A

Project Report

On

‘DEEP EXPLANATION MODEL FOR FACIAL EXPRESSION RECOGNITION THROUGH FACIAL ACTION CODING UNIT’

Submitted for partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SRUTHI NOONE	**16TP1A0568**
TASIKUL ISLAM	**16TP1A05B3**
SHIVARAMAKRISHNA VARKALA	**16TP1A05A0**
PARVEEN SHEIK	**16TP1A0589**

UNDER THE GUIDANCE OF

MR. B. MAHENDAR REDDY
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIDDHARTHA INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, New Delhi, Accreditted by NBA, Hyderabad)

Vinobhanagar (V), Ibrahimpatnam (M), R.R. Dist. 501506.

2019-2020

SIDDHARTHA INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, New Delhi, Accreditted by NBA, Hyderabad)

Vinobhanagar (V), Ibrahimpatnam (M), R.R. Dist. 501506.

CERTIFICATE

This is to certify that the project work entitled **"DEEP EXPLANATION MODEL FOR FACIAL EXPRESSION RECOGNITION THROUGH FACIAL ACTION CODING UNIT"** is being submitted by

SRUTHI NOONE	**16TP1A0568**
TASIKUL ISLAM	**16TP1A05B3**
SHIVARAMAKRISHNA VARKALA	**16TP1A05A0**
PARVEEN SHEIK	**16TP1A0589**

in fulfillment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING** is recorded of bonafide work carried out by them. The result embodied in this report have been verified and found satisfactory.

MR. B. MAHENDAR REDDY
Assistant Professor

INTERNAL GUIDE

PROF N. SAINATH
Professor

HEAD OF THE DEPARTMENT

G. UDAY KUMAR
Assistant Professor

PROJECT CO-ORDINATOR

EXTERNAL EXAMINER

DECLARATION

This is to certify that the work reported in titled **“ DEEP EXPLANATION MODEL FOR FACIAL EXPRESSION RECOGNITION THROUGH FACIAL ACTION CODING UNIT ”** submitted to the Department Computer Science and Engineering, SIET**,** Ibrahimpatnam, Ranga Reddy in partial fulfillment of degree for the award of Bachelor of Technology, is a bonafide work done by us.

No part of this project is copied from any books, journals and wherever the portion is taken, the same has duly need referred in the text. The reported results are based on the project work entirely done by me and not copied from any other source.

We further state to the best of our knowledge that the matter embedded in this project has not been submitted by me in full or partial there of the award of any degree to any other institution or University.

NAME	**H.T.NO**
SRUTHI NOONE	**16TP1A0568**
TASIKUL ISLAM	**16TP1A05B3**
SHIVARAMAKRISHNA VARKALA	**16TP1A05A0**
PARVEEN SHEIK	**16TP1A0589**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

We extend our deep sense of gratitude to **PRINCIPAL**, **DR. E.L.NAGESH**, Siddhartha Institute of Engineering and Technology Vinobha nagar, for permitting us to undertake this project.

We are also thankful to **PROF N.SAINATH**, **HEAD OF THE DEPARTMENT**, Computer Science and Engineering, Siddhartha Institute of Engineering and Technology Vinobha nagar, for his support and guidance throughout our project, and as well as our **PROJECT CO-ORDINATOR G.UDAY KUMAR**, Assistant Professor, in Computer Science and Engineering department for his valuable support.

We would like to express our sincere gratitude and indebtedness to our **PROJECT SUPERVISOR MR. B. MAHENDAR REDDY**, M.Tech, Computer Science and Engineering, Siddhartha Institute of Engineering and Technology Vinobhanagar, for his support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully to completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

NAME	H.T.NO
SRUTHI NOONE	**16TP1A0568**
TASIKUL ISLAM	**16TP1A05B3**
SHIVARAMAKRISHNA VARKALA	**16TP1A05A0**
PARVEEN SHEIK	**16TP1A0589**

ABSTRACT

Facial expression is the most powerful and natural non-verbal emotional communication method. Facial Expression Recognition (FER) has significance in machine learning tasks. Deep Learning models perform well in FER tasks, but it doesn’t provide any justification for its decisions. Based on the hypothesis that facial expression is a combination of facial muscle movements, we find that Facial Action Coding Units (AUs) and Emotion label have a relationship in CK+ Dataset. In this paper, we propose a model which utilises AUs to explain Convolutional Neural Network (CNN) model’s classification results. The CNN model is trained with CK+ Dataset and classifies emotion based on extracted features. Explanation model classifies the multiple AUs with the extracted features and emotion classes from the CNN model. Our experiment shows that with only features and emotion classes obtained from the CNN model, Explanation model generates AUs very well.

This thesis introduces new efficient hardware implementations for the Advanced Encryption Standard (AES) algorithm. Two main contributions are presented in this thesis, the first one is a high speed 128 bits AES encryption, and the second one is a new 32 bits AES design. In first contribution 128 bits loop unrolled sub-pipelined AES encryption is presented. In this encryption an efficient merging for the encryption process sub-steps is implemented after relocating them. The second contribution presents a 32 bits AES design. In this design, the S-BOX is implemented with internal pipelining and it is shared between the main round and the key expansion units. Also, the key expansion unit is implemented to work on the fly and in parallel with the main round unit. These designs have achieved higher FPGA (Throughput/Area) efficiency comparing to previous AES designs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATION

S.NO	ABBREVATION	EXPANSION
1.	**DB**	DataBase
2.	**JVM**	Java Virtual Machine
3.	**JSP**	Java Server Page
4.	**PWS**	Personalised Web Search
5.	**UPS**	User Personalised Search
6.	**JRE**	Java Runtime Environment

CHAPTER - 1
INTRODUCTION

1.1 OBJECTIVE

One of the most critical stage for building facial expression system is extraction of proper features which describe physical phenomena. Conventional studies on FER tasks commonly used feature descriptor for FER are Histograms of oriented gradients (HOG), Local Gabor features and Weber Local Descriptor (WLD).

To extend the usage of deep learning models further in critical tasks where human life depends on, deep learning model must be justifiable; It must explain basis for its decision. However, current deep learning model lack this function. Facial Expression Recognition is one example that can be applied to critical tasks such as driver drowsiness detection by providing justifications.

1.2 PROBLEM SPECIFICATION

1.2.1 Existing System

One of the most critical stage for building facial expression system is extraction of proper features which describe physical phenomena. Conventional studies on FER tasks commonly used feature descriptor for FER are Histograms of oriented gradients (HOG), Local Gabor features and Weber Local Descriptor (WLD). However, the dimension of extracted features is typically high. Therefore, common reduction techniques such as Principal Component Analysis (PCA), Local Binary Pattern (LBP), Non-Negative Matrix Factorization (NMF) are used.

Existing System Technique

- ➢ We symbolize facial expression as a combination of muscle movements.
- ➢ In other words, facial expressions can be encoded with combinations of the atomic movements of facial muscles.
- ➢ In the study of facial expression, there is a system for mapping the movements of the facial muscle called Facial Action Coding System (FACS). FACS is a system to taxonomize human facial movements by their appearance on the face.

Drawbacks

- Large number of parameters.
- Does not enforce any structure.

1.2.2 Proposed System

Unlike conventional models, deep learning model can learn how to extract features by its ability to learn relationships over hidden layers. Deep learning models automatically learn to extract features suitable for FER tasks. Autoencoder model is used as feature descriptor for FER task. Convolutional Neural Networks (CNNs) perform well as not only the feature extractor but also the classifier. DNN model, Support Vector Machines (SVMs), Dynamic Bayesian Network and Gaussian Mixture Models GMMs) are commonly used. Hidden Markov Models (HMMs) are utilised to extract frame level features from facial image sequences.

Proposed System Technique

- The difficulty of using side features (that is, any features beyond the query ID/item ID). As a result, the model can only be queried with a user or item present in the training set.
- Relevance of recommendations. As you saw in the first Colab, popular items tend to be recommended for everyone, especially when using dot product as a similarity measure. It is better to capture specific user interests.

Advantages

- Facial Action Coding Units.
- No sub sampling.

1.3 METHODOLOGIES

Unlike conventional models, deep learning model can learn how to extract features by its ability to learn relationships over hidden layers. Deep learning models automatically learn to extract features suitable for FER tasks. Autoencoder model is used as feature descriptor for FER task. Convolutional Neural Networks (CNNs) perform well as not only the feature extractor but also the classifier. DNN model, Support Vector Machines (SVMs), Dynamic Bayesian Network and Gaussian Mixture Models GMMs) are commonly used. Hidden Markov Models (HMMs) are utilized to extract frame level features from facial image sequences.

