibility assessment of the proposal should be made. By doing this, he ensured that there was no burden on the company's planning process. In order to perform the analysis, the basic requirements of the system must be known.

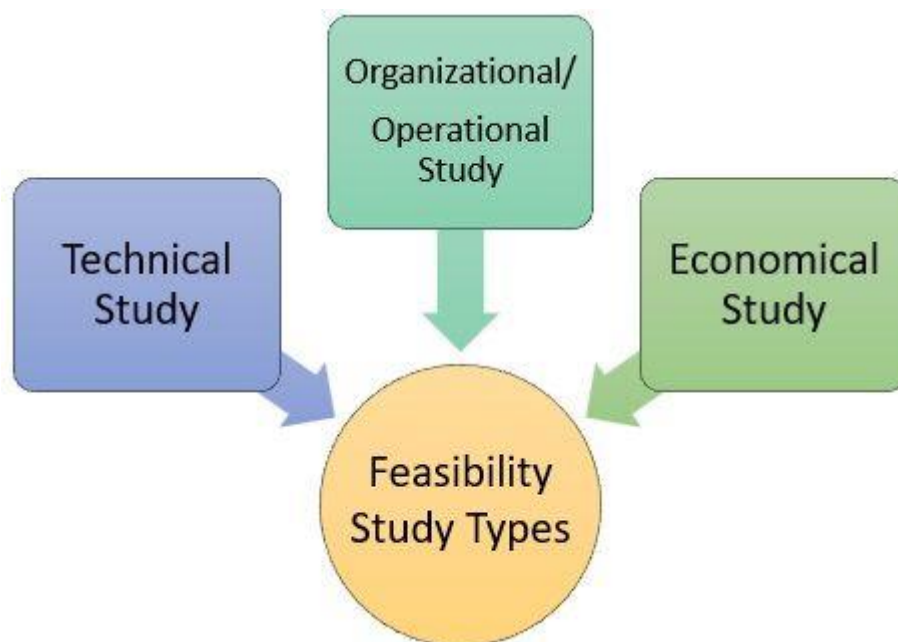


Fig:6.1. Types of Feasibility Study

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

6.1.1. Economical Feasibility

This study looks at what the company's financial impact will be on the company. Companies have less money for system development. Debts must be supported by evidence. As a result, the design process is within budget as most of the equipment is in the public domain. Only special items need to be purchased.

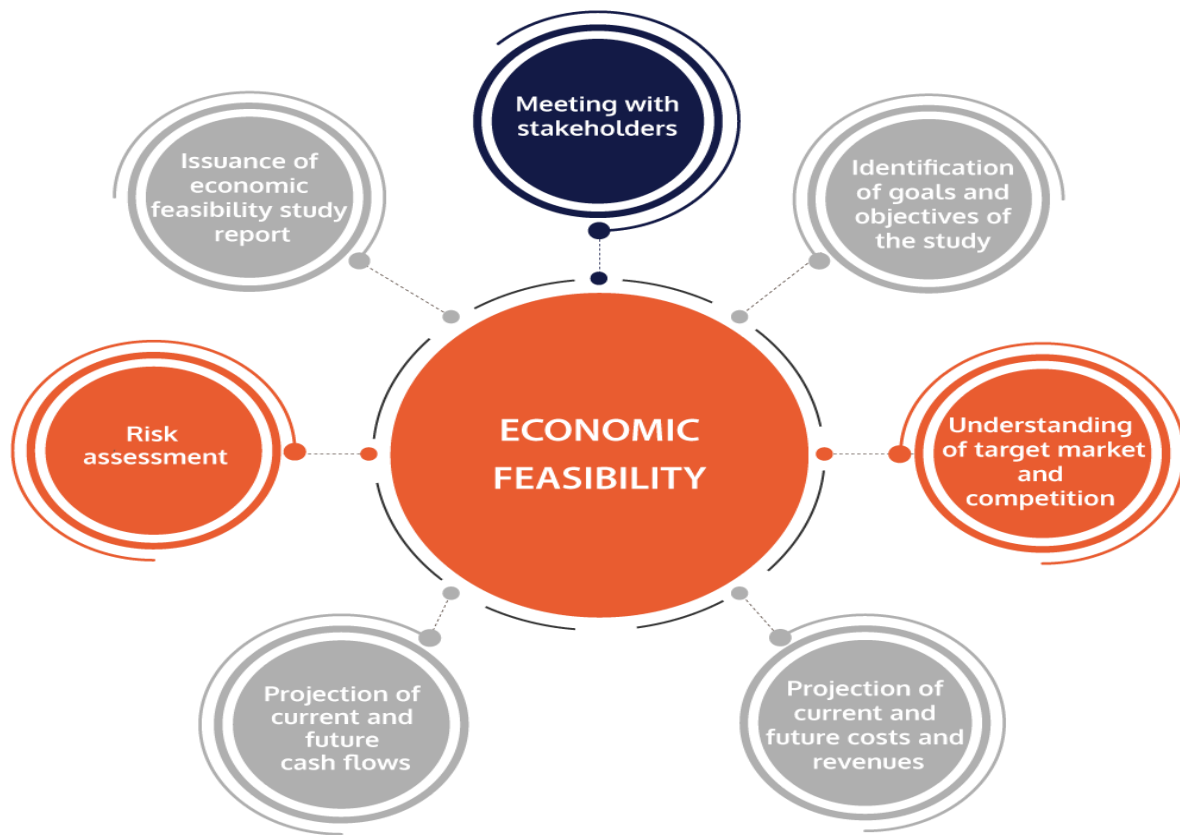


Fig:6.2. Economic Feasibility

6.1.2. Technical Feasibility

In other words, as part of this study, rules were analyzed to determine the effectiveness of the system. Any system design should not strain available resources. As a result, the market value will be very high. Therefore, customers will face high expectations. Because the system created by the application requires little or no modification, there should be very little demand.

