PROJECT REPORT
ON
"Music Recommendation Using Facial Expression"

AT

MODEL INSTITUTE OF ENGINEERING AND TECHNOLOGY

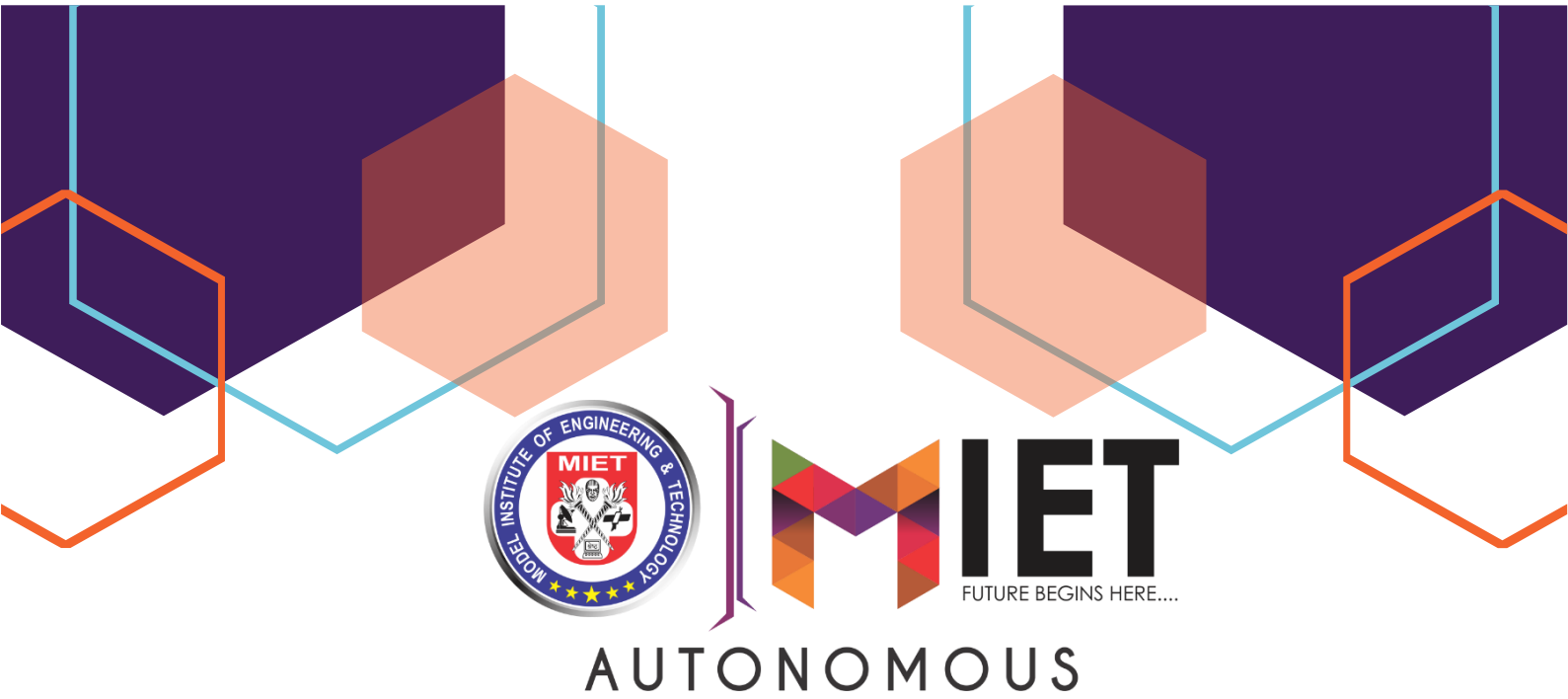
A PROJECT REPORT SUBMITTED

IN PARTIAL FULFILLMENT OF THE REQUIREMENTSFOR THE AWARD OF DEGREE OF

BACHELOR OF ENGINEERING
In
Computer Science

SUBMITTED BY

Archit Gupta	191203079
Saksham Mahajan	191203063
Aadish Bhat	191203011
Ansh Khajuria	191203009



SUBMITTED TO

Department Name
Model Institute of Engineering and Technology (Autonomous)
Jammu, India

2022

CANDIDATES' DECLARATION

We, **Archit Gupta, Saksham Mahajan, Ansh Khajuria, Aadish Bhat** of final semester B.Tech., in the department of Computer Science Engineering from MIET, Jammu, hereby declare that the project work entitled **MUSIC RECOMMENDATION USING FACIAL EXPRESSIONS** is carried out by us and submitted in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science Engineering , under Shafalika Vijayal, Model Institute of Engineering and Institute during the academic year 2019-2023and has not been submitted to any other university for the award of any kind of degree.

Archit Gupta	**191203079**
Saksham Mahajan	**191203063**
Aadish Bhat	**191203011**
Ansh Khajuria	**191203009**

Computer Science
Model Institute of Engineering and Technology (Autonomous) Kot Bhalwal, Jammu, India
(NAAC "A" Grade Accredited)

CERTIFICATE

This is to certify that the project report entitled "GENERATING A PLAYLIST USING FACIAL EXPRESSIONS" submitted by Archit Gupta (191203079), Ansh Khajuria (191203079), Saksham Mahajan (191203079), Aadish Bhat (191203079) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering of Model Institute of Engineering and technology.

ACKNOWLEDGEMENTS

We would like to express our deep gratitude to our project guide Shafalika Vijayal, Assistant Professor, Department of Computer Science and Engineering, MIET, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to DR Ashok Kumar, Head of the Department, Computer Science and Engineering, for providing us with the required facilities for the completion of the project work. We are very much thankful to the Principal and Management, MIET, Jammu, for their encouragement and cooperation to carry out this work. We thank all teaching faculty of Department of CSE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank Department of CSE, MIET for providing great assistance in accomplishment of our project. We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

Our Project Student: -

Archit Gupta	**191203079**
Saksham Mahajan	**191203063**
Aadish Bhat	**191203011**
Ansh Khajuria	**191203009**

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

We propose a new approach for playing music automatically using facial emotion. Most of the existing approaches involve playing music manually, using wearable computing devices, or classifying based on audio features. Instead, we propose to change the manual sorting and playing. We have used a Convolutional Neural Network for emotion detection. For music recommendations, Pygame & Tkinter are used. Our proposed system tends to reduce the computational time involved in obtaining the results and the overall cost of the designed system, thereby increasing the system's overall accuracy. Testing of the system is done on the FER2013 dataset. Facial expressions are captured using an inbuilt camera. Feature extraction is performed on input face images to detect emotions such as happy, angry, sad, surprise, and neutral. Automatically music playlist is generated by identifying the current emotion of the user. It yields better performance in terms of computational time, as compared to the algorithm in the existing literature.

1. INTRODUCTION

Music plays an important role in our daily life. Users have to face the task of manually browsing the music.

Computer vision is a field of study which encompasses on how computer see and understand digital images and videos.

Computer vision involves seeing or sensing a visual stimulus, make sense of what it has seen and also extract complex information that could be used for other machine learning activities.

We will implement our use case using the Haar Cascade classifier. Haar Cascade classifier is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, **"Rapid Object Detection using a Boosted Cascade of Simple Features"** in 2001.

This project recognizes the facial expressions of user and play songs according to emotion. Facial expressions are best way of expressing mood of a person. The facial expressions are captured using a webcam and face detection is done by using Haar cascade classifier.

The captured image is input to CNN which learn features and these features are analyzed to determine the current emotion of user then the music will be played according to the emotion. In this project, five emotions are considered for classification which includes happy, sad, anger, surprise, neutral. This project consists of 4 modules-face detection, feature extraction, emotion detection, songs classification. Face detection is done by Haar cascade classifier, feature extraction and emotion detection are done by CNN. Finally, the songs are played according to the emotion recognized.

Convolutional Neural Networks (CNN) is a specific type of Artificial Neural Network which are widely used for image classification.

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification.

A convolution layer plays a key role in CNN, which is composed of a stack of

mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

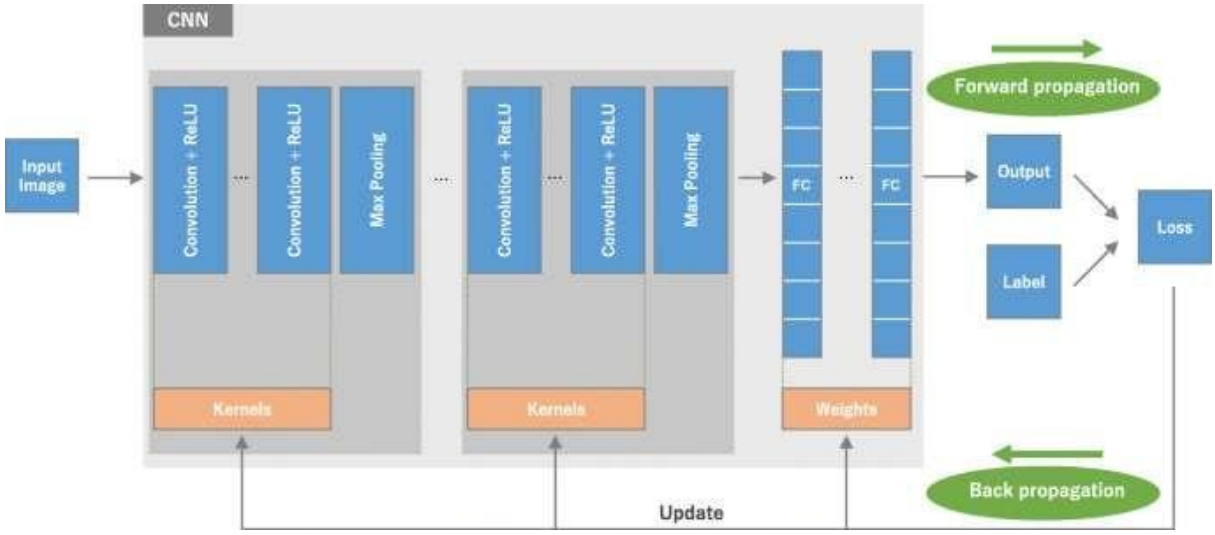


Fig. 1: Basic CNN

Applications of Computer Vision:

1. Autonomous Vehicles.
2. Facial Recognition.
3. Image Search and Object Recognition.

Advantages of Computer Vision:

1. Faster and simpler process
2. Better products and services
3. Cost-reduction

Disadvantages of Computer Vision:

1. Lack of specialists

2. Need for regular monitoring

Applications of CNN:

1. Decoding Facial Recognition.

2. Analyzing Documents.

3. Historic and Environmental Collections.

4. Understanding Climate.

5. Advertising.

Advantages of CNN:

1. Processing speed.
2. Flexibility.
3. Versatile in nature.
4. Dynamic Behavior.
5. Speed
6. Robustness.

Disadvantages of CNN:

3. CNN do not encode the position and orientation of object.

4. Lack of ability to be spatially invariant to the input data

5. Lots of training data is required

1.1 MACHINE LEARNING

Machine Learning is the most popular technique of predicting or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and analyze the historical data. Simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result. As it trains over the examples, again and again, it is able to identify patterns in order to make decisions more accurately. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

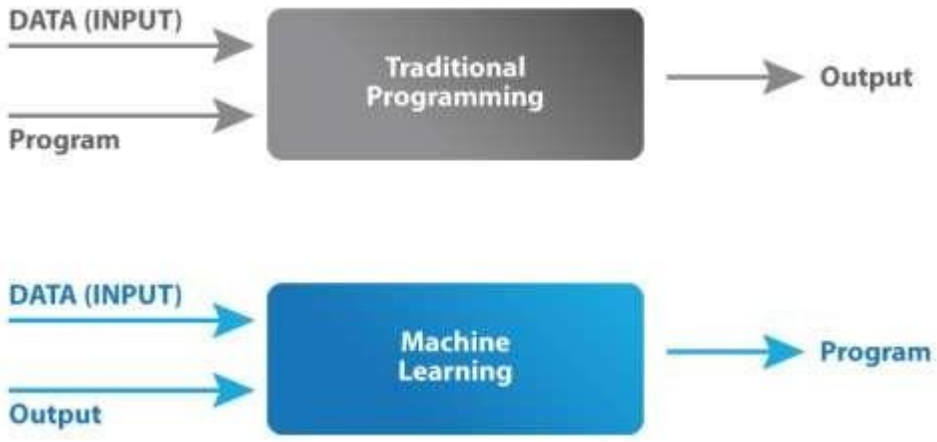


Fig. 2: Traditional Programming and Machine Learning

TYPES OF LEARNINGS:

