**FACIAL EMOTION RECOGNITION USING AI-DEEP LEARNING**

**A Project report Submitted in the partial fulfilment of the requirement the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**



**Submitted By**

**PINNIKA MANOHAR : 19NQ1A0582**

**PALLAPROLU RAKESH KUMAR : 19NQ1A0580**

**HANUMANTHU CHANDRA SEKHAR : 19NQ1A0545**

**NALLIBOYINA MOUNIKA : 19NQ1A0573**

**UNDER THE ESTEEMED GUIDANCE OF**

K. Chandra Sekhar Reddy , **M. Tech**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VIKAS COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Affiliated to JNTU Kakinada, Approved by AICTE)**

**Nunna, Vijayawada-521 212, Andhra Pradesh**

**(2019-2023)**

VIKAS COLLEGE OF ENGINEERING AND TECHNOLOGY

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the project report entitled “**FACIAL EMOTION RECOGNITION**

**USING AI-DEEP LEARNING**” being submitted by

**PINNIKA MANOHAR : 19NQ1A0582**

**PALLAPROLU RAKESH KUMAR : 19NQ1A0580**

**HANUMANTHU CHANDRA SEKHAR : 19NQ1A0545**

**NALLIBOYINA MOUNIKA : 19NQ1A0573**

### In partial fulfillment for the award of the Degree of bachelor of technology in computer science and engineering to the Jawaharlal Nehru Technological University, Kakinada is a record if bonafied work carried out under my guidance and supervision.

**Internal Guide Head of the Department**

K. Chandra Sekhar Reddy , M. Tech M. Ashok Kumar, M. Tech

Assistant Professor Associate Professor

**Project External**

**ACKNOWLEDGEMENT**

We are thank my chairperson Sri N. Narsi Reddy for providing the necessary infrastructure required for my project

We are thankful to Secretary Sri N. Satyanarayana Reddy for providing us excellent facilities in the college without which I would not have succeeded.

We are thankful to our principal Dr. B. Ramana for fostering an excellent environment in our college and helping us to all points for achieving our task.

We are grateful to M. Ashok Kumar, Head of the Department of computer science and engineering for his valuable guidance, which helped me to bring how this project successfully. His wise approach made me to learn the minute details of the subject. His matured and patient guidance paved a way for completing my project with sense of satisfaction and pleasure.

We are very much thankful to K.Chandra Sekhar Reddy, for his valuable guidance, which helped me to bring out this project successfully.

Finally, We are thank all the faculty of **COMPUTER SCIENCE AND ENGINEERING**  and Library of **VIKAS COLLEGE OF ENGINEERING AND TECHNOLOGY** for imparting knowledge to me throughout my course.

**PINNIKA MANOHAR : 19NQ1A0582**

**PALLAPROLU RAKESH KUMAR : 19NQ1A0580**

**HANUMANTHU CHANDRA SEKHAR : 19NQ1A0545**

**NALLIBOYINA MOUNIKA : 19NQ1A0573**

# DECLARATION

### We hereby declare that the dissertation entitled “FACIAL EMOTION RECOGNITION USING AI-DEEP LEARNING” submitted for the Bachelor of technology in computer science and engineering in our original work. The dissertation and results embodied in this project report has not been submitted to any other University or Institute for the award of any Degree, Associate ship or any other similar titles.

**PINNIKA MANOHAR : 19NQ1A0582**

**PALLAPROLU RAKESH KUMAR : 19NQ1A0580**

**HANUMANTHU CHANDRA SEKHAR : 19NQ1A0545**

**NALLIBOYINA MOUNIKA : 19NQ1A0573**

# Place: Nunna Date:

**ABSTRACT**

The user emotions using facial expressions will be detected. These expressions can be derived from the live feed via the system’s camera or any pre-existing image available in the memory .Emotions possessed by humans can be recognized and has a vast scope of study in the computer vision industry upon which several types of search have already been done. Facial emotional expression is a part of face recognition, it has always been an easy task for humans, but achieving the same with a computer algorithm is challenging. With the recent and continuous advancements in computer vision and machine learning, it is possible to detect emotions in images, videos, etc. A face expression recognition method based on the Deep Neural Networks especially the convolutional neural network (CNN) and an image edge detection is proposed. The edge of each layer of the image is retrieved in the convolution process after the facial expression image is normalized. To maintain the texture picture's edge structure information, the retrieved edge information is placed on each feature image. In this research, several datasets are investigated and explored for training expression recognition models. The purpose of this paper is to make a study on face emotion detection and recognition via Machine learning algorithms and deep learning. This research work will present deeper insights into Face emotion detection and Recognition. It will also highlight the variables that have an impact on its efficacy.

**TABLE OF CONTENT**

**CONTENT PAGE NO**

**ABSTRACT I**

**LIST OF FIGURES IV**

**1.INTRODUCTION**

1.1About Project 1

1.2Aim 2

1.3 Domain Description 2

**2. LITERATURE SURVEY**

2.1 Existing System 9

Disadvantages of Existing System 10

2.2 Proposed System 10

Advantages of proposed system 10

**3.SYSTEM ANALYSIS**

* 1. Software Requirements 12
  2. Hardware Requirements 12
  3. Modules 12
  4. Libraries 12

**4. SOFTWARE DESIGN**

4.1 Block Diagram 17

4.2 Date Flow Diagram 18

4.3 DFD Symbols 18

4.4 Context Level DFD 19

4.5 Top Level DFD 19

4.6 Detailed Level Diagrams 20

4.7 Unified Modelling Diagrams 21

4.8 Class Diagrams 22

4.9 Use Case Diagrams 23

4.10 Interaction Diagrams 23

4.11 Sequence Diagrams 23

4.12 Collaboration diagrams 24

4.13 Activity Diagrams 25

**5.IMPLEMENTATION**

5.1 Technology Used 27

5.2 Code Templates 33

**6.TESTING**

6.1 Introduction 37

6.2 Design Of Test Cases And Scenarios 37

6.3 Black Box Testing 37

6.4 White Box Testing 38

**7. OUTPUT SCREENS**  39

**8. CONCLUSION**  42

**9. FUTURE SCOPE AND ENHANCEMENT** 43

**10. BIBLIOGRAPHY** 44

**LIST OF FIGURES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Figure Number Name of Figure Page Number1.1 Computer Vision 3 | | 1.2 Patterns 4 | | 1.3 Facial Recognition 5 | | 1.4 Health Care 61.5 Deep Learning 61.6 Deep Learning Life Cycle 72.1 Demo Image 92.2 Expressions 10 | | 4.1 Block Diagram 17 | | 4.4 Context Level DFD 19 |     4.6 Detailed Level DFD 2D Array 21  4.7 Detailed Level DFD 22    4.8 Class Diagram 22  4.9 Use Case Diagram 23  4.10 Sequence Diagram 24  4.11 Collaboration Diagram 25  4.12 Activity Diagram 26  5.1 CNN Cycle 29  5.2 Pooling Layer 31  5.3 Fully Connected Layer 32 |

# CHAPTER 1

# INTRODUCTION

## OVERVIEW:

Humans often have different moods and facial expressions changes accordingly. Human emotion recognition plays a very important role in social relations. The automatic recognition of emotions has been an active analysis topic from early eras. In this deep learning system user’s emotions using its facial expression will be detected. Real-time detection of the face and interpreting different facial expressions like happy, sad, angry, afraid, surprise, disgust, and neutral. etc. This system can detect six different human emotions. The trained model is capable to detect all the mentioned emotions in real-time. An automatic facial expression Recognition system has to perform detection and site of faces during a cluttered scene, facial feature extraction, and facial expression classification. The facial expression recognition system is enforced victimization of Convolution Neural Network (CNN).

