

**HUMAN ACTIVITY RECOGNITION USING OPENCV AND
DEEP LEARNING**

**A project report submitted in partial fulfillment of the requirements for
the award of the degree of**

**BACHELOR OF TECHNOLOGY
IN
DATA SCIENCE**

Submitted by

Asha	20KP1A4452
Narasa Kumari	20KP1A4408
Khamarunnisa	20KP1A4434
Chandra Sekhar	20KP1A4441



Under the Esteemed Guidance of

Mr.B.Srinivas, MCA, M.Tech

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE-DATA SCIENCE

NRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, Affiliated to the JNTUK, Kakinada, A.P)

Visidala (P), Medikonduru (M), Guntur-522438

May 2024



DEPARTMENT OF COMPUTER SCIENCE-DATA SCIENCE

NRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, Affiliated to the JNTUK, Kakinada, A.P)

Visidala (P), Medikonduru (M), Guntur-522438

CERTIFICATE

This is certify that the project report entitled **“HUMAN ACTIVITY RECOGNITION USING OPENCV AND DEEP LEARNING”** is the

bonafide record submitted by **Asha (20KP1A4452), Narasa Kumari (20KP1A4408),Khamarunnisa(20KP1A4434),Chandrasekhar (20KP1A4441)** in partial fulfillment of the requirements for the Award of the Degree of **Bachelor of Technology**, in Computer Science and Engineering from the NRI INSTITUTE OF TECHNOLOGY.

Guide

Mr. B. Srinivas MCA, MTech

Assistant Professor

Head of the Department (HoD)

Mr.D.Koteswarao,M.Tech

Associate Professor

Department of Data Science

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I express my heartfelt thanks to **Prof. Kota. Srinivasu**, Principal, for his kind permission to carry out this project.

I wish to record my thanks to our **Mr.D.Koteswara Rao, H.O.D**, Department of Data Science, for his constant support, enthusiasm and motivation.

I wish to record my thanks to our Project Guide **Mr..B.Srinivas**, Assistant Professor, Department of Data Science, for his constant support, enthusiasm and motivation.

I wish to express thanks to all the staff members in the Department of **Data Science**, **NRI INSTITUTE OF TECHNOLOGY** for their valuable support throughout this project. I also thank all my friends for this moral support and suggestion for this work.

Finally, I thank one and all those who have rendered help directly or indirectly at various stages of the project.

Team Members

Asha	20KP1A4452
Narasakumari	20KP1A4408
Khamarunnisa	20KP1A4434
Chandra Sekhar	20KP1A4441

DECLARATION

We hereby declare that the work which is being presented in the Dissertation Entitled **“HUMAN ACTIVITY RECOGNITION USING OPENCV AND DEEP LEARNING”** submitted towards the partial fulfillment of requirements for the award of the degree in Bachelor of Technology and authentic record in Department of Data Science at NRI institute of Technology, Guntur.

The matter embodied in this dissertation report has not been submitted by us for the award of any degree. Further the technical details furnished in the various chapters in this report are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

PROJECT MEMBERS

ASHA	20KP1A4452
Narasa Kumari	20KP1A4408
Khamarunnisa	20KP1A4434
Chandra Sekhar	20KP1A4441



NRI INSTITUTE OF TECHNOLOGY

**(Approved by AICTE, Affiliated to the JNTUK, Kakinada, A.P) Visidala
(P), Medikonduru (M), Guntur-522438**

INSTITUTE VISION:

To become reputed institution of Engineering & Management programs, producing competitive, ethical & socially responsible professionals.

INSTITUTE MISSION:

IM 1: Provide quality education through best teaching & learning practices of committed staff.

IM 2: Establish a continuous interaction, participation & collaboration with industry to provide solutions.

IM 3: Provide the facilities that motivate/encourage faculty & students in research & development activities.

IM 4: Develop human values, professional ethics, & interpersonal skills amongst the individuals.

(ALAPATI RAJENDRA PRASAD)

SECRETARY & CORRESPONDENT

(DR. KOTA SRINIVASU)

PRINCIPAL



NRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF DATA SCIENCE

DEPARTMENT VISION:

To become a centre of excellence by bringing out the professional competence in the core areas of Data Science.

DEPARTMENT MISSION:

DM 1: To provide conducive environment that impart Data Science knowledge through quality teaching & self- learning.

DM 2: To serve the needs of Data Science , analytical communication & allied industries through industry interaction.

DM 3: To encourage innovative thinking, continuous learning among the stakeholders & create new techniques in Data Science.

DM 4: To groom students in communication & interpersonal skills.

DM 5: To inculcate human values & ethics to make learners sensitive towards social issues.



NRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF DATA SCIENCE

OUR VISION:

To become reputed institution of Engineering & Management programs with competitive, ethical & social responsibility

OUR MISSION:

IM1: Provide quality education through best teaching and learning practices of committed staff.

IM2: Establish a continuous interaction, participation and collaboration with industry to provide solutions

IM3: Provide the facilities that motivate/encourage faculty and students in research and development activities.

IM4: Develop human values, professional ethics and interpersonal skills amongst the individuals.



NRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF DATA SCIENCE

PROGRAMME EDUCATIONAL OBJECTIVES (PEO'S)

PEO1: Graduates will have solid basics in Mathematics, Programming, Machine Learning, Artificial Intelligence and Data Science fundamentals and advancements to solve technical problems.

PEO2: Graduates will have the capability to apply their knowledge and skills acquired to solve the issues in real world Data Science sector and to develop feasible and viable systems.

PEO3: Graduates will have the potential to participate in life-long learning through the successful completion of advanced degrees, continuing education, certifications and/or other professional developments.

PEO4: Graduates will have the ability to apply the gained knowledge to improve the society ensuring ethical and moral values.

PROGRAMME SPECIFIC OUTCOMES (PSO'S)

PSO1: Data Science graduates are able to become leaders in the Industry and Academia with the help of advanced knowledge and skill, which can empower them to analyse, design, develop and implement their learning to develop the society.

PSO2: The ability to develop skills to address and solve social and environmental problem with ethics and perform multidisciplinary projects with advance technologies and tools.

