ABSTRACT:

This project addresses the escalating prevalence of stress-related concerns by proposing a multi-modal human stress detection system combining facial recognition and emotion analysis. The method involves capturing facial images through a web camera using OpenCV's Haar cascade algorithm for robust face detection. Isolated faces undergo processing through a pre-trained Convolutional Neural Network (CNN) model, tailored for stress estimation. Simultaneously, emotional features are extracted using a k-Nearest Neighbors (KNN) algorithm. This multi-modal approach enhances stress level assessments by capturing both physiological and emotional cues. The system offers real-time, accurate stress monitoring, making it suitable for diverse applications, from workplaces to personal well-being initiatives. The integration of OpenCV's face detection, CNN, and KNN provides a versatile and potent stress detection solution, contributing to mental health monitoring and intervention efforts. The project's innovative combination of computer vision, deep learning, and machine learning techniques demonstrates a comprehensive and adaptable approach to addressing the pressing challenges of stress detection in contemporary society.

CHAPTER 1

INTRODUCTION:

In modern society, the escalating prevalence of emotion-related concerns has underscored the critical need for accurate and non-intrusive emotion detection methods. Conventional approaches often fall short in terms of precision and real-time capabilities. This project seeks to address these limitations by proposing a human emotion detection system utilizing facial recognition. The objective is to harness the capabilities of computer vision and deep learning to precisely assess emotion levels through facial features, offering a versatile solution for mental health monitoring and well-being initiatives.

BACKGROUND:

The increasing awareness of mental health and well-being has led to a growing demand for innovative technologies capable of monitoring and addressing emotion-related issues. Traditional emotion assessment methods, such as self-reporting or physiological measurements, face challenges in providing real-time and objective insights into an individual's emotion levels. Leveraging advancements in computer vision and deep learning, this project aims to pioneer a more accurate and efficient approach to emotion detection, focusing on facial cues as indicators of emotion.

MOTIVATION:

The motivation behind this work stems from the limitations of existing emotion detection methods and the potential impact on individual well-being. Accurate and timely identification of emotion is crucial for early intervention and personalized support. By combining the capabilities of OpenCV's face detection with a pre-trained Convolutional Neural Network (CNN) model, we aim to create a robust system that can adapt to diverse datasets and provide nuanced emotion assessments. The motivation is rooted in the potential of this technology to contribute significantly to mental health monitoring, offering individuals and healthcare professionals valuable insights for timely interventions.

PROPOSED SOLUTION:

The proposed method unfolds in two crucial steps. First, facial images are captured through a web camera, and the Haar cascade algorithm from OpenCV is employed for robust face detection. This initial step ensures the efficient isolation of facial regions for subsequent analysis. Subsequently, the isolated faces undergo processing through a meticulously trained pre-trained Convolutional Neural Network (CNN) model tailored explicitly for emotion estimation. This model, trained on a diverse dataset, is designed to identify nuanced facial cues indicative of emotion, thereby enhancing the accuracy of emotion level assessments. The amalgamation of OpenCV's face detection and the deep learning model creates a potent and adaptable emotion detection system, poised to revolutionize the field of mental health monitoring.

NECESSITY:

In contemporary society, the escalating prevalence of emotion-related concerns has brought forth a pressing need for advanced and non-intrusive emotion detection methods. Conventional approaches, often reliant on self-reporting or physiological measurements, face limitations in terms of precision and real-time capabilities. The challenges associated with these traditional methods underscore the necessity for a more accurate and efficient emotion detection system. Such a system would not only address the growing concerns surrounding mental well-being but also contribute to early intervention strategies, thereby mitigating the potential impact of prolonged emotion on individuals.

ADVANTAGES:

Real-time Monitoring:

The emotion detection system enables real-time monitoring of emotion levels, allowing for prompt identification and timely interventions.

Non-intrusive Assessment:

Leveraging facial recognition, the system offers a non-intrusive approach to emotion detection, enhancing user comfort and acceptability.

Objective and Quantifiable Results:

Integration of computer vision and deep learning provides an objective and quantifiable assessment, reducing subjectivity associated with traditional methods.

Adaptability to Diverse Datasets:

The pre-trained Convolutional Neural Network (CNN) model ensures adaptability to diverse datasets, improving accuracy across demographic groups and cultural contexts.

Potential for Early Intervention:

The system identifies subtle facial cues indicative of emotion, presenting an opportunity for early intervention and prevention of escalated emotion-related issues.

Versatility for Mental Health Initiatives:

The system's versatility positions it as a valuable tool for broader mental health initiatives, supporting remote monitoring and offering personalized support for improved mental well-being. In modern society, the escalating prevalence of stress-related concerns has highlighted the imperative need for accurate and non-intrusive stress detection methods. Conventional approaches often fall short in terms of precision and real-time capabilities. This project addresses these limitations by proposing a multi-modal human stress detection system that incorporates both facial recognition and emotion analysis. The goal is to leverage the power of computer vision, deep learning, and machine learning to precisely assess stress levels through facial features and emotional cues, offering a versatile solution for mental health monitoring and well-being initiatives.

The proposed method unfolds in three key steps. Initially, facial images are captured through a web camera, utilizing the Haar cascade algorithm from OpenCV for robust face detection. This ensures the efficient isolation of facial regions for subsequent analysis. Following this, the isolated faces undergo processing through a pre-trained Convolutional Neural Network (CNN) model, specifically tailored for stress estimation. The CNN is meticulously trained on a diverse dataset to identify nuanced facial cues indicative of stress, thereby enhancing the accuracy of stress level assessments.

In parallel, an additional dimension is added to the stress detection system by incorporating emotion analysis using a k-Nearest Neighbors (KNN) algorithm. This step aims to extract emotional features from facial expressions, contributing to a more comprehensive understanding of the individual's mental state. The combination of OpenCV's face detection, the deep learning model, and KNN for emotion analysis creates a potent and adaptable multi-modal stress detection system.

The integration of facial recognition, deep learning, and machine learning techniques provides a holistic approach to stress detection, capturing both physiological and emotional aspects. The proposed system holds promise for real-time stress monitoring in various settings, from workplaces to personal well-being applications. The multi-modal nature of the system enhances its versatility, making it a valuable tool for mental health initiatives and interventions.