A CNN model is trained on FER2013 dataset. FER2013 Kaggle faces expression dataset with six facial features labels as happy, sad, surprise, fear, anger, disgust, and neutral is used throughout this project. Compared to the other datasets, FER has more variation in the images, including face occlusion, partial faces, low-contrast images, and eyeglasses. This system has ability to monitor people emotions, to discriminate between emotions and label them appropriately and use that emotion information to guide thinking and behaviour of particular person

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix.

**1.2 Aim**

The aim of facial emotion recognition (FER) using deep learning is to develop an automated system that can accurately recognize facial expressions and associate them with specific emotions. This technology can have various applications in fields such as:

**Healthcare**: FER can be used to monitor patients' emotional states and detect early signs of mental health disorders such as depression and anxiety.

**Education**: FER can be used to analyze students' emotional responses to educational content and provide personalized feedback to enhance their learning experience.

**Marketing**: FER can be used to analyze customers' emotional responses to products and advertisements, and improve marketing strategies.

Overall, the aim of FER using deep learning is to develop a technology that can improve our understanding of human emotions and lead to more natural and effective human-machine interactions.

**1.3 Domine Description**

**What is Computer Vision?**

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image. Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyse thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

How does computer vision work?

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects. Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

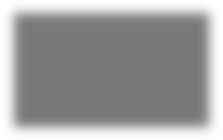


Figure : 1.1 Computer Vision

CNN helps a machine learning or deep learning model look by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is seeing. The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Applications Of Computer Vision:

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day.

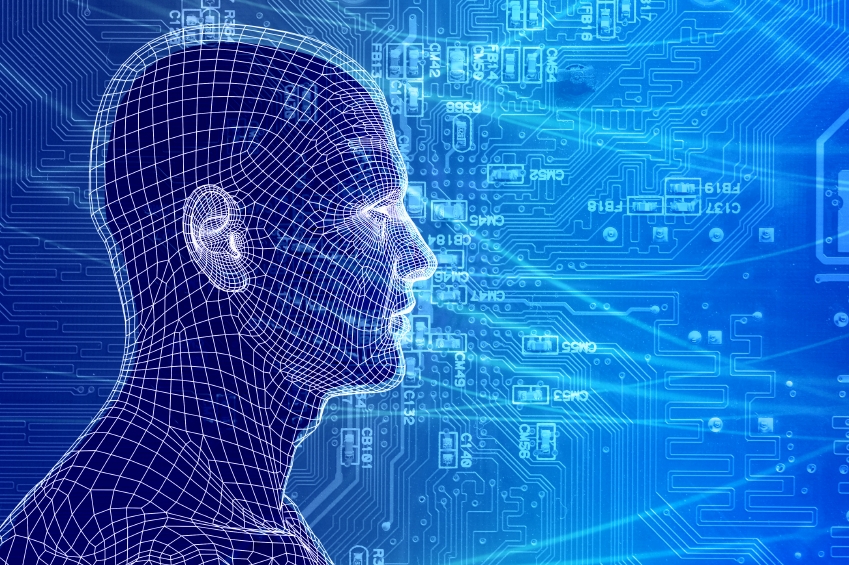


Figure : 1.2 Patterns

**How does computer vision work?**

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects. Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

CNN helps a machine learning or deep learning model look by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is seeing. The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

**Applications Of Computer Vision**

Computer vision is one of the areas in Machine Learning where core concepts are already being integrated into major products that we use every day.

**CV In Facial Recognition:**

Computer vision also plays an important role in facial recognition applications, the technology that enables computers to match images of peoples faces to their identities. Computer vision algorithms detect facial features in images and compare them with databases of face profiles. Consumer devices use facial recognition to authenticate the identities of their owners. Social media apps use facial recognition to detect and tag users. Law enforcement agencies also rely on facial recognition technology to identify criminals in video feed



Figure: 1.3 Facial Recognition

**CV In Healthcare**

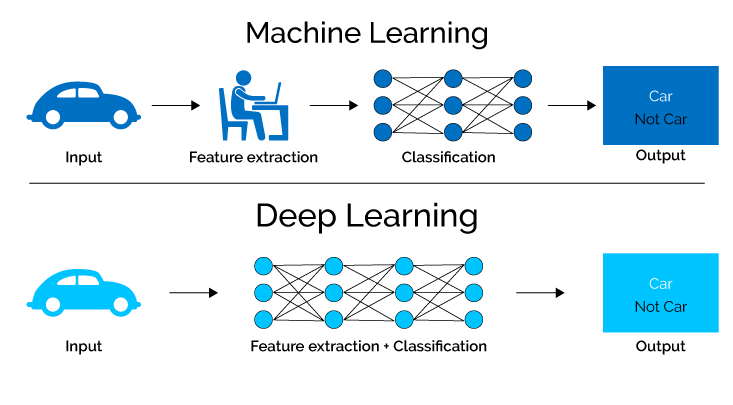
Computer vision has also been an important part of advances in health-tech. Computer vision algorithms can help automate tasks such as detecting cancerous moles in skin images or finding symptoms in x-ray and MRI scans.



Figure : 1.4 Health Care

**What is Deep learning?**

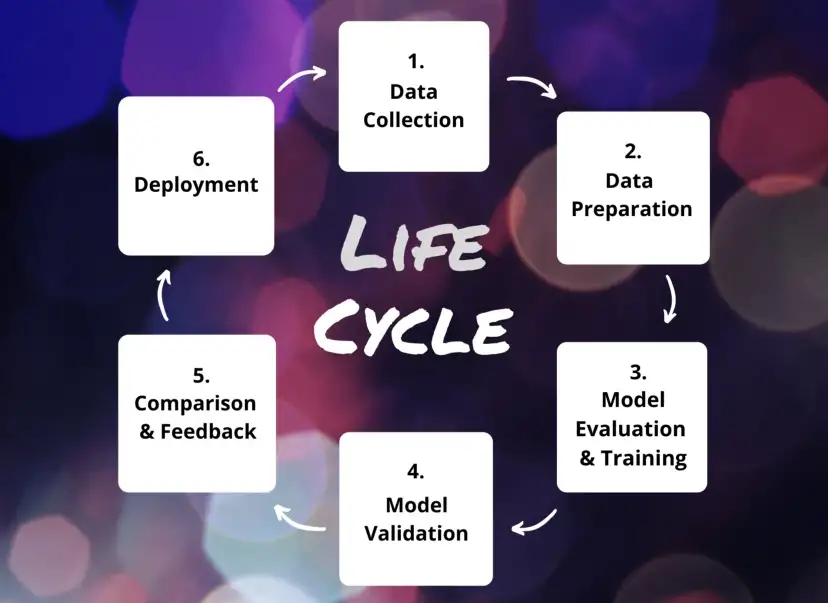
Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain albeit far from matching its ability allowing it to learn from large amounts of data.



## Figure : 1.5 Deep Learning

**Why it is used?**

Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts.