Machine Learning Algorithms can be classified into 3 types as follows:

1. Supervised learning
2. Unsupervised Learning
3. Reinforcement Learning

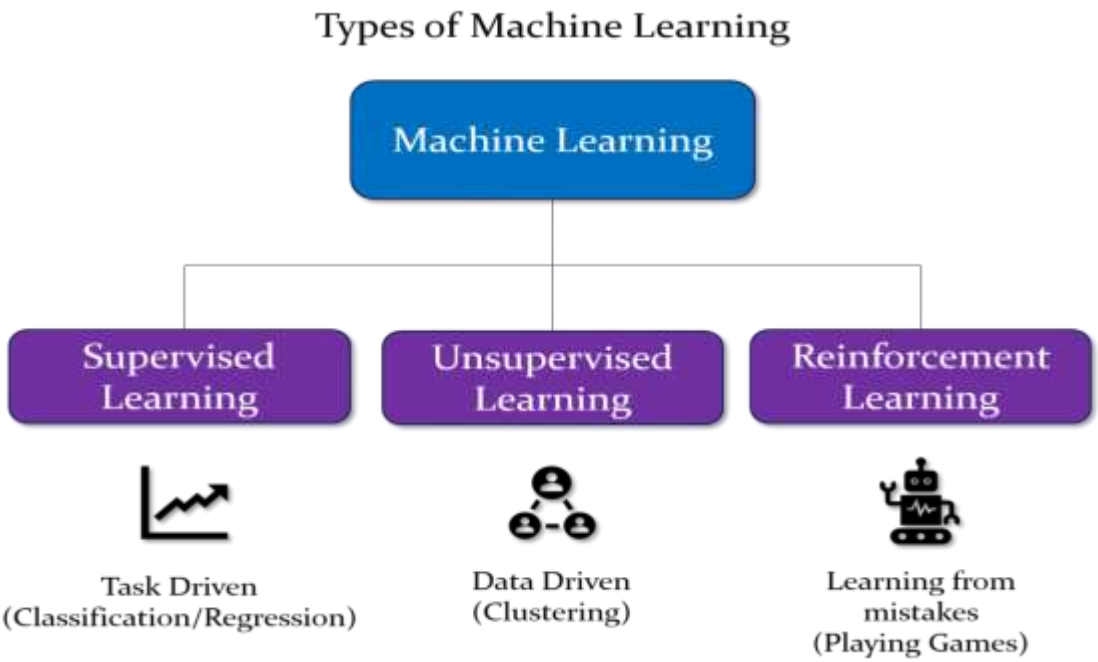


Fig. 3: Types of Machine Learning

SUPERVISED LEARNING:

Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. It is the task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labelled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. Supervised Learning is very similar to teaching a child with the given data and that data is in the form of examples with labels, we can feed a learning algorithm with these example-label pairs one by one, allowing the algorithm to predict the right answer or not. Over time, the algorithm will learn to approximate the exact nature of the relationship between examples and their labels. When fully trained, the supervised learning algorithm will be able to observe a new, never-before-seen example and predict a good label for it.

Most of the practical machine learning uses supervised learning. Supervised learning is where you have input variable (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(x)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for the data. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. Supervised learning is often described as task oriented. It is highly focused on a singular task, feeding more and more examples to the algorithm until it can accurately perform on that task. This is the learning type that you will most likely encounter, as it is exhibited in many of the common applications like Advertisement Popularity, Spam Classification, face recognition.

Two types of Supervised Learning are:

1. Regression:

Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Regression can be used to estimate/ predict continuous values (Real valued output). For example, given a picture of a person then we have to predict the age on the basis of the given picture.

2. Classification:

Classification means to group the output into a class. If the data is discrete or categorical then it is a classification problem. For example, given data about the sizes of houses in the real estate market, making our output about whether the house "sells for more or less than the asking price" i.e., Classifying houses into two discrete categories.

UNSUPERVISED LEARNING

Unsupervised Learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabeled data and looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human labelled data, unsupervised learning, also known as self-organization, allows for modelling of probability densities over inputs.

Unsupervised machine learning algorithms infer patterns from a dataset without reference to known, or labelled outcomes. It is the training of machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore, machine is restricted to find the hidden structure in unlabeled data by our-self. For example, if we provide some pictures of dogs and cats to the machine to categorized, then initially the machine has no idea about the features of dogs and cats so it categorizes them according to their similarities, patterns and differences. The Unsupervised Learning algorithms allows you to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning methods.

Unsupervised learning problems are classified into two categories of algorithms:

- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences. Machine mainly learns from past experiences and tries to perform best possible solution to a certain problem. It is the training of machine learning models to make a sequence of decisions. Though both supervised and reinforcement learning use mapping between input and output, unlike supervised learning where the feedback provided to the agent is correct set of actions for performing a task, reinforcement learning uses rewards and punishments as signals for positive and negative behaviour. Reinforcement learning is currently the most effective way to hint machine's creativity

1.2 NEURAL NETWORKS

Neural Network (or Artificial Neural Network) has the ability to learn by examples. ANN is an information processing model inspired by the biological neuron system. ANN biologically inspired simulations that are performed on the computer to do a certain specific set of tasks like clustering, classification, pattern recognition etc. It is composed of a large number of highly interconnected processing elements known as the neuron to solve problems. It follows the non-linear path and process information in parallel throughout the nodes. A neural network is a complex adaptive system. Adaptive means it has the ability to change its internal structure by adjusting weights of inputs.

Artificial Neural Networks can be best viewed as weighted directed graphs, where the nodes are formed by the artificial neurons and the connection between the neuron outputs and neuron inputs can be represented by the directed edges with weights. The ANN receives the input signal from the external world in the form of a pattern and image in the form of a vector. These inputs are then mathematically designated by the notations x(n) for every n number of inputs. Each of the input is then multiplied by its corresponding weights (these weights are the details used by the artificial neural networks to solve a certain problem). These weights typically represent the strength of the interconnection amongst neurons inside the artificial neural network. All the weighted inputs are summed up inside the computing unit (yet another artificial neuron).

If the weighted sum equates to zero, a bias is added to make the output non-zero or else to scale up to the system's response. Bias has the weight and the input to it is always equal to 1. Here the sum of weighted inputs can be in the range of 0 to positive infinity. To keep the response in the limits of the desired values, a certain threshold value is benchmarked. And then the sum of weighted inputs is passed through the activation function. The activation function is the set of transfer functions used to get the desired output of it. There are various flavours of the activation function, but mainly either linear or non-linear set of functions. Some of the most commonly used set of activation functions are the Binary, Sigmoid (linear) and Tan hyperbolic sigmoidal (non-linear) activation functions.

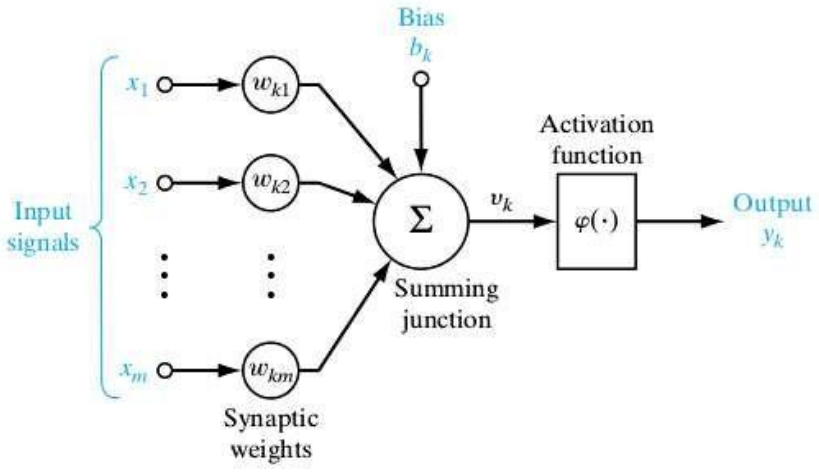


Fig. 4: Basic Neural Network

The Artificial Neural Network contains three layers:

1. **Input Layer:** The input layers contain those artificial neurons (termed as units) which are to receive input from the outside world. This is where the actual learning on the network happens or corresponding happens else it will process.

2. **Hidden Layer:** The hidden layers are mentioned hidden in between input and the output layers. The only job of a hidden layer is to transform the input into something meaningful that the output layer/unit can use in some way. Most of the artificial neural networks are all interconnected, which means that each of the hidden layers is individually connected to the neurons in its input layer and also to its output layer leaving nothing to hang in the air. This makes it possible for a complete learning process and also learning occurs to the maximum when the weights inside the artificial neural network get updated after each iteration.

3. **Output Layer:** The output layers contain units that respond to the information that is fed into the system and also whether it learned any task or not.

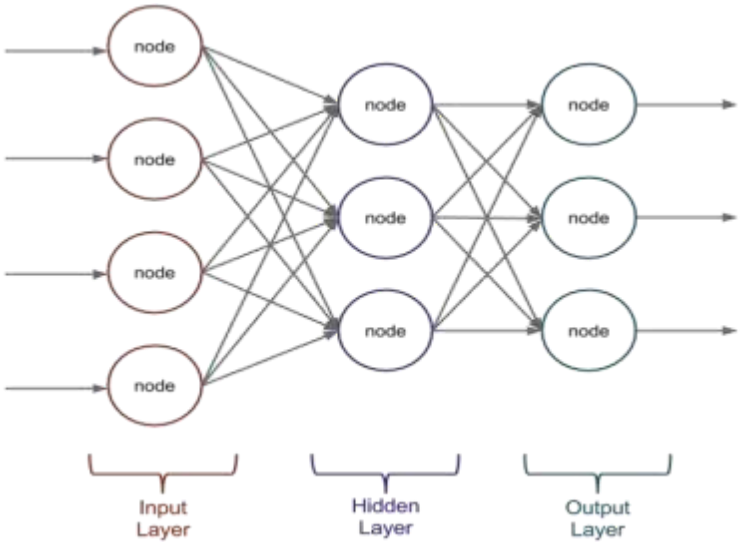


Fig. 5: Layers of Neural Network

Learning process of a neural network:

1. Start with values (often random) for the network parameters (wij weights and bj biases).
2. Take a set of examples of input data and pass them through the network to obtain their prediction.
3. Compare these predictions obtained with the values of expected labels and calculate the loss with them.
4. Perform the backpropagation in order to propagate this loss to each and every one of the parameters that make up the model of the neural network.
5. Use this propagated information to update the parameters of the neural network with the gradient descent in a way that the total loss is reduced, and a better model is obtained.
6. Continue iterating in the previous steps until we consider that we have a good model

1.3 DEEP LEARNING

Deep learning is a branch of machine learning which is completely based on artificial neural networks. Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. It has a greater number of hidden layers and known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing. However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unravelling this wealth of information and are increasingly adapting to AI systems for automated support.

Deep learning learns from vast amounts of unstructured data that would normally take humans decades to understand and process. Deep learning and utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

Deep Neural Network is a neural network with a certain level of complexity (having multiple hidden layers in between input and output layers). They are capable of modelling and processing non-linear relationships

1.4 MOTIVATION FOR THE WORK

As a music listener, I've always felt that music players should do way more things than just playing songs and allowing users to make play-lists. A music playlist should be intelligent and act in keeping with user's preferences. A music playlist should help users organize and list the songs automatically without putting much effort into selection and re-organization of songs. The Generating Playlist Using Facial Expressions provides a better platform to any or all the music listeners, and ensures automation of generating a playlist according the emotion of the user. This helps users to generate playlist according to their moods. It recommends the top playlist based on the mood of the user using Spotify API.

1.5 PROBLEM STATEMENT

Music plays an important part of our life. It gives us relief and reduce the stress. The goal of this project is to generate a music playlist based on the facial expressions of the users. In this we discuss how convolutional neural network (CNN) is used for generating music playlist according to the facial expressions of user.

2. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system.

2.1 Face Expression Recognition Using CNN & LBP:

Visual interaction is an effective means of communication for human beings as social beings. Even a simple change in facial expression signifies happiness, sorrow, surprise and anxiety. The facial expressions of every person should vary in various contexts such as lighting, posture and even background. All these factors still remain an issue while recognizing facial expressions. This paper hopes to bring out a fair comparison between two of the most commonly used face expression recognition [FER] techniques and to shed some light on their precision. The methods being used here are Local binary patterns [LBP] and Convolution neural networks [CNN]. The LBP is meant as a method only for the purpose of extracting features so the Support vector machine [SVM] classifier is being utilized for classifying the extracted features from LBP. The dataset used for the purpose of testing and training in this paper are CK+, JAFFE and YALE FACE.