1.4 CONTRIBUTIONS

Using FACS human coders can manually code nearly any automatically possible facial expression, deconstructing it into the specific action units (AU) and their temporal segments that produced the expression. As AUs are independent of any interpretation, they can be used for any higher order decision making process including recognition of basic emotions. Facial expression Recognition is one example that can be applied to critical tasks such as driver drowsiness detection by providing justification. So, we mainly contribute this technique to reduce accidents.

1.5 LAYOUT OF THE THESIS

In the chapter 1, we came to know the objectives of our project, problem specification, methodologies, Modules and Contribution. We got the idea of our project and came to know the entire description of the project.

A literature survey or a literature review is a project report is that section which shows the various analysis and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the intent of the project.

In the third chapter i.e., Requirement Specification, we describe about existing system along with its drawbacks and also the proposed system with its advantages.

In System Design, our deep explanation is about System architecture, UML diagram, Class Diagram, Object Diagram, State Diagram, Activity Diagram, Sequence Diagram, Collaboration Diagram, Data Flow Diagram, E-R diagrams and Modules along with their brief description.

In chapter 5, Implementation, we have discussed about Development Tools, given the Sample of Code that helps to work our project. I.e., for facial expression recognition, Technique for deep neural network models, Snapshots, Various Types of Testing and finally Test Cases

In the last chapter, we discuss about the Conclusion and Future Enhancements.

CHAPTER - 2
LITERATURE REVIEW

TITLE: The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression.

AUTHOR: Patrick Lucey , Jeffrey F. Cohn , Takeo Kanade.

YEAR: 2010

DESCRIPTION:

In 2000, the Cohn-Kanade (CK) database was released for the purpose of promoting research into automatically detecting individual facial expressions. Since then, the CK database has become one of the most widely used test-beds for algorithm development and evaluation. During this period, three limitations have become apparent: 1) While AU codes are well validated, emotion labels are not, as they refer to what was requested rather than what was actually performed, 2) The lack of a common performance metric against which to evaluate new algorithms, and 3) Standard protocols for common databases have not emerged. As a consequence, the CK database has been used for both AU and emotion detection (even though labels for the latter have not been validated), comparison with benchmark algorithms is missing, and use of random subsets of the original database makes meta-analyses difficult. To address these and other concerns, we present the Extended Cohn-Kanade (CK+) database. The number of sequences is increased by 22% and the number of subjects by 27%. The target expression for each sequence is fully FACS coded and emotion labels have been revised and validated. In addition to this, non-posed sequences for several types of smiles and their associated metadata have been added. We present baseline results using Active Appearance Models (AAMs) and a linear support vector machine (SVM) classifier using a leave-one-out subject cross-validation for both AU and emotion detection for the posed data. The emotion and AU labels, along with the extended image data and tracked landmarks will be made available July 2010.

TITLE: Do Deep Neural Networks Learn Facial Action Units When Doing Expression Recognition?

AUTHOR: Pooya Khorrami , Tom Le Paine.

YEAR: 2015

DESCRIPTION:

Despite being the appearance-based classifier of choice in recent years, relatively few works have examined how much convolutional neural networks (CNNs) can improve performance on accepted expression recognition benchmarks and, more importantly, examine what it is they actually learn. In this work, not only do we show that CNNs can achieve strong performance, but we also introduce an approach to decipher which portions of the face influence the CNN's predictions. First, we train a zero-bias CNN on facial expression data and achieve, to our knowledge, state-of-the-art performance on two expression recognition benchmarks: the extended Cohn-Kanade (CK+) dataset and the Toronto Face Dataset (TFD). We then qualitatively analyze the network by visualizing the spatial patterns that maximally excite different neurons in the convolutional layers and show how they resemble Facial Action Units (FAUs). Finally, we use the FAU labels provided in the CK+ dataset to verify that the FAUs observed in our filter visualizations indeed align with the subject's facial movements.

TITLE: Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order

AUTHOR: André TeixeiraLopes, Edilsonde Aguiar.

YEAR: 2017

DESCRIPTION:

Facial expression recognition has been an active research area in the past 10 years, with growing application areas including avatar animation, neuromarketing and sociable robots. The recognition of facial expressions is not an easy problem for machine learning methods, since people can vary significantly in the way they show their expressions. Even images of the same person in the same facial expression can vary in brightness, background and pose, and these variations are emphasized if considering different subjects (because of variations in shape, ethnicity among others). Although facial expression recognition is very studied in the literature, few works perform fair evaluation avoiding mixing subjects while training and testing the proposed algorithms. Hence, facial expression recognition is still a challenging problem in computer vision. In this work, we propose a simple solution for facial expression recognition that uses a combination of Convolutional Neural Network and specific image pre-processing steps. Convolutional Neural Networks achieve better accuracy with big data. However, there are no publicly available datasets with sufficient data for facial expression recognition with deep architectures. Therefore, to tackle the problem, we apply some pre-processing techniques to extract only expression specific features from a face image and explore the presentation order of the samples during training. The experiments employed to evaluate our technique were carried out using three largely used public databases (CK+, JAFFE and BU-3DFE). A study of the impact of each image pre-processing operation in the accuracy rate is presented. The proposed method: achieves competitive results when compared with other facial expression recognition methods – 96.76% of accuracy in the CK+ database – it is fast to train, and it allows for real time facial expression recognition with standard computers.

TITLE: Face expression recognition: A brief overview of the last decade

AUTHOR: Cătălin-Danicl Căleanu

YEAR: 2013

DESCRIPTION:

The huge research effort in the field of face expression recognition (FER) technology is justified by the potential applications in multiple domains: computer science, engineering, psychology, neuroscience, to name just a few. Obviously, this generates an impressive number of scientific publications. The aim of this paper is to identify key representative approaches for facial expression recognition research in the past ten years (2003-2012).

TITLE: Feature and label relation modeling for multiple-facial action unit classification and intensity estimation

AUTHOR: Shangfei Wang , Jiajia Yang.

YEAR: 2017

DESCRIPTION:

In this paper, we propose multiple facial Action Unit (AU) recognition and intensity estimation by modeling their relations in both feature and label spaces. First, a multi-task feature learning method is adopted to learn the shared features among the group of facial action units, and recognize or estimate their intensity simultaneously. Second, a Bayesian network is used to model the co-existent and mutual-exclusive semantic relations among action units. Finally, through probabilistic inference, the learned Bayesian network combines the results of the multi-task learning with the AU relations it captures to perform multiple AU recognition and AU intensity estimation. Experiments on the extended Cohn-Kanade database, the MMI database, the McMaster database and the DISFA database demonstrate the effectiveness of our method for both AU classification and AU intensity estimation.

TITLE: Autoencoder: Approach to the reduction of the dimension of the vector space with controlled loss of information

AUTHOR: Maxim V. Akinina , Natalya V. Akinina.

YEAR: 2015

DESCRIPTION:

The different ways of describing the characteristics of the image. The application of autoencoder for image classification. The results of experiments showing the effectiveness of autoencoder for solving pattern classification.

CHAPTER - 3
REQUIRMENT SPECIFICATION

3.1 GENERAL

There are two main approaches to solving the lack of explanations of deep learning model. The first is interpretable model, the second is explanation model. Studies on interpretable models consist of input modification methods, linking internal DNN activations with semantic concepts, aiming at discovering frequent mid-level visual patterns occurring in image collections.

3.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

• PROCESSOR	:	Intel Core 2 Duo.
• RAM	:	4GB DD RAM
• MONITOR	:	15" LCD,LED MONITOR
• HARD DISK	:	200 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Software Requirements

• FRONT END	:	python
• BACK END	:	CSV
• OPERATING SYSTEM	:	WINDOWS 7
• IDE	:	Spyder

3.4 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

3.5 NON-FUNCTIONAL REQUIREMENTS

Efficiency

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their multi-modal topics, but also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property of social event, which can effectively model social media documents including long text with related images and learn the correlations between textual and visual modalities to separate the visual-representative topics and non-visual-representative topics.

