A

MAJOR PROJECT REPORT

ON

Real-Time Fire and Gun Detection Using Deep Learning for Enhanced Surveillance

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

J. JEEVANTH KUMAR (21R91A0597)

Under the Guidance

Of

Mrs. Y. ShivaSree

(Assistant Professor)

Department of CSE

Department of Computer Science and Engineering

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

(An Autonomous Institution)

Medbowli, Meerpet, Saroornagar, Hyderabad – 500097

(Affiliated to JNTUH, Approved by AICTE, Accredited by NBA & NAAC)

(2021-2025)

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

(Sponsored by TKR Educational Society)

(Approved by AICTE, Affiliated by JNTUH, Accredited by NBA & NAAC)

Medbowli, Meerpet, Saroornagar, Hyderabad-500097

Phone:040-24092838 Fax:+91-040-24092555

E-mail: tkrec@rediffmail.com Website: www.tkrec.ac.in

Department of Computer Science & Engineering

College code: R9

CERTIFICATE

This is to certify that the Major Project report on **"Real-Time Fire and Gun Detection Using Deep Learning for Enhanced Surveillance"** is a bonafide work carried out by **J.Jeevanth Kumar(21R91A0597)** in partial fulfillment for the requirement of the award of B.Tech degree in Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Hyderabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The result of investigation enclosed in this report have been verified and found satisfactory. The results embodied in the project work have not been submitted to any other University for the award of any degree.

SUPERVISOR	**HEAD OF DEPARTMENT**
Mrs.Y. ShivaSree	Dr. CH. V. Phani Krishna
(Assistant Professor)	Professor
……………………………..	……………………………..
EXTERNAL EXAMINER	**PRINCIPAL**
	Dr. K. Venkata Murali Mohan
	Professor
……………………………….	………………………………….

College code: R9

DECLARATION

I hereby declare that the Major Project report entitled **“Real-Time Fire and Gun Detection Using Deep Learning for Enhanced Surveillance”** is done under the guidance of **Mrs.Y. SHIVA SREE**, Assistant Professor**,** Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, is submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** from **Jawaharlal Nehru Technological University**, Hyderabad.

This is a record of bonfire work carried out by us in **Teegala Krishna Reddy Engineering College** and the results embodied in this project have not been reproduced or copied from any source.

Submitted by

J. JEEVANTH KUMAR (21R91A0597)

ACKNOWLEDGEMENT

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

I am extend my deep sense of gratitude to **Dr. K. Venkata Murali Mohan**, **Principal**, Teegala Krishna Reddy Engineering College, Meerpet, for permitting me to undertake this project.

I am indebted to **Dr. CH. V. Phani Krishna, Professor & Head of the Department**, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his support and guidance throughout our project.

I am indebted to our guide **Mrs. Shivasree**, **Assistant Professor,** Computer Science Engineering,Teegala Krishna Reddy Engineering College, Meerpet for her support and guidance throughout our project.

I am indebted to the project coordinator **Mrs. P. Swetha, Assistant professor**, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for her support and guidance throughout our project.

Finally, I express thanks to one and all that have helped me in successfully completing this major project. Further I would like to thank my family and friends for their moral support and encouragement.

By

J.Jeevanth Kumar (21R91A0597)

ABSTRACT

Real-time object detection plays a crucial role in enhancing surveillance systems. With the increasing incidents of gun violence, mass shootings, and fire hazards, it is essential to develop automated detection mechanisms. This research focuses on detecting fire and guns in real-time video footage using Deep Learning and Computer Vision techniques. The system employs the YOLOv3 (You Only Look Once version 3) algorithm, which enables high-speed and accurate object detection. The proposed model processes each video frame and identifies potential threats by classifying objects as fire or gun. Upon detection, an instant alert is triggered for the concerned authorities to take immediate action. This solution enhances public safety by providing a rapid and efficient surveillance mechanism to prevent casualties and property damage. The final model has a validation loss of 0.2864, with a detection rate of 45 frames per second and has been benchmarked on datasets like IMFDB, UGR, and FireNet with accuracies of 89.3%, 82.6% and 86.5% respectively. Experimental result satisfies the goal of the proposed model and also shows a fast detection rate that can be deployed indoor as well as outdoors.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Fire accidents are a major risk in industries, crowded events, public gatherings, and heavily populated regions throughout India. Such incidents can lead to severe property loss, environmental harm, and can endanger both human and animal lives. According to the recent National Risk Survey Report [1], Fire stood at the third position overtaking corruption, terrorism, and insurgency thus posing a significant risk to our country's economy and citizens. The recent -fires in Australia reminded the world, the destructive capability of fire and the impending ecological disaster, by claiming millions of lives resulting in billions of dollars in damage. Early detection of fire accidents plays a crucial role in saving countless lives and preventing permanent damage to infrastructure, along with avoiding huge financial losses. To achieve high accuracy and reliability in densely populated urban settings, it becomes essential to rely on local surveillance systems. Traditional opto-electronic fire detection methods come with several drawbacks, such as the need for separate and sometimes redundant setups, hardware failures, frequent maintenance, and the risk of false alarms. Moreover, using sensors in hot and dusty industrial environments is often impractical. Given these challenges, detecting fires through surveillance video streams stands out as a highly practical and cost-effective alternative. It offers an efficient way to replace existing systems without requiring major infrastructure changes or heavy investments. However, current video-based machine learning models largely depend on domain expertise and manual feature engineering, which means they must constantly be updated to stay effective against emerging threats. We aim to develop a classification model using Deep learning and Transfer Learning to recognize fires in images/video frames, thus ensuring early detection and save manual work. This model is capable of detecting fires in surveillance video footage.Unlike existing systems, this neither requires special infrastructure for setup like hardware-based solutions, nor does it need domain knowledge and prohibitive computation for development.

Among the various computer-based methods for fire detection, the most prominent ones we identified include Artificial Neural Networks, Deep Learning, Transfer Learning, and Convolutional Neural Networks. Artificial Neural Network based approaches seen in paper uses Levenberg Marquardt training algorithm for a fast solution. The algorithm's accuracy varied between 61% and 92%, while false positives ranged from 8% to 51%. Although this method achieved high accuracy with a relatively low false positive rate, it still demands extensive domain expertise. In this paper, the author highlights that current hardware-based detection systems suffer from low accuracy and a high rate of false alarms, making them more prone to inefficiency. It is also not suitable for detecting fires breaking out in large areas such as s, warehouses, fields, buildings or oil reservoirs. with 12 layers. Image augmentation methods like rotation, contrast adjustment, zooming in and out, altering saturation, and changing aspect ratios were applied to generate multiple versions of each image, resulting in a total of 1,720 samples. The goal was to accurately place a bounding box around the flame area. This approach outperformed existing models, especially when the flame color differed from those seen during training.