****

## Figure : 1.6 Deep Learning Life Cycle

**1. Data Collection**

Machine learning models should solve a given problem on the basis of data. Therefore, everything starts with collecting enough samples with proper metadata.

**2. Data Preparation**

We clean the data by identifying noise, false or misleading data and correct or remove it from the training set. Additionally, we pre-process the data to normalize it. In our cases this mostly mean scaling or cropping images, converting them into a relevant format and creating a folder structure we can use for training. Collecting, cleaning, and pre-processing data are our biggest and most time-consuming challenges. It is not unusual to spend a major portion of the project time for these tasks.

**3. Model Evaluation and Training**

During model evaluation we take a closer look at different models and model architectures in order to find out which architectures work well with certain data and certain problems.

There are models that work well with text, e.g. translation, term classification. Other models work well with images, e.g. classification models, detection models, or localization models. Our experience, best practice orientation, and scientific research lead us to the appropriate model for our current project.

An example of how we split the training data, and where the data sets are used during the life cycle.

**4. Model Validation**

After finishing the training as described above, we assess the quality of the model. We work with the model to understand its behaviour: which aspects are already solved very well, and which are not. By inspecting the visual data, we interpret necessary changes to the training set in order to optimize result quality. An adjustment could be for example to collect or synthesize more data from a specific category.

**5. Comparison and Feedback**

In this step it is time to share the progress we’ve made so far with our customer. We present our findings on the quality and condition of the model; we show what worked and what did not work. A good teamwork with our customer is significant here. Together, we discuss possible improvements of the model, for example gathering more data and where to get this data from. In close cooperation, we plan the next iteration of model training.

**6. Deployment**

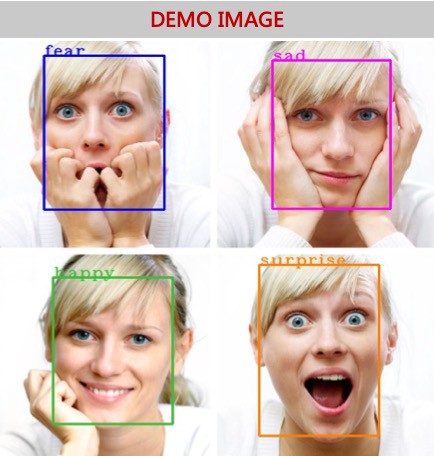
The deployment of our current model version acts as the quality base line for the following training iteration. If the model already adds value for the customer, it can be integrated in his prototype or even in production. Meanwhile, we begin the next iteration of training, the life cycle starts again.

## CHAPTER-2

**LITERATURE SURVEY**

### Existing System

Manually a human can recognize the facial emotions of other humans. For this, one can have proper eye condition. And also several approaches have been proposed to recognize facial emotions automatically by using a system run by some programming languages like C++.



### Figure: 2.1 Demo Image

### Disadvantages of Existing System

* Continuous detection is not possible.
* It is not possible to find accuracy rate.
* Consists of large code.

### 2.2.Proposed System

This proposed system is to detect continuous facial emotion expression for real time video. The proposed system We are using AI-Deep learning image processing Convolutional Neural Network Algorithm(CNN) with platform of python.

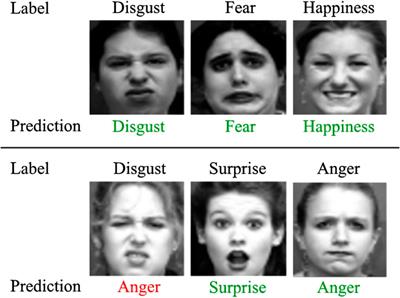


Figure 2.2 Expressions

### Advantages of Proposed System

* Our proposed system in python makes use of very less code.
* And also Computationally simple and fast.
* Can be used in multiple AI tools to get user feedback.
* No need of human intervention.
* High Speed Detection.
* It detects the emotions more accurately.

**CHAPTER-3**

**SYSTEM ANALYSIS**

### 3.1 SOFTWARE REQUIREMENTS

* IDE : PyCharm
* Programming language : Python
* Version : Python 3.6
* Technology used : AI – Deep learning
* Operating system : Windows – 64 bit

### 3.2 HARDWARE REQUIREMENT:

* RAM : 4GB (or) above.
* Processor : INTEL i3 CPU (or) above.
* Graphic card : 500MB(or)above.
* Hard disk : 400GB(or)500GB

## 3.3 MODULES

For convenience our project has been divided into four modules they are:

1. Access webcam
2. Detect face
3. Detect emotion
4. Display emotion

**1.ACCESS WEBCAME:**

* OpenCV is a vast library that helps in providing various functions for image and video operations.
* Below given are sequence of steps for accessing web cam through OpenCV.

>>>pip install OpenCV-Python \*\*\*\*OpenCV gets installed and then we should import it to Python libraries Import cv2

**2.DETECT FACE:**

* Face-detection algorithms focus on the detection of human faces.
* It is analogous to image detection in which the image of a person is matched bit by bit.
* Image matches with the image stores in dataset.
* The Haar Cascade algorithm is an Object Detection Algorithm used to identify faces in an image or a real time video.

**3 DETECT EMOTION:**

* By using CNN algorithm we can detect the emotion.
* CNN consists of some layers. They are:

1. Convolutional layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer

**4.DISPLAY EMOTION:**

* After predicting the emotion by CNN algorithm, it display that emotion to the user by using OpenCV.
* OpenCV is also used for displaying the emotion

**3.4 LIBRARIES**

**OpenCV :**

OpenCV (Open Source Computer Vision) is a popular open-source library of computer vision algorithms and tools. It provides a wide range of functions for image and video processing, including feature detection, object recognition, tracking, and more. OpenCV is written in C++, but also provides Python bindings.

Here are some of the key features and capabilities of OpenCV:

**Image and video processing**: OpenCV provides a wide range of functions for processing and manipulating images and videos. These include functions for resizing, cropping, blurring, filtering, and more.

**Object detection and recognition**: OpenCV includes several algorithms for object detection and recognition, such as Haar cascades and HOG (Histogram of Oriented Gradients). These algorithms can be used to detect faces, pedestrians, and other objects in an image or video stream.

**Feature detection and matching**: OpenCV includes algorithms for feature detection and matching, such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features). These algorithms can be used for tasks such as image stitching, object tracking, and more.

**Pillow :**

Pillow is a popular Python library for working with images. It provides a wide range of functions for loading, manipulating, and saving images in various formats. Some of the key features and capabilities of Pillow include:

**Image loading and saving**: Pillow can read and write images in a wide range of formats, including PNG, JPEG, BMP, and many more.

**Image manipulation**: Pillow provides a variety of functions for manipulating images, such as cropping, resizing, rotating, and flipping. It can also adjust image brightness, contrast, and color balance.

**Image filtering**: Pillow includes functions for applying various image filters, such as blur, sharpen, edge detection, and more.

**Image drawing**: Pillow provides functions for drawing on images, such as lines, rectangles, circles, and text.

**Color conversion**: Pillow can convert images between different color spaces, such as RGB, CMYK, and grayscale.

**Image metadata**: Pillow can read and write image metadata, such as EXIF data and IPTC data.

Pillow is built on top of the Python Imaging Library (PIL), which is no longer actively maintained. Pillow is a fork of PIL that provides continued development and support for the library. It is widely used in various applications, such as web development, computer vision, and scientific computing..