CHAPTER - 4
SYSTEM DESIGN

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

4.1 SYSTEM ARCHITECTURE

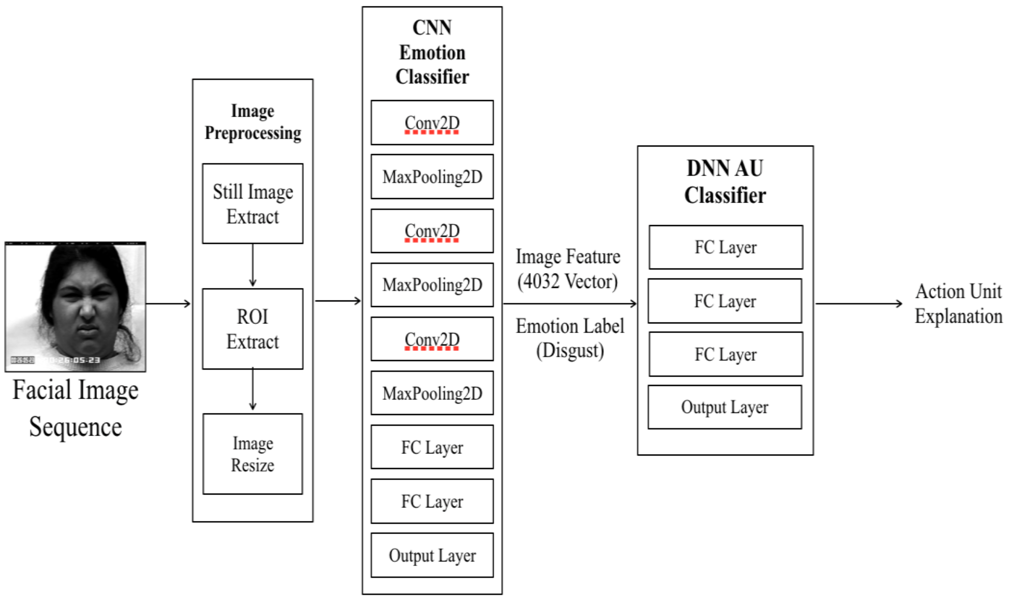


Fig 4.1 : System Architecture

Explanation :

A System Architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

4.2. Data Flow Diagram

Level-0:

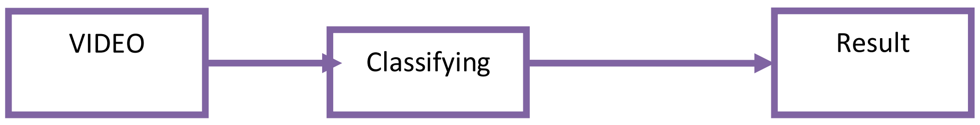


Fig 4.2.1 : Level-0

Level-1:

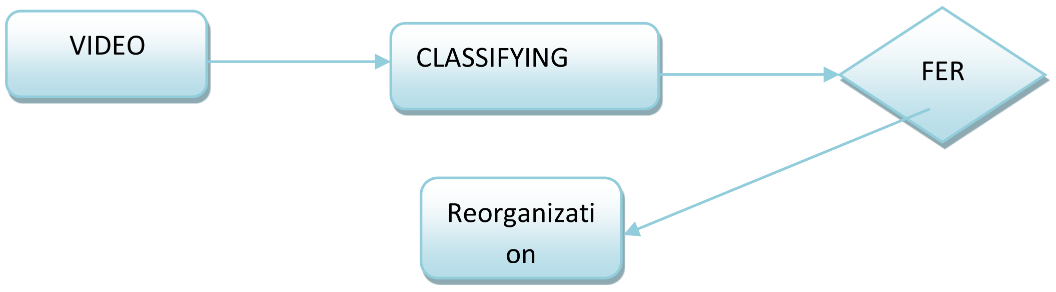


Fig 4.2.2 : Level-1

Level – 2:

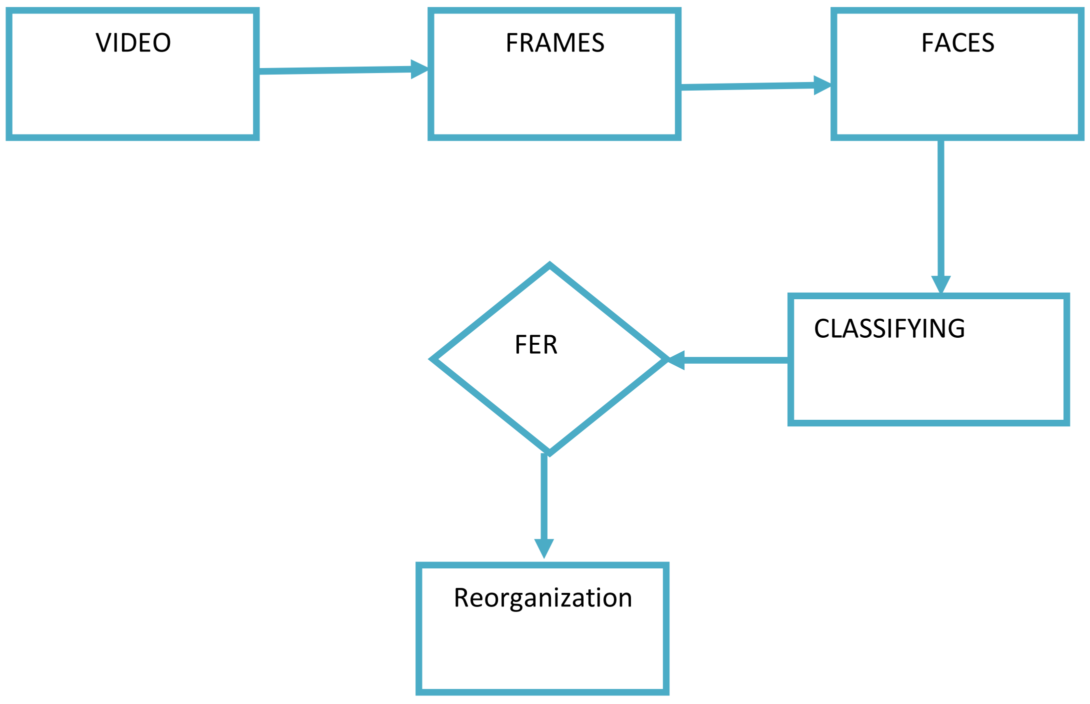


Fig 4.2.3 : Level-2

Explanation:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data 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 does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

4.3 UML DIAGRAMS

4.3.1 Use Case Diagram

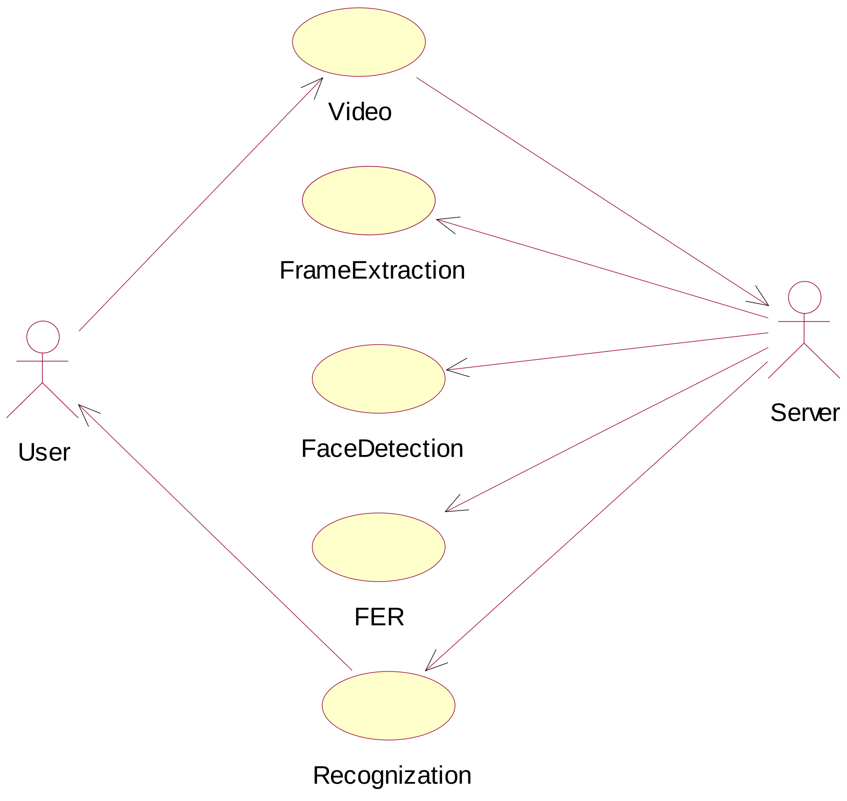


Fig 4.3.1 : Use Case Diagram

Explanation:

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

4.3.2 Class Diagram

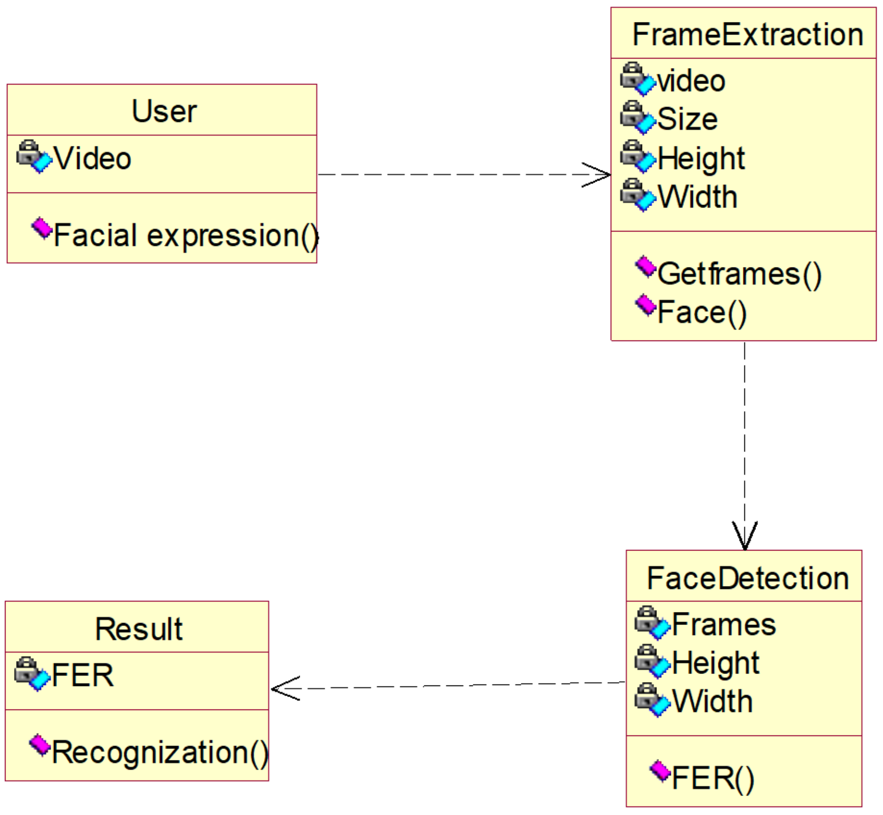


Fig 4.3.2 : Class Diagram

Explanation:

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

4.3.3 Object Diagram

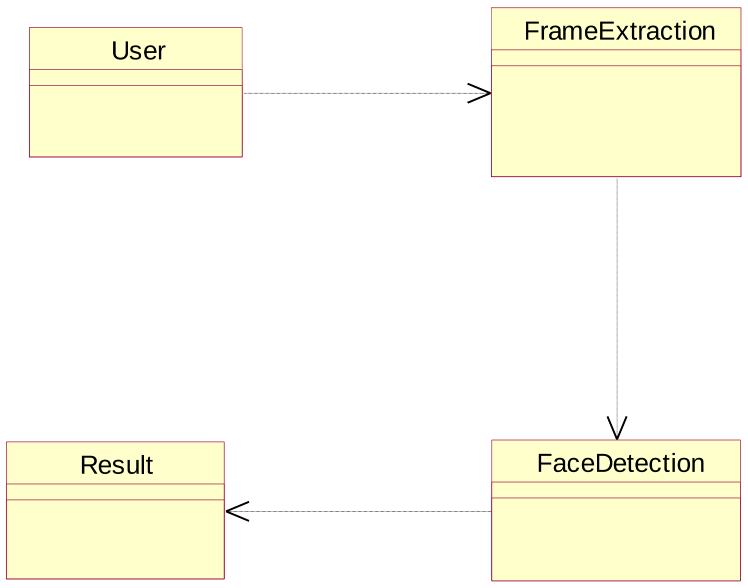


Fig 4.3.3 : Object Diagram

Explanation:

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

4.3.4 State Diagram

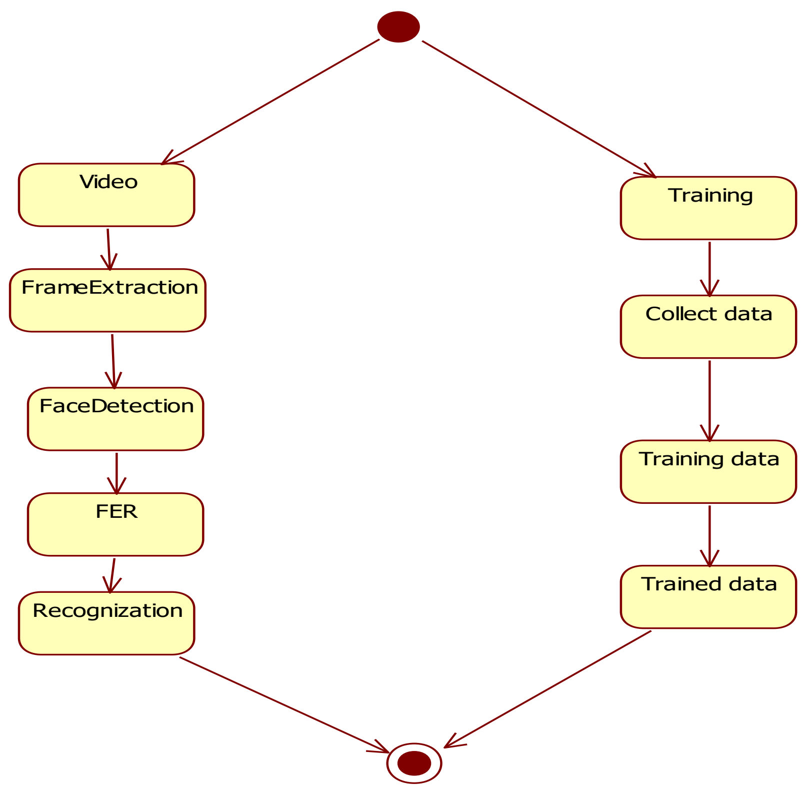


Fig 4.3.4 : State Diagram

Explanation:

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.