The main idea of our project is to create a system that monitors surveillance data of an area and sends alerts in case a fire or gun is detected. Closed Circuit Television (CCTV) cameras record video footage 24 hours of the day, however there isn't enough manpower to monitor each and every camera for various anomalous events. There are systems to detect fire using smoke sensors in many places like schools, educational institutes, etcHowever, there is a growing need for a cost-effective system that integrates both fire and gun detection for enhanced security. Surveillance systems such as closed-circuit television (CCTV) and drones are becoming increasingly common. Research also shows that the installation of CCTV systems helps to combat mass shooting incidents and are also extremely important for evidence collection.

The research work uses YOLO (You Only Look Once) object detection system which uses convolution neural networks for object detectionIt is among the fastest algorithms, delivering strong performance with minimal loss in accuracy.

The training of this model has been done on the cloud to save hundreds of hours of GPU time on a local runtime. Using hosted runtime has also been beneficial in fine-tuning our model to near perfection. The guns and fires found in CCTV videos in the dataset occupy only a small portion of the entire frame, hence our primary objective is to implement algorithm an that would accurately draw multiple bounding boxes in such low-quality videos. In an era where public safety and security are of paramount importance, traditional surveillance systems often fall short in detecting critical threats like fire outbreaks and firearm-related incidents in real time. The increasing number of violent crimes and accidental fires calls for intelligent, automated solutions capable of swift and accurate threat detection. To address this challenge, our project focuses on the development of a **real-time fire and gun detection system** powered by deep learning techniques.

This system leverages the capabilities of computer vision and convolutional neural networks (CNNs) to identify potential threats from live video feeds. By integrating state-of-the-art object detection algorithms such as YOLO (You Only Look Once) or SSD (Single Shot Detector), the model can accurately detect the presence of flames or firearms in various environments. Once a threat is detected, the system can send immediate alerts, allowing for faster response times and possibly preventing loss of life and property.

2.LITERATURE SURVEY

Celik et al.. Their system compares the characteristics of foreground objects with the statistical color features typically associated with fire. To achieve this, they developed a simple adaptive background model based on three Gaussian distributions, with each distribution representing the pixel statistics for one color channel. According to the first rule, the value of the red component of an RGB pixel must be greater than the mean of Red components of the entire image. The next rule states that the value of the red component of a pixel must be greater than the green component which must be greater than the blue component. The final rule takes into consideration the ratio of Red, Blue and Green components. All these rules complement the previous rules. Error is generated due to non-linearities in the fixed camera, sudden changes in lighting conditions and also due to some kind of materials producing different fire colors while burning. However, this method fails in case there is only smoke and no red-colored pixels.

Satellite-based systems can monitor a wide area, but satellite imagery resolution is low [2]. A fire is detected when it has grown quite a lot, so it is not possible to detect it in realtime. Such systems are also very expensive [3]. In satellite-based forest fire detection systems, weather conditions such as overcast skies or rainfall significantly reduce accuracy due to limitations from the satellites' long scanning intervals and low image resolution.. [4]

M. Trinath et al. [5]offers an IoT-based solution to the problem, utilizing temperature and smoke sensors for detection.

The biggest drawback of this system is that the sensors are costly and delicate and may be easily damaged due to various natural factors.

R-CNN based method for handheld gun detection [7] using a pre-trained VGG model is proposed. Weapons in the segmented images are identified using the Fast Retina Keypoint (FREAK) technique combined with the Harris Interest Point Detector.. Testing was done on a dataset built from the Internet Movie Firearm Database (IMFDB). The model could detect and classify three types of guns namely revolvers, rifles, and shotguns. However, for this method to detect guns, it has to be held by humans and not otherwise.

The visual gun detection system using SIFT (Scale Invariant Feature Transform) and Harris interest point detector was proposed which utilized color-based segmentation to take out a distinct object from an image using K-Means clustering algorithm.

Celik et al. introduced a new model for detecting fire and smoke using an image processing approach. Specific rules were defined for identifying fire pixels, which were then fed into a Fuzzy Inference System (FIS) operating in both the RGB and YCbCr color spaces.. Based on the probability value, a rule table is formed depending on which a pixel is considered to be fire. They report to have 99% accuracy but, this cannot be used for real-time monitoring.

Grega et al. proposed a method for the automatic detection of dangerous situations in CCTV systems by leveraging image processing and machine learning techniques. In the video footage, firearms and knives were detected using sliding window techniques, fuzzy classifiers, and Canny edge detectors.. The dataset and detection system constructed by the authors were made available.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- The system compares information about foreground objects with statistical color information of fire. A simple adaptive background model was created using three Gaussian distributions, with each distribution representing the pixel statistics of a specific color channel. Through adaptive background subtraction algorithms, the foreground information is extracted and then evaluated using a statistical fire color model to determine whether the foreground object is a potential fire candidate
- Satellite-based systems can monitor a wide area, but satellite imagery resolution is low . Fire is usually detected only after it has grown significantly, making real-time detection difficult.

Disadvantages:

1. SUDDEN CHANGES IN LIGHTING CONDITIONS
2. INEFFICIENCY IN COMPLEX BACKGROUNDS

3.2 PROPOSED SYSTEM

- In this project, the YOLO (You Only Look Once) object detection system is used, which applies convolutional neural networks (CNNs) for detecting objects. It is among the faster algorithms, maintaining high accuracy with minimal performance loss.

The proposed experiment utilizes the You Only Look Once (YOLO) v3 model, a deep learning framework built on Darknet, an open-source neural network platform developed in C.. YOLOv3 is considered the best option as it offers real-time detection while maintaining high accuracy.. The architecture used is darknet53 which consists of 53 convolutional layers each followed by Leaky ReLU activation functions and batch normalization layers are used, resulting in a fully convolutional network (FCN).

Advantages:

1. A deep learning model designed for real-time, frame-by-frame fire and gun detection has been created, delivering impressive accuracy.

2. Although the Darknet53 model is relatively bulky, it offers strong detection capabilities. Its performance in terms of detections per frame is well-suited for real-time monitoring and can be implemented on any GPU-based system.