Signature of HOD



NRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF DATA SCIENCE

PROGRAM OUTCOMES (PO'S)

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

INDEX

CONTENTS	Page No.
CHAPTER 1: INTRODUCTION	
1.1 OBJECTIVE OF THE PROJECT	2
1.2 EXISTING MODEL	2
1.3 PROPOSED MODEL	2
CHAPTER 2: SYSTEM ANALYSIS AND DESIGN	
2.1 SOFTWARE REQUIREMENT SPECIFICATION	4
2.2 SOFTWARE DESIGN	5
CHAPTER 3: SOFTWARE ENVIRONMENT	
A. WHAT IS PYTHON	15
B. WHAT IS DEEP LEARNING	21
C. NEED FOR DEEP LEARNING	23
D. PYTHON PACKAGES	30
E. HOW TO INSTALL PYTHON	34
CHAPTER 4: CODE	43
CHAPTER 5: SYSTEM TESTING	48
CHAPTER 6: SCREENS	53
CHAPTER 7: CONCLUSION	56
CHAPTER 8: FUTURE SCOPE	57
CHAPTER 9: REFERENCES	58

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE OF THE PROJECT

The project aims to develop a system using OpenCV and deep learning for accurate human activity recognition in real-time. Objectives include enhancing accuracy, achieving real-time performance, ensuring adaptability across domains, providing a user-friendly interface, and contributing to research in activity recognition technology. By leveraging advanced techniques, the project seeks to create a versatile tool applicable in healthcare, security, sports analytics, and human-computer interaction, ultimately advancing our understanding of human behavior in diverse environments

1.2 THE EXISTING SYSTEM

The existing system for Human Activity Recognition (HAR) typically involves traditional machine learning algorithms or simpler deep learning architectures. These systems often rely on handcrafted features extracted from sensor data (such as accelerometer and gyroscope data from wearable devices) and use classifiers like Support Vector Machines (SVM), Random Forests, or k-Nearest Neighbors (k-NN) to recognize human activities.

Disadvantages: Limitation- Handcrafted features may not capture the full complexity of human activities.

1.3 THE PROPOSED SYSTEM

The proposed HAR system combines OpenCV and Deep Learning to automatically learn features from raw sensor data or video frames, overcoming limitations of traditional methods. It includes data pre-processing, feature extraction, model training, real-time inference, and a user-friendly interface. This approach offers higher accuracy and versatility for HAR in real-world applications.

CHAPTER 2

2. SYSTEM ANALYSIS AND DESIGN

System analysis is the performance management and documentation of activities related to the life cycle phases of any software namely

- The Study Phase
- The Design Phase
- The Development Phase
- The Implementation Phase
- The Testing Phase

Software Analysis starts with a preliminary analysis and later switches on to a detailed one. During the preliminary analysis the Analyst takes a quick look at what is needed and whether the cost benefits. Detailed analysis studies in depth all the cornered factors, which builds and strengthens the software.

2.1 Software Requirement Specification

Software Requirement Specification (SRS) is a document that completely describes what the proposed should do, without describing how the software does it.

2.1.1 Performance Requirements

1. The operation time should be small and the throughput should be high..
2. It should produce timely and accurate result.

2.1.2 Software Quality Attributes

1. **Maintainability** – Since it is directly associated with the database, so there is very little maintainability problem with this application.
2. **Easy to Learn** – Since there are less number of forms, this application is very easy to learn with user-friendly screens.
3. **Flexibility** – This application is very much flexible for future enhancements.

2.1.3 Hardware Requirements

1. System : Pentium IV 2.4 GHz.
2. Hard Disk : 40 GB.
3. Ram : 512 Mb.

2.1.4 Software Requirements

1. Operating system : Windows.
2. Coding Language : Python

2.2 Software Design

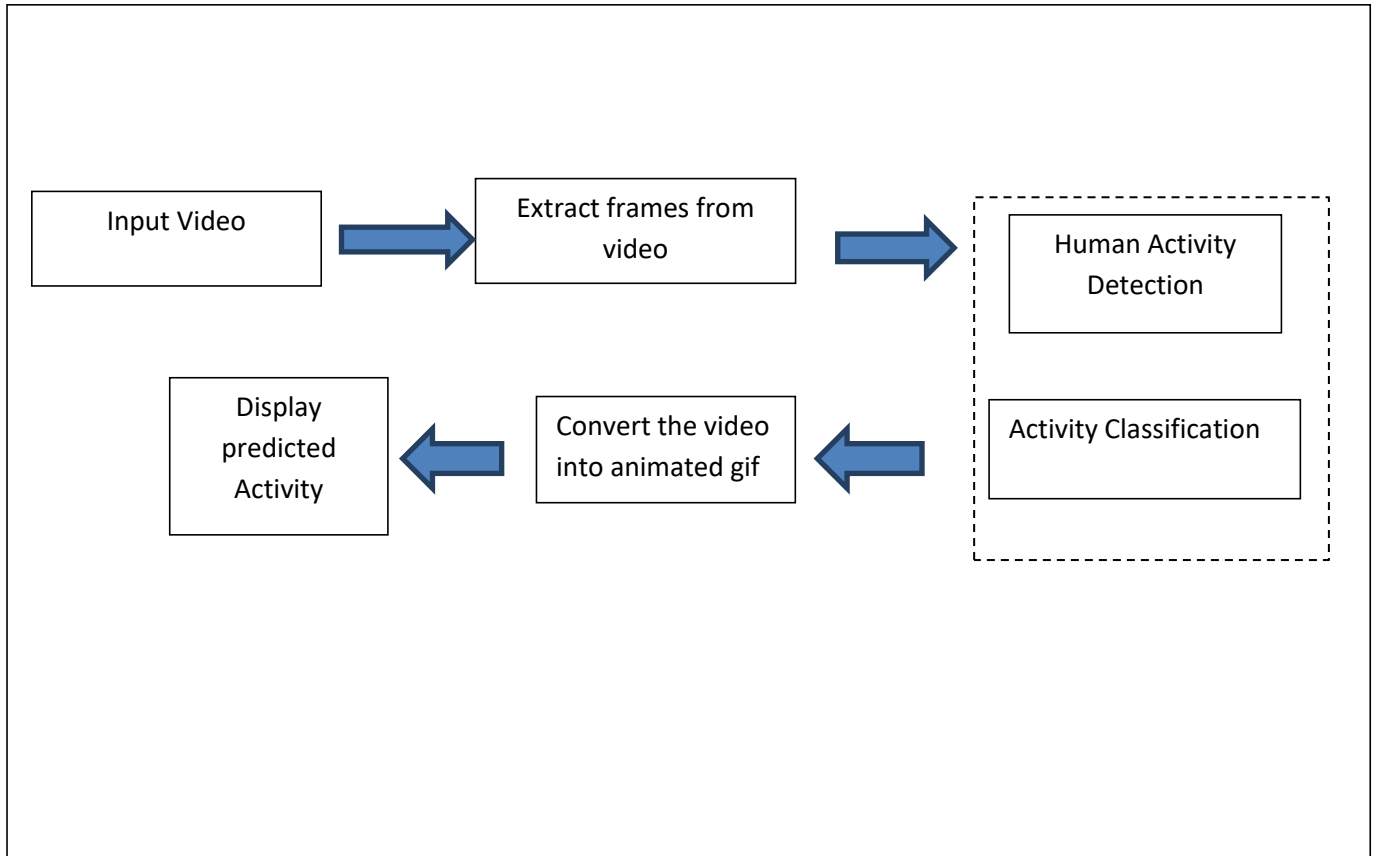
System design is the second step in the system life cycle, in which overall design of the system is achieved. The functionalities of the system is designed and studied in this phase. The first step is designing of program specification. This determines the various data inputs to the system, data flow and the format in which output is to be obtained.