CHAPTER 2

LITERATURE SURVEY

1.Title: Emotion Detection and Characterization using Facial Features

Authors: Charvi Jain,Kshitij Sawant,Mohammed Rehman,Rajesh Kumar,

Publication: 2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)

The human face has peculiar and specific characteristics, therefore it becomes difficult in understanding and identifying the facial expressions. It is easy to identify the facial expression of particular person in any image sequence. If we look to automated recognition system, however, the systems available are quite inadequate and incapable of accurately identify emotions. The area of facial expression identification has many important applications. It is an interactive tool between humans and computers. The user, without using the hand can go-ahead with the facial expressions. Presently, the research on facial expression are on the factors i.e. sad, happy, disgust, surprise, fear and angry. This paper aims to detect faces from any given image, extract facial features (eyes and lips) and classify them into 6 emotions (happy, fear, anger, disgust, neutral, sadness). The training data is passed through a series of filters and processes and is eventually characterized through a Support Vector Machine(SVM), refined using Grid Search. The testing data then tests the data and their labels and gives the accuracy of classification of the testing data in a classification report. Various approaches, including passing the training images through Gabor filter, or transforming images using Histogram of Oriented Gradients(HOG) and Discrete Wavelet Transform(DWT) for better classification of data are implemented. The best result achieved so far is by passing the training images through Histogram of Oriented Gradients(HOG), followed by characterization by SVM, which gives an average precision of 85%.",

2.Title: Facial Emotion Recognition using Deep Learning

Authors: Shubhanjay Pandey,Sonakshi Handoo,Yogesh,

Publication: 2022 International Mobile and Embedded Technology Conference (MECON)

As it is known, sentiments influence information processing, attitude formation, and decision making to a great extent in real-world scenarios. Several recent efforts have been published about FER or facial expression recognition, however, due to the diversity of human faces and fluctuations in pictures, reliable and robust FER systems remain a challenge. Till date, every study and work has proposed either a single network or an ensemble model. The accuracy of ensemble models is higher, but they were associated with many models and datasets and a few tweaked datasets to improve the accuracy, increasing the computing complexity. While the majority of research in this field focuses on improving accuracy, this study utilizes the proposed model to a real-world scenario in which a personu2019s face contains a mix of emotions, and a single-label sentiment can be highly noisy in such situations. In view of this scenario, we developed and tested 15-20 models and methods. In this paper, we propose a single standalone-based CNN model with its implementation on a real-time Intelligent System for Sentiment Recognition, which validates accuracy through transfer learning and performs tasks such as face detection, sentiment classification, and providing a live list of probabilistic labels in Realtime from a webcam feed in one blended step. The proposed model achieves an accuracy of 76.62 % outperforming all standalone-based models like VGG16, VGG19, EfficientNetB7, and other proposed models on the FER2013 dataset solely, which is a challenging and noisy crowd-sourced dataset, without using any other auxiliary dataset.",

3.Title: Facial Emotion Detection Using Deep Learning

Authors: Akriti Jaiswal,A. Krishnama Raju,Suman Deb,

Publication: 2020 International Conference for Emerging Technology (INCET)

Human Emotion detection from image is one of the most powerful and challenging research task in social communication. Deep learning (DL) based emotion detection gives performance better than traditional methods with image processing. This paper presents the design of an artificial intelligence (AI) system capable of emotion detection through facial expressions. It discusses about the procedure of emotion detection, which includes basically three main steps: face detection, features extraction, and emotion classification. This paper proposed a convolutional neural networks (CNN) based deep learning architecture for emotion detection from images. The performance of the proposed method is evaluated using two datasets Facial emotion recognition challenge (FERC-2013) and Japaness female facial emotion (JAFFE). The accuracies achieved with proposed model are 70.14 and 98.65 percentage for FERC-2013 and JAFFE datasets respectively.",

4.Title: Machine Learning Techniques for Real-Time Emotion Detection from Facial Expressions

Authors: Akshita Sharma,Vriddhi Bajaj,Jatin Arora,

Publication: 2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON)

Facial expressions recognition by emotion is a crucial component in many applications. This paper covers the recent trends in human emotion detection. An overview of various facial emotion recognition and its applications are presented. In the literature review, major machine-learning techniques used for facial emotion identification have been explored. Machine learning approaches are compared on the basis of their advantages, disadvantages, and their accuracy. Theoretical analysis of existing approaches shows that the algorithm providing the maximum accuracy should be used for facial emotion recognition. The existing approaches are also suffered from some challenges and those challenges should be addressed and considered for accurately predicting the users' emotional state. The application of emotion detection is also very vast and a few of the major applications are also discussed. Finally, a brief analysis of existing Machine learning approaches and their conclusion is given.",

5.Title: Facial Emotion Detection Using Convolutional Neural Networks

Authors: Vaishnavi Hosur,Ashwini Desai,

Publication: 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)

The most important aspect to understand human behavior is face reading. The expression speaks better than words. The expressions of face reflect human perspective and its mental state. The aim of this paper is to sight faces from image, extract facial expressions and classify them into different emotions like sad, happy, anger, and neutral. This paper discusses a technique named facial emotion recognition using convolutional neural networks (FERC). There are two parts in convolution neural networks (CNN) first removal of background from image and second facial feature extraction (EV). This application is used in medical treatment, teaching field, police investigation, human robot interface.",

6.Title: Human Emotion Recognition using Convolutional Neural Network in Real Time

Authors: Rohit Pathar,Abhishek Adivarekar,Arti Mishra,Anushree Deshmukh,

Publication: 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)

The human emotion recognition has attracted interest of many problem solvers in the field of artificial intelligence. The emotions on a human face say so much about our thought process and give a glimpse of what's going on inside the mind. Real time emotion recognition is to acquaint the machine with human like ability to recognize and analyse human emotions. This project aims to categorize a facial image into one of the seven emotions which we are considering in this study, by building a multi class classifier. In this paper we are using convolutional neural networks (CNNs) for training over gray scale images obtained from fer2013 dataset. We experimented with different depths and max pooling layers to get the best accuracy and ultimately achieving 89.98% accuracy. To combat overfitting, we have used technique like dropout. We are also analyzing the performance of different network architectures like shallow network and modern deep network in recognizing human emotion. We also present the real-time implementation of emotion recognition in web-camera which provides accurate results for multiple faces simultaneously. The results obtained from the research are quite interesting.",