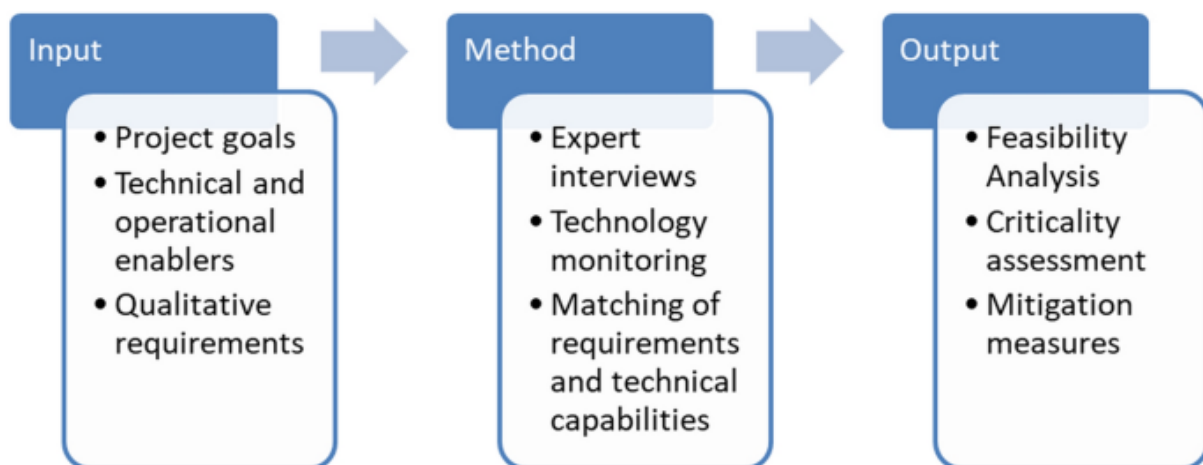


Fig:6.3. Technical Feasibility

6.1.3. Social Feasibility

The purpose of the research is to examine the user's acceptance of the system. This includes the training process for users to use the system correctly. Users should not be threatened by the system, but should be accepted as necessary. User acceptance is dependent on the application for the user to introduce and get to know the system. As he is the end user of the system, his confidence level should be raised so that he can still take some criticism.

CHAPTER 7

SYSTEM TESTING

Locating mistakes is the aim of trying out. The system of testing include seeking out any capability flaws or weaknesses in a chunk of work. It gives a way of evaluating the capability of elements, subassemblies, assemblies, and/or a completed excellent. in an effort to make certain that the software device satisfies its necessities and person expectancies and does now not fail in an unacceptable manner, it should be positioned via a manner known as software checking out. there are many distinctive take a look at types. each test kind specializes in a positive checking out requirement.

7.1. Types Of Tests

7.1.1. Unit Testing

Unit testing involves creating states that check that internal processes are working properly and program inputs produce usable products. All branch decisions and internal code flow must be defined. It is a test of individual software components of an application. It is done after each unit before integration. This is an evaluation model based on information about its structure and impact. Unit testing performs simple tests at the component level and checks the specific configuration of a system, application or business process. The testing unit verifies that each particular method of the business process works according to the published instructions and has the correct instructions and expected results.

7.1.2. Integration Testing

Integration software products are tested collaboratively to see if they work as a single program. Evaluation is event-driven and focuses more on the results of the screens or areas. Although the product combination has been tested successfully, the combination shows that the product combination is correct and consistent. The purpose of integration testing is to identify problems that arise when combining components.

7.1.3. Functional Test

Functional testing provides evidence that functional testing is compliant with business needs and requirements, system documentation, and user guides. performance measures focus on:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and planning of job tests focusing on needs, important tasks or specific situations. In addition, the assessment should consider the scope of the data area, the previous and subsequent procedures, and the work process. Additional experiments are discovered before the experimental work is completed and the results of existing experiments are evaluated.

7.1.4. System Test

System testing ensures that the integrated software is fully compliant with specifications. Evaluates settings to provide recognizable and visible results. Configuration-oriented system integration testing is an example of system testing. System evaluation is based on methods and descriptions that focus on the elements before integration and connection.

7.1.5. White Box Testing

White box testing is a type of testing where the software tester knows the inner workings, structure and language of the software, or at least what he wants to do. It has a purpose. It is used to measure inaccessible areas at the black box level.

7.1.6. Black Box Testing

In a "black box" software testing is done without knowledge of the inner workings, architecture or language of the model under test. Like many other tests, black box testing must be based on explicit information such as instructions or data requirements. It is a type of testing in which the tested software is treated like a black box. I can't "see" the inside. Regardless of the functionality of the software, tests generate input and respond to results.

7.2. Unit Testing

While it is not unusual for coding and unit testing to be performed as two separate phases, unit testing is often done as part of the mixed code and unit testing phase of the software lifecycle.

Test strategy and approach

An on-site assessment will be completed and full operational testing scheduled.

Test objectives

- All entries must be correct.
- The specified link must be used to open the page.
- Delayed access to screens, messages or replies.

Features to be tested

- Make sure the data entry is correct.
- Do not enter text.
- All links should direct the user to the correct page.

7.3. Integration Testing

Incremental testing of two or more integrated software components on a single platform to reduce the effects of interoperability is called "software integration testing". The purpose of integration is to ensure that software applications or devices, such as those found in the software system, or, at a higher level, enterprise-wide software applications do not work together badly.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.4 Acceptance Testing

User acceptance The testing phase of any project is important and requires the involvement of end users. It also ensures that the system meets certain requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 8

RESULTS AND DISCUSSION

The trained model is very fluid under shorter time pressure, can name feelings and recognise faces. Our model has the benefit of capturing 24 frames per second via a actual time cam feed, as opposed to more complicated models that go through an image in 2 seconds. Moreover, the trained model can recognize faces and name emotions very smoothly. The model was intensively trained to be able to classify emotions more effectively and accurately. The categorical cross-entropy is used as both a loss and accuracy metric to estimate the level of accuracy. As a result, we exceeded the maximum accuracy threshold of 95% and attained an average accuracy rate of 90%. Loss decreases significantly from epoch to epoch. This indicates that the model was trained more rigorously thanks to the increased number of datasets we provided. This improves the ability to express high precision. To train the model, we used the FER 2013 dataset. In addition, we used the aforementioned data augmentation technique to augment the dataset. Figure 8.1 displays a plot of the predictive model's precision and loss. Two lines are seen on each chart. The accuracy or loss during training is represented by one line, while the accuracy or loss during validation is represented by the other line. The first graph shows loss as a function of epochs. This graph displays the model's loss, which is a measure of how inaccurate the predictions are.