**Matplotlib :**

Matplotlib is a popular data visualization library for Python. It provides a wide range of tools and functions for creating high-quality, customizable plots and charts. With Matplotlib, you can create line plots, scatter plots, bar plots, histograms, 3D plots, and many other types of visualizations.

Matplotlib provides a wide range of customization options, allowing you to control every aspect of your plots. You can customize the colors, line styles, markers, fonts, and sizes of your plots, as well as add annotations, legends, and other elements to make your plots more informative and visually appealing.

Matplotlib is widely used in data science, machine learning, and scientific research, as it provides an easy-to-use interface for creating complex visualizations from data. Additionally, Matplotlib integrates well with other Python libraries such as NumPy, Pandas, and Seaborn, making it a versatile tool for data analysis and visualization.

**NumPy :**

NumPy is a powerful Python library for numerical computing. It provides a set of tools for creating and manipulating multi-dimensional arrays and matrices, which are the fundamental data structures for numerical computations in scientific computing and data analysis.

NumPy's main feature is its powerful array manipulation capabilities. It provides a comprehensive set of functions for performing basic and advanced mathematical operations on arrays, such as element-wise operations, matrix operations, statistical operations, and more.

NumPy also includes tools for linear algebra, Fourier analysis, and random number generation, as well as a powerful indexing system that allows for complex array manipulations.

One of the key benefits of NumPy is its speed and efficiency. It is implemented in C and optimized for performance, making it much faster than Python's built-in data structures for numerical computations. NumPy also provides a range of tools for parallel computing, which allows for even faster processing on multi-core CPUs or GPUs.

NumPy is widely used in scientific computing, data analysis, and machine learning, as it provides a fast and efficient platform for numerical computations. It also integrates well with other Python libraries, such as Pandas, SciPy, and Matplotlib, making it a versatile tool for data analysis and visualization.

**Keras :**

Keras is a high-level API for building and training machine learning models in TensorFlow. It provides a user-friendly and intuitive interface for creating deep learning models, allowing developers to focus on the design of their models rather than the low-level details of TensorFlow.

Keras supports a wide range of neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. It also provides a large collection of pre-built layers and models, which can be easily combined to create more complex models.

One of the key features of Keras is its ease of use. It provides a simple and consistent API for building models, and includes tools for data preprocessing, model evaluation, and visualization. Keras also supports a range of popular optimization algorithms, such as stochastic gradient descent (SGD) and Adam.

Keras can be used for a wide range of applications, including image classification, natural language processing, and time series analysis. It is also highly customizable, allowing developers to easily modify existing models or create new ones from scratch.

**TensorFlow:**

TensorFlow is a popular open-source machine learning framework developed by Google. It allows developers and data scientists to create and train machine learning models for a wide range of tasks, including image and speech recognition, natural language processing, and more.

One of the main features of TensorFlow is its ability to create and manipulate tensors, which are multi-dimensional arrays that represent data. TensorFlow provides a set of powerful tools and functions for performing mathematical operations on tensors, making it an efficient and scalable platform for machine learning.

TensorFlow supports both high-level and low-level APIs, allowing developers to choose the level of abstraction that best suits their needs. The high-level API, called Keras, provides a simple and user-friendly interface for building and training machine learning models, while the low-level API allows for more fine-grained control over the model's architecture and training process.

**CHAPTER-4**

**SYSTEM DESIGN**

**4.1. BLOCK DIAGRAM**

The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

Diagram

Description automatically generated

Figure : 4.1 Block Diagram for FER

**4.2.** **DATA FLOW DIAGRAMS**:

Data flow diagram (DFD) is a graphical representation of “flow” of data through an information system, modelling its process concepts. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD’s can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It doesn’t show information about timing of processes, or information about whether processes will operate in sequence or parallel. A DFD is also called as “bubble chart”.

**4.3 DFD Symbols:**

In the DFD, there are four symbols:

* A square define a source or destination of system data.
* An arrow indicates dataflow. It is the pipeline through which the information flows.
* A circle or a bubble represents transforms dataflow into outgoing dataflow.
* An open rectangle is a store, data at reset or at temporary repository of data.

**Dataflow:** Data move in a specific direction from an origin to a destination.

**Process:** People, procedures or devices that use or produce (Transform) data. The physical component is not identified.

**Sources**: External sources or destination of data, which may be programs, organizations or other entity.

**Data store:** Here data is stored or referenced by a process in the system’s #

|  |  |
| --- | --- |
|  |  |

In our project, we had built the data flow diagrams at the very beginning of business process modelling in order to model the functions that our project has to carry out and the interaction between those functions together with focusing on data exchanges between processes.

**4.****4 Context level DFD:**

A Context level Data flow diagram created using select structured systems analysis and design method (SSADM). This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are “owned” by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities. The Context level DFD is shown in fig.3.2.1



Figure 4.4 Context Level DFD for Customer churn prediction

The Context Level Data Flow Diagram shows the data flow from the application to the database and to the system.

**4.****5 Top level DFD:**

A data flow diagram is that which can be used to indicate the clear progress of a business venture. In the process of coming up with a data flow diagram, the level one provides an overview of the major functional areas of the undertaking. After presenting the values for most important fields of discussion, it gives room for Level two to be drawn.

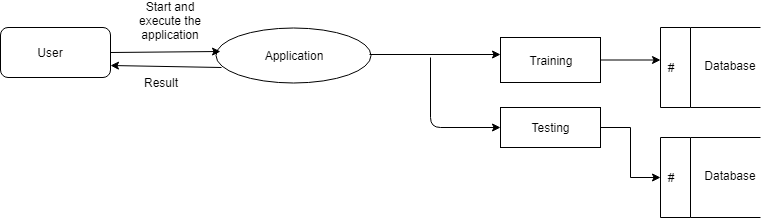


Figure 4.5 Top Level DFD

After starting and executing the application, training and testing the dataset can be done as shown in the above figure

**4.6 Detailed Level Diagram:**

This level explains each process of the system in a detailed manner. In first detailed level DFD (Generation of individual fields): how data flows through individual process/fields in it are shown.

In second detailed level DFD (generation of detailed process of the individual fields):

how data flows through the system to form a detailed description of the individual processes.

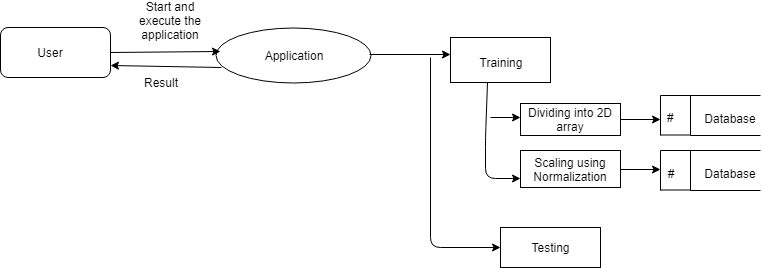
****

Figure : 4.6 Detailed level DFD 2D array

After starting and executing the application, training the dataset is done by using dividing into 2D array and scaling using normalization algorithms, and then testing is done.