2.2 A Machine Learning Based Music Player by Detecting Emotions:

This paper constitutes the implementation of Convolutional neural network for the emotion detection and thereby playing a song accordingly. In order to obtain minimal processing, multilayer perceptron are implemented by CNNs. In comparison to various algorithms for image classification, CNNs observed to have little-processing. This implies that the filters used in CNNs are advantageous when compared to traditional algorithm. The visualization of features directly can be less informative. Hence, we use the training procedure of back-propagation to

activate the filters for better visualization. The multiple actions such as capturing, detecting the emotion and classifying the same can all be confined as one step through the use of CNN.

2.3 Emotion-Based Music Player:

This paper proposed an emotion-based music player, which is able to suggest songs based on the user's emotions; sad, happy, neutral and angry. The application receives either the user's heart rate or facial image from a smart band or mobile camera. It then uses the classification method to identify the user's emotion. This paper presents 2 kinds of the classification method; the heart rate-based and the facial image-based methods. Then, the application returns songs which have the same mood as the user's emotion. The user and song emotions in this paper are divided into four types namely: neutral, happy, sad and angry. The experimental results present that detecting the happy emotion is the most precise with around 98%, while the accuracy of the sad mood detection is the lowest with 40%.

2.4 Automatic facial expression recognition using features of salient facial patches:

They proposed a system image from database is passed to the facial landmark detection stage to remove noise by applying Gaussian Filter or mask. Here itself they used Viola Jones technique of Haar-like features with Adaboost learning for face detection. The feature detection stage consists of Eyebrow corners detector, Eye detector, Noise detector, Lip corner detector. After these active facial patches are extracted, the classification of features is done by SVM (Support Vector Machine). While testing it will take the hundreds of images from the database and extract the features and classifies accordingly. They used CK+ (Cohn-Kanade) dataset and JAFEE dataset for training and testing the database. The training database consist of 329 images in total.

2.5 EXISTING SYSTEM

The features available in the existing Music players present in computer systems are as follows: i. Manual selection of Songs ii. Party Shuffle iii. Playlists iv. Music squares where user has to classify the songs manually according to particular emotions for only four basic emotions. Those are Passionate, Calm, Joyful and Excitement.

Using traditional music players, a user had to manually browse through his playlist and select songs that would soothe his mood and emotional experience .In today‘s world, with ever increasing advancements in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed (seek & time compression),local playback, streaming playback with multicast streams and including volume modulation, genre classification etc.

Although these features satisfy the user ‘s basic requirements, yet the user has to face the task of manually browsing through the playlist of songs and select songs based on his current mood and behaviour. That is the requirements of an individual, a user sporadically suffered through the need and desire of browsing through his playlist, according to his mood and emotions.

3. METHODOLOGY

3.1 Proposed System

Convolution neural network algorithm is a multilayer perceptron that is the special design for the identification of two-dimensional image information. It has four layers: an input layer, a convolution layer, a sample layer, and an output layer. In a deep network architecture, the convolution layer and sample layer may have multiple. CNN is not as restricted as the Boltzmann machine, it needs to be before and after the layer of neurons in the adjacent layer for all connections, convolution neural network algorithms, each neuron doesn't need to experience the global image, just feel the local region of the image. In addition, each neuron parameter is set to the same, namely, the sharing of weights, namely each neuron with the same convolution kernels to the deconvolution image. The key era of CNN is the local receptive field, sharing of weights, subsampling by using time or space, with a purpose to extract features and reduce the size of the training parameters. The advantage of CNN algorithm is to avoid the explicit feature extraction, and implicitly to learn from the training data. The same neuron weights on the surface of the feature mapping, thus the network can learn parallel, and reduce the complexity of the network Adopting sub-sampling structure by time robustness, scale, and deformation displacement. Input information and network topology can be a very good match. It has unique advantages in image processing.

3.2 FACE DETECTION:

The Viola-Jones Algorithm, developed in 2001 by Paul Viola and Michael Jones, the Viola-Jones algorithm is an object-recognition framework that allows the detection of image features in real-time. Viola-Jones is quite powerful and its application has proven to be exceptionally notable in real-time face detection. The framework is still a leading player in face detection alongside many of its CNNs counter parts. The Viola-Jones Object Detection Framework combines the concepts of **Haar-like Features, Integral Images, the AdaBoost Algorithm, and the Cascade Classifier** to create a system for object detection that is fast and accurate.

Viola-Jones was designed for frontal faces, so it is able to detect frontal the best rather than faces looking sideways, upwards or downwards. Before detecting a face, the image is converted into grayscale, since it is easier to work with and there's lesser data to process. The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.

Viola-Jones outlines a box (as you can see on the right) and searches for a face within the box. It is essentially searching for these haar-like features, which will be explained later. The box moves a step to the right after going through every tile in the picture. In this case, I've used a large box size and taken large steps for demonstration, but in general, you can change the box size and step size according to your needs. With smaller steps, a number of boxes detect face-like features (Haar-like features) and the data of all of those boxes put together, helps the algorithm determine where the face is.

3.2.1 Haar-like Features

Haar-like features are named after Alfred Haar, a Hungarian mathematician in the 19th century who developed the concept of Haar wavelets (kind of like the ancestor of haar-like features). The features below show a box with a light side and a dark side, which is how the machine determines what the feature is. Sometimes one side will be lighter than the other, as in an edge of an eyebrow. Sometimes the middle portion may be shinier than the surrounding boxes, which can be interpreted as a nose.

There are 3 types of Haar-like features that Viola and Jones identified in their research:

- Edge features
- Line-features
- Four-sided features

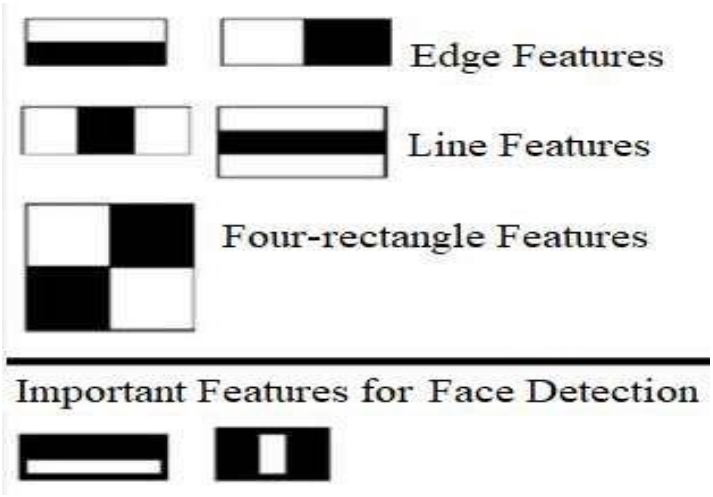


Fig. 6: Haar-like Features

These features help the machine understand what the image is. Imagine what the edge of a table would look like on a black and white image. One side will be lighter than the other, creating that edge like black and white image featureas you can see in the picture above. In the two important features for Face Detection, the horizontal andthe vertical features describe what eyebrows and the nose, respectively, look like to the machine. Additionally, when the images are inspected, each feature has a value of its own. It's quite easy to calculate: Subtract White area from the Black area. For example, look at the image below.

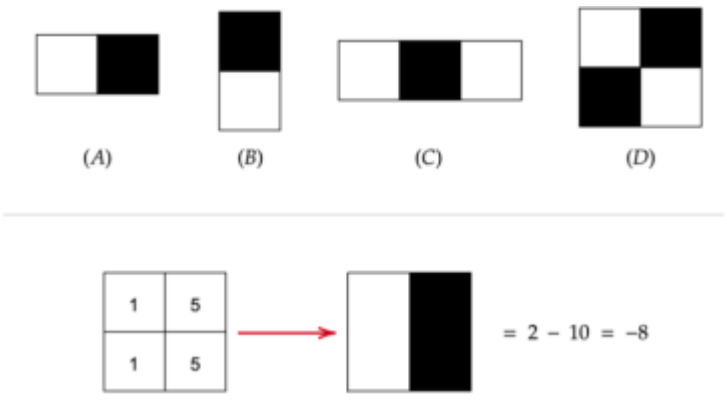


Fig. 7: Feature Value Calculation

3.2.2 Integral Image

we calculated the value of a feature. In reality, these calculations can be very intensive since the number of pixels would be much greater within a large feature. The integral image plays its part in allowing us to perform these intensive calculations quickly so we can understand whether a feature of a number of features fit the criteria. To calculate the value of a single box in the integral image, we take the sum of all the boxes to its left.

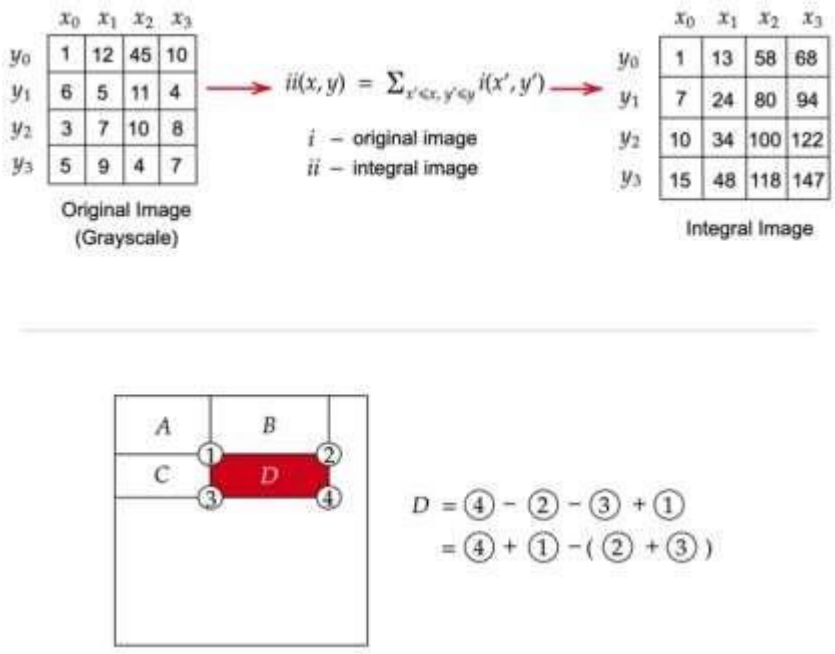


Fig. 8: Integral Image

Use of Integral Image

Haar-like features are actually rectangular, and the integral image process allows us to find a feature within an image very easily as we already know the sum value of a particular square and to find the difference between two rectangles in the regular image, we just need to subtract two squares in the integral image. So even if you had 1000 x 1000 pixels in your grid, the integral image method makes the calculations much less intensive and can save a lot of time for any facial detection model.

3.2.3 Adaptive Boosting (AdaBoost)

The AdaBoost (Adaptive Boosting) Algorithm is a machine learning algorithm for selecting the best subset of features among all available features. The output of the algorithm is a classifier (Prediction Function, Hypothesis Function) called a "Strong Classifier". A Strong Classifier is made up of a linear combination of "Weak Classifiers" (best features). From a high level, in order to find these weak classifiers the algorithm runs for T iterations where T is the number of weak classifiers to find and it is set by you. In each iteration, the algorithm finds the error rate for all features and then choose the feature with the lowest error rate for that iteration.

The algorithm learns from the images we supply it and is able to determine the false positives and true negatives in the data, allowing it to be more accurate. We would get a highly accurate model once we have looked at all possible positions and combinations of those features. Training can be super extensive because of all the different possibilities and combinations you would have to check for every single frame or image.

Let's say we have an equation for our features that determines the success rate (as seen in the image), with f1, f2 and f3 as the features and a1, a2, a3 as the respective weights of the features. Each of the features is known as a weak classifier. The left side of the equation F(x) is called a strong classifier. Since one weak classifier may not be as good, we get a strong classifier when we have a combination of two or three weak classifiers. As you keep adding, it gets stronger and stronger. This is called an ensemble.