7.Title: Emotion Detection through Facial Expression using DeepLearning

Authors: Manish Kumar,Swati Srivastava,

Publication: 2021 5th International Conference on Information Systems and Computer Networks (ISCON)

This paper aims to detect the emotion of a person through facial expression using artificial neural network. There is a wide range of applications where emotion detection has been used which witnesses the significance of emotion detection. There are 6 predefined facial expressions used in various researches. It includes happiness, surprise, sadness, disgust, fear and anger. Based on the above facial expressions, we determine the behavior and mood of the person. This proposal employs deep learning algorithm for emotion detection.",

8.Title: Emotion Recognition System from Speech and Visual Information based on Convolutional Neural Networks

Authors: Nicolae-Cu0103tu0103lin Ristea,Liviu Cristian Duu0163u,Anamaria Radoi,

Publication: 2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)

Emotion recognition has become an important field of research in the human-computer interactions domain. The latest advancements in the field show that combining visual with audio information lead to better results if compared to the case of using a single source of information separately. From a visual point of view, a human emotion can be recognized by analyzing the facial expression of the person. More precisely, the human emotion can be described through a combination of several Facial Action Units. In this paper, we propose a system that is able to recognize emotions with a high accuracy rate and in real time, based on deep Convolutional Neural Networks. In order to increase the accuracy of the recognition system, we analyze also the speech data and fuse the information coming from both sources, i.e., visual and audio. Experimental results show the effectiveness of the proposed scheme for emotion recognition and the importance of combining visual with audio data.",

9.Title: Analysis of Facial Emotion Recognition

Authors: Balaji Balasubramanian,Pranshu Diwan,Rajeshwar Nadar,Anuradha Bhatia,

Publication: 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)

Human beings rely a lot on non-verbal communication and facial emotion is a large part of it. In this review paper we cover the datasets and algorithms that are used for Facial Emotion Recognition (FER). The algorithms range from simple Support Vector Machines (SVM) to complex Convolutional Neural Network (CNN). We explain these algorithms through the fundamental research papers and go through their application to the task of FER.",

10.Title: Speech Emotion Recognition using Machine Learning

Authors: Kotikalapudi Vamsi Krishna,Navuluri Sainath,A. Mary Posonia,

Publication: 2022 6th International Conference on Computing Methodologies and Communication (ICCMC)

The aim of the paper is to detect the emotions which are elicited by the speaker while speaking. Emotion Detection has become a essential task these days. The speech which is in fear, anger, joy have higher and wider range in pitch whereas have low range in pitch. Detection of speech is useful in assisting human machine interactions. Here we are using different classification algorithms to recognize the emotions , Support Vector Machine , Multi layer perception, and the audio feature MFCC, MEL, chroma, Tonnetz were used. These models have been trained to recognize these emotions (Calm, neutral, surprise, happy, sad, angry, fearful, disgust). We got an accuracy of 86.5% and testing it with the input audio we get the same.",

11.Title: A Study on Human Emotion Detection

Authors: Shambhavi Bhardwaj,Ashutosh Dixit,Kshitij Yadav,Gokul Rajan V,

Publication: 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)

This paper centers around an arrangement of perceiving a human's feeling from an identified human's face. The dissected data is passed on by the locales of the eye and the mouth into a combined new picture in different outward appearances relating to six widespread essential facial feelings. The yield data got could be taken care of as a contribution to a machine fit for collaborating with social abilities, with regards to building socially wise frameworks. The approach utilizes an order method of data into another melded picture which is made out of two squares coordinated by the territory of the eyes and mouth, delicate regions to changes human's demeanor and that are especially pertinent for the disentangling of enthusiastic articulations. At long last we utilize the blended picture as a contribution to a feed-forward neural organization prepared by back-engendering. Such investigation of blended pictures causes it conceivable, to acquire important data through the mix of appropriate information in a similar picture and diminish the preparation set time while saving grouping rate. It is appeared by test results that the proposed calculation can distinguish feeling with great exactness.",

12.Title: Emotion Detection with Facial Feature Recognition Using CNN & OpenCV

Authors: Sarwesh Giri,Gurchetan Singh,Babul Kumar,Mehakpreet Singh,Deepanker Vashisht,Sonu Sharma,Prince Jain,

Publication: 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)

Emotion Detection through Facial feature recognition is an active domain of research in the field of human-computer interaction (HCI). Humans are able to share multiple emotions and feelings through their facial gestures and body language. In this project, in order to detect the live emotions from the human facial gesture, we will be using an algorithm that allows the computer to automatically detect the facial recognition of human emotions with the help of Convolution Neural Network (CNN) and OpenCV. Ultimately, Emotion Detection is an integration of obtained information from multiple patterns. If computers will be able to understand more of human emotions, then it will mutually reduce the gap between humans and computers. In this research paper, we will demonstrate an effective way to detect emotions like neutral, happy, sad, surprise, angry, fear, and disgust from the frontal facial expression of the human in front of the live webcam.",

13.Title: Emotion Detection using Deep Facial Features

Authors: Hari Kishan Kondaveeti,Mogili Vishal Goud,

Publication: 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI)

Emotion Detection became one of the most efficient and challenging activities in human interactions. In general, facial gestures are normal and clear means of expressing their feelings and intentions to human beings. The main features of non-verbal conversations are facial expressions. The study of the techniques of Face Emotion Recognition involves three key phases, like pre-processing, extracting of features and, classification techniques. This paper proposes the comparison of deep learning architectures available in Keras for emotion detection using the Deep Facial Features in images using Transfer Learning from famous pre-trained models like VGG-16, ResNet152V2, InceptionV3, and Xception and generating bottleneck features for our input images. The performance of these models is evaluated based on the dataset which is a combination of the Cohn-Kanade Dataset (CK+) and Japanese female facial emotion (JAFFE). For the above-mentioned architectures, the accuracies obtained are 83.16 %, 82.15 %, 77.1 %, 78.11 % respectively.",

14.Title: Facial Emotion Recognition Using Deep Convolutional Neural Network

Authors: E. Pranav,Suraj Kamal,C. Satheesh Chandran,M.H. Supriya,

Publication: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)