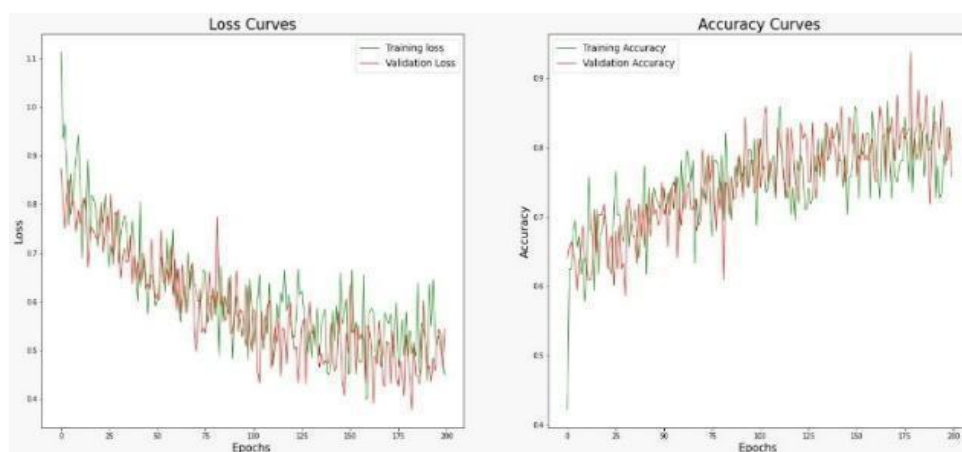


Fig. 8.1. Loss and Accuracy graphs for every epoch

The term "validation accuracy" refers to the accuracy presented during testing, "training accuracy" refers to the accuracy achieved during model training (using the FER dataset) as indicated by training accuracy. point. It has already exceeded the 95% validation accuracy threshold. This means that the model can achieve higher accuracy with more data. You can see this in the graph you're looking at now. This shows that after only 200 epochs we get an average accuracy of 80%. I paused the model after it completed 200 epochs and plotted the following chart. This is because the calculation of loss and precision takes a lot of time.

CHAPTER 9

CONCLUSION

Emotion recognition based on facial expressions has been the subject of extensive research and has attracted the interest of many academics and researchers because it can be used in a variety of ways in the real world. The difficulty of recognizing people's emotions is increasing at an alarming rate, and in direct response, numerous algorithmic solutions have been developed to address this problem. These emotion recognition algorithms are of great importance in the fields of medicine and human sciences, and researchers are constantly developing innovative approaches to extract people's emotions and use them therapeutically. This is one of the categories in which they are used. The proposed system has undergone extensive testing in real-time environments, demonstrating its ability to accurately capture user emotions. However, it should be tested under different lighting conditions to assess the resilience of the system. Despite this requirement, we found that our software performed admirably in capturing images of users and determining their emotions. Moreover, the proposed model successfully acquired new real-time images of users for the purpose of updating the Haar Cascade classifier and dataset.

Our findings show that our model has room for improvement in recognizing emotions. All it takes is a few more tweaks and tweaks. Our model was trained on her FER dataset, and in combination with this method, we are able to determine human emotions with over 95% accuracy while maintaining the best balance between speed and accuracy. increase. A new mapping algorithm has been developed that automatically links YouTube songs to CNN layers according to the detected emotions. The system was developed using the Mini-Xception architecture, which made the system lighter and consequently more efficient.

The current model has some shortcomings and fails to provide a high level of accuracy and predictability in operation. For example, to watch a music video, simply click the provided YouTube link. For better accuracy of the facial recognition system, the camera resolution should be his 360p or higher.

REFERENCES

- [1]. Gilda, Shlok, “Smart music player integrating facial emotion recognition and music mood recommendation.” 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). IEEE, 2017.
- [2]. Florence, S. Metilda, and M. Uma. “Emotional Detection and Music Recommendation System based on User Facial Expression.” IOP Conference Series: Materials Science and Engineering. Vol. 912. No. 6. IOP Publishing, 2020.
- [3]. Vinay p, Raj p, Bhargav S.K., et al. “Facial Expression Based Music Recommendation System", International Journal of Advanced Research in Computer and Communication Engineering, IJARCCCE.2021.10682, 2021.
- [4]. Samuvel, D. J., Perumal, B., & Elangovan, M. (2020). Music recommendation system based on facial emotion recognition. 3C Tecnología. Glosas de innovación aplicadas a la pyme. Special edition 261-271, March 2020.
- [5]. S. Bhat, V. S. Amith, N. S. Prasad and D. M. Mohan, “An Efficient Classification-Algorithm for Music Mood Detection in Western and Hindi Music Using Audio Feature Extraction,” 2014 Fifth International Conference on Signal and Image Processing, pp. 359-364, 2014.
- [6]. Londhe RR and Pawar DV 2012 Analysis of facial expression and recognition based on statistical approach, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2Issue-2, May 2012.
- [7]. Z. Zeng, M. Pantic, G. I. Roisman and T. S. Huang, “A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 1, pp. 39-58, Jan. 2009.
- [8]. Xianghua Fan, Fuyou Zhang, Haixia Wang and Xiao Lu, “The system of face detection based on OpenCV,” 24th Chinese Control and Decision Conference (CCDC), pp. 648- 651, 2012.
- [9]. Parul Tambe, Yash Bagadia, Taher Khalil and Noor UlAin Shaikh, Advanced Music Player with Integrated Face Recognition Mechanism, IJIRT, Volume 3, Issue 1, ISSN: 2349-6002,2015.