4.3.5 Activity Diagram

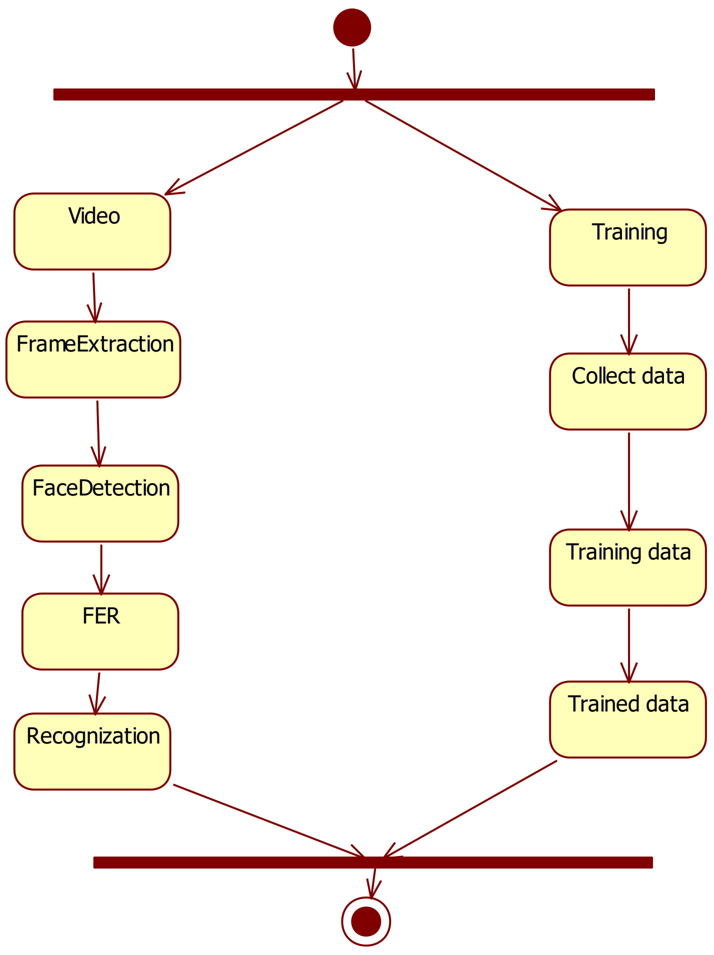


Fig 4.3.5 : Activity Diagram

Explanation:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

4.3.6 Sequence Diagram

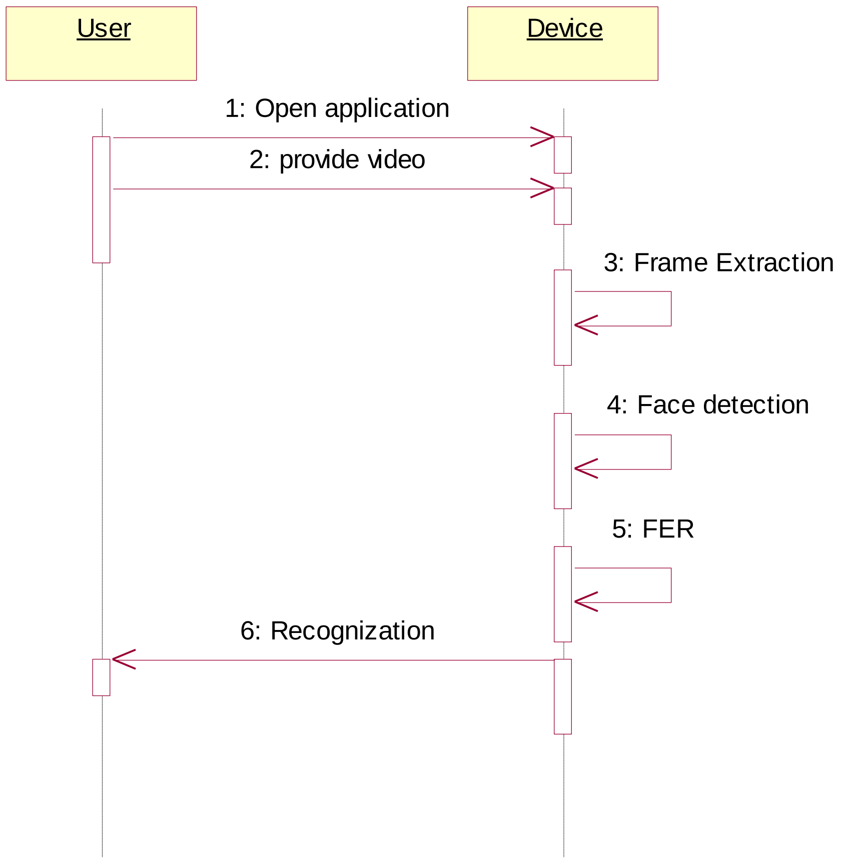


Fig 4.3.6 : Sequence Diagram

Explanation:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

4.3.7 Collaboration Diagram

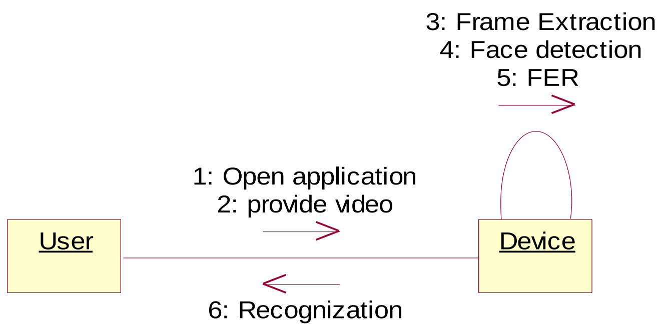


Fig 4.3.7 : Collaboration Diagram

Explanation:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

4.4. E-R Diagram

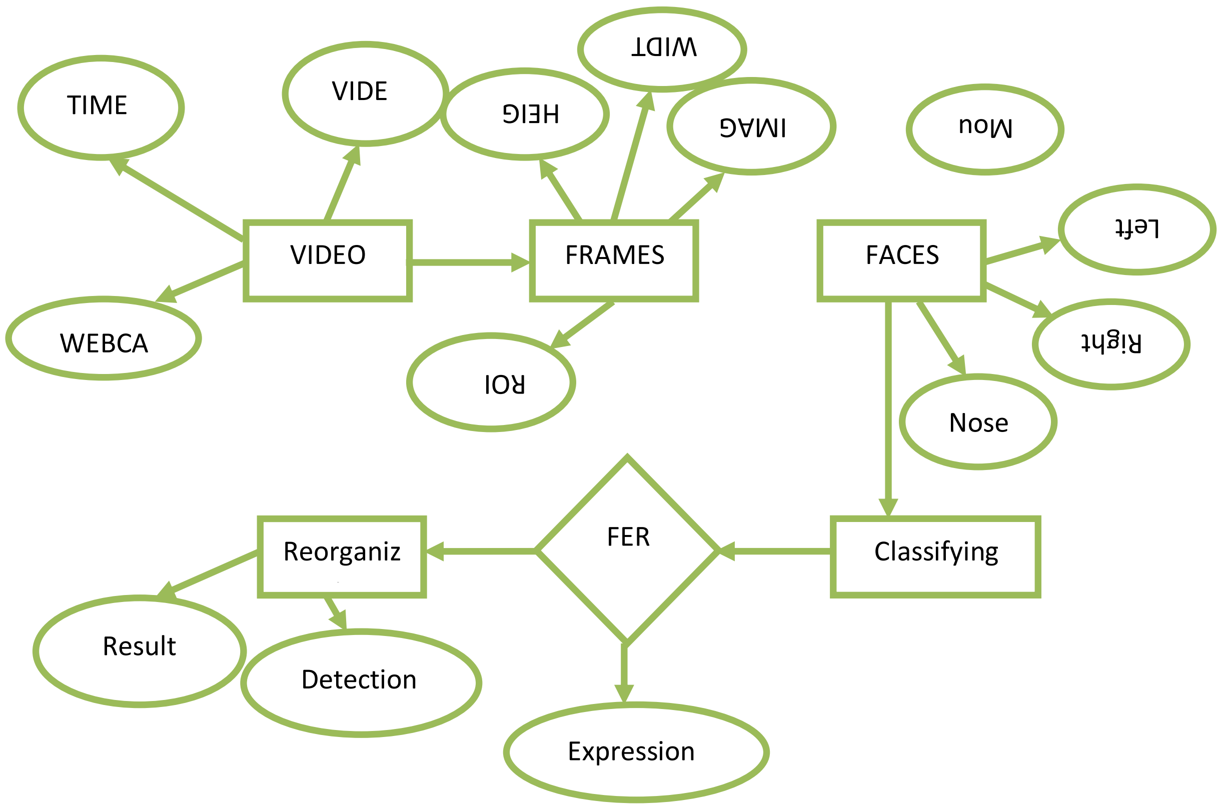


Fig 4.4 : E-R Diagram

Explanation:

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators.

4.5 MODULES

4.5.1. User Interface Design

This is the first module of our project. In this the application user's first create their account properly which are stored at the back end for verification or for providing security to the accounts. If user wants to get into his account first they have to submit their constraints such as username, password and so on…otherwise can't able to access the account. In our project according to actions they are performing we disperse the users as admin or normal application user.

4.5.2 FER (Facial Expression Recognition)

Facial expression recognition system is a computer-based technology and therefore, it uses algorithms to instantaneously detect faces, code facial expressions, and recognize emotional states. It does this by analyzing faces in images or video through computer powered cameras embedded in laptops, mobile phones, and digital signage systems, or cameras that are mounted onto computer screens.

4.5.3 Face Detection

Locating faces in the scene, in an image or video footage.

4.5.4 Facial Landmark Detection

Extracting information about facial features from detected faces. For example, detecting the shape of facial components or describing the texture of the skin in a facial area.

4.5.5 Facial Expression And Emotion Classification

Analyzing the movement of facial features and/or changes in the appearance of facial features and classifying this information into expression-interpretative categories such as facial muscle activations like smile or frown; emotion categories happiness or anger; attitude categories like (dis)liking or ambivalence.

4.5.6 Video Streaming

The video is recorded utilizing webcam and the casings are extricated and handled in a PC. Subsequent to removing the casings, picture preparing methods are applied on these 2D pictures. Directly, manufactured user information has been created.

CHAPTER - 5

IMPLEMENTATION

5.1 DEVELOPMENT TOOLS

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Importance of Python

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Features of Python

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Machine learning challenges that can solve by using python

- **Python for supervised learning**
  Supervised machine learning is one of the most widely-applied uses of AI. In supervised learning, an algorithm learns from a labeled dataset with the output already being known. Two of the main techniques within this category are classification and regression.

  Classification is used to categorize data into desired, distinct classes and predict a discrete value. It can serve to assess creditworthiness or help with medical diagnostics.