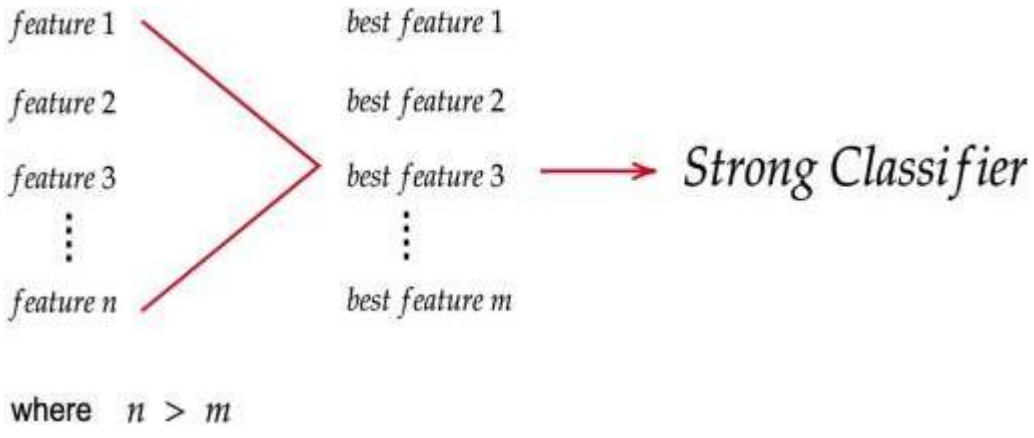


Fig. 9: Adaptive Boosting

3.2.4 The Cascade Classifier

A Cascade Classifier is a multi-stage classifier that can perform detection quickly and accurately. Each stage consists of a strong classifier produced by the AdaBoost Algorithm. From one stage to another, the number of weak classifiers in a strong classifier increases. An input is evaluated on a sequential (stage by stage) basis. If a classifier for a specific stage outputs a negative result, the input is discarded immediately. In case the output is positive, the input is forwarded onto the next stage. According to Viola & Jones (2001), this multi-stage approach allows for the construction of simpler classifiers which can then be used to reject most negative (non face) input quickly while spending more time on positive (face) input.

It is another sort of "hack" to boost the speed and accuracy of our model. So, we start by taking a sub window and within this sub window, we take our most important or best feature and see if it is present in the image within the sub window. If it is not in the sub window, then we don't even look at the sub window, we just discard it. Then if it is present, we look at the second feature in the sub window. If it isn't present, then we reject the sub window. We go on for the number of features have, and reject the sub windows without the feature. Evaluations may take split seconds but since you have to do it for each feature, it could take a lot of time. Cascading speeds up this process a lot, and the machine is able to deliver results much faster.

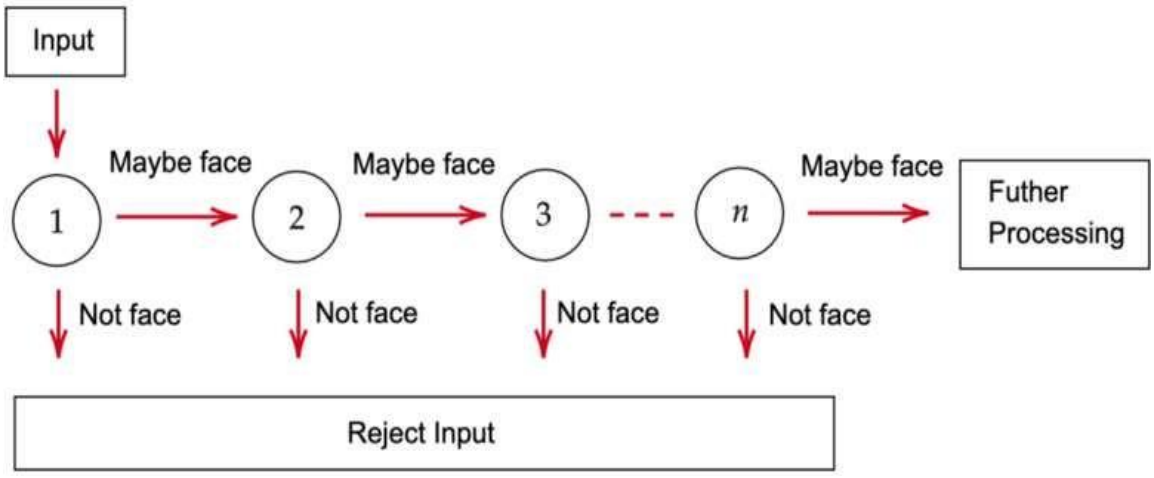


Fig. 10: Cascade Classifier

3.3 FACIAL FEATURE EXTRACTION:

3.3.1 Convolution Neural Network:

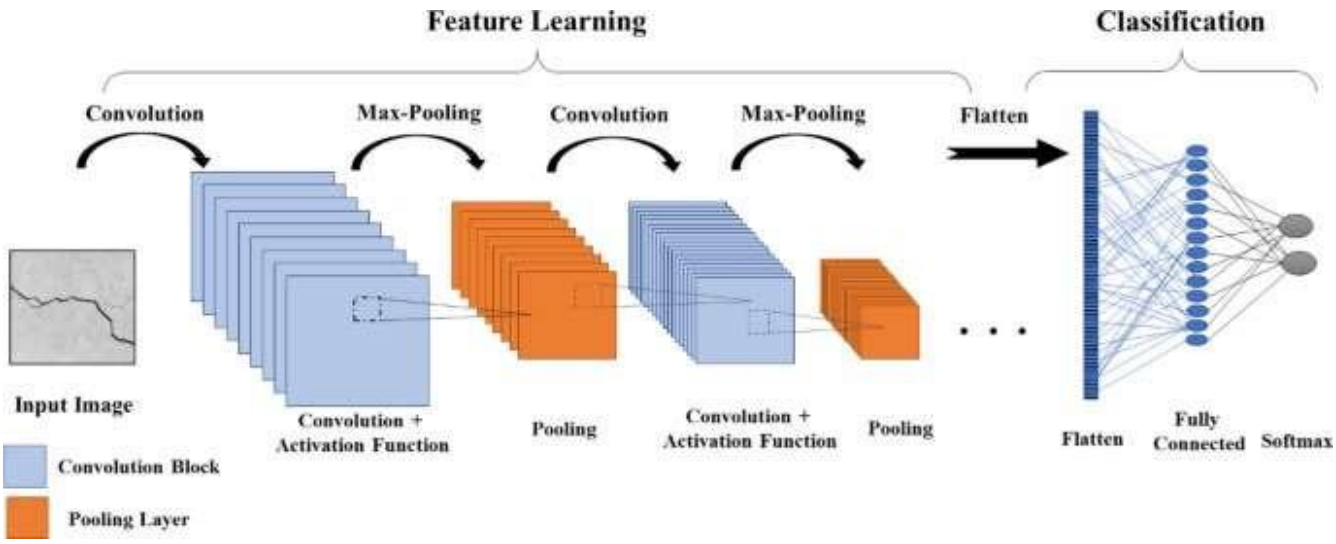


Fig. 11: CNN Architecture

Convolution neural network (CNN) is an efficient recognition algorithm which is widely used in pattern recognition and image processing. It has many features such as simple structure, less training parameters and adaptability.

CNN is a class of **deep learning neural networks**. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from facebook's photo tagging to self-driving cars. They're working hard behind the scenes in everything from healthcare to security. Image classification is the process of taking an **input** (like a picture) and outputting a **class** or a **probability** that the input is a particular class ("there's a 90% probability that this input is an image").

CNNs can be thought of automatic feature extractors from the image. It effectively uses adjacent pixel information to effectively down-sample the image. A CNN, in specific, has one or more layers of convolution units. A convolution unit receives its input from multiple units from the previous layer which together create a proximity. Therefore, the input units (that form a small neighborhood) share their weights.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

3.3.2 Convolution Layer — The Kernel:

The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter. The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well.

Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map. There are a lot of terms being used so let's visualize them one by one.

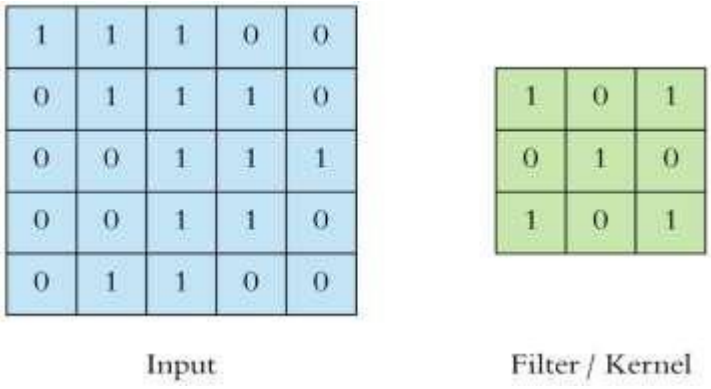


Fig. 12: Input and Filter

We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. The green area where the convolution operation takes place is called the receptive field. Due to the size of the filter the receptive field is also 3x3.

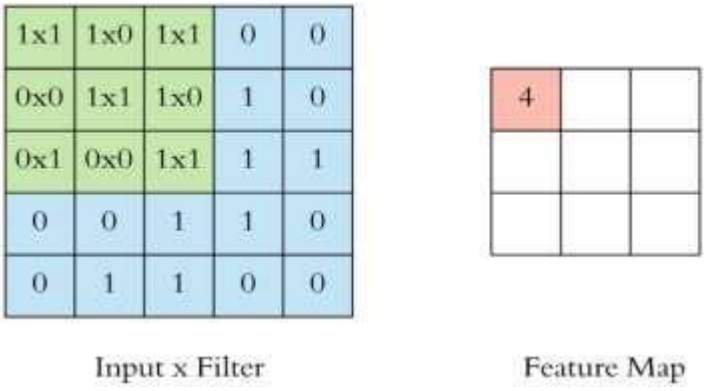


Fig. 13: Feature Map

ReLU (Rectified Linear Unit) activation function which is applied after the convolution operation. It is used for bringing non-linearity to the model. It simply converts the negative values present in the feature map to '0'.

3.3.3 Pooling Layer:

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

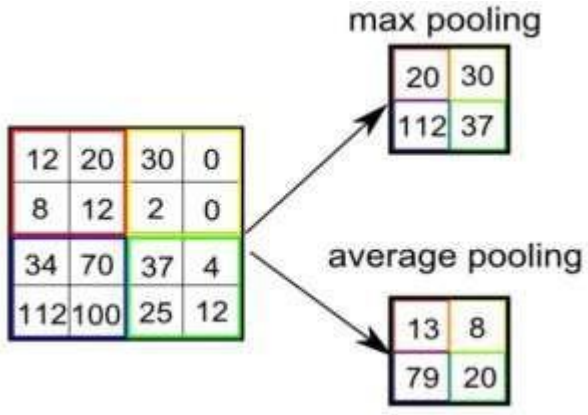


Fig. 14: Pooling

3.3.4 Classification — Fully Connected Layer (FC Layer):

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. Fully Connected Layer is also called as Dense Layer. It provides learning features from all the combinations of the features of the previous layer. The FC layer helps to map the representation between the input and the output.

The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the SoftMax Classification technique.

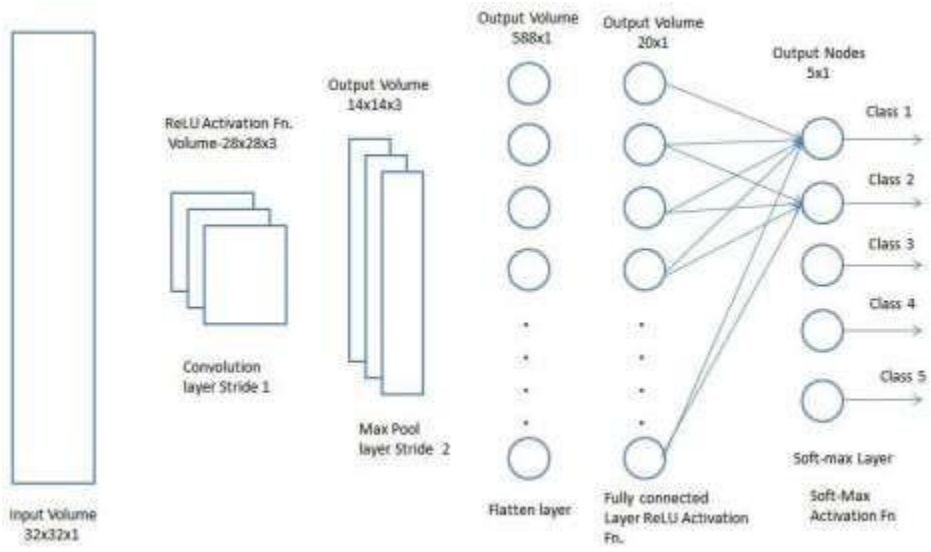


Fig. 15: Fully Connected Layer

WORKING:

1. The input 3-D volume is passed to this layer. The dimension would be H*W*C. H, W, and C represent height, width, and the number of channels respectively.
2. There can be K filter used where K represents the depth of an output volume. The dimension of all the K filters is the same which is f*f*C. f is the filter size and c is the number of channels input image has.
3. If we have configured padding then will add padding to the input volume. If padding is equal to same then will add one row and one column at each side of the dimension and the value would be zero. Padding is applicable only along the height and width of an input dimension and is applied along each layer.
4. After padding, the computation begins. Now we'll slide our filter starting from the top-left corner. The corresponding values of filter and input volume are multiplied and then the summation of all the multiplied value takes place. Now the filter is slide horizontally taking stride number of steps in each slide. So, if stride is 2, we'll slide 2 columns horizontally. The same process is repeated vertically until the whole image is covered.
5. After getting all the values from the filter computation they are passed through Relu activation which is max(0,x). In this, all the negative values obtained are replaced by zero as negative values have no significance in the pixel.
6. Step 4 & 5 generates just one layer of an output volume that is the 3-D input volume is transformed into a 2-D volume.
7. Now, step 4 & 5 gets repeated for K filters. And the output of each filter is stacked above one another and hence the depth of an output image is of dimension k.
8. Now, to calculate the dimension of an output volume we require all the hyperparameters of a convolutional layer. All the filters used at this layer needs to be trained and are initialized with random small numbers.