The rapid growth of artificial intelligence has contributed a lot to the technology world. As the traditional algorithms failed to meet the human needs in real time, Machine learning and deep learning algorithms have gained great success in different applications such as classification systems, recommendation systems, pattern recognition etc. Emotion plays a vital role in determining the thoughts, behaviour and feeling of a human. An emotion recognition system can be built by utilizing the benefits of deep learning and different applications such as feedback analysis, face unlocking etc. can be implemented with good accuracy. The main focus of this work is to create a Deep Convolutional Neural Network (DCNN) model that classifies 5 different human facial emotions. The model is trained, tested and validated using the manually collected image dataset.",

15.Title: Facial Emotion Detection using Machine Learning and Deep Learning Algorithms

Authors: Snehal Bhogan,Kedar Sawant,Nidhi Gondalekar,Rachel Carvalho,Vassant Kalangutkar,Alleena Mathew,

Publication: 2023 2nd International Conference on Edge Computing and Applications (ICECAA)

Emotion recognition plays a critical role in various domains, such as human-computer interaction, psychology, and market research. With the growing popularity of social media platforms and the increasing use of multimedia content, the ability to automatically recognize emotions from images has become a significant research area. In this research, through a pre-trained CNN model, rich and high-level features that effectively capture the emotional content present in the images are extracted. The dataset for the same is FER2013 which is downloaded through Kaggle and the data to test emotion will be taken live feed through a webcam. The extracted features are then fed into three classifiers namely: CNN, KNN, and random forest. Conducted experiments on a publicly available emotion recognition dataset named FER-2013. It illustrates the efficacy of change in accuracy by achieving a vast difference between the CNN, KNN, and Random Forest. Moreover, compared the performance with other state-of-the-art methods, demonstrating its superior performance in accurately recognizing emotions from images.",

CHAPTER 3

SYSTEM DESIGN

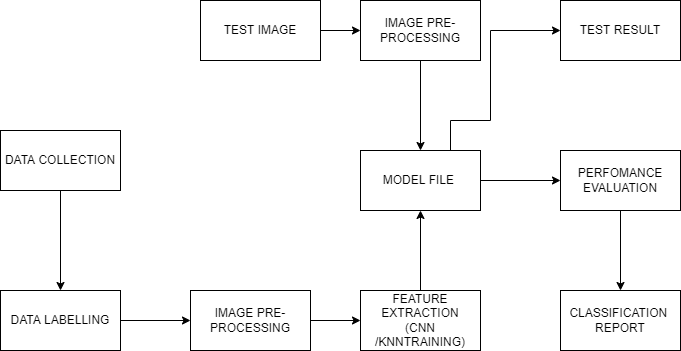
Existing System

Current stress detection methods predominantly rely on subjective self-reporting or physiological measurements, such as heart rate and cortisol levels. These approaches lack the precision and real-time capabilities required for effective stress monitoring. Moreover, they often necessitate intrusive measures, hindering their practicality in various settings. The absence of a non-intrusive, accurate, and real-time stress detection system limits the scope of proactive intervention and support for individuals experiencing stress. The need for a more advanced and versatile solution becomes evident as the prevalence of stress-related concerns continues to rise in contemporary society.

Proposed System

The proposed system represents a paradigm shift in stress detection by introducing a multi-modal approach that integrates facial recognition and emotion analysis. Leveraging computer vision, deep learning, and machine learning techniques, the system captures facial images through a web camera using OpenCV's Haar cascade algorithm for face detection. The isolated faces undergo stress estimation via a pre-trained Convolutional Neural Network (CNN), trained on a diverse dataset to recognize nuanced facial cues indicative of stress. Additionally, emotional features are extracted using a k-Nearest Neighbors (KNN) algorithm. This comprehensive approach not only enhances the accuracy of stress level assessments but also provides a real-time, non-intrusive, and adaptable solution suitable for various settings. The integration of OpenCV's face detection, CNN, and KNN creates a robust stress detection system, addressing the limitations of existing methods and contributing to proactive mental health monitoring and intervention initiatives.

BLOCK DIAGRAM



METHODOLOGY:

The proposed methodology for the stress detection system unfolds in a systematic two-step process. Firstly, facial images are captured in real-time using a web camera, and the Haar cascade algorithm from OpenCV is employed for efficient face detection, ensuring the isolation of facial regions. Subsequently, the isolated faces undergo stress estimation through a meticulously trained Convolutional Neural Network (CNN) model. This CNN is specifically designed to recognize subtle facial cues associated with stress, enhancing the precision of stress level assessments. In parallel, emotional features are extracted from the facial expressions using a k-Nearest Neighbors (KNN) algorithm, contributing an additional layer of understanding to the individual's mental state. The amalgamation of OpenCV's face detection, the CNN stress estimation model, and the KNN emotion analysis creates a powerful multi-modal system for stress detection. The methodology ensures the integration of both physiological and emotional aspects, providing a holistic approach to stress monitoring. This comprehensive methodology aims to deliver accurate, real-time, and non-intrusive stress assessments, making it a versatile tool for mental health initiatives and well-being applications.

CHAPTER 4

SOFTWARE DESCRIPTION

PYTHON:

PYTHON 3.7:

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in manya reason most platforms and

may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications. This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off- line as well. For a description of standard objects and modules, see library-index. Reference-index gives a more formal definition of the language. To write extensions in C or C++, read extending-index and c-api-index. There are also several books covering Python in depth. This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python’s most notes worthy features, and will give you a good idea of the language’s flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in library-index. If you do much work on computers, eventually you find that there’s some task you’d like

to automate. For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you’d like to write a small custom database, or a specialized

GUI application or a simple game. If you’re a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you’re writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you’ve written a program that could use an extension language, and you don’t want to design and implement a whole new language for your application.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit(). The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support read line. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. Ifnothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line. The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file. A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell’s -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote commands in its entirety with single quotes.Some Python modules are also useful as scripts. These can be invoked using python-m module [arg]...,which executes the source file for the module as if you had spelled out its full name on the command line. When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

There are tools which use doc strings to automatically produce online or printed documentation or to let the user interactively browse through code; it’s good practice to include doc strings in code that you write, so make a habit of it. The execution of a function introduces a new symbol table used for the local variables of the function. More precisely, all variable assignments in a functions to read the value in the local symbol table; whereas variable references first look in the local symbol table, then in the local symbol tables of enclosing functions, then in the global symbol table, and finally in the table of built-in names. Thus, global variables cannot be directly assigned a value within a function (unless named in a global statement), although they may be referenced. The actual parameters (arguments) to a function call are introduced in the local symbol table of the called function when it is called; thus, arguments are passed using call by value (where the value is always an object reference, not the value of the object).1 When a function calls another function, a new local symbol table is created for that call. A function definition introduces the function name in the current symbol table. The value of the function name has a type that is recognized by the interpreter as a user-defined function. This value can be assigned to another name which can then also be used as a function.