Design phase is a transmission phase because it is a transition from user oriented document to computer data. The activity in the design phase is the allocation of functions to manual operations, equipment and computer programs. Flow charts are prepared in the study time and is decomposed until all functions in the system perform evidently.

Design is a multi-step process that focuses on data structures, software architecture, procedural details(algorithms etc) and links between the modules. The design process goes through logical and physical stages. In logical design reviews are made linking existing system and specification gathered. The physical plan specifies any hardware and software requirement, which satisfies the local design. Modularization of task is made in this phase. The success of any integrated system depends on the planning of each and every fundamental module. Usually a project is revised in step by step sequence. Inter-phase management of such module is also important. Software design methodology changes continually as new methods, better analysis and broader understanding evolve. Various techniques for software design do exist with the availability of criteria for design quality. Software design leads three technical activities-design, code and test.

SYSTEM ARCHITECTURE

System Architecture:-



UML DIAGRAMS

UML Diagrams

UML stands for **Unified Modeling Language**. This object-oriented system of notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation. These renowned computer scientists fused their respective technologies into a single, standardized model. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling object oriented programs.

There are three classifications of UML diagrams:

Behavior diagrams. A type of diagram that depicts behavioral features of a system or business process. This includes activity, state machine, and use case diagrams as well as the four interaction diagrams.

Interaction diagrams. A subset of behavior diagrams which emphasize object interactions. This includes communication, interaction overview, sequence, and timing diagrams.

Structure diagrams. A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object, and package diagrams.

Types of UML Diagrams

UML defines nine types of diagrams: class (package), object, use case, sequence, collaboration, statechart, activity, component, and deployment.

Class Diagrams

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system.

Package Diagrams

Package diagrams are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.

Object Diagrams

Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.

Use Case Diagrams

Use case diagrams model the functionality of system using actors and use cases.

Sequence Diagrams

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Collaboration Diagrams

Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.

Statechart Diagrams

Statechart diagrams describe the dynamic behavior of a system in response to external stimuli. Statechart diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.

Activity Diagrams

Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

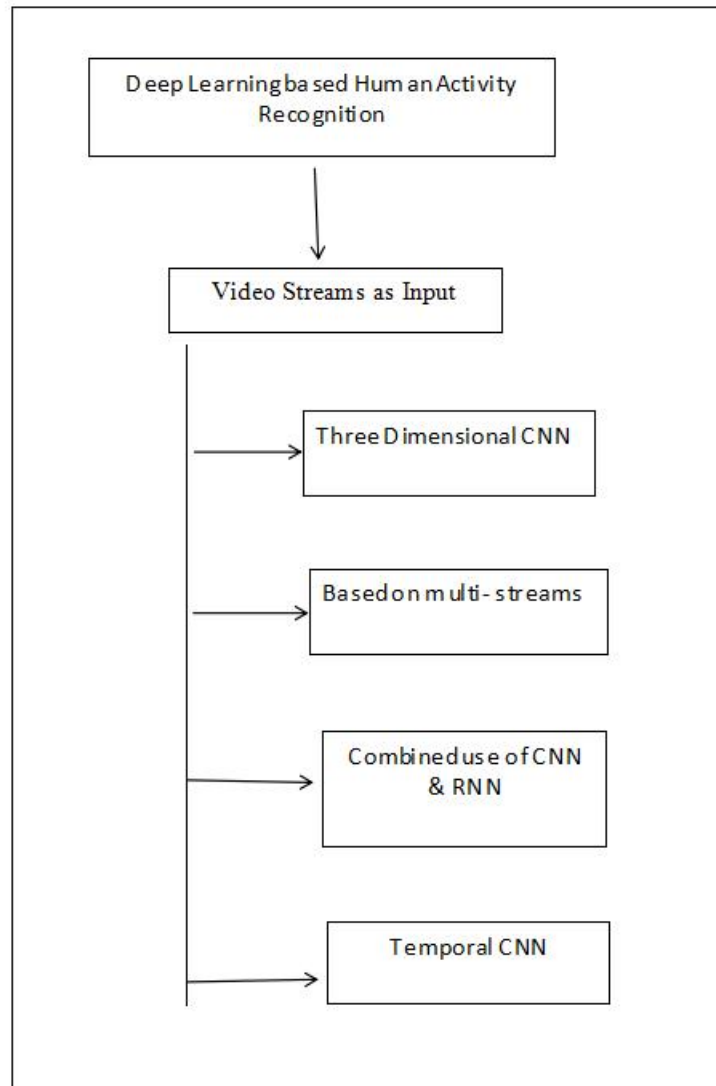
Component Diagrams

Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executables.