International Journal of Advance Research in Computer Science and Software Engineering, ISSN:2277 128X Volume 4, Issue 4, 2014.

- [11]. Aastha Joshi, Rajneet Kaur, A Study of speech emotion recognition methods, IJCSMC, Vol. 2, Issue. 4, pg.28 – 31, April 2013.
- [12]. Aditya et, Sundgren D, Rahmani R, Moran A, Bonet I and Larsson A 2015 Speech emotion recognition in emotional feedback for human-robot interaction, International Journal of Advanced Research in Artificial Intelligence, 2015.
- [13]. A habibzad, ninavin and Mir kamalMirnia A new algorithm to classify face emotions through eye and lip features by using particle swarm optimization, 4th International Conference on Computer Modeling and Simulation, IPCSIT Vol.22, 2012.
- [14]. L. Zahara, P. Musa, E. Prasetyo Wibowo, I. Karim and S. Bahri Musa, “The Facial Emotion Recognition (FER-2013) Dataset for Prediction System of Micro-Expressions Face using the Convolutional Neural Network (CNN) Algorithm based Raspberry Pi”, Proceedings of 5th International Conference on Informatics and Computing, pp. 1-9, 2020.
- [15]. Syed Aley Fatima, Ashwani Kumar, Syed Saba Raoof. Real Time Emotion Detection of Humans Using Mini-Xception Algorithm, IOP Conference Series: Materials Science and Engineering, Volume 1042, 2nd International Conference on Machine Learning, Security and Cloud Computing (ICMLSC 2020), December 2020.
- [16]. Ben Niu, Zhenxing Gao and Bingbing Guo, “Facial Expression Recognition with LBP and ORB Features”, Computational Intelligence and Neuroscience, Vol. 2021, pp. 1-16, 2021.
- [17]. Y.K. Bhatti, A. Jamil, N. Nida, M.H. Yousaf, S. Viriri and S.A. Velastin, “Facial Expression Recognition of Instructor using Deep Features and Extreme Learning Machine”, Computational Intelligence and Neuroscience, Vol. 2021, No. 1-14, 2021.
- [18]. J. Daihong, Hu Yuanzheng, D. Lei and P. Jin, “Facial Expression Recognition Based on Attention Mechanism”, Scientific Programming, Vol. 2021, pp. 1-18, 2021.

- [19]. Subarna B and Daleesha M Viswanathan, “Real Time Facial Expression Recognition Based on Deep Convolutional Spatial Neural Networks” International Conference on Emerging Trends and Innovations in Engineering and Technological Research (ICETIETR) 2018.
- [20]. Minjun Wang et al., “Face Expression Recognition Based on Deep Convolution Network” 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI) 2018.
- [21]. Bindu Verma and Ayesha Choudhary, “A Framework for Driver Emotion Recognition using Deep Learning and Grassmann Manifolds” 21st International Conference on Intelligent Transportation Systems (ITSC) 2018.
- [22]. Chieh-Ming Kuo et al., “A Compact Deep Learning Model for Robust Facial Expression Recognition” IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2018.
- [23]. Victor Codina, Francesco Ricci and Luigi Ceccaroni, “Distributional Semantic Pre-filtering in Context-Aware Recommender Systems”, 3rd ACM Conference on Recommender Systems, pp. 265 – 268, 2015.
- [24]. Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias Höllerer. Springer, “Advances in Location-Based Services”, “I’m Feeling LoCo: A Location-Based Context-Aware Recommendation System.”,2011.

APPENDIX - I

This section contains details on the language, software and packages used in our project.

This project is developed in Python, which is a general-purpose interpreted, interactive, object oriented and high-level programming language.

- **Numpy:** It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, etc. It is the fundamental package for scientific computing in Python.
- **Pandas:** It is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language.
- **Imgaug:** It is a library for image augmentation in machine learning experiments. It supports a wide range of augmentation techniques, allows to easily combine these and to execute them in random order or on multiple CPU cores, has a simple yet powerful stochastic interface and can not only augment images, but also keypoints/landmarks, bounding boxes, heatmaps and segmentation maps.
- **OpenCV-python:** OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a

common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

- **PIL:** The Python Imaging Library was developed by Fredrik Lundh and Contributors. It adds image processing capabilities to the Python interpreter and is designed for fast access to data stored in a few basic pixel formats.
- **Matplotlib:** It is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can create publication quality plots, make interactive figures that can zoom, pan, update, customize visual style and layout, export to many file formats and can be embedded in JupyterLab and Graphical User Interfaces.
- **Keras:** It is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.
- **Tensorflow:** It is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and develop ML powered application.

- **Scikit-learn:** It is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and Scipy.
- **SciPy:** It is a free and open-source Python library used for scientific computing and technical computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, signal and image processing, etc. The basic data structure used by SciPy is a multidimensional array provided by the NumPy module.
- **Tkinter:** It is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

APPENDIX - II

Training of the mini-xception model –

```

train_CNN.py - C:\Users\venka\Desktop\New folder\Proposed-Music-Mapping-Algorithm-Based-On-Human-Emotio
File Edit Format Run Options Window Help
"""
Description: Train emotion classification model
"""

from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from load_and_process import load_fer2013
from load_and_process import preprocess_input
from cnn import mini_XCEPTION
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# parameters
batch_size = 32
num_epochs = 10000
input_shape = (48, 48, 1)
validation_split = .2
verbose = 1
num_classes = 7
patience = 50
base_path = '/'

# data generator
data_generator = ImageDataGenerator(
    featurewise_center=False,
    featurewise_std_normalization=False,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=.1,
    horizontal_flip=True)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

# callbacks
log_file_path = base_path + 'emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)