  Regression is used in problems that involve continuous numbers, including demand and financial forecasting, as well as property price estimation. The predicted outcome here is an estimation of a numeric value.

  Both classification and regression problems can be solved thanks to a large number of Python libraries.

- **Python for unsupervised learning**

  Clustering and matrix factorization are two of the most common unsupervised machine learning methods. Frequently applied in user segmentation and recommender systems, both methods are used to group elements based on the similarity between object properties.

  Some of the most popular libraries used in clustering and recommendation system engines are:
  Surprise (neighborhood-based methods, SVD, PMF, SVD++, NMF).
  LightFM (hybrid latent representation recommender with matrix factorization).
  Spotlight (uses PyTorch to build recommender models).

- **Python for reinforcement learning**
  Reinforcement learning algorithms learn to modify their behavior to make the right decisions after receiving feedback. They have been **tested in self-learning solutions,** including video games and traffic light control systems.Problems posed by reinforcement learning are often highly specific and finding solutions to them may prove quite challenging.

Libraries Used In Python

- **NumPy** - mainly useful for its N-dimensional array objects.
- **pandas** - Python data analysis library, including structures such as dataframes.
- **matplotlib** - 2D plotting library producing publication quality figures.
- **scikit-learn** - Scikit-learn is the best known and arguably most popular Python library for machine learning. Built on SciPy and NumPy—and designed to interoperate with them—scikit-learn is open source, accessible to all, and reusable in a number of contexts.

  The library features a wide variety of algorithms for: classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. These algorithms include: support-vector machines, random forests, gradient boosting, k-means, and DBSCAN.

  Yet despite the barrage of options scikit-learn provides, the data mining and data analysis tools it offers are both simple and efficient.

- **TensorFlow** - TensorFlow was originally developed by engineers and researchers at Google to meet their needs for a system that can build and train neural networks to find and decipher correlations and patterns. The process was designed as analogous to the ways humans reason and learn.

  The flexible, high-performance architecture of the open-source library makes it easy to deploy numerical computation across multiple platforms, as well as from desktops to server clusters to mobile devices.

  TensorFlow is used by companies such as Uber, Dropbox, eBay, Snapchat, or Coca Cola—not to mention Google, of course. This pretty much speaks for itself.

- **nilearn** - Nilearn is a high-level Python library for easy and fast statistical learning on neuroimaging data. The library leverages scikit-learn for a plethora of advanced machine learning techniques, such as pattern recognition or multivariate statistics. The applications of this include predictive modelling and connectivity analysis, among others.

  Constructing domain-specific feature engineering is the highest value nilearn holds for machine-learning experts. This means shaping neuroimaging data into a matrix of features perfect for statistical learning, or the other way around.

- **mlpy -** Mlpy is a high-performance Python library for predictive modeling, built on top of SciPy, NumPy, and the GNU Scientific Libraries. Multiplatform and open source, mlpy aims to provide solutions for supervised and unsupervised problems, offering an extensive range of cutting-edge methods.

  Finding a sensible compromise between efficiency, modularity, reproducibility, maintainability, and usability is the prime goal of mlpy.

Fig - 5.1 : NumPy, Pandas, Matplotlib, Scikit-learn

Reasons for Choosing Python

Readable and Maintainable Code

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

Multiple Programming Paradigms

Like other modern programming languages, Python also supports several programming paradigm. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.

Compatible with Major Platforms and Systems

At present, Python is supports many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.

Robust Standard Library

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols. You can even gather information about various modules by browsing through the Python Standard Library documentation.

Many Open Source Frameworks and Tools

As an open source programming language, Python helps you to curtail software development cost significantly. You can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost. You even have option to choose from a wide range of open source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy. Likewise, you can accelerate desktop GUI application development using **Python GUI frameworks** and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.

Simplify Complex Software Development

Python is a general purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many Python developers even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

Adopt Test Driven Development

Python to create prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. You can easily write the required tests before writing code and use the tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code.

However, Python, like other programming languages, has its own shortcomings. It lacks some of the built-in features provided by other modern programming language. Hence, you have to use Python libraries, modules, and frameworks to accelerate custom software development. Also, several studies have shown that Python is slower than several widely used programming languages including Java and C++. You have to speed up the Python application by making changes to the application code or using custom runtime. But you can always use Python to speed up software development and simplify software maintenance.

The Fields which is Using Python

Python is everywhere. You may not even realize how widespread it is.The prominence of Guido van Rossum's creation can be attributed to a number of factors.Most of all, Python is easy to learn, clear to read, and simple to write in. This speeds up development without sacrificing reliability or scalability.

Python for Web Development

In the current market, a business without a website might as well not exist. Moreover, the trends are pushing for more and more impressive web apps that, among others, include:flawless mobile and desktop versions,asymmetrical layouts,Progressive Web Apps,integrated animations,ML-powered chatbots.

Nowadays, more than ever, it's important to choose the right tools when the time comes to build (or, quite likely, rebuild) your own website or web app.

Python has a large selection of pre-built libraries for just about anything.Scientific computing, image processing, data processing, machine learning, deep learning you name it, Python has it.

Python code takes less time to write due to its simple and clean syntax.Because ofthis,code written in Python lends itself very well to creating quick prototypes.

Python accelerates the ROI of commercial projects.The reason behind this is similar to the previous point: you can write and ship your code faster. This is especially important for startups.

Python has a built-in framework for unit tests.This helps you ship bug-free code.In addition to Python's standard features, one of its major strengths in web development is the variety of web frameworks it offers.

With a large selection of well-supported frameworks, you can find the right starting point for any project. Python gives you the tools to get the job done reliably, no matter whether you’re looking to focus on fast-to-implement and out-of-the-box solutions,solutions that require many specialized microservices working together,an app where performance is crucial.

Python for the Internet of Things

Python's popularity is a considerable asset.The language is supported by a large, helpful community, which has led to the creation of an extensive set of pre-written libraries, making it easier to implement and deploy working solutions.

Python is portable, expandable, and embeddable.This makes Python not system-dependent and allows it to support many of the single-board computers currently available on the market, regardless of the architecture or operating system.

Python works great for managing and organizing complex data.For IoT systems that are particularly data-heavy, this is especially useful.

Python is easy to learn without forcing you to get acquainted with many formatting standards and compiling options.The most immediate consequence of this are faster results.

Python code is easily readable and compact thanks to its clean syntax.This is helpful on small devices with limited memory and computational power. Additionally, the syntax is partly responsible for Python's growing popularity, further strengthening its community.

Python's close relation to scientific computing has allowed it to gain ground in IoT development.

Python is the language of choice for the Raspberry Pi.Raspberry Pi is one of the most popular microcontrollers on the market.Python offers tools that streamline the IoT development process, such as webrepl.This gives you the option to use your browser to run Python code for IoT. Additionally, the mqtt messaging protocol lets you update your code/config.

Since Python is an interpreted language, you can easily test your solution without compiling the code or flashing the device.With a C program, you would have to compile the code on your PC, then upload it to your "thing."Python allows you to log into the interpreter directly on your "thing," making it easier to test various solutions.

Python for Fintech

Hedge fund and investment banking industries have long decided that Python is an ideal choice for fintech because the language addresses many of their highly specific needs.

Creating platforms for risk and trade management.Quantitative rate problem solving.Simplifying data regulation, compliance, and analytics by leveraging the abundance of Python libraries.

Fintech belongs with Python for a number of reasons

Python uses Clean syntax.Python code is easy to understand, because it resembles actual English. This allows developers to learn it quickly and to become fairly competent in it within a short time.

Python is a dynamically typed language, making development in it quicker than in statically typed languages such as Java.When writing in Python, you need less code, which in turn allows for faster deployment.

Python Having helpful libraries. Python boasts an extensive array of libraries for a plethora of purposes; a lot of those are excellent for fintech and finance.

5.2 TECHNIQUE USED

Deep Neural Network Models

The difficulty of using side features (that is, any features beyond the query ID/item ID). As a result, the model can only be queried with a user or item present in the training set.

Relevance of recommendations. As you saw in the first Colab, popular items tend to be recommended for everyone, especially when using dot product as a similarity measure. It is better to capture specific user interests.