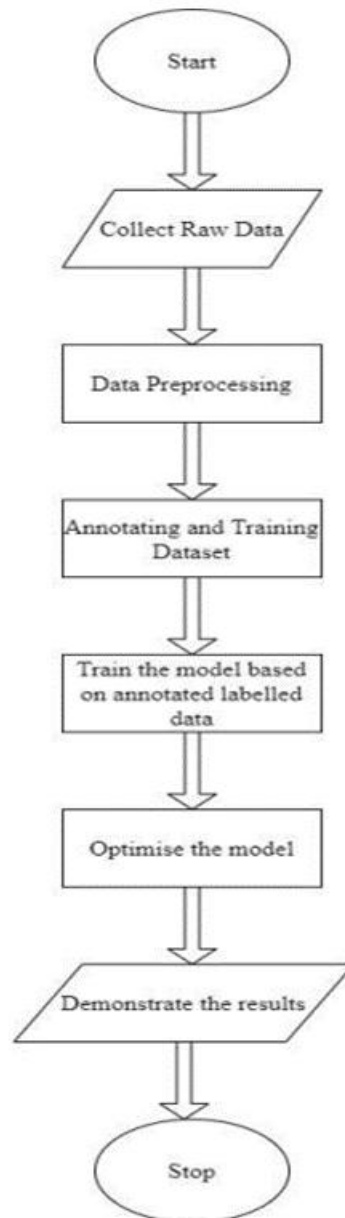
Deployment Diagrams

Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

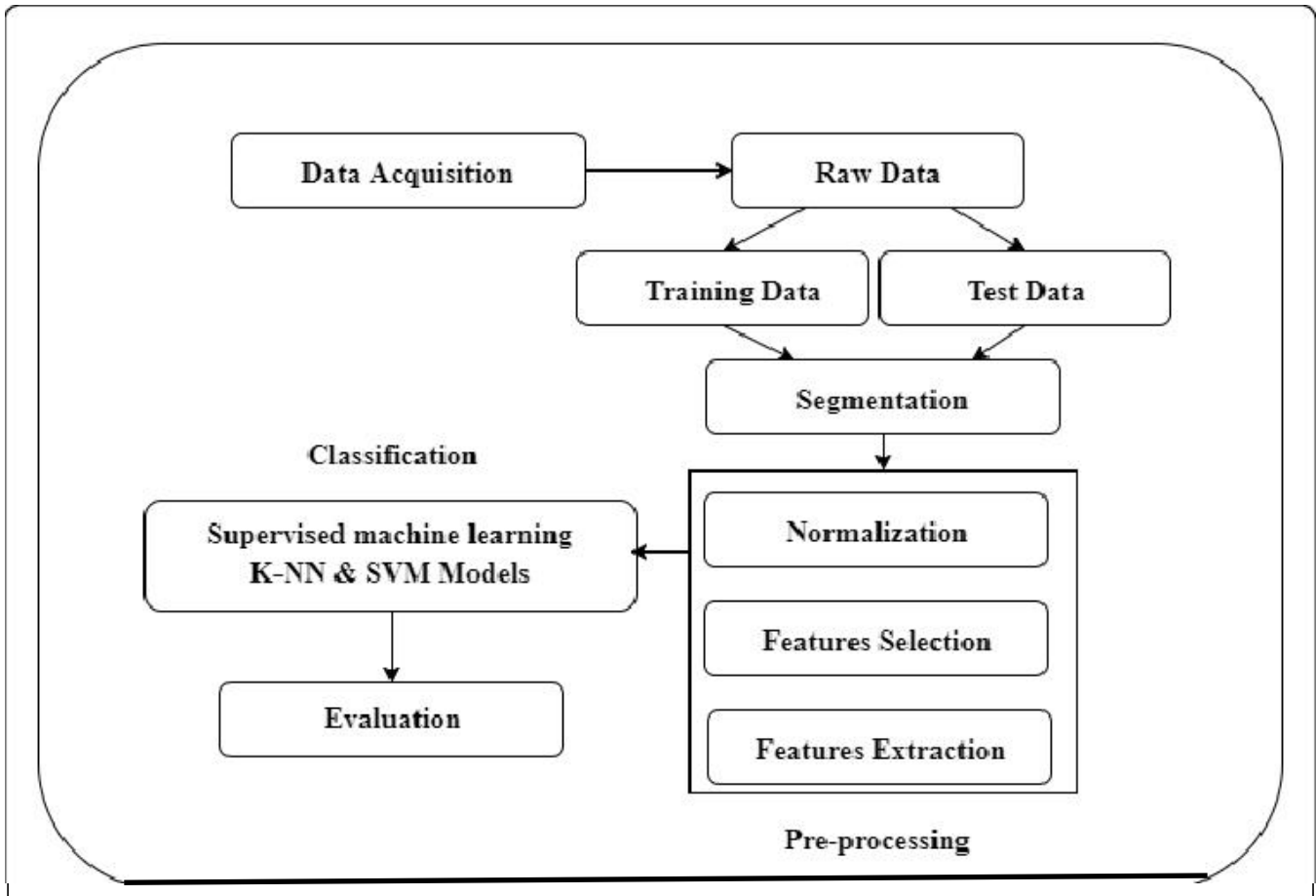
2.2.1.1 Use Case Diagram



Sequence Diagram



Activity Diagram



SOFTWARE ENVIRONMENT

SOFTWARE ENVIRONMENT

A. What is Python

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

1. Machine Learning
2. GUI Applications (like Kivy, Tkinter, PyQt etc.)
3. Web frameworks like Django (used by YouTube, Instagram, Dropbox)
4. Image processing (like Opencv, Pillow)
5. Web scraping (like Scrapy, BeautifulSoup, Selenium)
6. Test frameworks
7. Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite **easy to learn, understand, and code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python :-

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido

van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

B. What is Deep Learning Learning : -

Deep learning is a subset of machine learning that utilizes artificial neural networks, inspired by the structure and function of the human brain's neural networks. These networks consist of multiple layers of interconnected nodes (neurons) that process data in a hierarchical manner. What sets deep learning apart is its ability to automatically learn intricate patterns and representations directly from raw data. This is achieved through training on large datasets using techniques like back-propagation, where the network adjusts its internal parameters to minimize the error between its predictions and the actual targets. Deep learning has significantly impacted various domains, showcasing remarkable performance in tasks such as image recognition, natural language processing, speech recognition, and more. Its success is attributed to advancements in hardware, particularly GPUs, which enable faster training of complex models, as well as the availability of vast amounts of labeled data. Through convolutional neural networks (CNNs), recurrent neural networks (RNNs), generative adversarial networks (GANs), and other architectures, deep learning continues to push the boundaries of what machines can achieve in understanding and processing complex data.

Categories Of Deep Learning :-

Deep learning can be categorized into several subfields based on the types of problems they address and the architectures they employ. Some key categories include:

Supervised Learning: This category involves training deep learning models on labeled datasets, where the input data is paired with corresponding output labels. Examples include image classification, object detection, and language translation.

Unsupervised Learning: Here, the deep learning models learn patterns and structures from unlabeled data. Clustering, dimensionality reduction, and generative modeling are common tasks in unsupervised learning.

Semi-Supervised Learning: This category lies between supervised and unsupervised learning, where models are trained on a combination of labeled and unlabeled data. This approach is particularly useful when labeled data is scarce or expensive to obtain.

Reinforcement Learning: In reinforcement learning, agents learn to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Deep reinforcement learning involves using deep neural networks to approximate the value functions or policy functions necessary for decision-making.

Convolutional Neural Networks (CNNs): CNNs are specialized deep learning architectures designed for tasks involving grid-like data, such as images and videos. They leverage convolutional layers to automatically learn hierarchical representations of features.

Recurrent Neural Networks (RNNs): RNNs are designed to handle sequential data by maintaining a memory of past inputs. They are commonly used in tasks such as natural language processing, time series prediction, and speech recognition.