   The height and weight of an output volume is given by
   height, weight = floor((W+2*P-F)/S +1)
   depth = K (number of filters used)

4. DESIGN

4.1 Structure Chart:

A structure chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name.

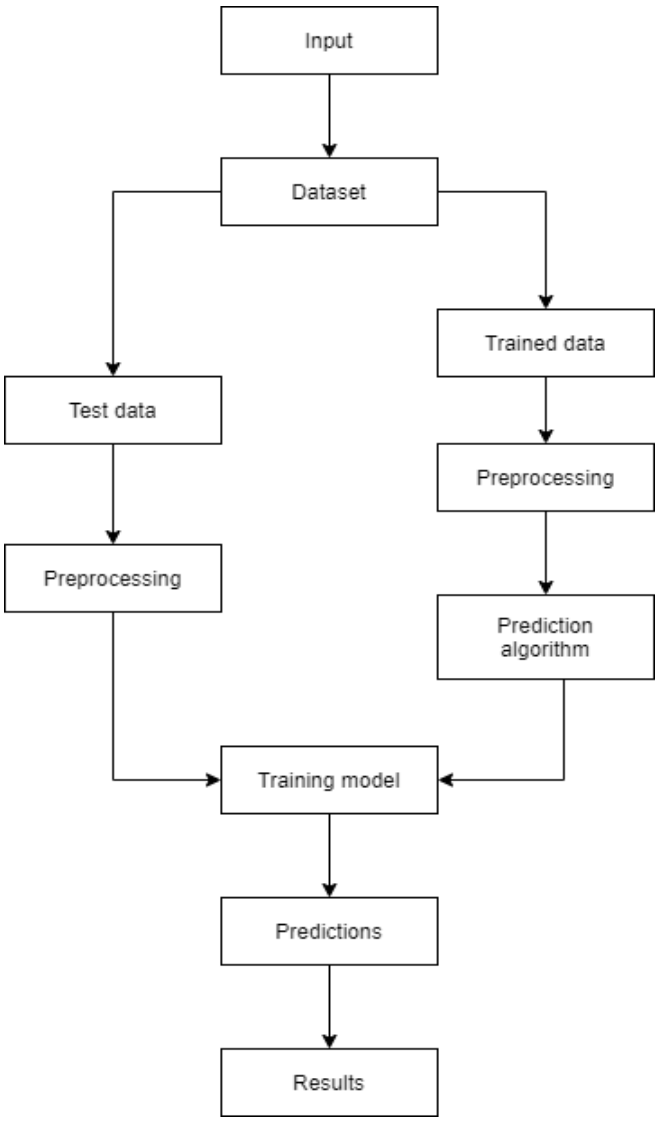


Fig. 16: Structure Chart

4.2 UML Diagrams

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

4.2.1 USE CASE DIAGRAM:

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external system
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

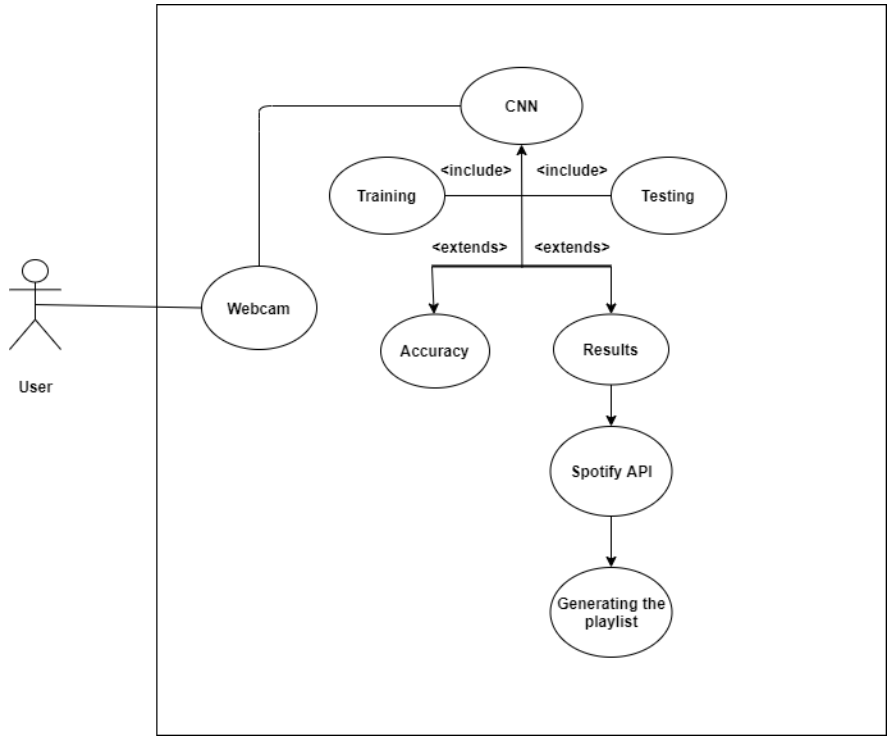


Fig. 17: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM:

In the Unified Modelling Language (UML), A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

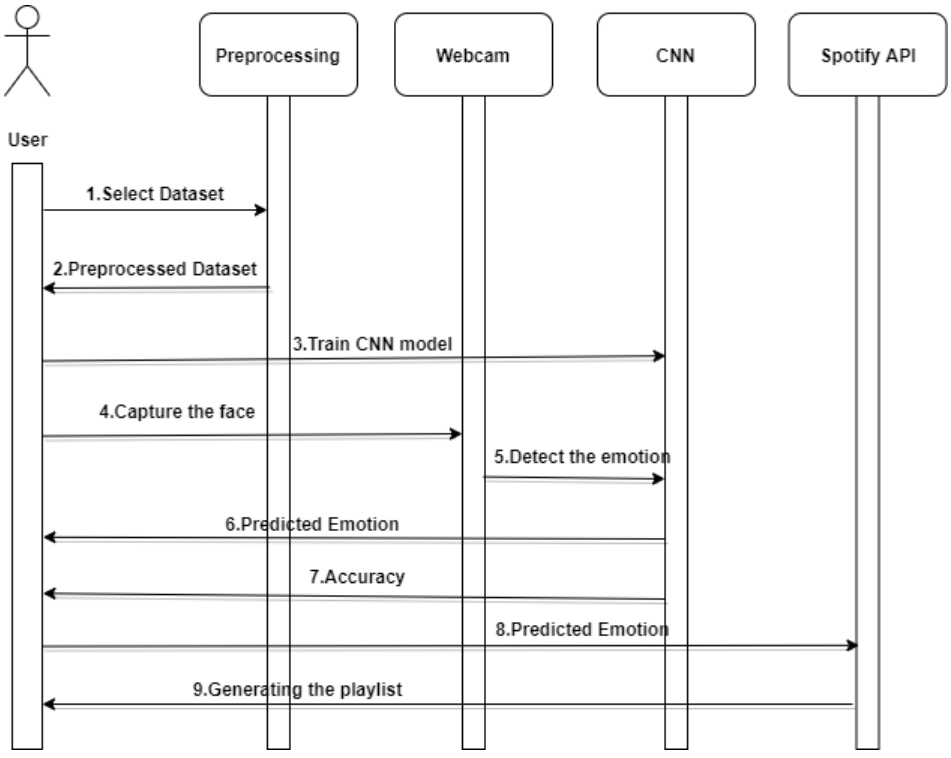


Fig. 18: Sequence Diagram

4.2.3 ACTIVITY DIAGRAM:

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

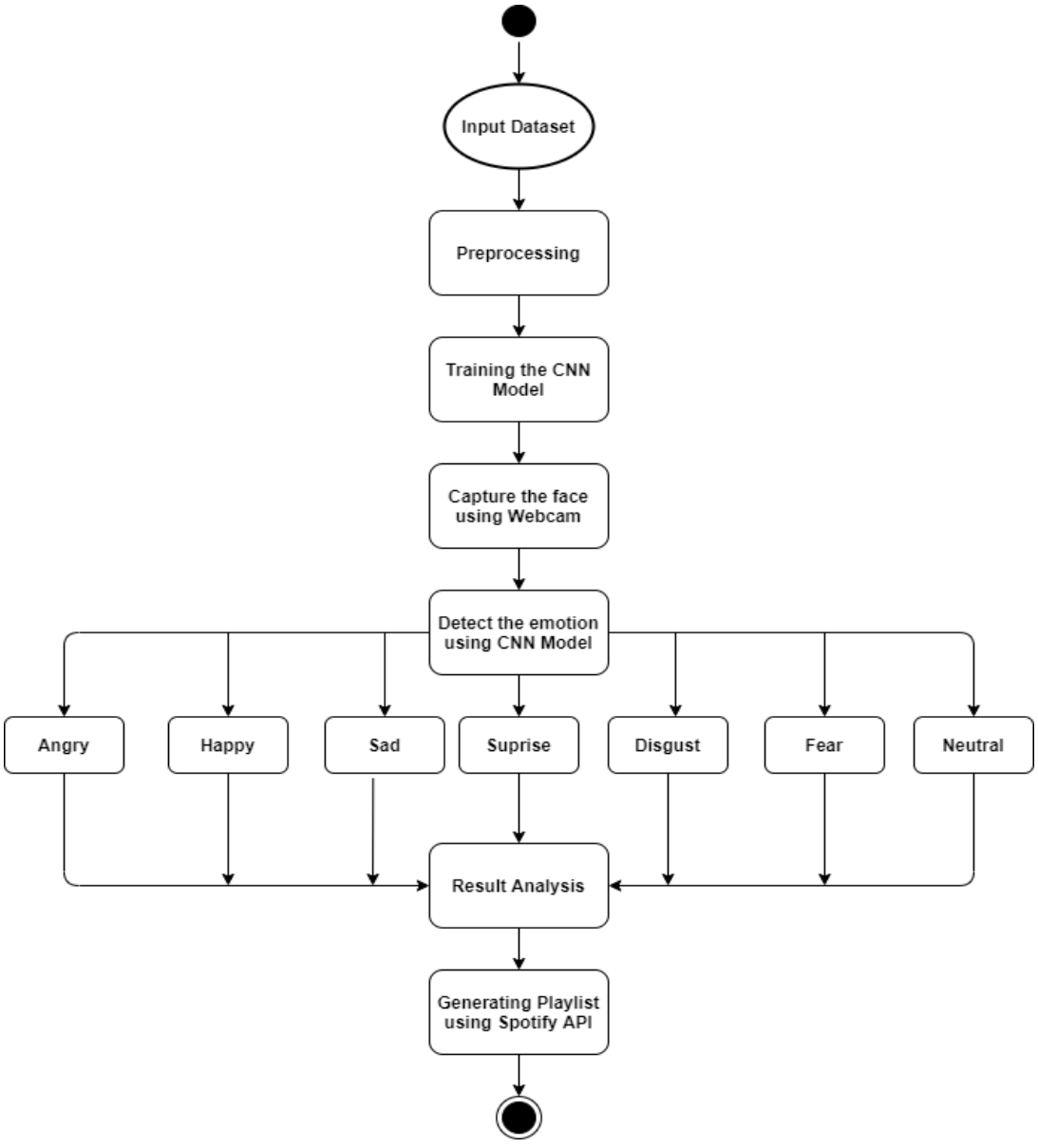


Fig. 19: Activity Diagram

4.2.4 COLLABORATION DIAGRAM:

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different. The collaboration diagrams are best suited for analyzing use cases.