3.3 SYSTEM REQUIREMENTS

HARDWARE

• SYSTEM	:	Core i3 12th generation
• RAM	:	4GB
• HARD DISK	:	100GB

SOFTWARE

• OS	:	Windows 10/11
• PROGRAMMING LANGUAGES	:	Python

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

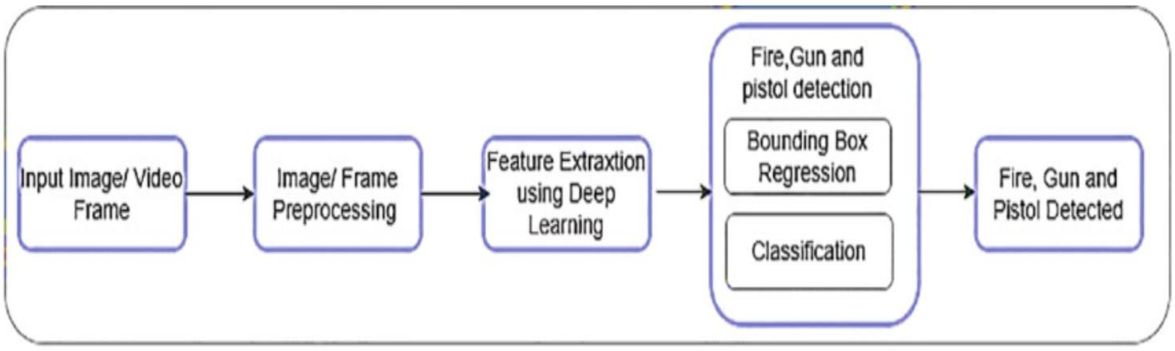

Input Image/ Video Frame
Image/ Frame Preprocessing
Feature Extraxtion using Deep Learning
Fire,Gun and pistol detection
Bounding Box Regression
Classification
Fire, Gun and Pistol Detected

Fig.4.1: System Architecture

Explanation of System Architecture:

The system architecture for real-time fire and gun detection using deep learning follows a structured pipeline designed to process input data and accurately identify potential threats. It begins with capturing input in the form of an image or video frame, which is then passed through a preprocessing stage to enhance quality and standardize the format for further analysis. This involves tasks like resizing, normalizing, and reducing noise. Once preprocessed, the data is fed into a deep learning model that performs feature extraction, identifying crucial visual patterns like edges, shapes, and textures. These extracted features are then used in the detection phase, which consists of two components: bounding box regression and classificationBounding box regression identifies the object's location within the frame by drawing a box around it, while the classification process determines what the object is—whether it is fire, a gun, or a pistol. The final output presents the detected objects with labeled bounding boxes, enabling real-time alerts and responses to enhance surveillance and public safety.

Input Acquisition:

The system begins by acquiring input in the form of either a static image or real-time video frame. These frames are captured from surveillance cameras or video streams and serve as the raw data for the detection pipeline.

Preprocessing:

After receiving the input, it is preprocessed to maintain consistency and enhance the model's performance. This involves resizing the frame to a fixed resolution, normalizing pixel values, reducing noise, and converting color formats if needed. These steps help in standardizing the input for reliable feature extraction.

Feature Extraction Using Deep Learning:

After preprocessing, the image is passed into a deep learning model—typically a convolutional neural network (CNN)—to extract meaningful features. These features capture key visual details like edges, textures, and patterns, which are essential for object recognition and classification.

Detection Module:

The detection module consists of two main components: bounding box regression and classification. Bounding box regression is responsible for predicting the coordinates of objects in the frame by drawing boxes around them. The classification part of the model then analyzes each detected object and identifies it as fire, a gun, or a pistol based on the extracted features.

Output Generation:

In the final stage, the system displays the results by overlaying labeled bounding boxes on the original frame, indicating the presence and type of detected objects. If a threat is identified, the system can trigger real-time alerts, thereby enabling quick and effective response in surveillance scenarios.

Fire, Gun, and Pistol Detection:

This module is divided into two sub-processes:

- **Bounding Box Regression:** This component of the network predicts the coordinates for bounding boxes that precisely enclose the detected objects, helping the system pinpoint the exact location of the fire, gun, or pistol within the frame..
- **Classification**: Once the bounding boxes are identified, the classification network assigns a label (fire, gun, or pistol) to each detected object based on the extracted features.

4.2 UML DIAGRAMS

UML is a standardized language used to visualise, specify, assemble, and file software program systems and their architectural designs. There are various types of UML diagrams, each serving a unique cause, whether created before implementation or used later on as a part of the documentation method. The broadest categories that encompass all other sorts are Behavioural UML diagram and Structural UML diagramAs the name implies, some UML diagrams focus on studying and illustrating the shape of a device or manner, whilst others constitute the system's behavior, its actors, and its additives. He numerous types of UML diagrams can be classified as follows.

• Use Case Diagram

• Sequence Diagram

• Activity Diagram

• Class Diagram

Just like with anything else, attaining a success results calls for the proper tools. When documenting software program, procedures, or structures, it's essential to have equipment that provide UML annotations and templates for developing UML diagrams. Various software equipment are to be had to assist in drawing those diagrams, and that they typically fall into two important categories:

1. Paper and Pen – Pick up some paper and a pen, then use a web UML syntax cheat sheet to start sketching the diagram you.
2. Online Tools – There are many online packages designed for drawing UML diagrams. Most provide unfastened trials or a constrained number of diagrams on their unfastened tiers. For long-time period use, buying a top class subscription for this type of gear is mostly a greater realistic solution.

4.2.1 USE CASE DIAGRAM

Use case diagrams are designed to seize a system's necessities, thinking about both inner and external factors. At its core, a use case diagram showcases how a user engages with the system, highlighting the connections between the user and the different use cases.

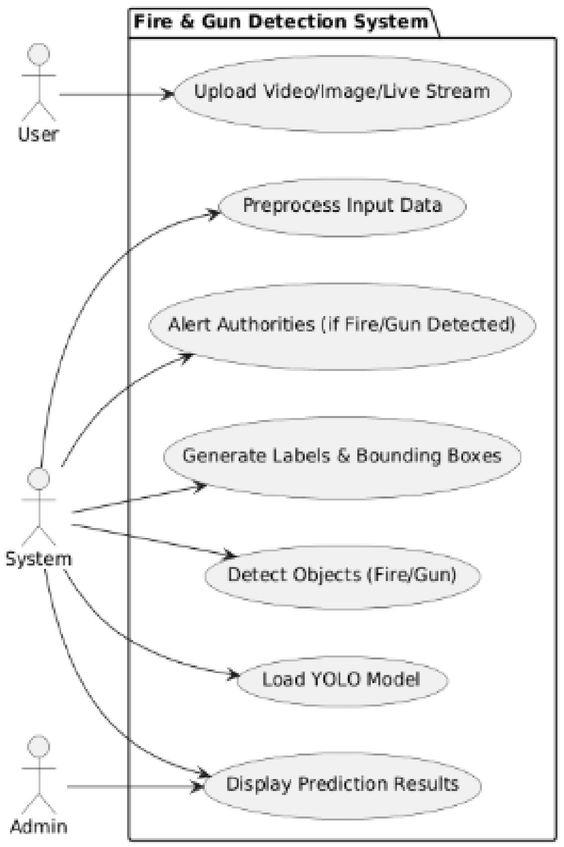

Fire & Gun Detection System
User
Upload Video/Image/Live Stream
Preprocess Input Data
Alert Authorities (if Fire/Gun Detected)
Generate Labels & Bounding Boxes
System
Detect Objects (Fire/Gun)
Load YOLO Model
Admin
Display Prediction Results

Fig 4.2.1: Use case Diagram

4.2.2 CLASS DIAGRAM

Class diagrams serve as the foundational elements of any object-oriented design approach. They help visualize classes along with their relationships, interfaces, associations, and collaborations within a system. In UML (Unified Modeling Language), class diagrams are standardized to accurately depict the structure of an application built around object-oriented principles.