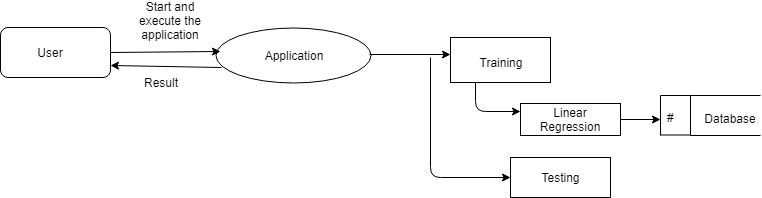
****

Figure 4.7 Detailed level DFD

After starting and executing the application, training the dataset is done by using linear regression and then testing is done.

**4.7 UNIFIED MODELLING LANGUAGE DIAGRAMS**:

The Unified Modelling Language (UML) is a Standard language for specifying, visualizing, constructing and documenting the software system and its components. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* **User Model View** 
  1. This view represents the system from the user’s perspective.
  2. The analysis representation describes a usage scenario from the end-users perspective.
* **Structural Model View** 
  1. In this model the data and functionality are arrived from inside the system.
  2. This model view models the static structures.
* **Behavioural Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

**4.8 Class Diagram:**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

Diagram

Description automatically generated

Figure 4.8 CLASS DIAGRAM

**4.9 Use Case Diagram:**

Use case diagrams are one of the five diagrams in the UML for modelling the dynamic aspects of the systems (activity diagrams, sequence diagram, state chart diagram, collaboration diagram are the four other kinds of diagrams in the UML for modelling the dynamic aspects of systems).Use case diagrams are central to modelling the behavior of the system, a sub-system, or a class. Each one shows a set of use cases and actors and relations.

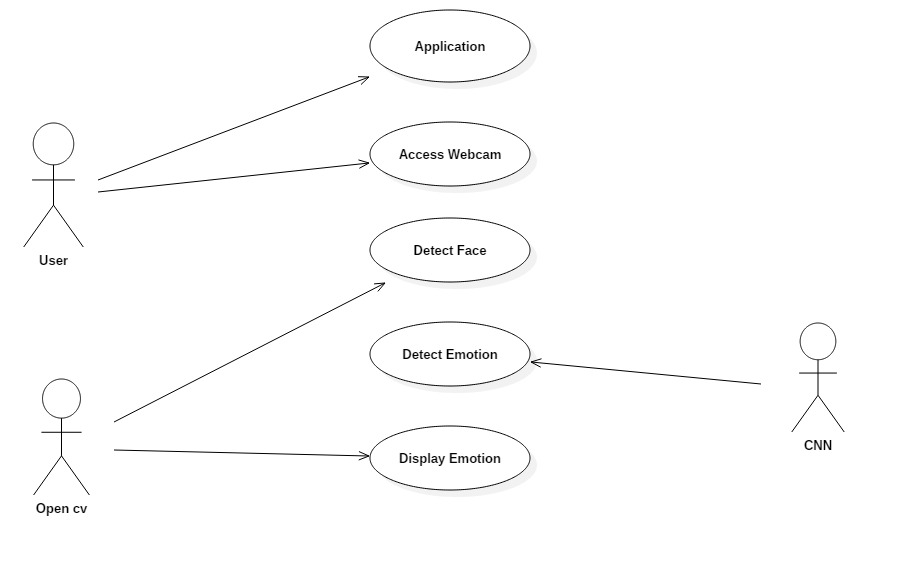


Figure 4.9 USECASE DIAGRAM

### 4.10 Interaction Diagrams

### From the term Interaction, it is clear that the diagram is used to describe some type of interactions among the different elements in the model. This interaction is a part of dynamic behavior of the system. This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purpose of both the diagrams are similar. Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

**4.11 Sequence Diagram:**

Sequence diagram is an interaction diagram which is focuses on the time ordering of messages. It shows a set of objects and messages exchanged between these objects. This diagram illustrates the dynamic view of a system.

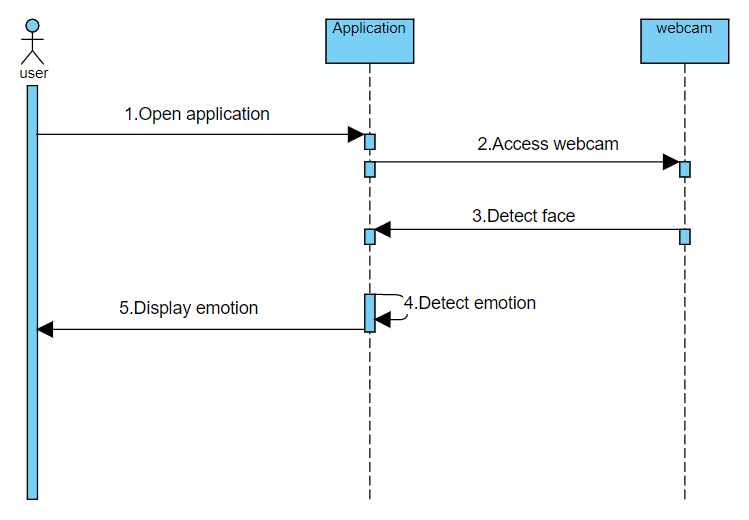


Figure 4.11 SEQUENCE DIAGRAM

**4.12 Collaboration Diagram:**

Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Collaboration diagram and sequence diagram are isomorphic.

Diagram

Description automatically generated

Figure 4.12 COLLABORATION DIAGRAM

* 1. **Activity Diagram:**

An Activity diagram shows the flow from activity to activity within a system it emphasizes the flow of control among objects**.** This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are “owned” by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities.

Diagram

Description automatically generated

Figure 4.13 ACTIVITY DIAGRAM

**CHAPTER-5**

**IMPLEMENTATION**

**5.1 Technology Used**

**PYTHON:**

### Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It is known for its simplicity, readability, and flexibility, and is widely used in web development, scientific computing, data analysis, and artificial intelligence.

### Some key features of Python include:

### Easy to learn: Python's syntax is straightforward and easy to understand, making it an ideal language for beginners.

### Interpreted: Python is interpreted, which means that code can be executed immediately, without the need for compiling.

### Object-oriented: Python is an object-oriented language, which means that it supports object-oriented programming concepts such as inheritance, polymorphism, and encapsulation.

### Dynamically typed: Python is dynamically typed, which means that data types are determined at runtime, rather than being declared explicitly in code.

### Large standard library: Python has a large standard library, which provides a wide range of functions and modules that can be used for various purposes.

### Platform-independent: Python can run on different operating systems, including Windows, Linux, and macOS.

### To write and run Python code, you can use an integrated development environment (IDE) such as PyCharm, Visual Studio Code, or Spyder, or you can simply use a text editor like Sublime Text or Atom. To run Python code, you need to have Python installed on your system, which can be downloaded from the official Python website

### why Python?

**Easy to learn**: Python's syntax is simple and straightforward, making it an ideal language for beginners to learn. The language emphasizes readability, so even those with little experience in programming can quickly grasp the basics.

**Versatile**: Python is a versatile language that can be used for a wide range of applications, including web development, data analysis, artificial intelligence, scientific computing, and more.

**Large community**: Python has a large and active community of developers who are constantly contributing to the language's development, sharing resources, and providing support to each other.

**Rich libraries**: Python has a large standard library that provides a wide range of functions and modules, which makes it easy to perform various tasks without having to write code from scratch.

**Cross-platform**: Python can run on various operating systems, including Windows, Linux, and macOS.

**High-level language**: Python is a high-level language that allows developers to write code that is easy to read, understand, and maintain.