5.3 SAMPLE OF CODE

```
# load json and create model

from __future__ import division

from keras.models import Sequential

from keras.layers import Dense

from keras.models import model_from_json

import numpy

import os

import numpy as np

import cv2



#loading the model

json_file = open('fer.json', 'r')

loaded_model_json = json_file.read()

json_file.close()
```

```
loaded_model = model_from_json(loaded_model_json)


# load weights into new model

loaded_model.load_weights("fer.h5")

print("Loaded model from disk")

#setting image resizing parameters

WIDTH = 48

HEIGHT = 48

x=None

y=None

labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']


#loading image

full_size_image = cv2.imread("inputs/im6.jpg")

print("Image Loaded")

gray=cv2.cvtColor(full_size_image,cv2.COLOR_RGB2GRAY)

face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

faces = face.detectMultiScale(gray, 1.3  , 10)
```

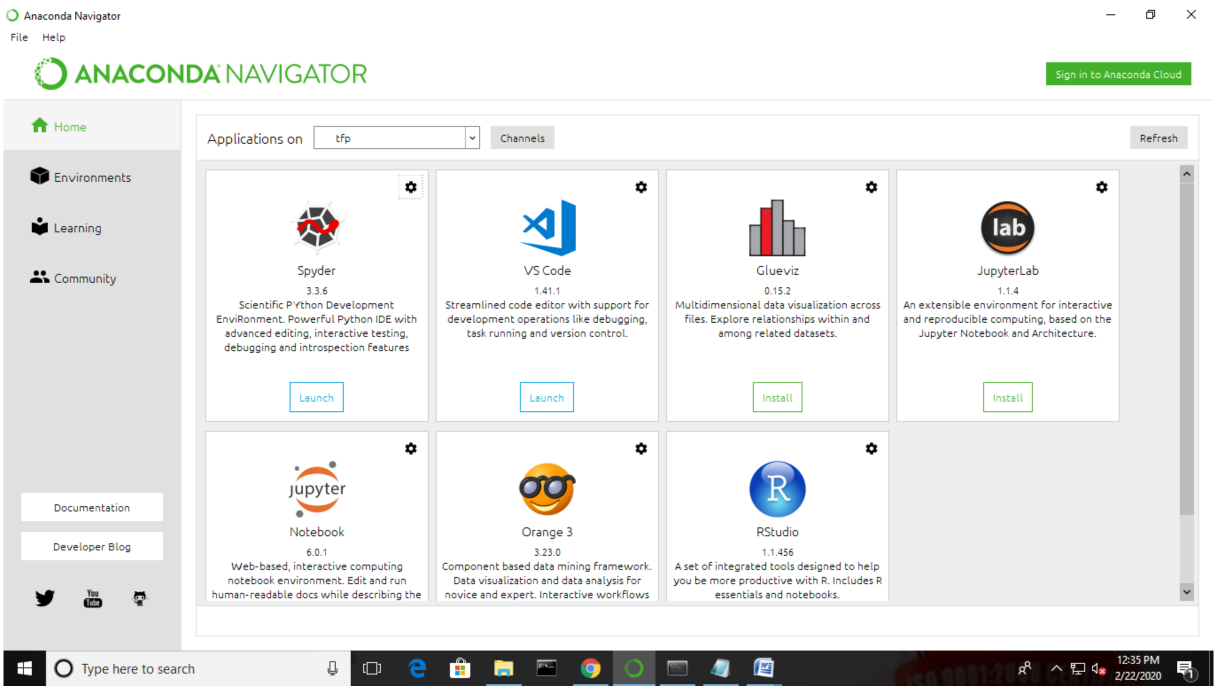
```
#detecting faces

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

    roi_gray = gray[y:y + h, x:x + w]

    cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)

  cv2.normalize(cropped_img, cropped_img, alpha=0, beta=1,
norm_type=cv2.NORM_L2, dtype=cv2.CV_32F)

    cv2.rectangle(full_size_image, (x, y), (x + w, y + h), (0, 255, 0), 1)

#predicting the emotion

 yhat= loaded_model.predict(cropped_img)

 cv2.putText(full_size_image, labels[int(np.argmax(yhat))], (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1, cv2.LINE_AA)

 print("Emotion: "+labels[int(np.argmax(yhat))])

cv2.imshow('Emotion', full_size_image)

cv2.waitKey()

cv2.destroyAllWindows()
```

Train.py

```
import sys, os

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split
from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Conv2D, MaxPooling2D, BatchNormalization

from keras.losses import categorical_crossentropy

from keras.optimizers import Adam

from keras.regularizers import l2

from keras.callbacks import ReduceLROnPlateau, TensorBoard, EarlyStopping,
ModelCheckpoint

from keras.models import load_model

from keras.models import model_from_json

num_features = 64

num_labels = 7

batch_size = 64

epochs = 100

width, height = 48, 48

x = np.load('./fdataX.npy')

y = np.load('./flabels.npy')

x -= np.mean(x, axis=0)

x /= np.std(x, axis=0)
```

```
#for xx in range(10):

#    plt.figure(xx)

#    plt.imshow(x[xx].reshape((48, 48)), interpolation='none', cmap='gray')

#plt.show()



#splitting into training, validation and testing data

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=42)

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.1,
random_state=41)



#saving the test samples to be used later

np.save('modXtest', X_test)

np.save('modytest', y_test)



#desinging the CNN

model = Sequential()

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', input_shape=(width,
height, 1), data_format='channels_last', kernel_regularizer=l2(0.01)))

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Dropout(0.5))

model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Dropout(0.5))

model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='relu',
padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3), activation='relu',
padding='same'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Dropout(0.5))
model.add(Flatten())

model.add(Dense(2*2*2*num_features, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(2*2*num_features, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(2*num_features, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num_labels, activation='softmax'))


#model.summary()

#Compliling the model with adam optimixer and categorical crossentropy loss

model.compile(loss=categorical_crossentropy,

              optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-7),

              metrics=['accuracy'])


#training the model

model.fit(np.array(X_train), np.array(y_train),

          batch_size=batch_size,

          epochs=epochs,

          verbose=1,

          validation_data=(np.array(X_valid), np.array(y_valid)),
          shuffle=True)

#saving the  model to be used later

fer_json = model.to_json()

with open("fer.json", "w") as json_file:

    json_file.write(fer_json)

model.save_weights("fer.h5")

print("Saved model to disk")
```

5.4 SNAPSHOTS



Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.41.1
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

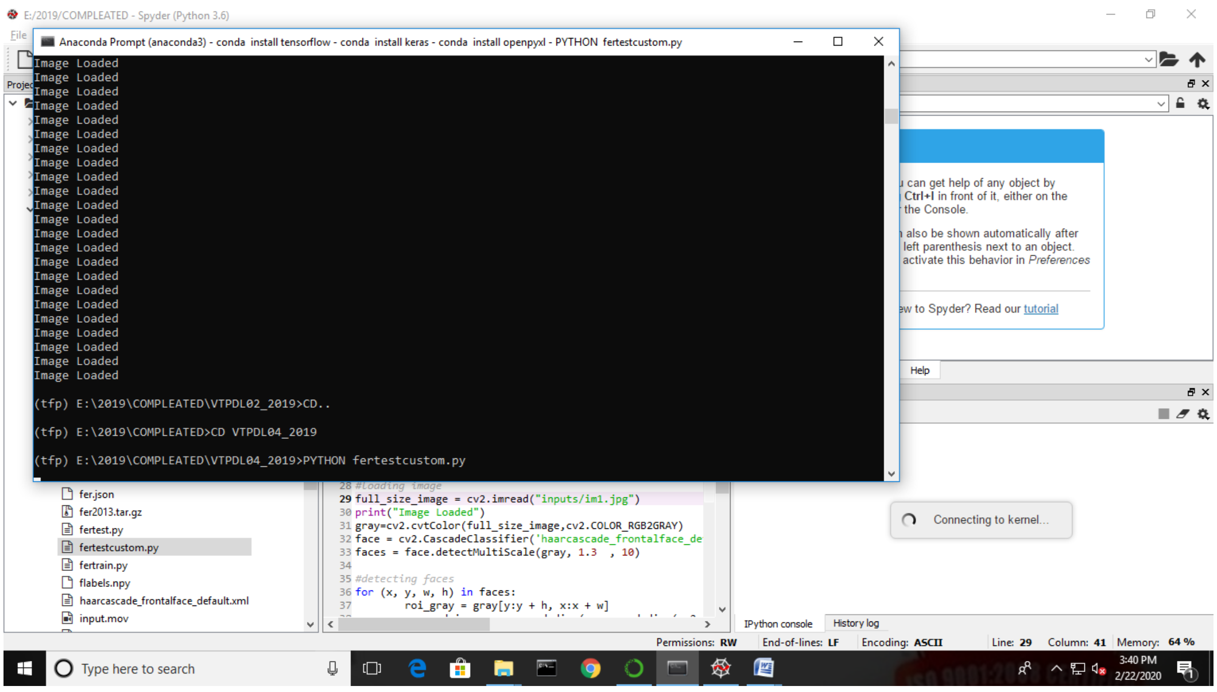
JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the