Since classes form the core of an object-oriented application, class diagrams provide a clear and organized way to represent classes, inheritance hierarchies, relationships, and other key OOP concepts. They outline different types of objects along with the static relationships that exist among them.

The **primary purposes** of using class diagrams are:

- To support the development of component and deployment diagrams.
- To graphically depict the static structure of the system.

- In a class diagram, every class is shown as a rectangle split into three parts: the class name, its attributes, and its operations (or methods).
  Additionally, three types of visibility modifiers are commonly used to control access to attributes and operations.

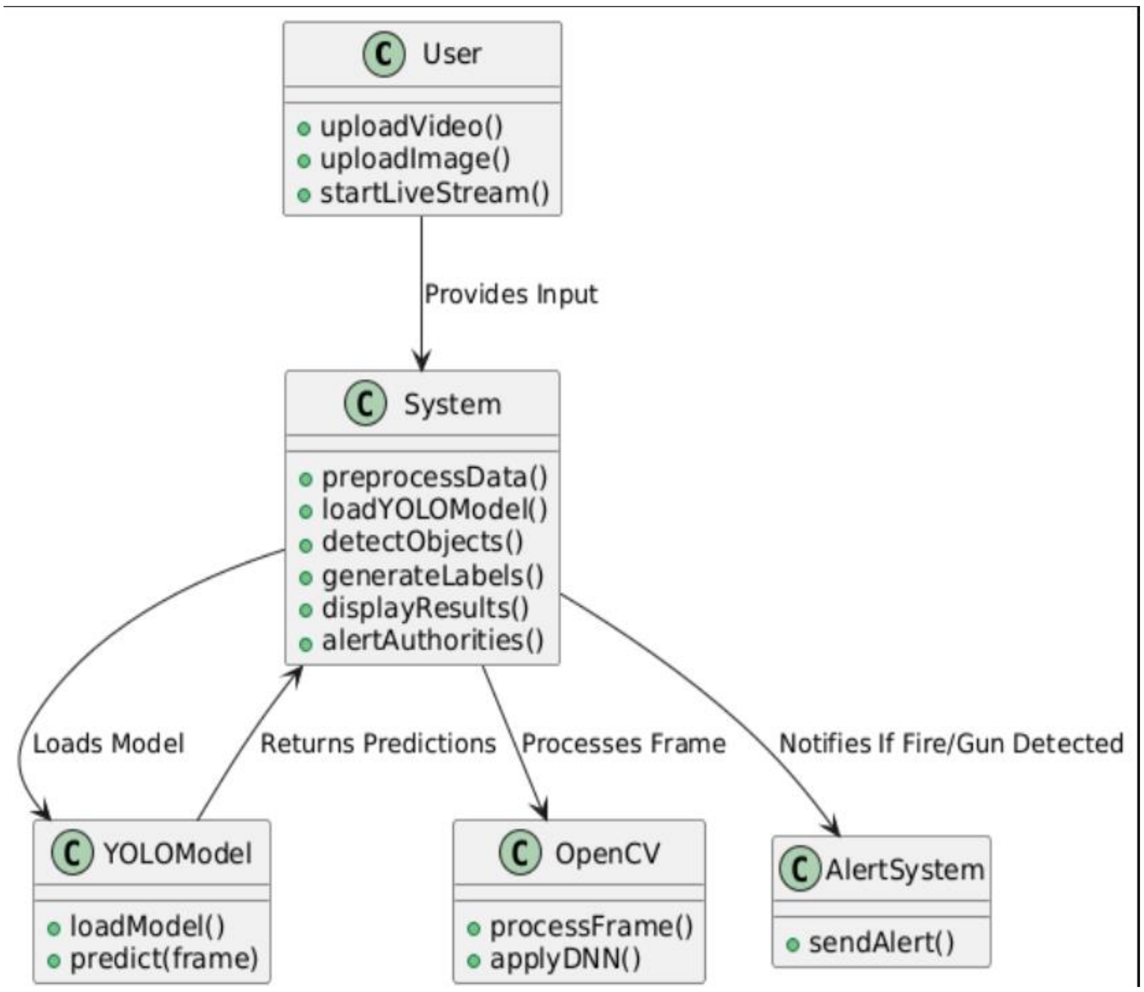


Fig 4.2.2: Class Diagram

4.2.3 SEQUENCE DIAGRAM

UML Sequence Diagrams are use to symbolize the glide of messages, occasions, and movements, between the gadgets or components of a machine. Time is represented within the vertical path showing the collection of interactions of the header factors, which can be displayed horizontally at the pinnacle of the diagram. Sequence Diagrams are used on the whole to layout, record and validate the structure, interfaces and common sense of the gadget by means of describing the collection.Moves that need to be executed to complete a venture or scenario. UML collection diagrams are beneficial design gear because they provide a dynamic view of the machine behaviour which can be hard to extract from static diagrams or specs.