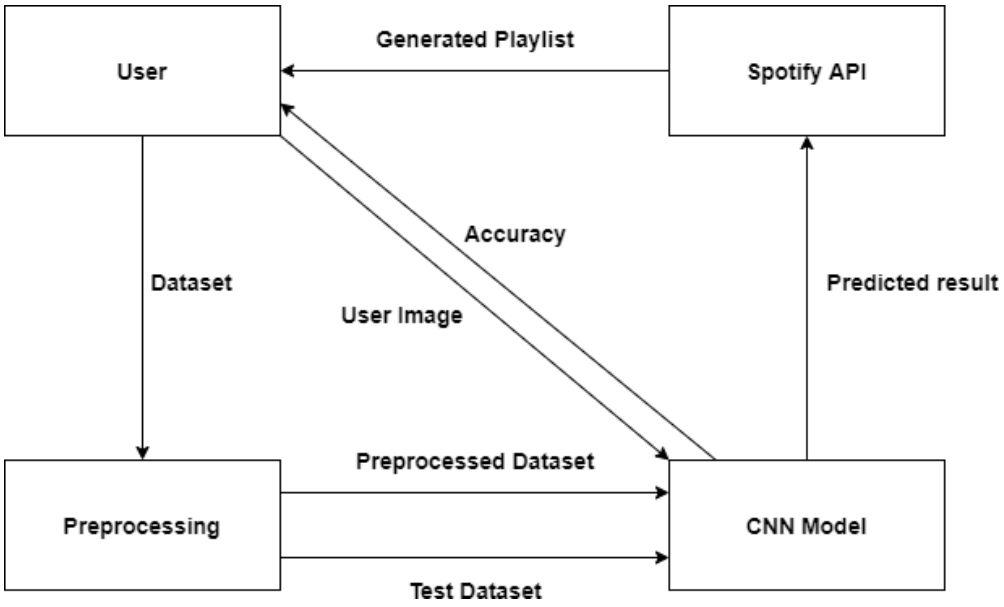


Fig. 20: Collaboration Diagram

4.2.5 FLOW CHART:

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

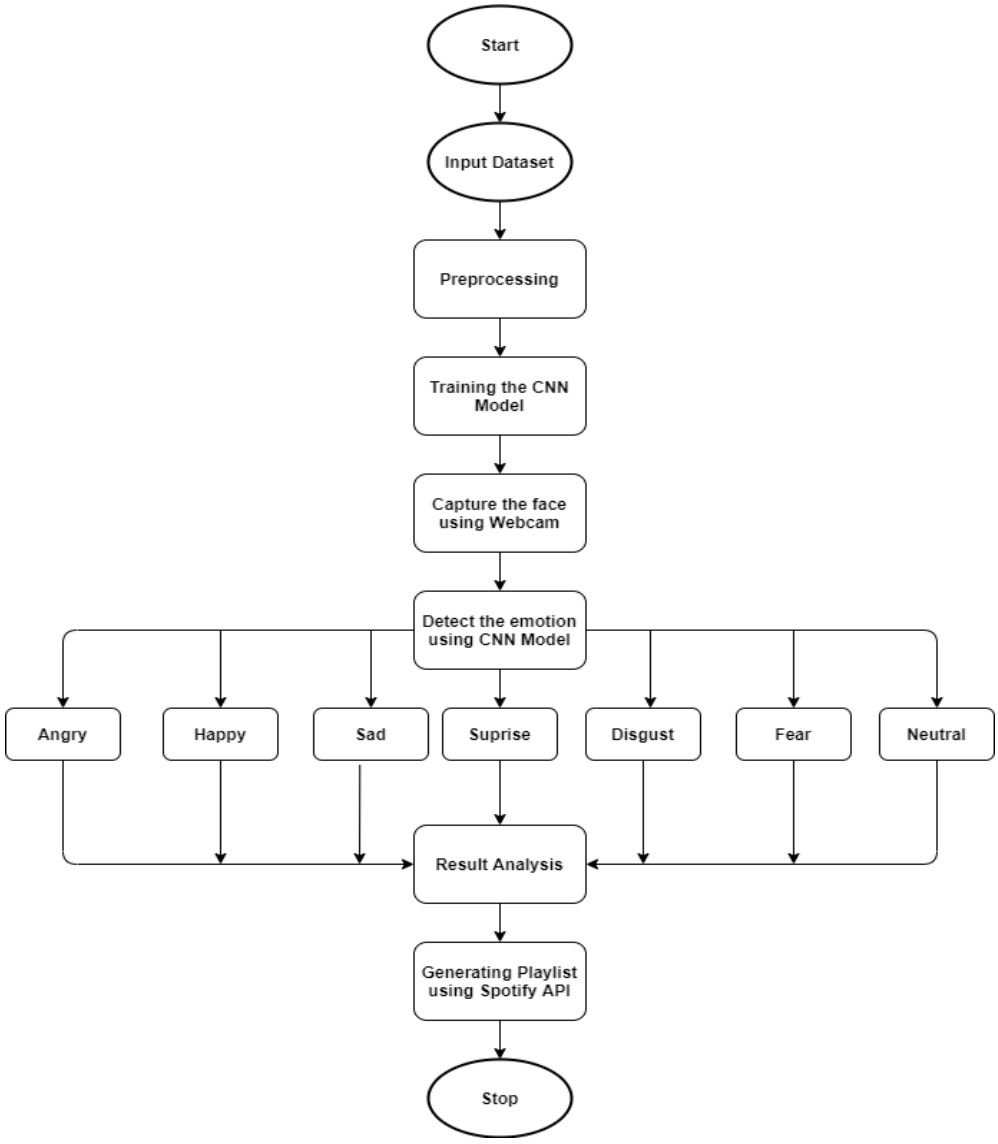


Fig. 21: Flow Chart

4.2.6 COMPONENT DIAGRAM:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

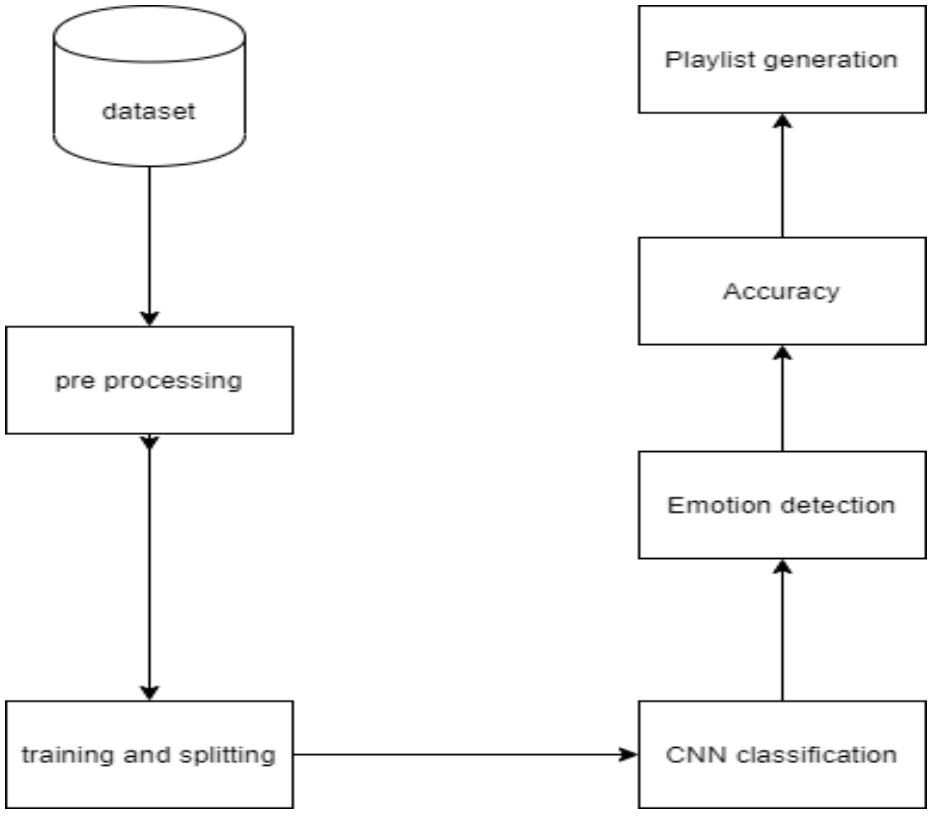


Fig. 22: Component Diagram

5. DATASET DETAILS

Fer-2013 dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest.

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

Fig. 23: Fer-2013.csv

The train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

This dataset consists of 35,887 grayscale images. The training set consists of 28,709 examples. The public test set consists of 3,589 examples.

Fig. 24: Fer-2013 Sample Images

Emotion labels in the dataset:

0: -4593 images- Angry

1: -547 images- Disgust

2: -5121 images- Fear

3: -8989 images- Happy

4: -6077 images- Sad

5: -4002 images- Surprise

6: -6198 images- Neutral

6. EXPERIMENTAL ANALYSIS AND RESULTS

6.1 SYSTEM CONFIGURATION

6.1.1 Software requirements

These are the software configurations used

Operating system: windows 10, Mac OS, Linux.

Libraries: OpenCV, DeepFace, Matplotlib

Python: Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability with its notable use of significant Whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Deep Face: DeepFace is a facial recognition system developed by Facebook's AI research team. It gained significant attention when it was introduced in 2014 due to its impressive accuracy in face recognition tasks. The underlying technology of DeepFace relies on deep learning algorithms, specifically convolutional neural networks (CNNs).The primary objective of DeepFace is to identify and verify human faces in images and videos. It goes beyond simple face detection and delves into facial feature extraction, analysis, and recognition. By leveraging its deep neural network architecture, DeepFace learns hierarchical representations of faces, enabling it to capture complex facial patterns and variations.

OpenCV: OpenCV (*Open-source computer vision*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open-source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules. which is a portable format, compatible with all other formats.

Matplotlib: Matplotlib is a widely-used plotting library in the Python programming language. It provides a flexible and comprehensive set of tools for creating various types of visualizations, ranging from simple line plots to complex 3D plots. Matplotlib is often used in scientific computing, data analysis, and data visualization tasks.

6.1.2 Hardware requirements

These are the Hardware interfaces used

Processor: Intel Pentium 4 or equivalent

RAM: Minimum of 4 GB or higher

HDD: 100 GB or higher

Architecture: 32-bit or 64-bit

Monitor: 15'' or 17'' color monitor

Mouse: Scroll or optical mouse

Keyboard: Standard 110 keys keyboard

6.2 SAMPLE CODE

1. For accessing web cam and taking screenshots of user:

```
import cv2
def screenshot():
    cam = cv2.VideoCapture(0)
    cv2.namedWindow("test")
    #img_counter = 0
    while True:
        ret, frame = cam.read()
        if not ret:
            print("failed to grab frame")
            break
        cv2.imshow("test", frame)

        k = cv2.waitKey(1)
        if k%256 == 27:
            # ESC pressed
            print("Escape hit, closing...")
            break
        elif k%256 == 32:
            # SPACE pressed
            img_name = "test.jpg"
            cv2.imwrite(img_name, frame)
            print("Picture Captured!")
```

```
    cam.release()
    cv2.destroyAllWindows()

if__name__== '__main__':

    screenshot()
```

2. For Preprocessing and Creating the CNN model:

Preprocessing:

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Dense, Activation, Dropout, Flatten

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

#variables
num_classes = 7 #angry, disgust, fear, happy, sad, surprise, neutral
batch_size = 128
epochs = 100
#------------------------------------
```

```
with open("fer2013/fer2013.csv") as f:
  content = f.readlines()

lines = np.array(content)

num_of_instances = lines.size
print("number of instances: ",num_of_instances)
print("instance length: ",len(lines[1].split(",")[1].split(" ")))

#------------------------------
#initialize trainset and test set
x_train, y_train, x_test, y_test = [], [], [], []

#------------------------------
#transfer train and test set data
for i in range(1,num_of_instances):
    try:
        emotion, img, usage = lines[i].split(",")

        val = img.split(" ")

        pixels = np.array(val, 'float32')

        emotion = keras.utils.to_categorical(emotion, num_classes)

        if 'Training' in usage:
```

```
            y_train.append(emotion)
            x_train.append(pixels)
        elif 'PublicTest' in usage:
            y_test.append(emotion)
            x_test.append(pixels)
    except:
      print("", end="")