Generative Models: These models aim to generate new data samples that resemble the training data. Generative adversarial networks (GANs), variational autoencoders (VAEs), and autoregressive models are examples of architectures used for generative modeling.

These categories represent a broad overview of the diverse applications and architectures within the field of deep learning, each tailored to address specific types of data and problems.

C. Need for Deep Learning

Deep learning's necessity arises from its capacity to handle vast amounts of data efficiently, enabling insights extraction from complex datasets. By autonomously learning intricate patterns and representations, deep learning obviates the need for manual feature engineering, making it particularly adept at tasks with opaque or large-scale data. It has revolutionized fields like image and speech recognition, achieving human-level performance and fostering advancements in healthcare, autonomous vehicles, and security. Moreover, its prowess extends to natural language processing, empowering machines to comprehend and generate human language, leading to innovations in chatbots, language translation systems, and sentiment analysis tools. Beyond practical applications, deep learning also contributes to scientific research by accelerating processes like drug discovery and analyzing complex datasets in fields such as genetics and climate science. In essence, the demand for deep learning stems from its capability to tackle intricate problems, process extensive datasets, and deliver cutting-edge results across diverse domains

The need for deep learning is its potential for automation and efficiency enhancement across various industries. By leveraging deep learning models, organizations can streamline processes, optimize resource allocation, and reduce operational costs. For instance, deep learning algorithms can optimize supply chains by predicting demand more accurately, detect anomalies in manufacturing processes to prevent downtime, and enhance energy efficiency by optimizing consumption patterns. Moreover, deep learning-driven automation can improve customer service through chatbots, personalize recommendations in e-commerce platforms, and optimize marketing campaigns by analyzing vast amounts of consumer data. Thus, the demand for deep learning arises from its capacity to automate tasks, enhance operational efficiency, and drive innovation across sectors.

Challenges in Deep Learning :-

While deep learning offers promising solutions to many complex problems, it also faces several challenges:

Data Quality and Quantity: Deep learning models require large amounts of labeled data for effective training. Obtaining and labeling such data can be costly and time-consuming. Moreover, the quality of the data can significantly impact the performance of the models, as noisy or biased data can lead to inaccurate predictions.

Overfitting: Deep learning models, especially those with a large number of parameters, are prone to overfitting, where they memorize the training data instead of learning generalizable patterns. Regularization techniques and proper validation strategies are often necessary to mitigate this issue.

Computational Resources: Training deep learning models can be computationally intensive, requiring specialized hardware such as GPUs or TPUs. Additionally, scaling up models to handle larger datasets or more complex tasks can further increase the computational demands.

Interpretability: Deep learning models are often considered "black boxes," making it challenging to interpret how they arrive at their predictions. This lack of interpretability can be problematic, especially in fields where decisions need to be explainable, such as healthcare or finance.

Transfer Learning and Generalization: While deep learning models excel at learning representations from data, transferring these learned representations to new tasks or domains can be challenging. Ensuring that models generalize well across different datasets and scenarios remains an ongoing research area.

Ethical and Societal Implications: As deep learning technologies become more pervasive, concerns about privacy, bias, and fairness arise. Biased training data can lead to biased models, resulting in unfair outcomes for certain demographic groups. Addressing these ethical considerations is crucial for the responsible development and deployment of deep learning systems.

Applications of Deep Learning :-

Deep learning finds application across a wide range of fields due to its ability to learn complex patterns directly from data. Some notable applications include:

1. Computer Vision
2. Natural Language Processing (NLP)
3. Speech Recognition
4. Healthcare
5. Finance
6. Autonomous Systems
7. Recommendation Systems
8. Genomics and Bio informatics
9. Manufacturing and Industry 4.0
10. Environmental Monitoring

How to Start Learning Deep Learning?

1.Foundation in Mathematics and Programming:

A.Mathematics: Deep learning relies heavily on mathematical concepts such as linear algebra, calculus, probability, and statistics. Ensure you have a solid understanding of these topics, as they form the basis of many deep learning algorithms.

Programming: Python is the predominant language in deep learning due to its simplicity and the availability of powerful libraries. Learn Python fundamentals, including data structures, control flow, functions, and object-oriented programming.

Introduction to Machine Learning:

Before diving into deep learning, it's beneficial to grasp the basics of machine learning. Understand different types of machine learning algorithms (supervised, unsupervised, reinforcement learning), evaluation metrics, and the general workflow of building and training models.

Deep Learning Libraries:

Familiarize yourself with popular deep learning libraries such as TensorFlow, PyTorch, and Keras. These libraries provide high-level APIs for building and training deep neural

networks. Start with TensorFlow or PyTorch, as they offer extensive documentation and community support.

Explore introductory tutorials and documentation provided by these libraries to understand their functionalities, basic syntax, and how to build simple models.

Deep Learning Concepts:

Study fundamental concepts of deep learning, including neural networks, activation functions, loss functions, optimization algorithms (e.g., gradient descent, Adam), regularization techniques (e.g., dropout, L2 regularization), and hyperparameter tuning.

Understand common architectures like feedforward neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their applications in various domains.

Online Courses and Tutorials:

Enroll in online courses specifically focused on deep learning. Platforms like Coursera, Udacity, and edX offer courses taught by experts in the field. Recommended courses include Andrew Ng's "Deep Learning Specialization" on Coursera and the "Deep Learning Nanodegree" on Udacity.

Explore free tutorials and resources available on websites like TensorFlow's official website, PyTorch's documentation, and blogs such as Towards Data Science on Medium.

Practical Projects and Hands-on Learning:

Apply theoretical knowledge to practical projects by working on hands-on exercises and building deep learning models from scratch. Start with simple projects like image classification on datasets like MNIST or CIFAR-10.

Participate in coding challenges and competitions on platforms like Kaggle to apply your skills, learn from others, and benchmark your performance against peers.

Read Research Papers and Stay Updated:

Read seminal papers in deep learning to understand foundational concepts and cutting-edge techniques. Explore papers published in top conferences such as NeurIPS, ICML, CVPR, and ACL.

Follow reputable researchers, practitioners, and organizations on platforms like arXiv, Twitter, and LinkedIn to stay updated with the latest advancements, research trends, and best practices in deep learning.

Experimentation and Iteration:

Experiment with different architectures, hyperparameters, optimization algorithms, and regularization techniques. Learn from both successful experiments and failures, as they provide valuable insights into model behavior and performance.

Iterate on your models by continuously refining your approaches, incorporating feedback, and adapting to new challenges and datasets.