Overall, Python's simplicity, versatility, large community, rich libraries, cross-platform compatibility, and high-level nature make it an excellent choice for many programming applications.

Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

**DEEP LEARNING:**

### Deep learning is a subfield of machine learning that focuses on training artificial neural networks to learn from data and make predictions or decisions. It involves the use of complex neural networks with multiple layers, which are capable of automatically learning hierarchical representations of data.

### Deep learning has become a popular approach for a wide range of tasks, such as image and speech recognition, natural language processing, and even game playing. Some of the key features and capabilities of deep learning include

### Neural network architecture: Deep learning models typically involve complex neural networks with many layers. These networks can be trained to learn hierarchical representations of data, allowing them to recognize patterns and features at multiple levels of abstraction.

### Automatic feature extraction: Deep learning models can automatically extract features from raw data, reducing the need for manual feature engineering. This can be particularly useful in tasks where it is difficult to define explicit rules or features for the problem domain.

### Large datasets: Deep learning models typically require large datasets for training, as they have many parameters that need to be learned. However, with large datasets, deep learning models can achieve state-of-the-art performance on a wide range of tasks.

### Training algorithms: Deep learning models are typically trained using backpropagation algorithms, which involve computing the gradients of the model's parameters with respect to a loss function. These gradients are then used to update the model's parameters in the direction that minimizes the loss function.

### Transfer learning: Deep learning models can be trained on one task and then transferred to another task with similar characteristics. This can be useful in situations where there is limited data available for the target task.

### Deep learning has enabled significant advances in many areas of artificial intelligence, including computer vision, natural language processing, and robotics. It has also led to the development of new techniques for data analysis, such as unsupervised learning and generative modeling.

**CONVOLUTIONAL NEURAL NETWORK:**

### A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

### 

### Figure : 5.1 CNN Cycle

### Convolution Layer :

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a CNN generally has the following attributes:

Convolutional filters/kernels defined by a width and height (hyper-parameters).

The number of input channels and output channels (hyper-parameters). One layer's input channels must equal the number of output channels (also called depth) of its input.

Additional hyperparameters of the convolution operation, such as: padding, stride, and dilation.

### Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[14] Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.[15] For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks.[16][17] Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

**ReLu Layer :**

ReLU is an activation function. But, what is an activation function?

**Rectified Linear Unit**(ReLU) transform function only activates a node if the input is above a certain quantity, while the input is below zero, the output is zero, but when the input rises above a certain threshold, it has a linear relationship with the dependent variable

### Pooling layer :

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map.[18][19] There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map,[20][21] while average pooling takes the average value.

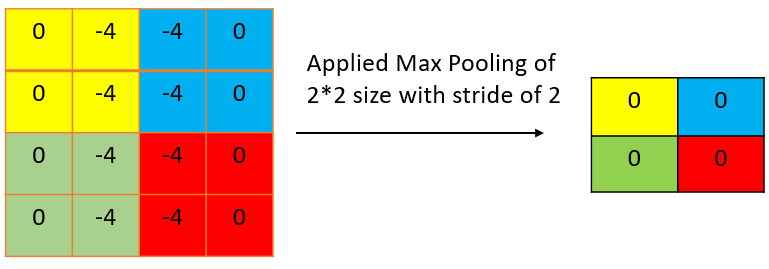


Figure : 5.2 Pooling Layer

### Fully Connected Layer :

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. Typically the area is a square (e.g. 5 by 5 neurons). Whereas, in a fully connected layer, the receptive field is the entire previous layer. Thus, in each convolutional layer, each neuron takes input from a larger area in the input than previous layers. This is due to applying the convolution over and over, which takes into account the value of a pixel, as well as its surrounding pixels. When using dilated layers, the number of pixels in the receptive field remains constant, but the field is more sparsely populated as its dimensions grow when combining the effect of several layers.

Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called filters and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.[22]

### 

### Figure : 5.3 Fully Connected Layer

**Different Types Of CNN**

**VGG:**

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The deep refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers.

**Xception**:

Xception is a convolutional neural network that is 71 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

**ResNet:**

A residual neural network (ResNet) is an artificial neural network. It is a gatelesor open-gated variant of the Highway Net, the first working very deep feed forward neural network with hundreds of layers, much deeper than previous neural networks.

### 5.2 Code Templates

**Cv\_cam\_facial\_expression.py:**

#import python modules

import tensorflow as tf

from tensorflow import keras

import numpy as np

import cv2

emotion = ['Anger', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

model = keras.models.load\_model("model\_35\_91\_61.h5")

font = cv2.FONT\_HERSHEY\_SIMPLEX

cam = cv2.VideoCapture(0)

face\_cas = cv2.CascadeClassifier('./cascades/haarcascade\_frontalface\_default.xml')

while True:

ret, frame = cam.read()

if ret==True:

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

#gray = cv2.flip(gray,1)

faces = face\_cas.detectMultiScale(gray, 1.3,5)

for (x, y, w, h) in faces:

face\_component = gray[y:y+h, x:x+w]

fc = cv2.resize(face\_component, (48, 48))

inp = np.reshape(fc,(1,48,48,1)).astype(np.float32)

inp = inp/255.

prediction = model.predict\_proba(inp)

em = emotion[np.argmax(prediction)]

score = np.max(prediction)

cv2.putText(frame, em+" "+str(score\*100)+'%', (x, y), font, 1, (0, 255, 0), 2)

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)

cv2.imshow("image", frame)

if cv2.waitKey(1) == 27:

break

else:

print ('Error')

cam.release()

cv2.destroyAllWindows()

**facial\_expression.py:**

import tensorflow as tf

from tensorflow import keras

#from tensorflow.keras.models import Sequential

#from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten

#import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

#---------------------------------------------------------------------------------------------------------------------------------

def generate\_dataset():

"""generate dataset from csv"""

df = pd.read\_csv("./fer2013/fer2013.csv")

train\_samples = df[df['Usage']=="Training"]

validation\_samples = df[df["Usage"]=="PublicTest"]

test\_samples = df[df["Usage"]=="PrivateTest"]

y\_train = train\_samples.emotion.astype(np.int32).values

y\_valid = validation\_samples.emotion.astype(np.int32).values

y\_test = test\_samples.emotion.astype(np.int32).values

X\_train =np.array([ np.fromstring(image, np.uint8, sep=" ").reshape((48,48)) for image in train\_samples.pixels])

X\_valid =np.array([ np.fromstring(image, np.uint8, sep=" ").reshape((48,48)) for image in validation\_samples.pixels])

X\_test =np.array([ np.fromstring(image, np.uint8, sep=" ").reshape((48,48)) for image in test\_samples.pixels])

return X\_train, y\_train, X\_valid, y\_valid, X\_test, y\_test

#---------------------------------------------------------------------------------------------------------------------------------

def generate\_model(lr=0.001)

"""training model""

with tf.device('/gpu:0'):

model = keras.models.Sequential()

model.add(keras.layers.Conv2D(64,(3,3), input\_shape=(48,48, 1), padding="same"))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.MaxPooling2D())

model.add(keras.layers.Dropout(0.20))

model.add(keras.layers.Conv2D(128,(5,5), padding='same'))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.MaxPooling2D())

model.add(keras.layers.Dropout(0.20))

model.add(keras.layers.Conv2D(512,(3,3), padding="same"))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.MaxPooling2D())

model.add(keras.layers.Dropout(0.20))

model.add(keras.layers.Conv2D(512,(3,3)))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.MaxPooling2D())

model.add(keras.layers.Dropout(0.25))

model.add(keras.layers.Conv2D(256,(3,3), activation='relu'))

model.add(keras.layers.Conv2D(128,(3,3), padding='same', activation='relu'))

model.add(keras.layers.MaxPooling2D())

model.add(keras.layers.Dropout(0.25))