Annotations are stored in the annotations attribute of the function as a dictionary and haven o effect on any other part of the function. Parameter annotations are defined by a colon after the parameter name, followed by an expression evaluating to the value of the annotation. Return annotationsare defined by a literal ->, followed by an expression, between the parameter list and the colon denoting the end of the def statement.

The comparison operators in and not in check whether a value occurs (does not occur) in a sequence. The operator is and does not compare whether two objects are really the same object; this only matters for mutable objects like lists. All comparison operators have the same priority, which is lower than that of all numerical operators. Comparisons can be chained. For example,a<b==ctestswhetheraislessthanbandmoreoverbequalsc. Comparisons may be combined using the Boolean operators and the outcome of a comparison (or of any other Boolean expression) may be negated with not. These have lower priorities than comparison operators; between them, not has the highest priority and or the lowest, so that A and not B or C is equivalent to (A and (not B)) or C. As always, parentheses can be used to express the desired composition. The Boolean operators and are so-called short-circuit operators: their arguments are evaluated from left to right, and evaluation stops as soon as the outcome is determined. For example, if A and C are true but Bis false, A and B and C does not evaluate the expression C. When used as a general value and not as a Boolean, the return value of a short-circuit operator is the last evaluated argument.

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. Compared with other programming languages, Python’s class mechanism adds classes with a minimum of new syntax and semantics. It is a mixture of the class mechanisms found in C++ and Modula-3. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation. In C++ terminology, normally class members (including the data members) are public (except see below Private Variables), and all member functions are virtual. A sin Modula-3, there are no short hands for referencing the object’s members from its methods: the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call. A sin Small talk, classes themselves are objects. This providesSemantics for importing and renaming. Unlike C++ and Modula-3, built-in types can be used as base classes for extension by the user. Also, like in C++, most built-in operators with special syntax (arithmetic operators, sub scripting etc.) can be redefined for class instances.(Lacking universally accepted terminology to talk about classes, I will make occasional use of Smalltalk and C++ terms. I would use Modula-3 terms, since its object- oriented semantics are closer to those of Python than C++, but I expect that few readers have heard of it.)

Objects have individuality, and multiple names (in multiple scopes) can be bound to the same object. This is known as aliasing in other languages. This is usually not appreciated on a first glance at Python, and can be safely ignored when dealing with immutable basic types (numbers, strings, tuples).However, aliasing has a possibly surprising effect on these mantic of Python code involving mutable objects such as lists, dictionaries, and most other types. This is usually used to the benefit of the program, since aliases behave like pointers in some respects. For example, passing an object is cheap since only a pointer is passed by the implementation; and if a function modifies an object passed as an argument, the caller will see the change — this eliminates the need for two different argument passing mechanisms as in Pascal.

A namespace is a mapping from names to objects. Most name spaces are currently implemented as Python dictionaries, but that’s normally not noticeable in any way (except for performance), and it may change in the future. Examples of name spaces are: these to f built-in names (containing functions such as abs(), and built-in exception names); the global names in a module; and the local names in a function invocation. In a sense the set of attributes of an object also form a namespace. The important thing to know about namespaces is that there is absolutely no relation between names in different namespaces; for instance, two different modules may both define a function maximize without confusion — users of the modules must prefix it with the module name. By the way, I use the word attribute for any name following a dot — for example, in the expression z. real, real is an attribute of the object z. Strictly speaking, references to names in modules are attribute references: in the expression modname.funcname, modname is a module object and funcname is an attribute of it. In this case there happens to be a straight forward mapping between the module’s attributes and the global names defined in the module: they share the same namespace!1 Attributes may be read-only or writable. In the latter case, assignment to attributes is

possible. Module attributes are writable: you can

write modname.the\_answer = 42. Writable attributes may also be deleted with the del statement. For example, del mod name .the\_ answer will remove the attribute the\_answer from the object named by mod name. Namespaces are created at different moments and have different lifetimes. The namespace containing the built-in names is created when the Python interpreter starts up, and is never deleted. The global namespace for a module is created when the module definition is read in; normally, module namespaces also last until the interpreter quits.The statements executed by the top-level invocation of the interpreter, either read from a script file or interactively, are considered part of a module called main, so they have their own global namespace.(The built-in names actually also live in a module; this is called built ins.) The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function. (Actually, forgetting would be a better way to describe what actually happens.) Of course, recursive invocations each have their own local namespace.

To speed uploading modules, Python caches the compiled version

of each module in the pycache directory under the name

module.version.pyc, where the version encodes the format of the compiled

file; it generally contains the Python version number. For example, in CPython release 3.3 the compiled version of spam.py would be cached as

pycache/spam.cpython-33.pyc. This naming

convention allows compiled modules from different releases and different versions of Python to coexist. Python checks the modification date of the source against the compiled version to see if it’s out of date and needs to be recompiled. This is a completely automatic process. Also, the compiled modules are platform-independent, so the same library can be shared among systems with different architectures. Python does not check the cache in two circumstances. First, it always recompiles and does not store the result for the module that’s loaded directly from the command line. Second, it does not check the cache if there is no source module. To support anon-source (compiled only) distribution, the compiled module must be in the source directory, and there must not be a source module. Some tips for experts:

You can use the -O or -OO switches on the Python command to reduce the size of a compiled module. The -O switch removes assert statements, the -OO switch removes both assert statements and doc

strings. Since some programs may rely on having these available, you should only use this option if you know what you’re doing. “Optimized”

modules have an opt- tag and are usually smaller. Future releases may change the effects of optimization.

A program doesn’t run any faster when it is read from a .pyc file than when it is read from a .py file; the only thing that’s faster about .pyc files is the speed with which they are loaded.

The module compile all can create .pyc files for all modules in a directory.