#------------------------------
#data transformation for train and test sets
x_train = np.array(x_train, 'float32')
y_train = np.array(y_train, 'float32')
x_test = np.array(x_test, 'float32')
y_test = np.array(y_test, 'float32')

x_train /= 255 #normalize inputs between [0, 1]
x_test /= 255

x_train = x_train.reshape(x_train.shape[0], 48, 48, 1)
x_train = x_train.astype('float32')
x_test = x_test.reshape(x_test.shape[0], 48, 48, 1)
x_test = x_test.astype('float32')

print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

CNN Structure:

```
#construct CNN structure
model = Sequential()

#1st convolution layer
model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(5,5), strides=(2, 2)))

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(AveragePooling2D(pool_size=(3,3), strides=(2, 2)))

model.add(Flatten())

#fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_classes, activation='softmax'))
```

```
#.....

#batch process

gen = ImageDataGenerator()

train_generator = gen.flow(x_train, y_train, batch_size=batch_size)


#.....


model.compile(loss='categorical_crossentropy'

    , optimizer=keras.optimizers.Adam()

    , metrics=['accuracy']

)


#.....


#model.fit_generator(x_train, y_train, epochs=epochs) #train for all trainset

model.fit_generator(train_generator, steps_per_epoch=batch_size, epochs=epochs) #train for randomly selected one


#Saving the model

model.save('model100.h5')
```

Evaluation:

```
#Evaluation
train_score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', train_score[0])
print('Train accuracy:', 100*train_score[1])

test_score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', test_score[0])
print('Test accuracy:', 100*test_score[1])
```

3. Using CNN model for detecting the emotion of the user:

```
from keras.models import load_model
model = load_model('model100.h5')

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
import cv2

import spot as sp

def emotion_analysis(emotions):
    objects = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
    y_pos = np.arange(len(objects))

    plt.bar(y_pos, emotions, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('percentage')
    plt.title('emotion')

    res=(max(emotions))
    j=0
    for i in emotions:
      if(i==res) : break
```

```
        else : j=j+1

    Emotion=str(objects[j])
    print('Emotion Detected : ' + Emotion)
    print('Accuracy : '+ str(res*100))

    plt.show()

    return Emotion

def facecrop(image):
    facedata = "haarcascade_frontalface_default.xml"
    cascade = cv2.CascadeClassifier(facedata)

    img = cv2.imread(image)

    try:

        minisize = (img.shape[1],img.shape[0])
        miniframe = cv2.resize(img, minisize)

        faces = cascade.detectMultiScale(miniframe)

        for f in faces:
            x, y, w, h = [ v for v in f ]
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
```

```
            sub_face = img[y:y+h, x:x+w]



            cv2.imwrite('capture.jpg', sub_face)
            #print ("Writing: " + image)

    except Exception as e:
        print (e)

    #cv2.imshow(image, img)


if__name__== '__main__':

#Testing a file.

    facecrop('test.jpg')

    file = 'capture.jpg'
    true_image = image.load_img(file)
    img = image.load_img(file, grayscale=True, target_size=(48, 48))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    x /= 255
```

```
custom = model.predict(x)
    final_Emotion=emotion_analysis(custom[0])

    x = np.array(x, 'float32')
    x = x.reshape([48, 48]);

    plt.gray()
    plt.imshow(true_image)
    plt.show()

    print('\n.....\n')

    print('Playlists Generated By Using The Emotion : ' + final_Emotion)
    print('\n.....\n')

    final_list = sp.songs_by_emotion(final_Emotion)
    sp.print_songs(final_list)
```

4. Using Spotify API for generating the playlist according to user emotion:

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

sp=spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id="4e7b220ab55e4887a
6f275a639cd08a7", client_secret="a181a5078b0143b5a43d7f7e2883497f"))

playlist_limit = 5
song_limit_per_playlist = 20

def songs_by_emotion(emotion):
    results = sp.search(q=emotion,type='playlist', limit=playlist_limit)
    gs = []
    for el in results['playlists']['items']:
        temp = {}
        temp['playlist_name'] = el['name']
        temp['playlist_href'] = el['href']
        temp['playlist_id'] = el['id']
        temp['playlist_spotify_link'] = el['external_urls']['spotify']
        gs.append(temp)

    fnl_playlist_songs = gs
    for i in range(0,len(gs)):
        res = sp.playlist(playlist_id = gs[i]['playlist_id'])
        srn = res['tracks']['items'][0:song_limit_per_playlist]
        tlist = []
```

```
        for el in srn:
            tlist.append(el['track']['name'])
        fnl_playlist_songs[i]['playlist_songs'] = tlist
    return fnl_playlist_songs


def print_songs(fnl_playlist_songs):
    for el in fnl_playlist_songs:
        print('playlist_name : ' + str(el['playlist_name']))
        print('playlist_href : ' + str(el['playlist_href']))
        print('playlist_spotify_link : ' + str(el['playlist_spotify_link']))
        print('playlist_songs : ' )
        for i in range(0,len(el['playlist_songs'])):
            print(str(i+1) + ') ' + el['playlist_songs'][i])
        print('-----------------------------------------------------------')
```

5. Main Code:

```
from pywebio import start_server
import pywebio.input as webin
import pywebio.output as webout
from pywebio.session import info as session_info
import base64
import json
from PIL import Image
import io

from keras.models import load_model
model = load_model('model100.h5')

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
import cv2

import spot as sp

def emotion_analysis(emotions):
    objects = ('Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral')
    y_pos = np.arange(len(objects))
```

```
    plt.bar(y_pos, emotions, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('percentage')
    plt.title('emotion')
    plt.savefig('graph.png')

    res=(max(emotions))
    j=0
    for i in emotions:
      if(i==res) : break
      else : j=j+1

    Emotion=str(objects[j])
    accuracy = str(res*100)

    print('Emotion Detected : ' + Emotion)
    print('Accuracy : '+ str(res*100))

    return Emotion, accuracy

def facecrop(image):

    image = image[23:]
    facedata = "haarcascade_frontalface_default.xml"
    cascade = cv2.CascadeClassifier(facedata)

    img_b64decode = base64.b64decode(image) # base64 decoding
```

```
img_array = np.fromstring(img_b64decode,np.uint8) # convert np sequence
img=cv2.imdecode(img_array,cv2.COLOR_BGR2RGB) # Convert Opencv format

try:

    minisize = (img.shape[1],img.shape[0])
    miniframe = cv2.resize(img, minisize)

    faces = cascade.detectMultiScale(miniframe)
    print(faces)
    for f in faces:
        x, y, w, h = [ v for v in f ]
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

        sub_face = img[y:y+h, x:x+w]



        cv2.imwrite('capture.jpg', sub_face)
        #print ("Writing: " + image)

except Exception as e:
    print (e)
```

```
def main():
    webout.put_html("<center style='font-size: 40px;font-weight:bold;margin-bottom:
50px;'>Generating Playlist Using Facial Expression</center>")
    data = webin.file_upload("Select a image:", accept="image/*")
    data = data['dataurl']

    # img = json.loads(info)['dataurl']
    # webout.put_image(data)

    facecrop(data)

    file = 'capture.jpg'
    true_image = image.load_img(file)
    img = image.load_img(file, grayscale=True, target_size=(48, 48))

    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)

    x /= 255

    custom = model.predict(x)

    final_Emotion, final_Accuracy=emotion_analysis(custom[0])

    html = ""
    # webout.put_image(open('graph.png' , 'rb').read())
    blob_img = None
```

```
    with open("graph.png", "rb") as fh:
        blob_img = base64.b64encode(fh.read()).decode()

    img_html = "<center> <h4 style='font-size:30px;font-weight:bold'>Output: </h4> <br/> <br/> <img src='"+data+"' style='float:left;height:350px;width:45%;box-shadow: 10px 10px 5px grey;' /><img src='"+blob_img+"' style='height:350px;width:45%;box-shadow: 10px 10px 5px grey;' /></center><br/><br/><br/>"
    # img_html = "<img src='"+current_directory+"\\graph.png' />"
    html += img_html

    # print(custom[0])

    emotion_html = "<center style='font-size:20px;font-weight:bold'>Emotion Detected: " + final_Emotion + "<br />Accuracy : "+ final_Accuracy +"</center><br />"
    html += emotion_html


    final_list = sp.songs_by_emotion(final_Emotion)

    playlist_html = ""

    playlist_html += '<center><br/>.....<br />'

    playlist_html += "<h2 style='font-weight:bold'>PLAYLISTS GENERATED BY USING THE EMOTION : " + final_Emotion.upper() + "</h2>"
```

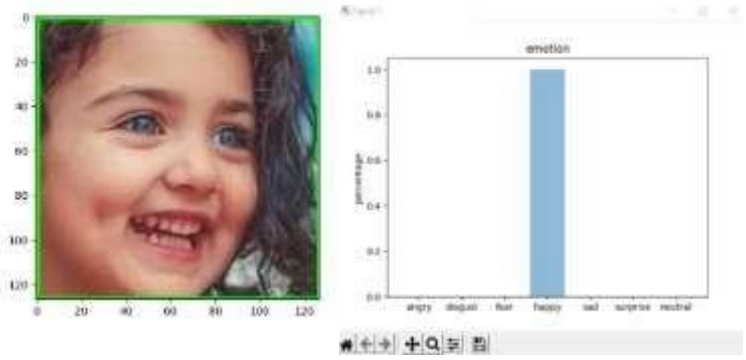
```
    playlist_html +='<br /><center>'

    currentPlaylist = 0

    for el in final_list:
        currentPlaylist += 1

        playlist_html+= "<h2 style='text-align:center;'>PLAYLIST - " +str(currentPlaylist)+ ":</h2>"

        playlist_html+= "<table style='width:80%;padding: 10px;border: 1px solid black;border-collapse: collapse;box-shadow: 10px 10px 5px grey;'>"
        playlist_html+= "<tr>"
        playlist_html+= "<th style='width:8%;text-align:center;border: 1px solid black;border-collapse: collapse;padding: 10px;'><h4>S.no</h4></th>"
        playlist_html+= "<th style='width:92%;text-align:center;border: 1px solid black;border-collapse: collapse;padding: 10px;'><h4>Songs</h4></th>"
        playlist_html+= "</tr>"

        for i in range(0,len(el['playlist_songs'])):

            playlist_html+= "<tr>"
            playlist_html+= "<td style='width:8%;text-align:center;border: 1px solid black;border-collapse: collapse;padding: 10px;'>" + str(i+1) + '. ' "</td>"
            playlist_html+= "<td style='width:92%;text-align:center;border: 1px solid black;border-collapse: collapse;padding: 10px;'>" + el['playlist_songs'][i] + "</td>"
            playlist_html+= "</tr>"
```

```
        playlist_html+= "</table>"

    html += playlist_html
    webout.put_html(html)