#model.add(keras.layers.GlobalAveragePooling2D())

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(256))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(512, activation='relu'))

model.add(keras.layers.BatchNormalization())

model.add(keras.layers.Activation('relu'))

model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(7,activation='softmax'))

model.compile(loss="sparse\_categorical\_crossentropy", optimizer=keras.optimizers.Adam(lr=lr) , metrics=['accuracy'])

return model

#--------------------------------------------------------------------------------------------------------------------------------

if \_\_name\_\_=="\_\_main\_\_":

#df = pd.read\_csv("./fer2013/fer2013.csv")

X\_train, y\_train, X\_valid, y\_valid, X\_test, y\_test = generate\_dataset()

X\_train = X\_train.reshape((-1,48,48,1)).astype(np.float32)

X\_valid = X\_valid.reshape((-1,48,48,1)).astype(np.float32)

X\_test = X\_test.reshape((-1,48,48,1)).astype(np.float32)

X\_train\_std = X\_train/255.

X\_valid\_std = X\_valid/255.

X\_test\_std = X\_test/255.

model = generate\_model(0.01)

with tf.device("/gpu:0"):

history = model.fit(X\_train\_std, y\_train,batch\_size=128,epochs=35, validation\_data=(X\_valid\_std, y\_valid), shuffle=True)

model.save("my\_model.h5")

### 

### CHAPTER-6

**TESTING**

### 6.1 Introduction

### software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

### 6.2 DESIGN OF TEST CASES AND SCENARIOS

It is the process of testing the functionality and it is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as at undiscovered error. A successful test is one that uncovers an as at undiscovered error. Software testing is usually performed for one of two reasons:

* Defect Detection
* Reliability estimation

**6.3 BLACK BOX TESTING:**

The base of the black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming to route the testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test application by the developer. The following are different types of techniques involved in black box testing. They are:

* Decision Table Testing
* All pairs testing
* State transition tables testing
* Equivalence Partitioning

Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through. Validation is the process of checking what has been specified is what the user actually wanted.

Validation: Are we doing the right job?

Verification: Are we doing the job right?

In order to achieve consistency in the Testing style, it is imperative to have and follow a set of testing principles. This enhances the efficiency of testing within SQA team members and thus contributes to increased productivity. The purpose of this document is to provide overview of the testing, plus the techniques. Here, after training is done on the training dataset, testing is done.

**6.4 WHITE BOX TESTING:**

White box testing requires access to source code. Though white box testing can be performed any time in the life cycle after the code is developed, it is a good practice to perform white box testing during unit testing phase.

In designing of database the flow of specific inputs through the code, expected output and the functionality of conditional loops are tested.

At SDEI, 3 levels of software testing is done at various SDLC phases

* **UNIT TESTING**: in which each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented
* **INTEGRATION TESTING**: in which progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a whole.
* **SYSTEM TESTING**: in which the software is integrated to the overall product and tested to show that all requirements are met. A further level of testing is also done, in accordance with requirements:
* **REGRESSION TESTING**: is used to refer the repetition of the earlier successful tests to ensure that changes made in the software have not introduced new bugs/side effects.
* **ACCEPTANCE TESTING:** Testing to verify a product meets customer specified requirements. The acceptance test suite is run against supplied input data. Then the results obtained are compared with the expected results of the client.

**CHAPTER-7**

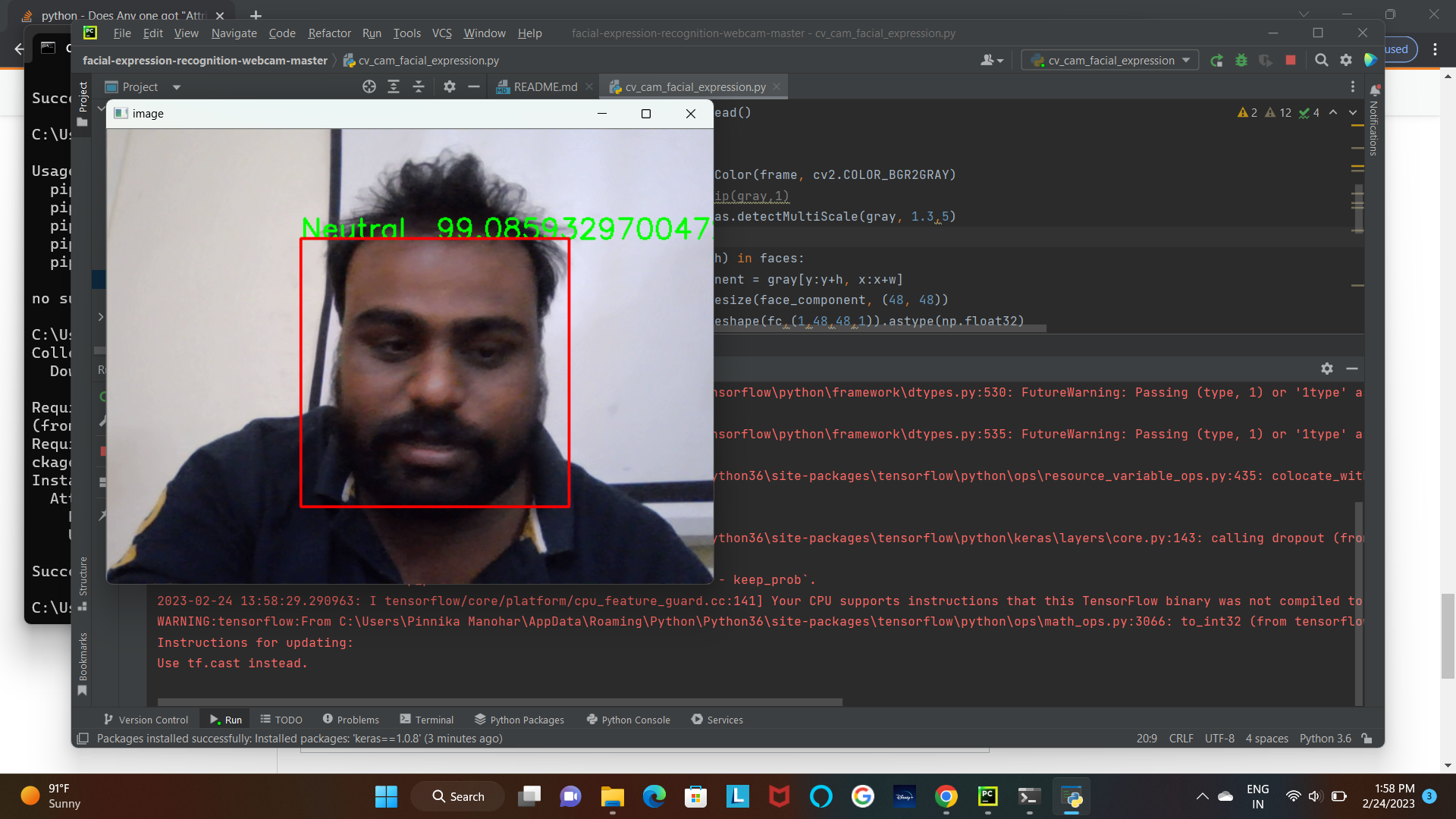
**OUTPUT SCREENS**

### 

### 

### 

### 

****

**CHAPTER-8**

**CONCLUSION**

In this what we found is during small datasets in some other cases most of time decision trees direct us to a solution which is not accurate, but when we look at Logistic Regression results We are getting more accurate results with probabilities of all other possibilities but due to guidance to only one solution decision trees may miss lead. Finally we can say by this experiment that Logistic Regression is more accurate if the input data is cleaned and well maintained even though ID3 can clean it self, it cannot give accurate results every time, and in this same way Logistic Regression also will not give accurate results every time we need to consider results of different algorithms and by all its results if a prediction is made it will be accurate. But we can use Logistic Regression consider variables as individual we can use combination of algorithms like Logistic Regression and K-means to get accuracy.