Join Communities and Seek Support:

Engage with the deep learning community by joining online forums, discussion groups, and social media channels. Participate in discussions, ask questions, share insights, and collaborate with others.

Attend local meetups, workshops, and conferences to network with peers, attend talks and workshops, and gain exposure to real-world applications and case studies.

Build a Portfolio and Showcase Your Work:

Create a portfolio showcasing your deep learning projects, code repositories, and contributions to the community. Use platforms like GitHub, Kaggle, and personal websites to showcase your skills, demonstrate your expertise, and attract potential collaborators or employers.

Continuously update and expand your portfolio with new projects, insights, and learnings as you progress in your deep learning journey.

Remember that learning deep learning is a continuous process that requires dedication, perseverance, and a willingness to learn and adapt. Stay curious, keep exploring new ideas and techniques, and don't hesitate to seek guidance and support from the vibrant deep learning community.

Terminologies of Deep Learning

Neural Network: Computational model inspired by the brain, composed of interconnected nodes (neurons).

Activation Function: Non-linear function applied to neuron outputs, enabling complex pattern learning.

Loss Function: Measures the difference between predicted and actual outputs during training.

Gradient Descent: Optimization algorithm that adjusts network parameters to minimize loss.

Backpropagation: Technique for computing gradients and updating parameters in neural networks.

Feedforward Neural Network: Information flows from input to output without feedback connections.

Convolutional Neural Network (CNN): Specialized network for grid-like data (e.g., images).

Recurrent Neural Network (RNN): Handles sequential data by maintaining internal memory.

Batch Normalization: Normalizes inputs of each layer to stabilize and speed up training.

Dropout: Regularization technique that randomly drops neurons during training to prevent overfitting.

Epoch: One complete pass through the entire training dataset during training.

Overfitting: Model memorizes training data and performs poorly on new data.

Underfitting: Model is too simple to capture underlying patterns in data.

Hyperparameters: Parameters set before training that influence model learning.

Transfer Learning: Technique using pre-trained models for related tasks to improve performance.

Advantages of Deep learning :-

Finds Important Stuff by Itself: Deep learning can look at data and figure out what's important without us telling it.

Likes Lots of Data: The more data you give it, the smarter it becomes. It learns better with more examples.

Sees Complex Patterns: It's good at spotting patterns, even really complicated ones. It looks at data in layers, like peeling an onion, to understand everything better.

Works Really Well: Deep learning is like having a super smart assistant. It's great at recognizing things in pictures, understanding speech, and lots more.

Can Do Many Things: Deep learning isn't just for one thing; it can learn lots of different tasks and adapt to new challenges easily.

Saves Time and Money: It can reuse what it's learned before to help with new tasks, saving time and resources.

Makes Quick Decisions: It's fast at making decisions, which is useful for things like self-driving cars and helping doctors diagnose diseases.

Getting Easier to Understand: People are working on making it easier to understand how deep learning works, so everyone can benefit from it.

Disadvantages of Deep Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps :-

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced.

This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x.

The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python :

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

D. PYTHON PACKAGES :

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated functions
- Tools for integration C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number Capabilities.

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- **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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows

E. How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7

device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.







Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	Release Notes
Python 3.6.9	July 2, 2019	 Download	Release Notes
Python 3.7.3	March 25, 2019	 Download	Release Notes
Python 3.4.10	March 18, 2019	 Download	Release Notes
Python 3.5.7	March 18, 2019	 Download	Release Notes
Python 2.7.16	March 4, 2019	 Download	Release Notes
Python 3.7.2	Dec. 24, 2018	 Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	6PG
Clipped source tarball	Source release		68111671e5b2db4aef709ab01bf099be	23017963	SG
XZ compressed source tarball	Source release		d33e4aae6097051c2mca45ee3604803	17131432	SG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4f675b3daf1a442c8a1ce08e6	34898436	SG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd625c38217a45773b75e4a93b243f	28082845	SG
Windows help file	Windows		063999573a2c9682ac56c4de6b47cd2	8111761	SG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9605c3c7b89ec08abe0318aa40729a2	7504381	SG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76d8bd03043a553e563400	26688368	SG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c5088bd73ae8e63a3bd351b4bd2	1362904	SG
Windows x86 embeddable zip file	Windows		9fab3bd18841879fda94133574139d8	6741626	SG
Windows x86 executable installer	Windows		33cc802942a54446a3d6451476394789	25663848	SG
Windows x86 web-based installer	Windows		1b670cfa5cd117df92c30983ea371d87c	1324608	SG

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

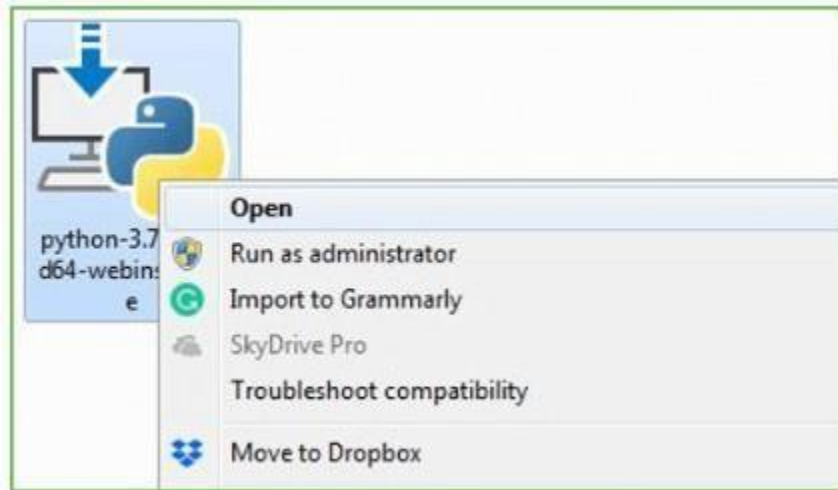
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



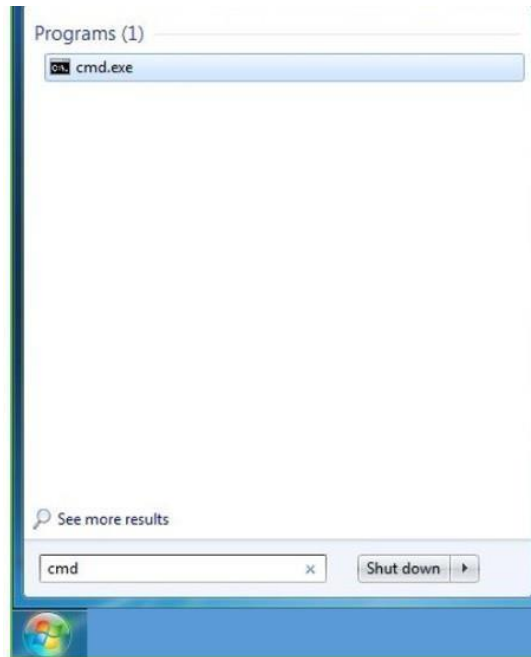
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