```

Fig A1.1. Training the model

```

train_CNN.py - C:\Users\venka\Desktop\New folder\Proposed-Music-Mapping-Algorithm-Based-On-Human-Emotions-main\train_CNN.py
File Edit Format Run Options Window Help

# callbacks
log_file_path = base_path + 'emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                             patience=int(patience/4), verbose=1)

trained_models_path = base_path + 'mini_XCEPTION'
model_names = trained_models_path + '_(epoch:02d)-(val_acc:.2f).hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
                                   save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

# loading dataset
faces, emotions = load_fer2013()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
xtrain, xtest, ytrain, ytest = train_test_split(faces, emotions, test_size=0.2, shuffle=True)
history = model.fit_generator(data_generator.flow(xtrain, ytrain,
                                                  batch_size=batch_size,
                                                  steps_per_epoch=len(xtrain) / batch_size,
                                                  epochs=num_epochs, verbose=1, callbacks=callbacks,
                                                  validation_data=(xtest, ytest)))

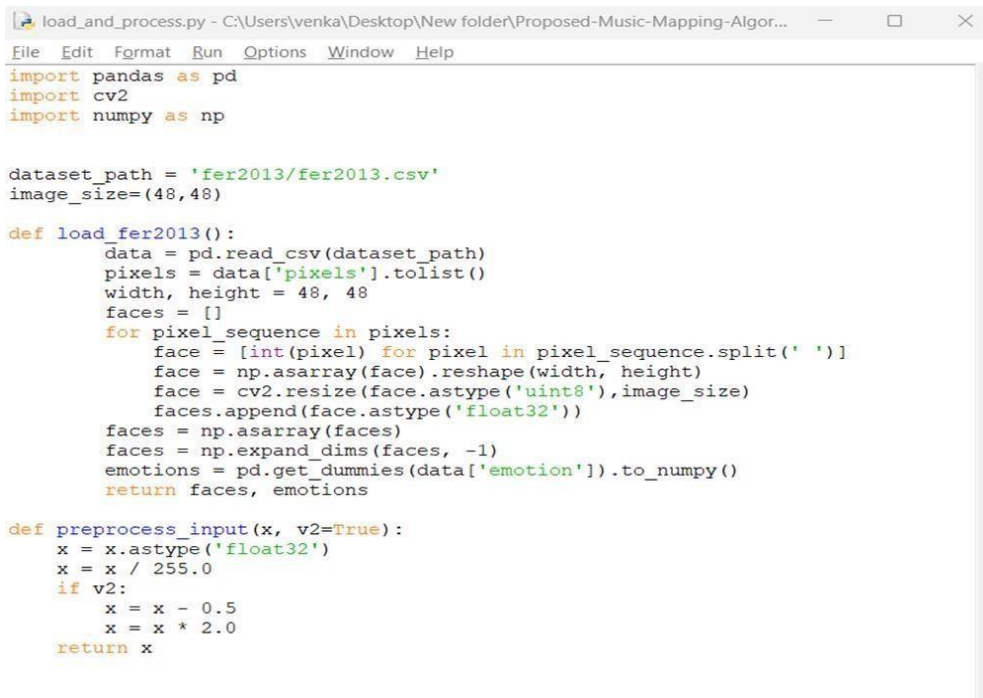
# Loss Curves
plt.figure(figsize=(25, 10))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], 'g', linewidth=1.0)
plt.plot(history.history['val_loss'], 'r', linewidth=1.0)
plt.legend(['Training Loss', 'Validation Loss'], fontsize=14)
plt.xlabel('Epochs', fontsize=16)
plt.ylabel('Loss', fontsize=16)
plt.title('Loss Curves', fontsize=22)

# Accuracy Curves
plt.subplot(1, 2, 2)
plt.plot(history.history['acc'], 'g', linewidth=1.0)
plt.plot(history.history['val_acc'], 'r', linewidth=1.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'], fontsize=14)
plt.xlabel('Epochs', fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.title('Accuracy Curves', fontsize=22)
plt.show()

```

Fig A1.2. Training the model

Loading the dataset-



```
load_and_process.py - C:\Users\venka\Desktop\New folder\Proposed-Music-Mapping-Algor...
File Edit Format Run Options Window Help

import pandas as pd
import cv2
import numpy as np

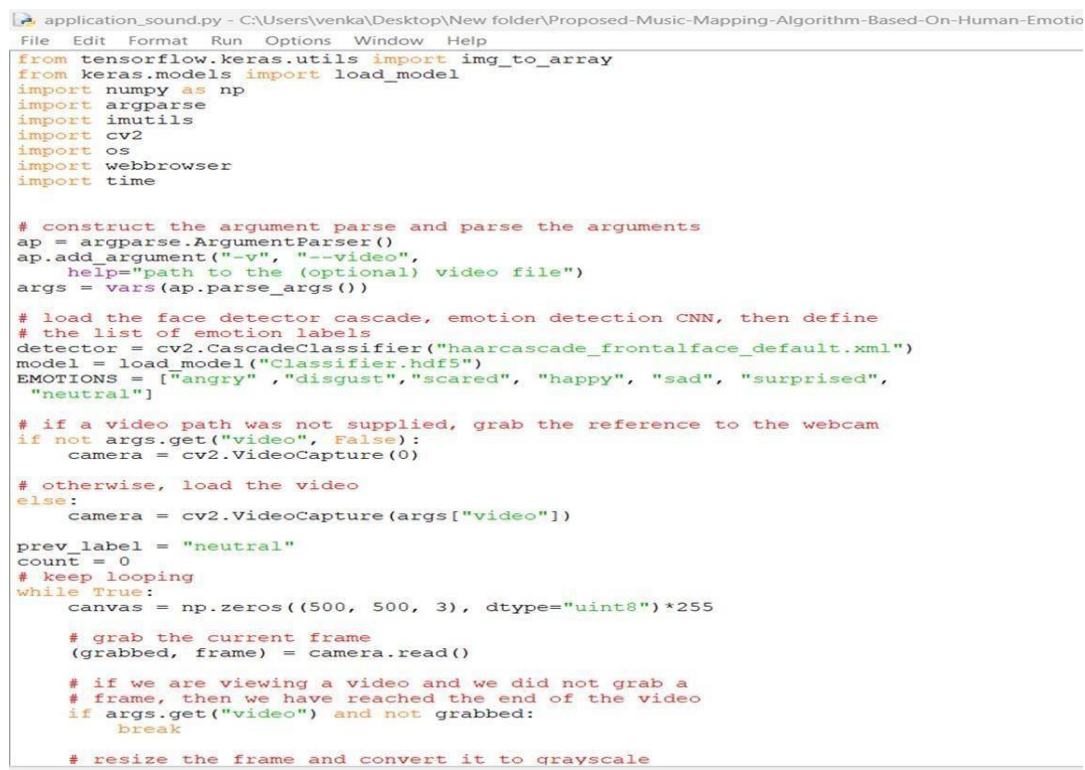
dataset_path = 'fer2013/fer2013.csv'
image_size=(48,48)

def load_fer2013():
    data = pd.read_csv(dataset_path)
    pixels = data['pixels'].tolist()
    width, height = 48, 48
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        face = cv2.resize(face.astype('uint8'), image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    faces = np.expand_dims(faces, -1)
    emotions = pd.get_dummies(data['emotion']).to_numpy()
    return faces, emotions

def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
    return x
```