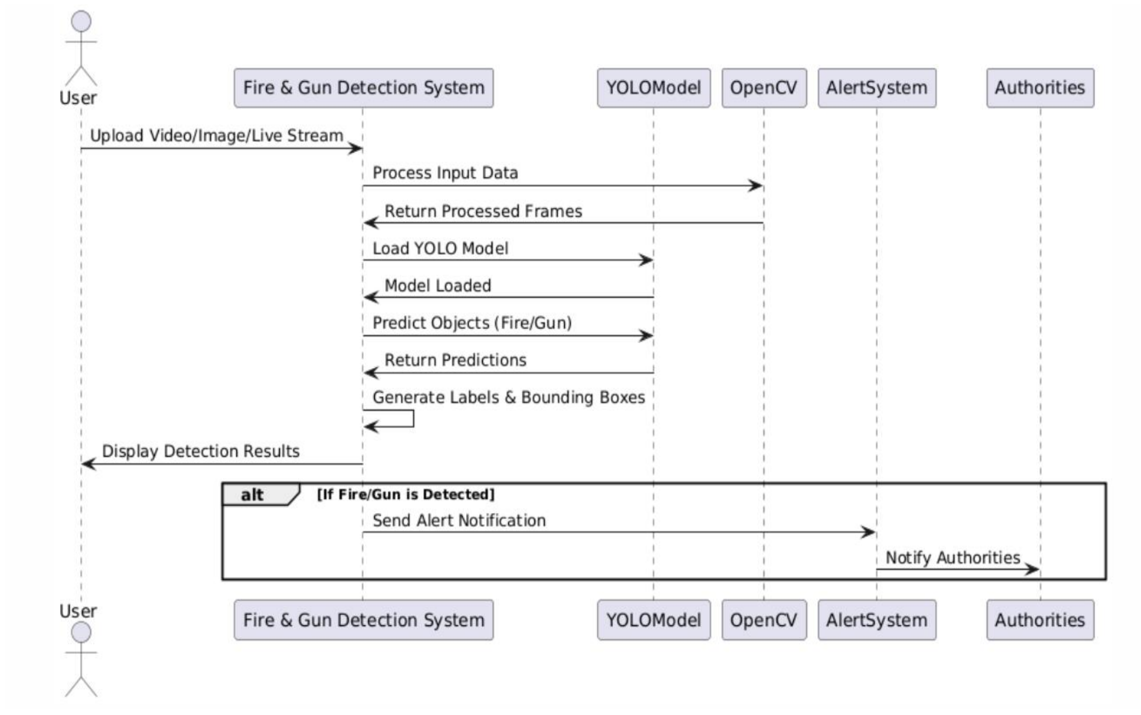


Fig4.2.3Sequence Diagram

actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behaviour which can be difficult to extract from static diagrams or specifications.

4.2.4 ACTIVITY DIAGRAM

Activity diagrams provide a visual representation of workflows, capturing the sequence of activities and actions while supporting decisions, iterations, and parallel processes. In UML, they are often used to model the business and operational workflows of a system, illustrating how various components interact step-by-step.

Each rectangular box within an activity diagram represents a specific activity or action performed by the system. Arrows are used to indicate the direction of control flow from one activity to another. Diamond-shaped nodes symbolize decision points where the flow can branch based on certain conditions, allowing multiple possible paths. Rounded rectangles mark the starting and ending points of the workflow.

Activity diagrams are especially valuable because they clearly depict the control flow within a process, helping stakeholders visualize how tasks are organized and executed within the system.

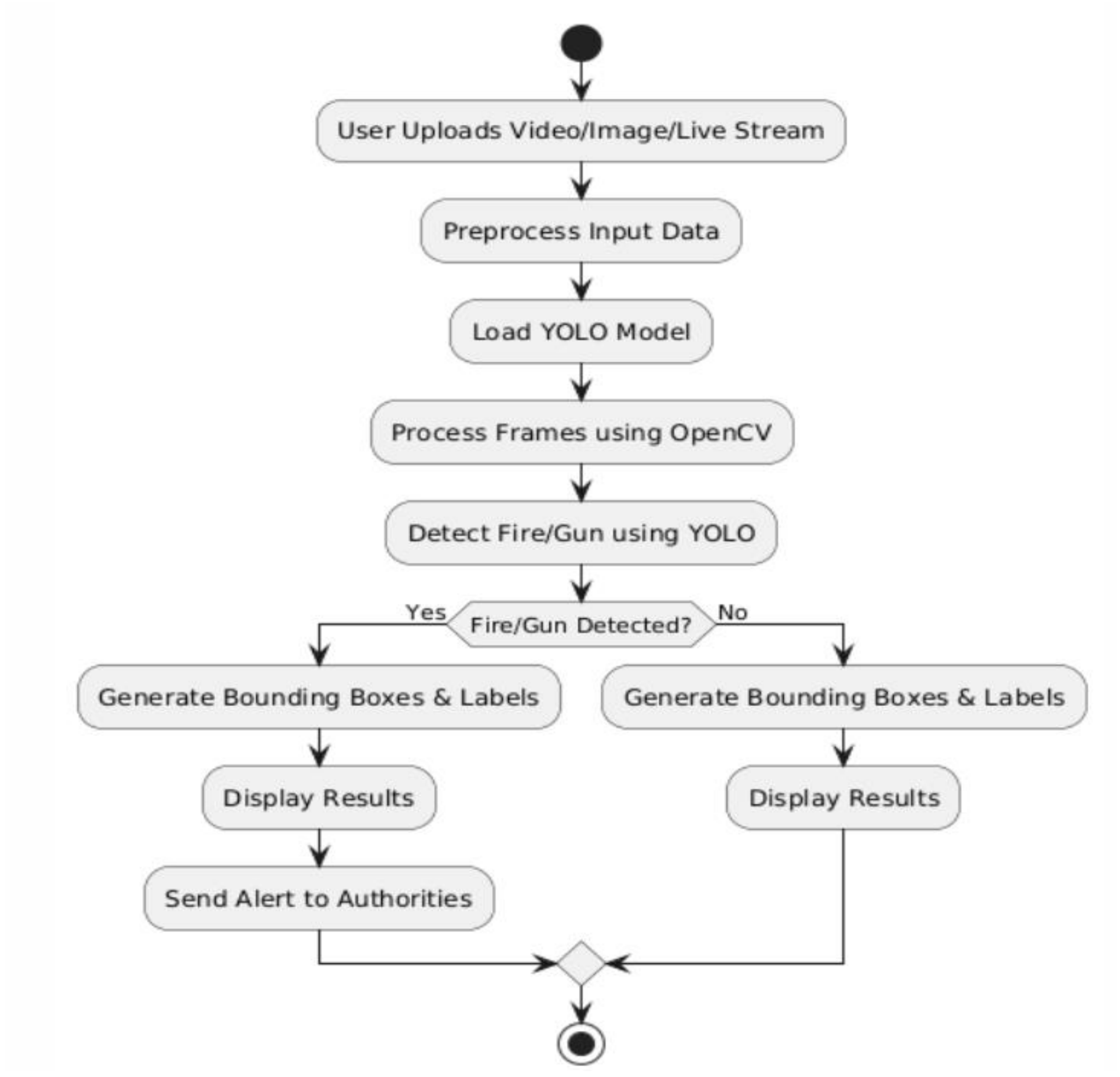

User Uploads Video/Image/Live Stream
Preprocess Input Data
Load YOLO Model
Process Frames using OpenCV
Detect Fire/Gun using YOLO
Fire/Gun Detected?
Yes
No
Generate Bounding Boxes & Labels
Generate Bounding Boxes & Labels
Display Results
Display Results
Send Alert to Authorities

Fig 4.2.4: Activity Diagram

5. IMPLEMENTATION

5.1 INTRODUCTION

Fire accidents pose a serious hazard to industries, crowded events, social gatherings, and densely populated areas which might be determined throughout India. These styles of incidents can also reason harm to property, surroundings, and pose a risk to human and animal existence. According to the current National Risk Survey Report [1], Fire stood on the third role overtaking corruption, terrorism, and insurgency consequently posing a widespread chance to our united states's financial system and residents. The latest -fires in Australia reminded the sector, the unfavorable functionality of fireplace and the impending ecological catastrophe, through claiming tens of millions of lives ensuing in billions of greenbacks in damage. Early detection of hearth-injuries can keep innumerable lives at the side of saving residences from permanent infrastructure damage and the ensuing financial losses. In order to obtain high accuracy and robustness in dense urban regions, detection via local surveillance is essential and also powerful. Traditional opto-electronic fireplace detection systems have foremost hazards: Requirement of separate and regularly redundant systems, fault-prone hardware systems, everyday renovation, fake alarms and so forth. Usage of sensors in warm, dusty business conditions is likewise now not viable. Thus, detecting fires through surveillance video circulation is one of the maximum possible, fee-powerful answer suitable for substitute of current systems without the want for huge infrastructure set up or funding. Main website. The customer can then compare prices of merchandise that are to be had on e-commerce web sites.