Orange 3
3.23.0
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Fig 5.4.1 : Anaconda Navigator



```
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded

(tfp) E:\2019\COMPLEATED\VTPDL02_2019>CD..

(tfp) E:\2019\COMPLEATED>CD VTPDL04_2019

(tfp) E:\2019\COMPLEATED\VTPDL04_2019>PYTHON fertestcustom.py
```

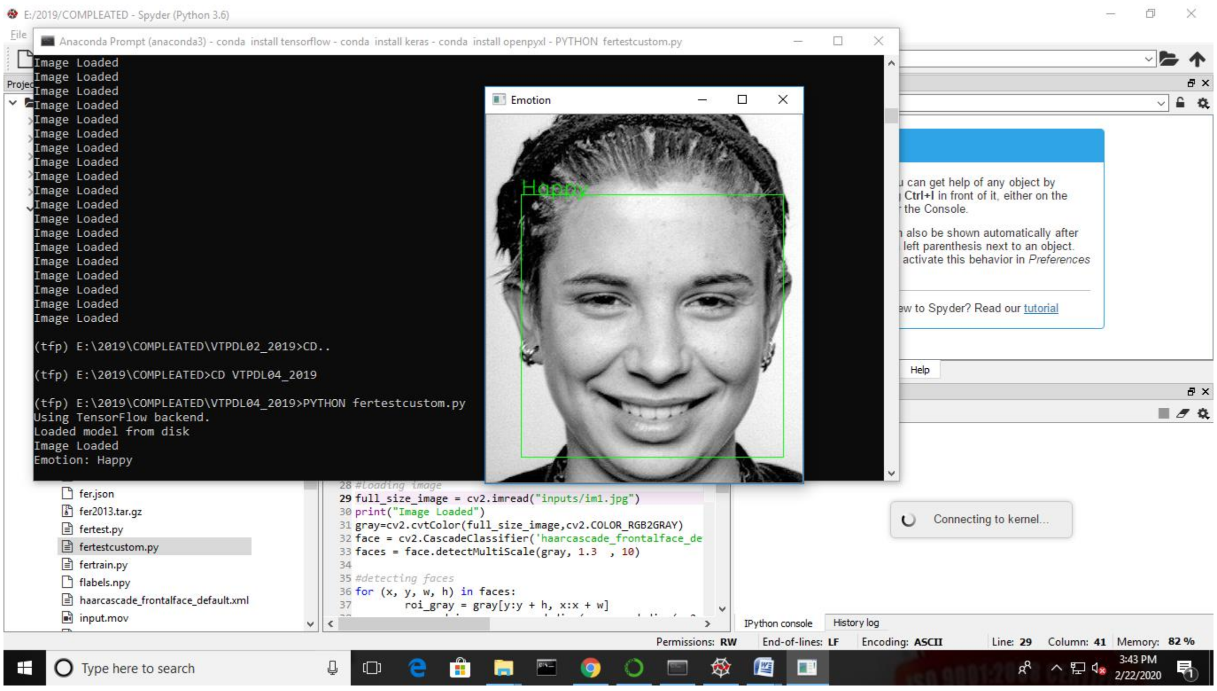
```
#loading image
full_size_image = cv2.imread("inputs/im1.jpg")
print("Image Loaded")
gray=cv2.cvtColor(full_size_image,cv2.COLOR_RGB2GRAY)
face = cv2.CascadeClassifier('haarcascade_frontalface_de
faces = face.detectMultiScale(gray, 1.3  , 10)

#detecting faces
for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
```

u can get help of any object by Ctrl+I in front of it, either on the the Console.

also be shown automatically after left parenthesis next to an object. activate this behavior in *Preferences*

w to Spyder? Read our tutorial

Fig 5.4.2 : Loading Images on Anaconda Terminal


E:/2019/COMPLEATED - Spyder (Python 3.6)
Anaconda Prompt (anaconda3) - conda install tensorflow - conda install keras - conda install openpyxl - PYTHON fertestcustom.py
Emotion
Happy

```
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded
Image Loaded

(tfp) E:\2019\COMPLEATED\VTPDL02_2019>CD..

(tfp) E:\2019\COMPLEATED>CD VTPDL04_2019

(tfp) E:\2019\COMPLEATED\VTPDL04_2019>PYTHON fertestcustom.py
Using TensorFlow backend.
Loaded model from disk
Image Loaded
Emotion: Happy
```

```
#loading image
full_size_image = cv2.imread("inputs/im1.jpg")
print("Image Loaded")
gray=cv2.cvtColor(full_size_image,cv2.COLOR_RGB2GRAY)
face = cv2.CascadeClassifier('haarcascade_frontalface_de
faces = face.detectMultiScale(gray, 1.3  , 10)

#detecting faces
for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
```

Fig 5.4.3: Loading Model from Disk

CHAPTER 6

SOFTWARE TESTING

6.1 GENERAL

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, sub-assemblies, 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 and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce 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 a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

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

6.2.3 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 of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.2.4 Performance Test

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

6.2.5 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

6.2.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Acceptance testing for Data Synchronization

- The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
- The Route add operation is done only when there is a Route request in need
- The Status of Nodes information is done automatically in the Cache Updation process.

6.3 TEST CASES

Test Case 1

Given Input	**:** abc
Correct Input	**:** abc
Result	**:** Accepted

Test Case 2

Given Input	**:** acb
Correct Input	**:** abc
Result	**:** Rejected

6.4 TEST RESULTS

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

CHAPTER 7

CONCLUSIONS & FUTURE ENHANCEMENTS

CONCLUSIONS

This paper suggests a novel model to solve lack of human intertable explanation for deep learning classifiers. Based on the hypothesis that facial expression is a combination of facial muscle movements, our DNN model explains the decision of CNN model that classify the emotion from the facial images through AUs. DNN Explanation model generates AUs with assumption that features and emotion labels are the product of CNN's decision making. Our model achieves better performance than existing AU classifier. With the Explanation model, we can know why the facial images are classified.

FUTURE ENHANCEMENTS

Two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network.

REFERENCES

[1] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, “The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression”, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, San Francisco, CA, 2010, pp. 94–101.

[2] Pooya Khorrami, Tom Le Paine, and Thomas S. Huang. “Do Deep Neural Networks Learn Facial Action Units When Doing Expression Recognition?.”, In Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW) (ICCVW ’15), IEEE Computer Society, Washington. DC, USA, 2015, pp. 19–27.

[3] P. Ekman and W. V. Friesen. Facial action coding system. 1977. 1, 2

[4] Andr Teixeira Lopes, Edilson de Aguiar, Alberto F. De Souza, Thiago Oliveira-Santos, Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order”, Pattern Recognition 61, 2017, pp. 610-628

[5] Shangfei Wang, Jiajia Yang, Zhen Gao, and Qiang Ji. 2017. Feature and label relation modeling for multiple-facial action unit classification and intensity estimation. Pattern Recognition. 65, C (May 2017), 71–81. DOI: https://doi.org/10.1016/j.patcog.2016.12.007

[6] G. Littlewort, J. Whitehill, T. Wu, I. Fasel, M. Frank, J. Movellan, M. Bartlett, The computer expression recognition toolbox (cert), in: Proceedings of the 2011 IEEE International Conference on,Automatic Face Gesture Recognition and Workshops FG 201), march 2011, pp. 298-305.

[7] Jyoti Kumari, R. Rajesh, K.M. Pooja, Facial Expression Recognition: A Survey, Procedia Computer Science, Volume 58, 2015, Pages 486-491

[8] Muhammad Usman , Siddique Latif, Junaid Qadir, “Using Deep Autoencoders for Facial Expression Recognition”, 2017 13th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 2017, pp. 1–6

[9] Cleanu, Ctlin. “Face expression recognition: A brief overview of the last decade.” SACI 2013 - 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, Proceedings. 157- 161. 10.1109/SACI.2013.6608958. 2013.

[10] M. V. Akinin, N. V. Akinina, A. I. Taganov, and M. B. Nikiforov, “Autoencoder: Approach to the reduction of the dimension of the vector space with controlled loss of information,” in Embedded Computing (MECO), 2015 4th Mediterranean Conference on. IEEE, 2015, pp. 171– 173.