Fig A2.1. Loading the dataset

Facial recognition and music mapping algorithm-



```
application_sound.py - C:\Users\venka\Desktop\New folder\Proposed-Music-Mapping-Algorithm-Based-On-Human-Emotio
File Edit Format Run Options Window Help

from tensorflow.keras.utils import img_to_array
from keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2
import os
import webbrowser
import time

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video",
    help="path to the (optional) video file")
args = vars(ap.parse_args())

# load the face detector cascade, emotion detection CNN, then define
# the list of emotion labels
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
model = load_model("Classifier.hdf5")
EMOTIONS = ["angry", "disgust", "scared", "happy", "sad", "surprised",
    "neutral"]

# if a video path was not supplied, grab the reference to the webcam
if not args.get("video", False):
    camera = cv2.VideoCapture(0)

# otherwise, load the video
else:
    camera = cv2.VideoCapture(args["video"])

prev_label = "neutral"
count = 0
# keep looping
while True:
    canvas = np.zeros((500, 500, 3), dtype="uint8")*255

    # grab the current frame
    (grabbed, frame) = camera.read()

    # if we are viewing a video and we did not grab a
    # frame, then we have reached the end of the video
    if args.get("video") and not grabbed:
        break

    # resize the frame and convert it to grayscale
```

Fig A3.1. Facial recognition and music mapping algorithm

```

# clone the frame so we can draw on it
frameClone = frame.copy()

# detect faces in the input frame, then clone the frame so that
# we can draw on it
rects = detector.detectMultiScale(gray, scaleFactor=1.1,
    minNeighbors=5, minSize=(30, 30),
    flags=cv2.CASCADE_SCALE_IMAGE)

if len(rects) > 0:

    # determine the largest face area
    rect = sorted(rects, reverse=True,
        key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
    (fx, fy, fw, fh) = rect

    # extract the face ROI from the image, then pre-process
    # it for the network
    roi = gray[fy:fy + fh, fx:fx + fw]
    roi = cv2.resize(roi, (64, 64))
    roi = roi.astype("float") / 255.0
    roi = img_to_array(roi)
    roi = np.expand_dims(roi, axis=0)

    # make a prediction on the ROI, then lookup the class
    # label
    preds = model.predict(roi)[0]
    label = EMOTIONS[preds.argmax()]

    #for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
    #    # construct the label text
    #    text = "{}: {:.2f}%".format(emotion, prob * 100)
    #    prob = max(preds)
    #    prob = prob*100
    #    final_text = str(label) + " - " +str(prob) + " %"

    cv2.putText(frameClone, final_text, (fx, fy - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
    cv2.rectangle(frameClone, (fx, fy), (fx + fw, fy + fh),
        (0, 0, 255), 2)

cv2.imshow('Face', frameClone)

if len(rects) > 0:

```

Fig A3.2. Facial recognition and music mapping algorithm

```

if len(rects) > 0:
    print (label,prev_label)
    if (str(prev_label) == str(label)):
        count += 1
        print (count)
        #prev_label = label
    else :
        count = 0
        prev_label = label

    if count > 10:
        if (label == "happy"):
            cv2.putText(canvas, 'Playing Happy Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow("Face", canvas)
            cv2.waitKey(5000)
            cv2.destroyAllWindows()
            count = 0
            print ("playing Happy songs")
            webbrowser.open('https://youtu.be/DAYSzemgPxc')
            time.sleep(2)

        elif (label == "angry"):
            cv2.putText(canvas, 'Playing Angry Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow("Face", canvas)
            cv2.waitKey(5000)
            cv2.destroyAllWindows()
            count = 0
            print ("Playing angry songs")
            webbrowser.open('https://youtu.be/qrF_Er_OqDI')
            time.sleep(2)

        elif (label == "sad"):
            cv2.putText(canvas, 'Playing Sad Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow("Face", canvas)
            cv2.waitKey(5000)
            cv2.destroyAllWindows()
            count = 0
            webbrowser.open('https://youtu.be/vuek5YvKtHU')
            time.sleep(2)

        elif (label == "scared"):
            cv2.putText(canvas, 'Playing scared Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow("Face", canvas)
            cv2.waitKey(5000)
            cv2.destroyAllWindows()
            count = 0

```

Fig A3.3. Facial recognition and music mapping algorithm

```

if len(rects) > 0:
    print (label,prev_label)
    if (str(prev_label) == str(label)):
        count += 1
        print (count)
        #prev_label = label
    else :
        count = 0
    prev_label = label

if count > 10:
    if (label == "happy"):
        cv2.putText(canvas, 'Playing Happy Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow("Face", canvas)
        cv2.waitKey(5000)
        cv2.destroyAllWindows()
        count = 0
        print ("playing Happy songs")
        webbrowser.open('https://youtu.be/DAYszmgPxc')
        time.sleep(2)

    elif (label == "angry"):
        cv2.putText(canvas, 'Playing Angry Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow("Face", canvas)
        cv2.waitKey(5000)
        cv2.destroyAllWindows()
        count = 0
        print ("Playing angry songs")
        webbrowser.open('https://youtu.be/qrF_Er_OqDI')
        time.sleep(2)

    elif (label == "sad"):
        cv2.putText(canvas, 'Playing Sad Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow("Face", canvas)
        cv2.waitKey(5000)
        cv2.destroyAllWindows()
        count = 0
        webbrowser.open('https://youtu.be/vuek5YvKtHU')
        time.sleep(2)

    elif (label == "scared"):
        cv2.putText(canvas, 'Playing scared Songs', (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow("Face", canvas)
        cv2.waitKey(5000)
        cv2.destroyAllWindows()
        count = 0