**CHAPTER-9**

**FUTURE SCOPE AND ENHANCEMENT**

There are currently no regulations in the United States expressly covering the biometric data of a person. Facial recognition devices are already being tested or implemented for airport protection, and it is reported that their faceprint has now been produced by more than half the United States populace. Information may be collected and processed by a facial recognition program, and a person does not even recognize it. Then, a hacker might reach the details, and the knowledge of a person would propagate without even realizing it. Government entities or marketers may use this data to monitor individuals too. Worse still, a false positive may include a person for a crime they are not.

Hundreds of companies have embraced face recognition. Integrating and installing is reasonably straightforward, but it has also provided users a feeling of utilizing a system that is more sophisticated and safer than passwords or PINs, thereby increasing user experience. Nonetheless, plenty is often unclear on the road to implementing what many deem the ideal biometric approach, causing several relatively severe blunders along the way.

**CHAPTER-10**

**BIBLIOGRAPHY**

**1**.Sonam Nikhar, A.M.Karandikar “Prediction of Heart Disease Us-ing Machine Learning Algorithms” in International Journal of Advanced Engineering, Management and Science (IJAEMS) June-2016 vol-2

**2**.Deeanna Kelley “Heart Disease: Causes, Prevention, and Current Research” in JCCC Honors Journal

**3**.Nabil Alshurafa, Costas Sideris, Mohammad Pourhomayoun, Haik Kalantarian, Majid Sarrafzadeh "Remote Health Monitoring Out-come Success Prediction using Baseline and First Month Interven-tion Data" in IEEE Journal of Biomedical and Health Informatics

**4**.PonrathiAthilingam, Bradlee Jenkins, Marcia Johansson, Miguel Labrador "A Mobile Health Intervention to Improve Self-Care in Patients With Heart Failure: Pilot Randomized Control Trial" in JMIR Cardio 2017, vol. 1, issue 2, pg no:1

**5**.DhafarHamed, Jwan K. Alwan, Mohamed Ibrahim, Mohammad B. Naeem "The Utilisation of Machine Learning Approaches for Med-ical Data Classification" in Annual Conference on New Trends in Information & Communications Technology Applications - march-2017

**6**.Applying k-Nearest Neighbour in Diagnosing Heart Disease Pa-tients Mai Shouman, Tim Turner, and Rob Stocker International Journal of Information and Education Technology, Vol. 2, No. 3, June 2012

**7**.Amudhavel, J., Padmapriya, S., Nandhini, R., Kavipriya, G., Dha-vachelvan, P., Venkatachalapathy, V.S.K., "Recursive ant colony optimization routing in wireless mesh network", (2016) Advances in Intelligent Systems and Computing, 381, pp. 341-351.

**8**.Alapatt, B.P., Kavitha, A., Amudhavel, J., "A novel encryption algorithm for end to end secured fiber optic communication", (2017) International Journal of Pure and Applied Mathematics, 117 (19 Special Issue), pp. 269-275.

**9.**Amudhavel, J., Inbavalli, P. Bhuvaneswari B., Anandaraj, B. Vengattaraman, T., Premkumar, K., "An effective analysis on har-mony search optimization approaches", (2015) International Journal of Applied Engineering Research, 10 (3), pp. 2035-2038.

**10**.Amudhavel, J., Kathavate, P., Reddy, L.S.S., BhuvaneswariAadharshini, A., "Assessment on authentication mechanisms in dis-tributed system: A case study", (2017) Journal of Advanced Re-search in Dynamical and Control Systems, 9 (Special Issue 12), pp. 1437-1448.

**11**.Amudhavel, J., Kodeeshwari, C., Premkumar, K., Jaiganesh, S., Rajaguru, D., Vengattatraman, T., Haripriya, R., "Comprehensive analysis on information dissemination protocols in vehicular ad hoc networks", (2015) International Journal of Applied Engineering Re-search, 10 (3), pp. 2058-2061.

**12**.Amudhavel, J., Kathavate, P., Reddy, L.S.S., Satyanarayana, K.V.V., "Effects, challenges, opportunities and analysis on security based cloud resource virtualization", (2017) Journal of Advanced Research in Dynamical and Control Systems, 9 (Special Issue 12), pp. 1458-1463.

**13**.Amudhavel, J., Ilamathi, R., Moganarangan, N., Ravishankar, V., Baskaran, R., Premkumar, K., "Performance analysis in cloud au-diting: An analysis of the state-of-the-art", (2015) International Journal of Applied Engineering Research, 10 (3), pp. 2043-2046.

### REFERENCES

* [https:/deeplearningmastery.com/bagging-and-random-forest-ensemble-](https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/) [algorithms-for-deep-learning/](https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/)
* Nobuyuki Otsu (1979). A threshold selection method from gray-level histograms. IEEE Trans. Sys., Man., Cyber. 9: 62-66.
* R. Vanaga and B. Sloka, “Financial and capital market commission financing: aspects and challenges, “Journal of Logistics, Informatics and Service Science, vol. 7, no.1, pp. 17-30, 2020.
* R. Qiao, “Stock prediction model based on neural network,”Operations Research and Management Science, vol. 28, no. 10, pp.132-140, 2019. [7] C. Jung and R. Boyd, “Forecasting UK stock prices,” Applied Financial Economics, vol.6, no.3, pp. 279-286,1996
* C. jung and R. Boyd, “Forecasting UK stock prices,” Applied Financial Economics, vol.6, no. 3, pp.279-286, 1996J.Li, S. Pan, L. Huang, and X. Zhu, “A machine learning based method for customer behavior prediction,” TehnickiVjesnikTechnical Gazette, vol.26, no.6,pp. 1670-1676, 2019.
* OpenCV: <http://docs.opencv.org/3.1.0/df/d9d/tutorialpycolorspaces.html>
* B. Sventlana and T. Ioannis, “An ensemble of LSTM neural networks for high-frequency stock market classification, “Journal of Forecasting, vol.38, no. 6, pp. 600-619, 2019.
* Coser, M. M. Maer-Matei, and C. Albu, “Predictive models for loan default risk assessment,” Economic Computation And 8 Economic Cybernetics Studies And Research, vol.53, no.2, pp. 149-165, 2019
* Pattern Recognition: <https://en.wikipedia.org/wiki/Pattern_recognition>