if__name__== '__main__':
    start_server(main, debug=True, port=8080, cdn=False)
```

6.3 SAMPLE INPUTS AND OUTPUTS:

1. **Input:** **Output:**

Fig. 25: Input 1



Emotion Detected: Happy

Accuracy: 99.924

S.no	Songs
1.	The Best Days (feat. Tabitha)
2.	Fly Away
3.	Anyone
4.	Lasting Lover
5.	At Least I Had Fun
6.	Good Vibes
7.	Dancing in the Moonlight (feat. NEIMY)
8.	Slow Dance (feat. Ava Max) - Sam Feldt Remix
9.	Holy (feat. Chance The Rapper)
10.	Love You Better
11.	Wonder
12.	Together
13.	Dive
14.	I Don't Know Why
15.	Drown (feat. Clinton Kane)
16.	Watermelon Sugar
17.	Get To Know You
18.	I Found You
19.	Run
20.	Lifestyle (feat. Adam Levine)

Fig. 26: Output 1

2. Input: Output:

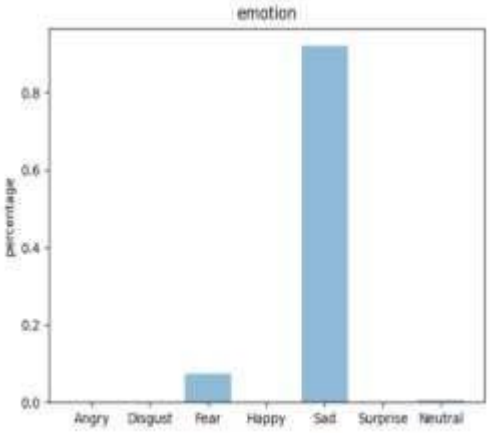


Fig. 27: Input 2

Emotion Detected: Sad

Accuracy: 92.054

S.no	Songs
1.	Love Is Gone - Acoustic
2.	Walked Through Hell
3.	enough for you
4.	Astronomy
5.	Broken
6.	all i need (the distance song)
7.	Arcade
8.	Hold Me Like You Used To
9.	Changes
10.	America's Sweetheart
11.	Remember That Night?
12.	The Night We Met (feat. Phoebe Bridgers)
13.	deja vu
14.	Unlearn (with Gracie Abrams)
15.	The Worst Of You
16.	Unstable (feat. The Kid LAROI)
17.	Move It to the Side
18.	Cry Over Boys
19.	Talking to Myself
20.	Closure

Fig. 28: Output 2

6.4 PERFORMANCE MEASURE

- Number of instances: 35888.
- Instance length: 2304
- 28709 train samples
- 3589 test samples

Accuracy Measure:

1. **Emotion Detected: Surprise**
   Accuracy: 99.489

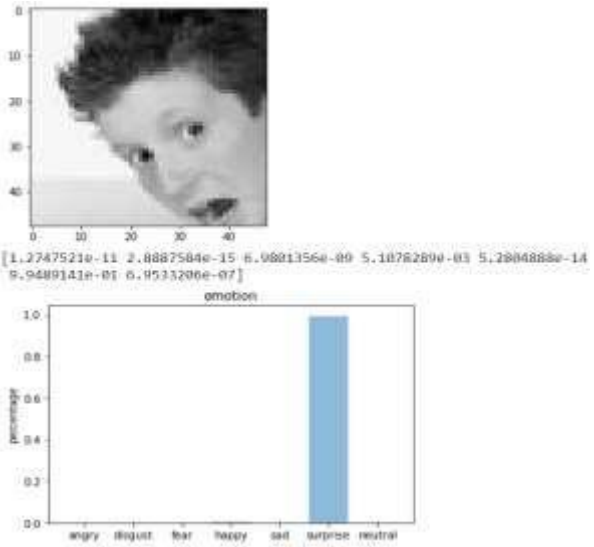


Fig. 29: Accuracy 1

2. **Emotion Detected: Sad**
   Accuracy: 99.99

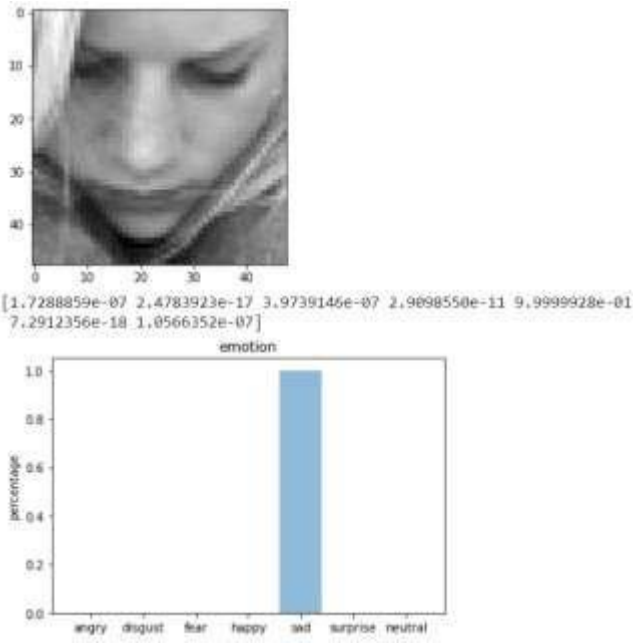


Fig. 30: Accuracy 2

3. **Emotion Detected: Angry**

   Accuracy: 99.612

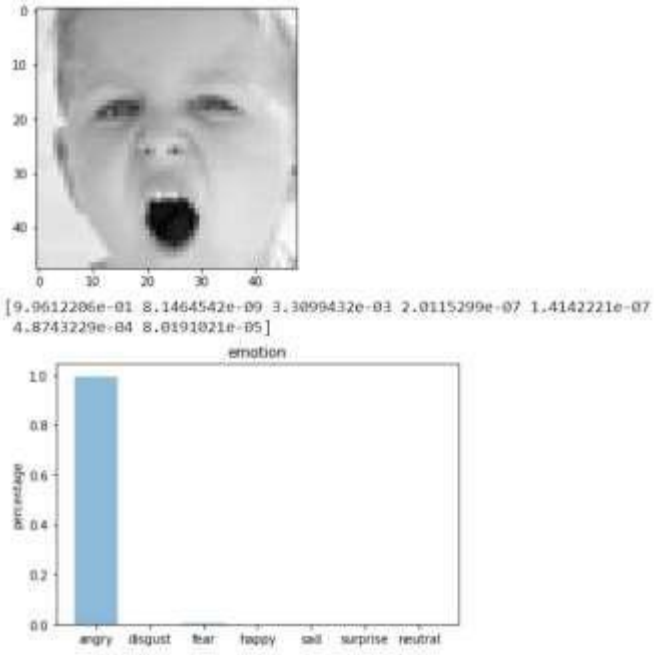


Fig. 31: Accuracy 3

4. **Emotion Detected: Fear**

   Accuracy: 84.611

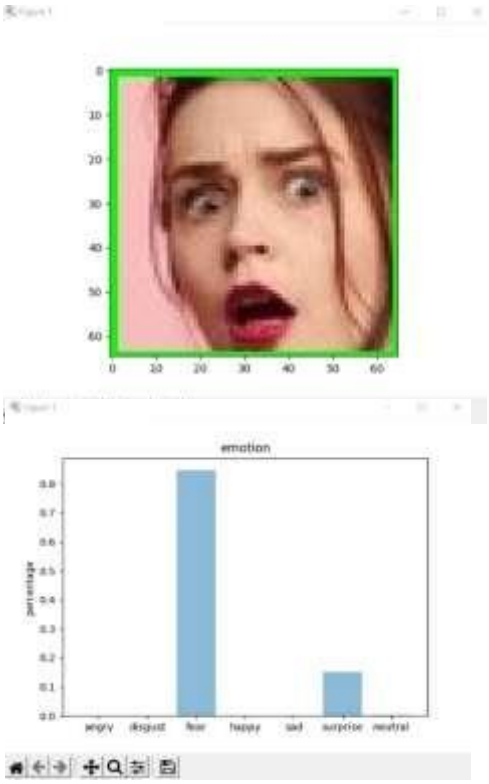


Fig. 32: Accuracy 4

In the accuracy, the term ETA usually refers to estimated time of arrival but in technology realm it refers as estimated completion time of a computation process in general. The problem is too specific to estimating completion time of a batch of long scripts running parallel to each other, processing data and preparing some lists. The running time for each can be varied depending on the past data we considered.

The term Epoch is once all the images are processed one time individually of forward and backward to the network. Usually, we feed a neural network the training data for more than one epoch in different patterns by which a better generalization can be there when an unseen input data is given. If there is a large but finite training dataset then it gives the network a chance to see the previous data to readjust the model parameters so that the model is not biased towards the last few data points during training.

The term Loss, is nothing but a prediction error of neural network and the method to calculate the loss is called loss function. A loss function is used to optimize the machine learning algorithm. The loss is calculated on training and validation sets and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each example in training or validation sets. Loss value implies how poorly or well a model behaves after each iteration of optimization.

An accuracy metric is used to measure the algorithm’s performance in an interpretable way. Accuracy of a model is usually determined after the model parameters and is calculated in the form of percentage. It is the measure of how accurate the model prediction is compared to the true data i.e., training data.

6.5 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and or/a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations does not fail in unacceptable manner. There are various types of tests. Each test type addresses a specific requirement.

TYPES OF TESTINGS

1. Unit Testing

Unit testing involves the design of test cases that validate the internal program logic is functioning properly, and that program inputs procedure valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is structural testing that relies on knowledge of its construction and is invasive. Unit test perform basic test at component level and test a specific business, application and/or system configuration Unit test ensures that each unique path of princess performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing in event driven and more concerned with the basic outcome of screens or fields. Integration test demonstrate that although the components were individually satisfaction, as shown successfully by unit testing the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination, of components

3. Functional Testing

Functional test provides systematic demonstrations that function tests are available as specified by the business and technical requirements requirement's, system documentation and user manuals.

Functional testing is centered on the following items:

Valid input: identify classes of valid input must be accepted.

Invalid Input: Identify classes of id pus must be rejected,

Functions: Identified be exercised identities function must be exercised

Output: identify classes of application outputs must be exercised

Procedures: interfacing systems or procedures must be invoked

Organization and preparation of functions test is focused on requirements, key functions or special test cases in addition, systematic coverage pertaining to identify business process flow; data fields, predefined process and successive process must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current test is determined

4. System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results An example on site testing is configuration oriented system integration test.

5. White box Testing

It is a testing in which the software tester has and of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

6. Black Box Testing

Black Box Testing a testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kind of tests must be written from a definitive source document, such as specification requirements document. It is a testing in which the software under test is treated as black box you cannot see into it. The test provides inputs and responds to outputs without considering how the software works.

Test plan:

A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process. Follow the below steps to create a test plan as per IEEE 829

Analyze the system: A system/product can be analyzed only when the tester has any information about it i.e., how the system works, who the end users are, what software/hardware the system uses, what the system is for etc.

Design the Test Strategy: Designing a test strategy for all different types of functioning, hardware by determining the efforts and costs incurred to achieve the objectives of the system. For any project, the test strategy can be prepared by

Define the Test Objectives: Test objective is the overall goal and achievement of the test execution. Objectives are defined in such a way that the system is bug-free and is ready to use

by the end-users. Test objective can be defined by identifying the software features that are needed to test and the goal of the test, these features need to achieve to be noted as successful.

Define Test Criteria: Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There are two such test criteria: Suspension criteria where if the specific number of test cases are failed, then the tester should suspend all the active test cycle till the criteria is resolved, exit criteria which specifies the criteria that denote a successful completion of a test phase.

Resource Planning: Resource plan is a detailed summary of all types of resources required to complete the project task. Resource could be human, equipment and materials needed to complete a project.

Plan Test Environment: A testing environment is a setup of software and hardware on which the testing team is going to execute test cases.

Schedule & Estimation: Preparing a schedule for different testing stages and estimating the time and man power needed to test the system is mandatory to mitigate the risk of completing the project within the deadline. It includes creating the test specification, test execution, test report, test delivery.

CONCLUSION AND FUTURE WORK

CONCLUSION:

In this project, we are generating the playlist according the emotion of the user,we developed an application for predicting the emotion of the user using Convolution neural networks and for generating the playlist we have used Spotify API. We have applied it on various images and achieved an accuracy of above 80%.

FUTURE WORK:

We want to extend our work by creating a real time music player which generates playlist and play songs according to the mood of the user

APPENDIX

Python:

Python is an interpreted, high-level, general purpose programming language created by Guido VanRossum and first released in 1991, Python's design philosophy emphasizes code Readability withits notable use of significant Whitespace. Its language constructs and object-oriented approach aimto help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

OpenCV:

OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garagethen Itseez (which was later acquired by Intel). The library is cross platform and free for use underthe open-source BSD license. OpenCV supports some models from deep learning frameworks likeTensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules which is a portable format, compatible with all other formats.

Deepface :

DeepFace is a facial recognition system developed by Facebook's AI research team. It gained significant attention when it was introduced in 2014 due to its impressive accuracy in face recognition tasks. The underlying technology of DeepFace relies on deep learning algorithms, specifically convolutional neural networks (CNNs).The primary objective of DeepFace is to identify and verify human faces in images and videos. It goes beyond simple face detection and delves into facial feature extraction, analysis, and recognition. By leveraging its deep neural network architecture, DeepFace learns hierarchical representations of faces, enabling it to capture complex facial patterns and variations.

Matplotlib:

Matplotlib is a widely-used plotting library in the Python programming language. It provides a flexible and comprehensive set of tools for creating various types of visualizations, ranging from simple line plots to complex 3D plots. Matplotlib is often used in scientific computing, data analysis, and data visualization tasks.

REFERENCES

[1] S.L.Happy and A. Routray,"Automatic facial expression recognition using features of salient facial patches," in IEEE Transactions on Affective Computing, vol.6, no. 1, pp. 1-12, 1 Jan.-March 2015.

[2] Rahul ravi, S.V Yadhukrishna, Rajalakshmi, Prithviraj, "A Face Expression Recognition Using CNN & LBP",2020 IEEE.

[3] Krittrin Chankuptarat, Raphatsak Sriwatanaworachai, Supannada ChotipantEmotion-Based Music Player,978- 1-7281-0067-8/19/$31.00 ©2019 IEEE.

[4] Karthik Subramanian Nathan, manasi arun, Megala S kannan," EMOSIC — Anemotion-based music player for Android,2017 IEEE

[5] S.Deebika, K.A.Indira, Jesline, "A Machine Learning Based Music Player byDetecting Emotions",2019 IEEE.

7.