```

Fig A3.4. Facial recognition and music mapping algorithm


```

In [ ]: from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
        from keras.callbacks import ReduceLRonPlateau
        from keras.preprocessing.image import ImageDataGenerator
        from load_and_process import load_fer2013
        from load_and_process import preprocess_input
        from cnn import mini_XCEPTION
        from sklearn.model_selection import train_test_split
        import matplotlib.pyplot as plt

# parameters
batch_size = 32
num_epochs = 10000
input_shape = (48, 48, 1)
validation_split = .2
verbose = 1
num_classes = 7
patience = 50
base_path = 'models/'

# data generator
data_generator = ImageDataGenerator(
    featurewise_center=False,
    featurewise_std_normalization=False,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=.1,
    horizontal_flip=True)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

# callbacks
log_file_path = base_path + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLRonPlateau('val_loss', factor=0.1,
                              patience=int(patience/4), verbose=1)
trained_models_path = base_path + '_mini_XCEPTION'
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
                                   save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

```

```

# Loading dataset
faces, emotions = load_fer2013()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
xtrain, xtest, ytrain, ytest = train_test_split(faces, emotions, test_size=0.2, shuffle=True)
history = model.fit_generator(data_generator.flow(xtrain, ytrain,
                                                  batch_size),
                             steps_per_epoch=len(xtrain) / batch_size,
                             epochs=num_epochs, verbose=1, callbacks=callbacks,
                             validation_data=(xtest, ytest))

# Loss Curves
plt.figure(figsize=(25, 10))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], '-g', linewidth=1.0)
plt.plot(history.history['val_loss'], 'r', linewidth=1.0)
plt.legend(['Training loss', 'Validation Loss'], fontsize=14)
plt.xlabel('Epochs ', fontsize=16)
plt.ylabel('Loss', fontsize=16)
plt.title('Loss Curves', fontsize=22)

# Accuracy Curves
plt.subplot(1, 2, 2)
plt.plot(history.history['acc'], '-g', linewidth=1.0)
plt.plot(history.history['val_acc'], 'r', linewidth=1.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'], fontsize=14)
plt.xlabel('Epochs ', fontsize=16)

plt.ylabel('Accuracy', fontsize=16)
plt.title('Accuracy Curves', fontsize=22)
plt.show()

```

```

In [ ]: import pandas as pd
import cv2
import numpy as np

dataset_path = 'fer2013/fer2013.csv'
image_size=(48,48)

def load_fer2013():
    data = pd.read_csv(dataset_path)
    pixels = data['pixels'].tolist()
    width, height = 48, 48
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        face = cv2.resize(face.astype('uint8'), image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    faces = np.expand_dims(faces, -1)
    emotions = pd.get_dummies(data['emotion']).to_numpy()
    return faces, emotions

def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
    return x

```

[illegible]

```

In [ ]: from keras.layers import Activation, Convolution2D, Dropout, Conv2D
        from keras.layers import AveragePooling2D, BatchNormalization
        from keras.layers import GlobalAveragePooling2D
        from keras.models import Sequential
        from keras.layers import Flatten
        from keras.models import Model
        from keras.layers import Input
        from keras.layers import MaxPooling2D
        from keras.layers import SeparableConv2D
        from keras import layers
        from keras.regularizers import l2

def simple_CNN(input_shape, num_classes):

    model = Sequential()
    model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same',
                            name='image_array', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

```

```

model.add(Convolution2D(filters=256, kernel_size=(1, 1), padding='same'))
model.add(BatchNormalization())
model.add(Convolution2D(filters=num_classes, kernel_size=(3, 3),
                        strides=(2, 2), padding='same'))

model.add(Flatten())
#model.add(GlobalAveragePooling2D())
model.add(Activation('softmax', name='predictions'))
return model

def tiny_XCEPTION(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(5, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
              use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(5, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
              use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(8, (1, 1), strides=(2, 2),
                     padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(8, (3, 3), padding='same',
                       kernel_regularizer=regularization,
                       use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(8, (3, 3), padding='same',
                       kernel_regularizer=regularization,
                       use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

```

```

# module 2
residual = Conv2D(16, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(16, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(16, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 3
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

```

```

# module 4
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

x = Conv2D(num_classes, (3, 3),
           #kernel_regularizer=regularization,
           padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax', name='predictions')(x)

model = Model(img_input, output)
return model

def mini_XCEPTION(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
              use_bias=False)(img_input)

    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
              use_bias=False)(x)

    x = BatchNormalization()(x)
    x = Activation('relu')(x)

```



```

# module 1
residual = Conv2D(16, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(16, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(16, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 2
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

```

```

# module 3
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 4
residual = Conv2D(128, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(128, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(128, (3, 3), padding='same',
                   kernel_regularizer=regularization,
                   use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

x = Conv2D(num_classes, (3, 3),
           #kernel_regularizer=regularization,
           padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax', name='predictions')(x)

model = Model(img_input, output)
return model

```

```

def big_XCEPTION(input_shape, num_classes):
    img_input = Input(input_shape)
    x = Conv2D(32, (3, 3), strides=(2, 2), use_bias=False)(img_input)
    x = BatchNormalization(name='block1_conv1_bn')(x)
    x = Activation('relu', name='block1_conv1_act')(x)
    x = Conv2D(64, (3, 3), use_bias=False)(x)
    x = BatchNormalization(name='block1_conv2_bn')(x)
    x = Activation('relu', name='block1_conv2_act')(x)

    residual = Conv2D(128, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block2_sepconv1_bn')(x)
    x = Activation('relu', name='block2_sepconv2_act')(x)
    x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block2_sepconv2_bn')(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    residual = Conv2D(256, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = Activation('relu', name='block3_sepconv1_act')(x)
    x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block3_sepconv1_bn')(x)
    x = Activation('relu', name='block3_sepconv2_act')(x)
    x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block3_sepconv2_bn')(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])
    x = Conv2D(num_classes, (3, 3),
               #kernel_regularizer=regularization,
               padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax', name='predictions')(x)

    model = Model(img_input, output)
    return model