There is more detail on this process, including a flow chart of the decisions

THONNY IDE:

Thonny is as mall and light weight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME, so Thonny only requires the GTK2 toolkit and therefore you only need the GTK2 runtime libraries installd to run it.

For compiling Thonny yourself, you will need the GTK (>= 2.6.0) libraries and header files. You will also need the Pango, Gliband ATK libraries and header files. All these files are available at [http://www.gtk.org.](http://www.gtk.org/) Furthermore you need, of course, a C compiler and the Make tool; a C++ compiler is also required for the included Scintilla library. The GNU versions of these tools are recommended.

Compiling Thonny is quite easy. The following should do it:

% ./configure

% make

% make install

The configure script supports several common options, for a detailed list, type

% ./configure --help

There are also some compile time options which can be found in src/Thonny .h. Please see Appendix C for more information. In the case that your system lacks dynamic linking loader support, you probably want to pass the option --disable-vte to the configure script. This prevents

compiling Thonny with dynamic linking loader support to automatically load libvte.so.4 if available. Thonny has been successfully compiled and tested under Debian 3.1 Sarge, Debian 4.0 Etch, Fedora Core 3/4/5, Linux From Scratch and FreeBSD 6.0. It also compiles under Microsoft Windows

At startup, Thonny loads all files from the last time Thonny was launched. You can disable this feature in the preferences dialog (see Figure 3-4). If you specify some files on the command line, only these files will be opened, but you can find the files from the last session in the file menu under the "Recent files" item. By default this contains the last 10 recently opened files. You can change the amount of recently opened files in the preferences dialog. You can start several instances of Thonny , but only the first will load files from the last session. To run a second instance of Thonny , do not specify any file names on the command-line, or disable opening files in a running instance using the appropriate command line option.

Thonny detects an already running instance of itself and opens files from the command-line in the already running instance. So, Thonny can be used to view and edit files by opening them from other programs such as a file

manager. If you do not like this for some reason, you can disable using the first instance by using the appropriate command line option

If you have installed libvte.so in your system, it is loaded automatically by Thonny , and you will have a terminal widget in the notebook at the bottom. If Thonny cannot find libvte.so at startup, the terminal widget will not be loaded. So there is no need to install the package containing this file in order to run Thonny . Additionally, you can disable the use of the terminal widget by command line option, for more information see Section3.2.You can use this terminal (from now on called VTE) nearly as an usual terminal program like xterm. There is basic clipboard support. You can paste the contents of the clipboard by pressing the right mouse button to open the popup menu and choosing Paste. To copy text from the VTE, just select the desired text and then press the right mouse button and choose Copy from the pop up menu. On systems running the X Window System you can paste the last selected text by pressing the middle mouse button in the VTE (on 2-button mice, the middle button can often be simulated by pressing both mouse buttons together).

As long as a project is open, the Make and Run commands will use the project’s settings, instead of the defaults. These will be used whichever document is currently displayed. The current project’s settings

are saved when it is closed, or when Thonny is shut down. When restarting Thonny , the previously opened project file that was in use at the end of the last session will be reopened.

Execute will run the corresponding executable file, shell script or interpreted script in a terminal window. Note that the Terminal tool path must be correctly set in the Tools tab of the Preferences dialog - you can use any terminal program that runs a Bourne compatible shell and accept the "-e" command line argument to start a command. After your program or script has finished executing, you will be prompted to press the return key. This allows you to review any text output from the program before the terminal window is closed.

By default the Compile and Build commands invoke the compiler and linker with only the basic arguments needed by all programs. Using Set Includes and Arguments you can add any include paths and compile flags for the compiler, any library names and paths for the linker, and any arguments you want to use when running Execute.

Thonny has basic printing support. This means you can print a file by passing the filename of the current file to a command which actually prints the file.

However, the printed document contains no syntax highlighting.

CHAPTER 5  
WORKING

PREPROCESSING:

Images come in different shapes and sizes. They also come through different sources.

Taking all these variations into consideration, we need to perform some pre-processing on any image data. RGB is the most popular encoding format, and most “natural images” . Also, among the first step of data pre-processing is to make the images of the same size.

Here we have used auto resizing for training to make all the images in the dataset to convert in to same resolution.

FEATURE EXTRACTION:

The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine’s efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

CNN:

In deep learning, a convolutional neural network (CNN) is a type of deep neural networks, which deals with the set of data to extract information about that data. Like images, sounds or videos etc. can be used in the CNN for the data extraction. There are mainly three things in CNN. First one is local receptive field and then shared weight and biases and the last one is activation and pooling. In CNN, first the neural networks are trained using a heavy set of data so that the CNN can extract the feature of given input. When the input is given, first image preprocessing is done then the feature extraction occurs on the basis of set of data stored and then the classification of data is done and output is shown as the result.

The CNN can deal with those input only for what the neural network is trained and the data is saved.

They are used in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing

DATASET:

One major advantage of using CNNs over NNs is that you do not need to flatten the input images to 1D as they are capable of working with image data in 2D. This helps in retaining the “spatial” properties of images.

PRE-PROCESSING STEPS

The pre-processing steps were conducted with resizing, patch, and augmentation steps. The first pre-processing step normalizes the size of the input images. Almost all the radiographs were rectangles of different heights and too large (median value of matrix size ≥1,800). Accordingly, we resized all images to a standardized 224×224 pixel square, through a combination of preserving their aspect ratios and using zero-padding. The investigation of deep learning efficiency depends on the input data; therefore, in the second processing step, input images were pre-processed by using a patch (a cropped part of each image). Finally, data augmentation was conducted for just the training dataset, using mirror images that were reversed left to right and rotated −30, −10, 10, and 30 degrees.

IMAGE LABELING AND DATASET DISTRIBUTIONS:

All subjects were independently labeled twice. Labeling was first evaluated with the original images on a picture archiving communication system (PACS) and secondly with the resized images that were used for the actual learning data. Datasets were defined as the internal dataset and temporal dataset, with the temporal dataset used to evaluate the test. The internal dataset was randomly split into training (70%), validation (15%), and test (15%) subsets.

Activation function

Activation function serves as a decision function and helps in learning of intricate patterns. The selection of an appropriate activation function can accelerate the learning process. In literature, different activation functions such as sigmoid, tanh, maxout, SWISH, ReLU, and variants of ReLU, such as leaky ReLU, ELU, and PReLU are used to inculcate non-linear combination of features

Operations using NumPy:

Using NumPy, a developer can perform the following operations:

• Mathematical and logical operations on arrays.