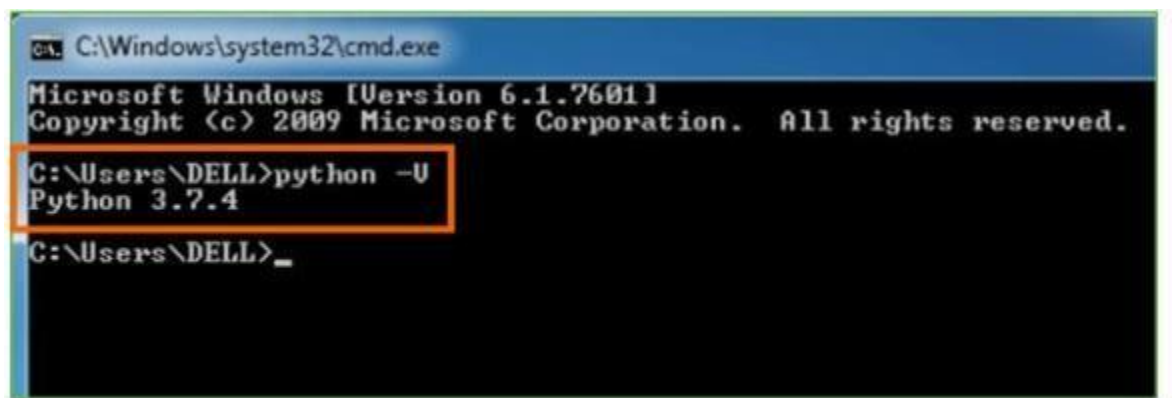
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.



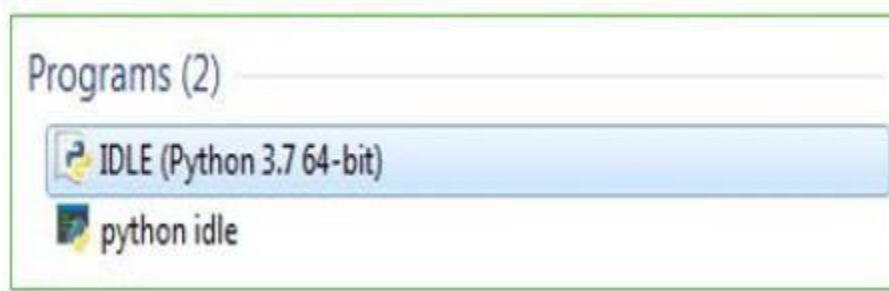
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

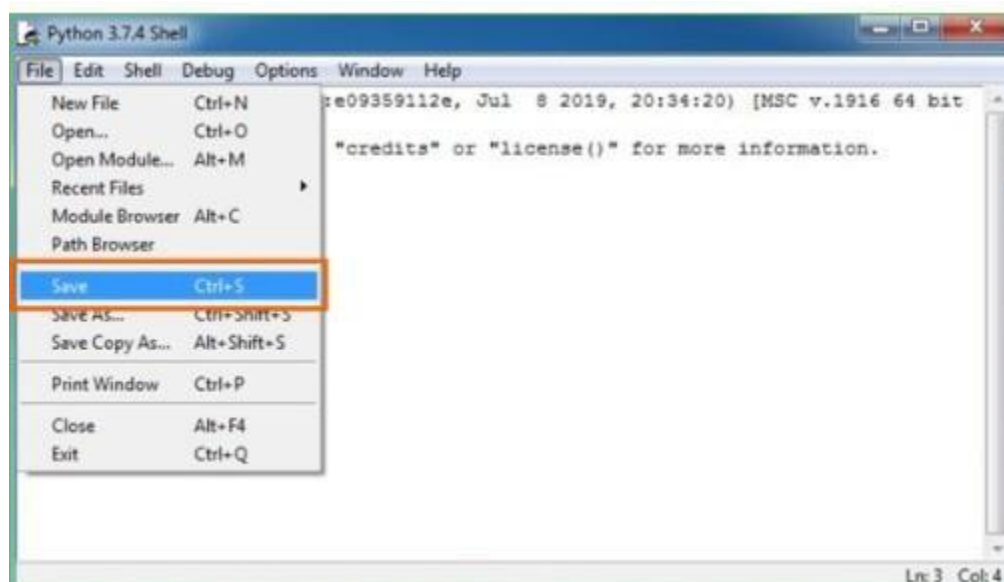
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

CODE

CODE

```
#@title Import the necessary modules

# TensorFlow and TF-Hub modules.

from absl import logging


import tensorflow as tf

import tensorflow_hub as hub

from tensorflow_docs.vis import embed


logging.set_verbosity(logging.ERROR)


# Some modules to help with reading the UCF101 dataset.

import random

import re

import os

import tempfile

import ssl

import cv2

import numpy as np


# Some modules to display an animation using imageio.

import imageio

from IPython import display
```

```

from urllib import request # requires python3

#@title Helper functions for the UCF101 dataset


# Utilities to fetch videos from UCF101 dataset

UCF_ROOT = "https://www.crcv.ucf.edu/THUMOS14/UCF101/UCF101/"

_VIDEO_LIST = None

_CACHE_DIR = tempfile.mkdtemp()

# As of July 2020, crcv.ucf.edu doesn't use a certificate accepted by the
# default Colab environment anymore.

unverified_context = ssl._create_unverified_context()


def list_ucf_videos():

    """Lists videos available in UCF101 dataset."""

    global _VIDEO_LIST

    if not _VIDEO_LIST:

        index = request.urlopen(UCF_ROOT, context=unverified_context).read().decode("utf-8")

        videos = re.findall("(v_[\w_]+\avi)", index)

        _VIDEO_LIST = sorted(set(videos))

    return list(_VIDEO_LIST)


def fetch_ucf_video(video):

    """Fetches a video and cache into local filesystem."""

    cache_path = os.path.join(_CACHE_DIR, video)

    if not os.path.exists(cache_path):

        urlpath = request.urljoin(UCF_ROOT, video)

```

```

print("Fetching %s => %s" % (urlpath, cache_path))

data = request.urlopen(urlpath, context=unverified_context).read()

open(cache_path, "wb").write(data)

return cache_path