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

Python a flexible programming language doesn't come pre-hooked up for your laptop gadgets. Python became first launched in the 12 months 1991 and until nowadays it's far a totally famous excessive-stage programming language. Its design philosophy strongly values code readability, notably achieved through its distinctive use of significant whitespace. The item-oriented approach and language construct supplied by using Python enables programmers to put in writing both clean and logical code for projects. This software program does no longer come pre-packaged with Windows.

A guide on how to install Python on both Windows and Mac :

There had been numerous updates inside the Python model over time. The query is the way to install Python? It might be complicated for the amateur who's willing to start learning Python but this tutorial will solve your query. The trendy or the most modern version of Python is version 3.12.0 or in different phrases, it's miles Python 12.

Note: The python model 3.12.0 cannot be used on Windows XP or earlier devices. Before you start with the set up technique of Python. First, you want to recognise approximately your System Requirements. Based in your system kind i.E. Operating device and primarily based processor, you ought to down load the python version. My computer is running a 64-bit version of Windows python 12. Download the Python Cheat sheet right here.The steps on the way to set up Python on Windows 10, eight and 7 are divided into four parts to help understand higher.

Download the Correct model into the machine

Step 1: Go to the legitimate website to download and installation python the use of Google Chrome or any other net browser. OR Click on the following hyperlink: https://www.Python.Org

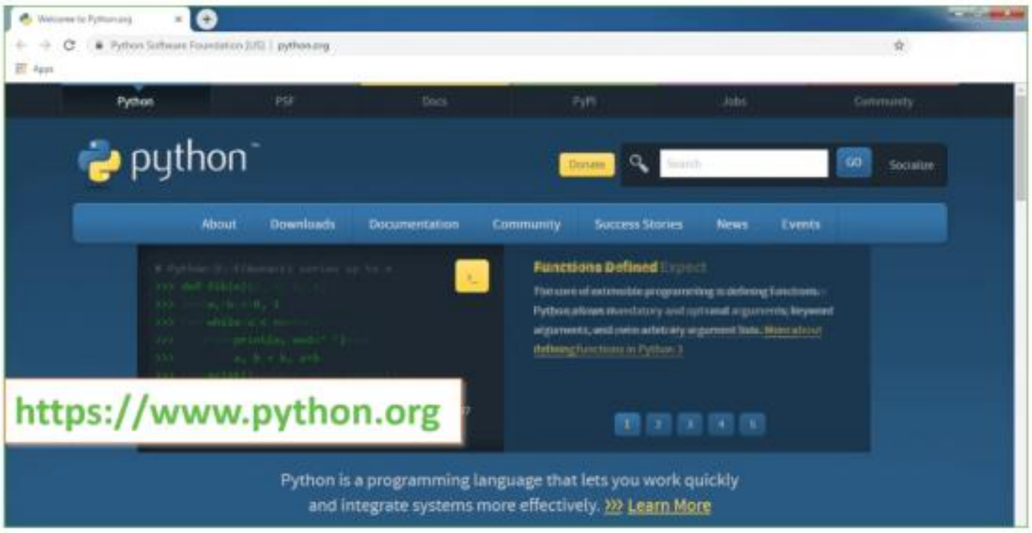
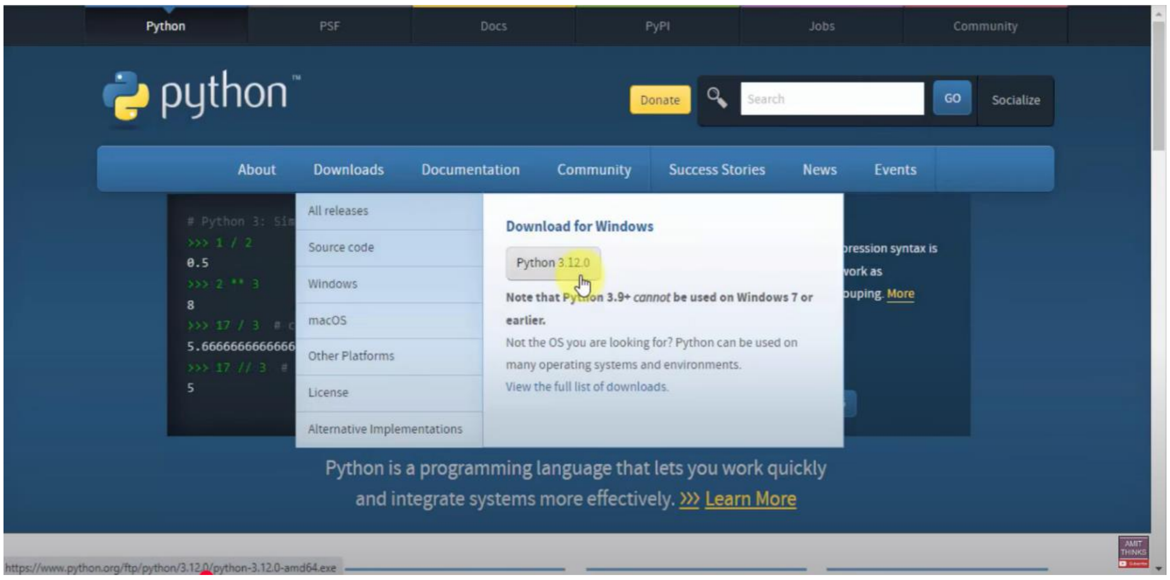


Fig:5.2.1

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

Step2: Click on the Download Tab



Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

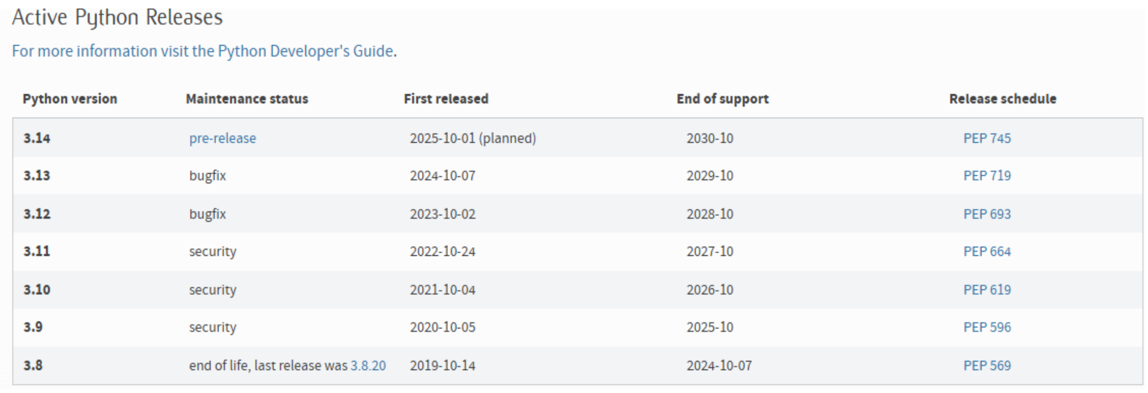
Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

https://www.python.org/ftp/python/3.12.0/python-3.12.0-amd64.exe

Step 3: You can either pick the Download Python for home windows three.12.Zero button in Yellow Color or you can scroll further down and click on on download with respective to their model. Here, we're downloading the most latest python model for home windows three.12.Zero