• Fourier transforms and routines for shape manipulation.

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

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index. Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Data Type Objects(dtype)

A data type object describes interpretation of fixed block of memory corresponding to an array, depending on the following aspects:

• Type of data (integer, float or Python object)

• Size of data

• Byte order (little-endian or big-endian)

• In case of structured type, the names of fields, data type of each field and part of the memory block taken by each field

• If data type is a subarray, its shape and data type The byte order is decided by prefixing '<' or '>' to data type. '<' means that encoding is littleendian (least significant is stored in smallest address)

TENSORFLOW:

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery. This paper describes the TensorFlow interface and an implementation of that interface that we have built at Google. The TensorFlow API and a reference implementation were released as an open-source package under the Apache 2.0 license in November, 2015 and are available at [www.tensorflow.org](http://www.tensorflow.org).

Based on our experience with Disbelief and a more complete understanding of the desirable system properties and requirements for training and using neural networks, we have built TensorFlow, our second-generation system for the implementation and deployment of largescale machine learning models. TensorFlow takes computations described using a dataflow-like model and maps them onto a wide variety of different hardware platforms, ranging from running inference on mobile device platforms such as Android and iOS to modest sized training and inference systems using single machines containing one or many GPU cards to large-scale training systems running on hundreds of specialized machines with thousands of GPUs. Having a single system that can span such a broad range of platforms significantly simplifies the real-world use of machine learning system, as we have found that having separate systems for large-scale training and small-scale deployment leads to significant maintenance burdens and leaky abstractions. TensorFlow computations are expressed as stateful dataflow graphs (described in more detail in Section 2), and we have focused on making the system both flexible enough for quickly experimenting with new models for research purposes and sufficiently high performance and robust for production training and deployment of machine learning models. For scaling neural network training to larger deployments, TensorFlow allows clients to easily express various kinds of parallelism through replication and parallel execution of a core model dataflow graph, with many different computational devices all collaborating to update a set of shared parameters or other state. Modest changes in the description of the computation allow a wide variety of different approaches to parallelism to be achieved and tried with low effort [14, 29, 42]. Some TensorFlow uses allow some flexibility in terms of the consistency of parameter updates, and we can easily express and take advantage of these relaxed synchronization requirements in some of our larger deployments. Compared to Disbelief, TensorFlow’s programming model is more flexible, its performance is significantly better, and it supports training and using a broader range of models on a wider variety of heterogeneous hardware platforms.

In a TensorFlow graph, each node has zero or more inputs and zero or more outputs, and represents the instantiation of an operation. Values that flow along normal edges in the graph (from outputs to inputs) are tensors, arbitrary dimensionality arrays where the underlying element type is specified or inferred at graph-construction time. Special edges, called control dependencies, can also exist in the graph: no data flows along such edges, but they indicate that the source node for the control dependence must finish executing before the destination node for the control dependence starts executing. Since our model includes mutable state, control dependencies can be used directly by clients to enforce happens before relationships. Our implementation also sometimes inserts control dependencies to enforce orderings between otherwise independent operations as a way of, for example, controlling the peak memory usage.

TENSORFLOW IMPLEMENTATION:

The main components in a TensorFlow system are the client, which uses the Session interface to communicate with the master, and one or more worker processes, with each worker process responsible for arbitrating access to one or more computational devices (such as CPU cores or GPU cards) and for executing graph nodes on those devices as instructed by the master. We have both local and distributed implementations of the TensorFlow interface. The local implementation is used when the client, the master, and the worker all run on a single machine in the context of a single operating system process (possibly with multiple devices, if for example, the machine has many GPU cards installed). The distributed implementation shares most of the code with the local implementation, but extends it with support for an environment where the client, the master, and the workers can all be in different processes on different machines. In our distributed environment, these different tasks are containers in jobs managed by a cluster scheduling system . These two different modes are illustrated . Most of the rest of this section discusses issues that are common to both implementations, while This discusses some issues that are particular to the distributed implementation.

Data Parallel Training

One simple technique for speeding up SGD is to parallelize the computation of the gradient for a mini-batch across mini-batch elements. For example, if we are using a mini-batch size of 1000 elements, we can use 10 replicas of the model to each compute the gradient for 100 elements, and then combine the gradients and apply updates to the parameters synchronously, in order to behave exactly as if we were running the sequential SGD algorithm with a batch size of 1000 elements. In this case, the TensorFlow graph simply has many replicas of the portion of the graph that does the bulk of the model computation, and a single client thread drives the entire training loop for this large graph. The TensorFlow system shares some design characteristics with its predecessor system, Disbelief , and with later systems with similar designs like Project Adam and the Parameter Server project . Like Disbelief and Project Adam, TensorFlow allows computations to be spread out across many computational devices across many machines, and allows users to specify machine learning models using relatively high-level descriptions. Unlike DistBelief and Project Adam, though, the general-purpose dataflow graph model in TensorFlow is more flexible and more amenable to expressing a wider variety of machine learning models and optimization algorithms.

LIMITATIONS:

The VGG-16 and VGG-19 architecture consist of large kernel-size filters with multiple 3×3 kernel-size filters, one after another. Within a given receptive field (the effective area size of an input image on which output depends), multiple stacked smaller sized kernels are better than a single larger sized kernel because multiple non-linear layers increase the depth of the network, enabling it to learn more complex features at a lower cost. As a result, the 3×3 kernels in the VGG architecture help to retain more fine details of an image . The ResNet architecture is similar to the VGG model, consisting mostly of 3×3 filters. Additionally, the ResNet model has a network depth of as large as 152. Therefore, it achieves better accuracy than VGG and GoogleNet, while being computationally more efficient than VGG. While the VGG and ResNet models achieve phenomenal accuracy, their deployment on even the most modest sized GPUs is a problem because of the massive computational requirements, both in terms of memory and time. There are several limitations to this study. First, the external test dataset in multiple medical centers did not be included for reproducibility. In the case of X-ray equipment, there is a relatively small difference in performance compared to other medical imaging equipment, depending on the manufacturer or model. In this study, therefore, the external test dataset in other medical centers did not be included. In the case of a local medical center using relatively old equipment; however, an additional performance evaluation is also required to utilize artificial intelligence (AI) assistive software. Second, the proposed majority decision algorithm was optimized to evaluate only maxillary sinusitis. Therefore, there is a limitation to evaluate sinusitis in frontal, ethmoid, and sphenoid. In order to utilize AI based assistive software in the future, further study is underway because it is necessary to evaluate sinusitis at other locations as well as maxillary. Third, it lacks pattern recognition and representation methods that can solve black-box in deep learning. It needs to determine a reasonable consensus for solving the black-box problem. The feature recognition based activation map was used to solve the black-box problem in deep learning. As it can be shown from the results, not only classification but also lesion localization can be expressed as a result. It helps medical doctors make a reasonable inference about the deep learning analysis. However, it is not enough to understand all deep leaning procedures. For example, it is difficult to understand the pattern of each learned CNN model. By understanding the pattern recognition capabilities of each model, we can understand the advantages and disadvantages of each model and achieve the optimization of the overall AI system. To overcome this limitation, a feature connectivity representation should be available for each layer to determine which feature weights are strong. In addition to feature representation, text-based description algorithm can be applied to overcome the black-box limitation in a medical application using the convolutional recurrent neural network (CRNN) that is the combination CNN and recurrent neural network (RNN) (20,21). A majority decision algorithm with multiple CNN models was shown to have high accuracy and significantly more accurate lesion detection ability compared to individual CNN models. The proposed deep learning method using PNS X-ray images can be used as an adjunct tool to help improve the diagnostic accuracy of maxillary sinusitis.

KNN ALGORITHM:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.

Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following −

3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − Next, it will choose the top K rows from the sorted array.

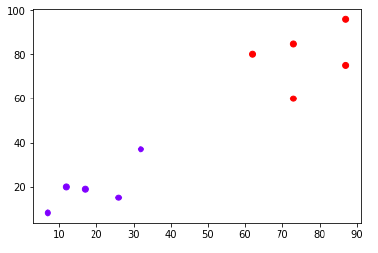
3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 − End

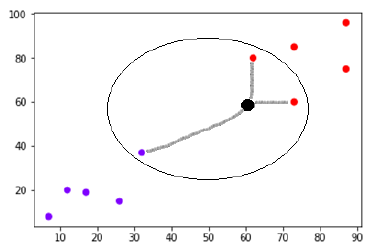
Example

The following is an example to understand the concept of K and working of KNN algorithm −

Suppose we have a dataset which can be plotted as follows −



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

CHAPTER 6

RESULTS

CHAPTER 7

CONCLUSION

In conclusion, this project introduces a pioneering multi-modal approach to stress detection, leveraging the capabilities of facial recognition, deep learning, and machine learning. The integration of OpenCV's face detection, a meticulously trained Convolutional Neural Network (CNN) for stress estimation, and a k-Nearest Neighbors (KNN) algorithm for emotion analysis creates a versatile and accurate stress detection system. The methodology captures both physiological and emotional cues, providing a holistic understanding of an individual's stress levels. The real-time, non-intrusive nature of the system makes it adaptable to various settings, from workplace environments to personal well-being applications. By addressing the limitations of existing stress detection methods, this project contributes to proactive mental health monitoring and intervention initiatives. The innovative combination of computer vision and machine learning techniques showcased in this project holds promise for advancing the field of stress detection, offering a valuable tool for promoting individual well-being and mental health awareness in contemporary society.

REFERENCES

1.Charvi Jain,Kshitij Sawant,Mohammed Rehman,Rajesh Kumar,"Emotion Detection and Characterization using Facial Features",2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)

2.Shubhanjay Pandey,Sonakshi Handoo,Yogesh,"Facial Emotion Recognition using Deep Learning",2022 International Mobile and Embedded Technology Conference (MECON)

3.Akriti Jaiswal,A. Krishnama Raju,Suman Deb,"Facial Emotion Detection Using Deep Learning",2020 International Conference for Emerging Technology (INCET)

4.Akshita Sharma,Vriddhi Bajaj,Jatin Arora,"Machine Learning Techniques for Real-Time Emotion Detection from Facial Expressions",2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON)

5.Vaishnavi Hosur,Ashwini Desai,"Facial Emotion Detection Using Convolutional Neural Networks",2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)

6.Rohit Pathar,Abhishek Adivarekar,Arti Mishra,Anushree Deshmukh,"Human Emotion Recognition using Convolutional Neural Network in Real Time",2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)

7.Manish Kumar,Swati Srivastava,"Emotion Detection through Facial Expression using DeepLearning",2021 5th International Conference on Information Systems and Computer Networks (ISCON)

8.Nicolae-Cu0103tu0103lin Ristea,Liviu Cristian Duu0163u,Anamaria Radoi,"Emotion Recognition System from Speech and Visual Information based on Convolutional Neural Networks",2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)

9.Balaji Balasubramanian,Pranshu Diwan,Rajeshwar Nadar,Anuradha Bhatia,"Analysis of Facial Emotion Recognition",2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)

10.Kotikalapudi Vamsi Krishna,Navuluri Sainath,A. Mary Posonia,"Speech Emotion Recognition using Machine Learning",2022 6th International Conference on Computing Methodologies and Communication (ICCMC)

11.Shambhavi Bhardwaj,Ashutosh Dixit,Kshitij Yadav,Gokul Rajan V,"A Study on Human Emotion Detection",2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)

12.Sarwesh Giri,Gurchetan Singh,Babul Kumar,Mehakpreet Singh,Deepanker Vashisht,Sonu Sharma,Prince Jain,"Emotion Detection with Facial Feature Recognition Using CNN & OpenCV",2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)

13.Hari Kishan Kondaveeti,Mogili Vishal Goud,"Emotion Detection using Deep Facial Features",2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI)

14.E. Pranav,Suraj Kamal,C. Satheesh Chandran,M.H. Supriya,"Facial Emotion Recognition Using Deep Convolutional Neural Network",2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)

15.Snehal Bhogan,Kedar Sawant,Nidhi Gondalekar,Rachel Carvalho,Vassant Kalangutkar,Alleena Mathew,"Facial Emotion Detection using Machine Learning and Deep Learning Algorithms",2023 2nd International Conference on Edge Computing and Applications (ICECAA)