```

Utilities to open video files using CV2

```

def crop_center_square(frame):

    y, x = frame.shape[0:2]

    min_dim = min(y, x)

    start_x = (x // 2) - (min_dim // 2)

    start_y = (y // 2) - (min_dim // 2)

    return frame[start_y:start_y+min_dim,start_x:start_x+min_dim]

```

```

def load_video(path, max_frames=0, resize=(224, 224)):

```

```

    cap = cv2.VideoCapture(path)

    frames = []

    try:

        while True:

            ret, frame = cap.read()

            if not ret:

                break

            frame = crop_center_square(frame)

            frame = cv2.resize(frame, resize)

            frame = frame[:, :, [2, 1, 0]]

            frames.append(frame)

```

```

    if len(frames) == max_frames:

        break

    finally:

        cap.release()

    return np.array(frames) / 255.0

def to_gif(images):

    converted_images = np.clip(images * 255, 0, 255).astype(np.uint8)

    imageio.mimsave('./animation.gif', converted_images, duration=40)

    return embed.embed_file('./animation.gif')

# Get the list of videos in the dataset.

ucf_videos = list_ucf_videos()

categories = {}

for video in ucf_videos:

    category = video[2:-12]

    if category not in categories:

        categories[category] = []

    categories[category].append(video)

print("Found %d videos in %d categories." % (len(ucf_videos), len(categories)))

for category, sequences in categories.items():

    summary = ", ".join(sequences[:2])

    print("%-20s %4d videos (%s, ...)" % (category, len(sequences), summary))

a# Get a sample cricket video.

video_path = fetch_ucf_video("v_CricketShot_g04_c02.avi")

```



```

sample_video = load_video(video_path)

sample_video.shape

i3d = hub.load("https://tfhub.dev/deepmind/i3d-kinetics-400/1").signatures['default']

def predict(sample_video):

    # Add a batch axis to the sample video.

    model_input = tf.constant(sample_video, dtype=tf.float32)[tf.newaxis, ...]

    logits = i3d(model_input)['default'][0]

    probabilities = tf.nn.softmax(logits)

    print("Top 5 actions:")

    for i in np.argsort(probabilities)[::-1][:5]:

        print(f' {labels[i]:22}: {probabilities[i] * 100:5.2f}%")

predict(sample_video)

curl -O https://upload.wikimedia.org/wikipedia/commons/8/86/End_of_a_jam.ogv

video_path = "End_of_a_jam.ogv"

sample_video = load_video(video_path)[:100]

sample_video.shape

to_gif(sample_video)

predict(sample_video)

```

SYSTEM TESTING

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, 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.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

Integration testing

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

Functional testing

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.

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

System Testing

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.

White Box Testing

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

Black Box Testing

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

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

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.

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

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.

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

Test Case Description	Expected Outcome	Result
1. Correct input video file provided	Human activities recognized accurately	Pass
2. Incorrect input video file provided	Error message displayed indicating invalid input	Fail
3. Various lighting conditions in the video	Human activities recognized accurately under all conditions	Pass
4. Video with occlusions (partial obstruction)	Human activities recognized accurately despite occlusions	Pass
5. Video with multiple persons performing actions	Correctly identifies and distinguishes between multiple persons	Pass
6. Video with no human activities	No activities recognized	Pass
7. Different camera angles and perspectives	Accurately recognizes activities from different viewpoints	Pass
8. Low-resolution video provided	Still accurately recognizes human activities	Pass
9. Corrupted video file provided	Error message displayed indicating corrupted file	Fail

Test Case Description	Expected Outcome	Result
10. Incorrect model parameters provided	Error message displayed indicating invalid parameters	F

SCREENS

Top 5 actions:

roller skating	:	96.85%
playing volleyball	:	1.63%
skateboarding	:	0.21%
playing ice hockey	:	0.20%
playing basketball	:	0.16%



CONCLUSION

CONCLUSION

The project “Human Activity Recognition Using OpenCV and Deep Learning” provides a versatile solution for accurately identifying human activities. Its real-time performance and high accuracy make it applicable across various domains like healthcare, security, and sports analytics. Future work may focus on refining model architecture and enhancing computational efficiency for wider deployment. Overall, this project signifies a notable advancement in activity recognition technology with broad potential applications.

FUTURE SCOPE

Human activity recognition (HAR) using OpenCV and deep learning holds significant promise across various domains. In healthcare, it offers solutions for remote patient monitoring, fall detection for the elderly, and rehabilitation progress assessment. Smart environments stand to benefit from HAR through automation of homes and offices based on human behavior, optimizing energy usage, and enhancing convenience. Additionally, gesture recognition, a subset of HAR, enables intuitive human-computer interaction, with applications spanning gaming, virtual reality, and sign language recognition.

In security and surveillance, HAR contributes to detecting anomalies, identifying intruders, and monitoring crowd behavior. It also finds application in sports and fitness for analyzing performance metrics, technique assessment, and personalized training feedback. Further, in industrial settings, HAR enhances safety protocols by identifying potential hazards and optimizing workflows. Beyond practical applications, HAR holds promise in education for personalized learning experiences and in research and development for driving advancements in computer vision, deep learning algorithms, and sensor technologies. Overall, the future scope of HAR using OpenCV and deep learning is expansive, with potential applications touching nearly every aspect of society and industry.

REFERENCES

1. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12- 10). "Deep Residual Learning for Image Recognition".
2. Srivastava, Rupesh Kumar; Greff, Klaus; Schmidhuber, Jürgen (2015-05-02). "Highway Networks".
3. Huang, Gao; Liu, Zhuang; Weinberger, Kilian Q.; van der Maaten, Laurens (2016-08-24). "Densely Connected Convolutional Networks
4. S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. arXiv preprint, arXiv:1609.08675, 2016.
5. Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In Proceedings of the International Conference on Computer Vision ICCV), 2017.
6. L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory- pooled deep-convolutional descriptors. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4305-4314, 2015.
7. Bishoy Sefen, Sebastian Baumbach et.(Human Activity Recognition Using Sensor Data of Smart phones and Smart watches ICAART 2016).
8. un X, Chen C, Manjunath BS , —Probabilistic motion para-meter models for human activity recognition,|| In: Proceedings of 16th international conference on pattern recognition, pp 443–450.
9. Kwon W, Lee TW, — Phoneme recognition using ICAbased feature extraction and transformation, Signal Process 84(6):1005– 1019, 2004.
10. Lee SI, Batzoglu S, —Application of independent compo-nent analysis to microarrays,|| Genome Biol
11. <https://github.com/techycs18/human-activity-recognition.git>
12. Udemy - Deep Learning for Computer Vision with Python and OpenCV