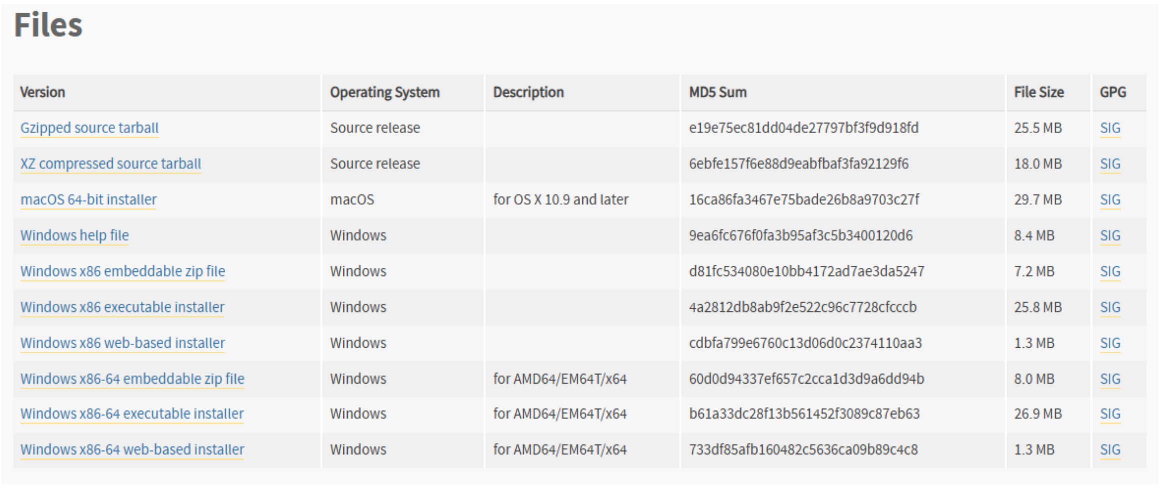
Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.14	pre-release	2025-10-01 (planned)	2030-10	PEP 745
3.13	bugfix	2024-10-07	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	security	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	end of life, last release was 3.8.20	2019-10-14	2024-10-07	PEP 569

Step 4: Scroll down the web page until you discover the Files choice.

Step five: Here you notice a distinct model of python in conjunction with the working gadget



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e19e75ec81dd04de27797bf3f9d918fd	25.5 MB	SIG
XZ compressed source tarball	Source release		6ebfe157f6e88d9eabfbaf3fa92129f6	18.0 MB	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	16ca86fa3467e75bade26b8a9703c27f	29.7 MB	SIG
Windows help file	Windows		9ea6fc676f0fa3b95af3c5b3400120d6	8.4 MB	SIG
Windows x86 embeddable zip file	Windows		d81fc534080e10bb4172ad7ae3da5247	7.2 MB	SIG
Windows x86 executable installer	Windows		4a2812db8ab9f2e522c96c7728cfcccb	25.8 MB	SIG
Windows x86 web-based installer	Windows		cdbfa799e6760c13d06d0c2374110aa3	1.3 MB	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	60d0d94337ef657c2cca1d3d9a6dd94b	8.0 MB	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	b61a33dc28f13b561452f3089c87eb63	26.9 MB	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	733df85afb160482c5636ca09b89c4c8	1.3 MB	SIG

• To download Windows 32-bit python, you may pick out any individual from the three alternatives: Windows x86 embeddable zip report, Windows x86 executable installer or Windows x86 net-primarily based installer.

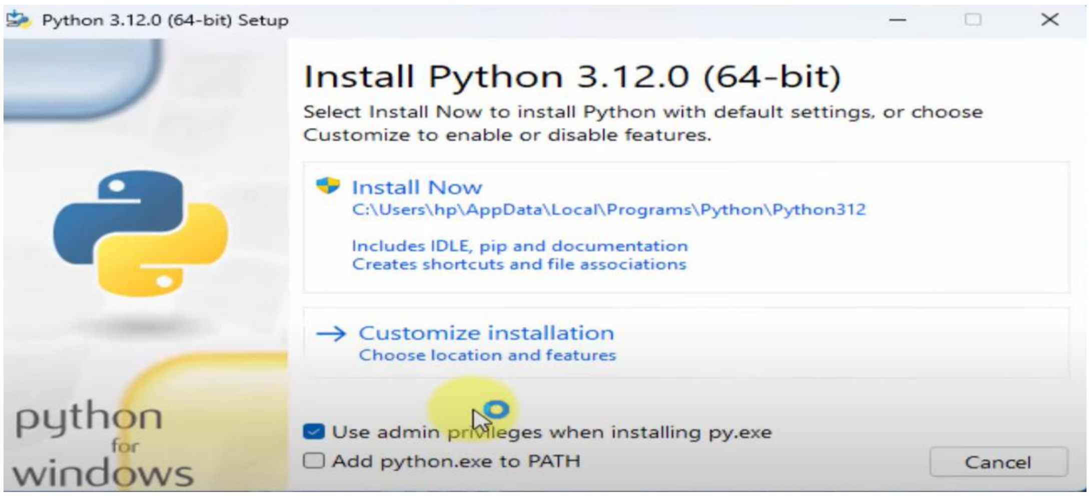
•To down load Windows sixty four-bit python, you can pick any one from the three alternatives: Windows x86-64 embeddable zip report, Windows x86-64 executable installer or Windows x86-sixty four net-based totally installer. Here we can install Windows x86-sixty four net-based totally installer. Here your first part concerning which model of python is to be downloaded is completed. Now we circulate ahead with the second element in putting in python i.E. Installation

Note: To understand the modifications or updates that are made within the model you may click at the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to perform the set up method

Before you click on on Install Now, Make sure to place a tick on Add Python 3.12.0 to PATH