```



```

# keep looping
while True:
    canvas = np.zeros((500, 500, 3), dtype="uint8")*255

    # grab the current frame
    (grabbed, frame) = camera.read()

    # if we are viewing a video and we did not grab a
    # frame, then we have reached the end of the video
    if args.get("video") and not grabbed:
        break

    # resize the frame and convert it to grayscale
    frame = imutils.resize(frame, width=1000)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # clone the frame so we can draw on it
    frameClone = frame.copy()

    # detect faces in the input frame, then clone the frame so that
    # we can draw on it
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)

    if len(rects) > 0:
        # determine the largest face area
        rect = sorted(rects, reverse=True,
            key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
        (fX, fY, fW, fH) = rect

        # extract the face ROI from the image, then pre-process
        # it for the network
        roi = gray[fY:fY + fH, fX:fX + fW]
        roi = cv2.resize(roi, (64, 64))
        roi = roi.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)

        # make a prediction on the ROI, then lookup the class
        # label
        preds = model.predict(roi)[0]
        label = EMOTIONS[preds.argmax()]

```

```

    #for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
        # construct the label text
    #text = "{:}. {:.2f}%".format(emotion, prob * 100)
    prob = max(preds)
    prob = prob*100
    final_text = str(label) + " - " +str(prob) + " %"

    cv2.putText(frameClone, final_text, (fx, fy - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
    cv2.rectangle(frameClone, (fx, fy), (fx + fw, fy + fh),
                    (0, 0, 255), 2)

cv2.imshow('Face', frameClone)

# if the 'q' key is pressed, stop the loop
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

camera.release()
cv2.destroyAllWindows()

```

<p align="center">SRM INSTITUTE OF SCIENCE AND TECHNOLOGY (Deemed to be University u/s 3 of UGC Act, 1956)</p>		
<p align="center">Office of Controller of Examinations</p>		
<p align="center">REPORT FOR PLAGIARISM CHECK ON THE SYNOPSIS/THESIS/DISSERTATION/PROJECT REPORTS</p>		
1	Name of the Candidate (IN BLOCK LETTERS)	CHANDAN SINGH V. HIMAYANTH
2	Address of the Candidate	D-50, Ashokpuram Colony, Daffi, Varanasi, Uttar Pradesh – 221011 Do.No.23-10-71/B, New Maruti Nagar, TIRUPATI, ANDHRA PRADESH – 517501 Mobile Number: 6390131033, 9553763384
3	Registration Number	RA1911031010076 RA1911031010088
4	Date of Birth	19 September 2000 04 December 2001
5	Department	Department of Networking and Communications
6	Faculty	Engineering and Technology
7	Title of the Dissertation/Project	Design of Music Recommendation System Based on Facial Recognition using Mini-Xception CNN

8	Name and address of the Supervisor / Guide (if any)	Dr. B. BALAKIRUTHIGA Assistant Professor, Department of Networking and Communications SRM Institute of Science and Technology Kattankulatur - 603 203. Mail ID: balakirb@srmist.edu.in Mobile Number: 99767 67096
9	Name and address of the Co- Supervisor / Co- Guide (if any)	NIL Mail ID: Mobile Number:
10	Software Used	Turnitin
11	Date of Verification	21 - April - 2023

12	Plagiarism Details: (to attach the final report)			
Chapter	Title of the Chapter	Percentage of similarity index (Including self citation)	Percentage of similarity index (Excluding self-citation)	% of plagiarism after excluding Quotes, Bibliography, etc.,
1	CHAPTER 1	1	1	<1
2	CHAPTER 2	1	1	1
3	CHAPTER 3	1	1	1
4	CHAPTER 4	1	1	1
5	CHAPTER 5	1	1	1
6	CHAPTER 6	1	<1	1
7	CHAPTER 7	1	1	1

8	CHAPTER 8	<1	1	<1
9	CHAPTER 9	<1	<1	<1
Thesis abstract		0	0	0
Appendices		0	0	0
I declare that the above information has been verified and found true to the best of my knowledge.				
Signature of the Candidate		Name & Signature of the Staff (Who uses the plagiarism check software)		
Name & Signature of the Supervisor/Guide		Name & Signature of the Co-Supervisor/Co- Guide		
Name & Signature of the HOD				

Chandan_2

ORIGINALITY REPORT

9%

SIMILARITY INDEX

8%

INTERNET SOURCES

4%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.ijraset.com

Internet Source

5%

2

Submitted to SRM University

Student Paper

2%

3

www.mdpi.com

Internet Source

1%

4

"Deep Sciences for Computing and Communications", Springer Science and Business Media LLC, 2023

Publication

1%

5

scholarworks.waldenu.edu

Internet Source

<1%

Exclude quotes On

Exclude matches < 10 words

Exclude bibliography On

PROOF OF PUBLICATION

Paper submitted to Third International Conference on Artificial Intelligence, 5G Communications and Network Technologies (ICA5NT 2023).



ICA5NT 2023 <ica5nt2023@velammalitech.edu.in>
to me ▾

Fri, Mar 10, 9:41 PM ☆ ↶ ⋮

Dear Authors,

Thanks for your Registration.

We received your payment and Camera ready paper . We will share the Paper Presentation schedule details and virtual presentation link on 21.03.2023.

With Regards,
REGISTRATION TEAM
ICA5NT 2023
Velammal Institute of Technology
www.velammalitech.edu.in

Thank you for your response.

Thanks a lot.

Thank you for the information.

↶ Reply

➦ Forward