Python 3.12.0 (64-bit) Setup

Install Python 3.12.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

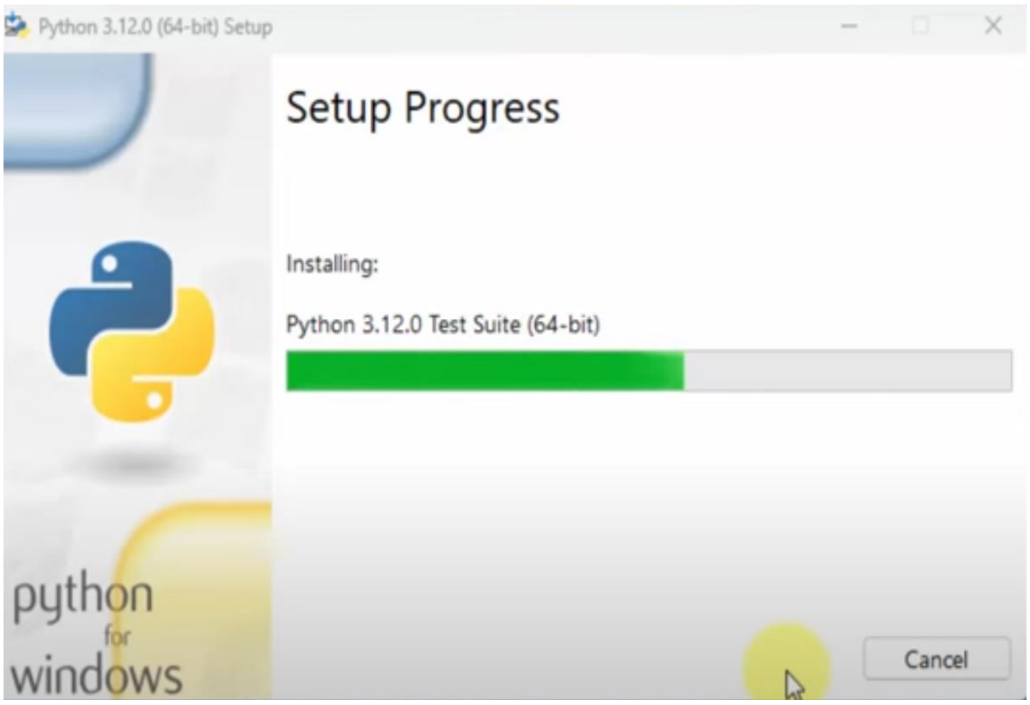
Install Now
C:\Users\hp\AppData\Local\Programs\Python\Python312

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☐ Add python.exe to PATH

Step 2: Click on Install NOW After the installation is successful. Click on Close.



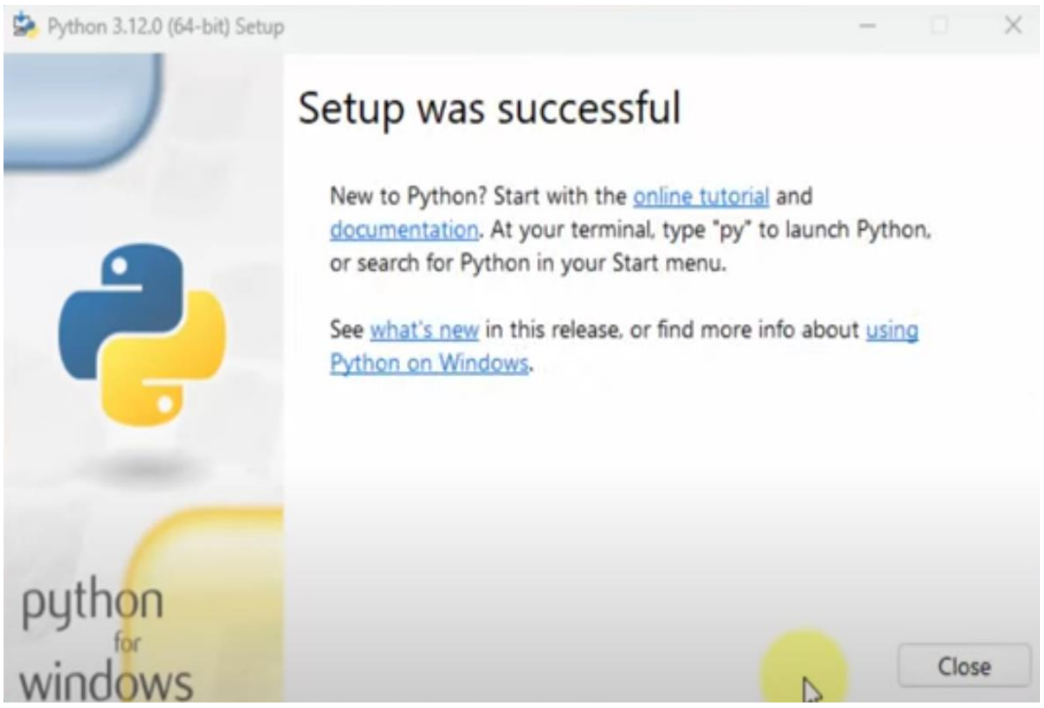
Python 3.12.0 (64-bit) Setup

Setup Progress

Installing:

Python 3.12.0 Test Suite (64-bit)

Fig:5.2.2


Python 3.12.0 (64-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

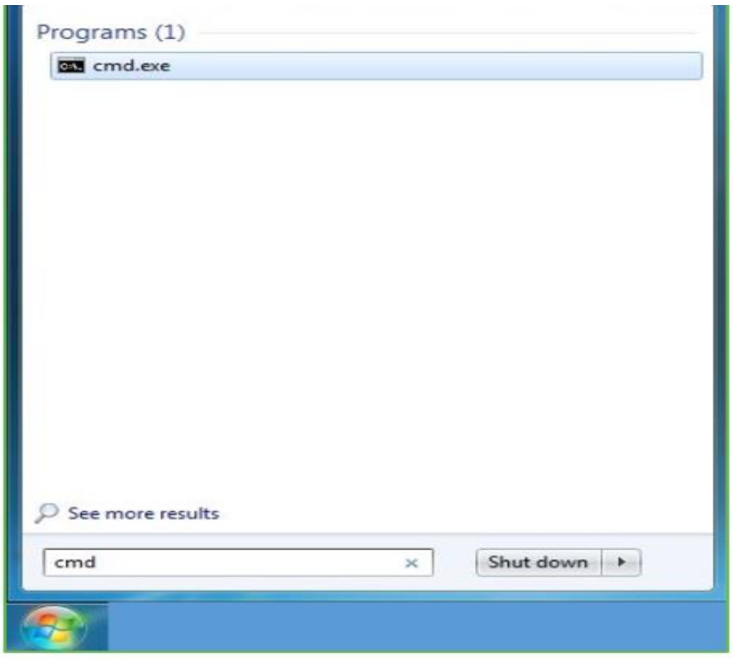
With these above 3 steps on python installation, you have efficaciously and efficaciously hooked up Python. Now is the time to verify the set up.

Note: The set up process may take a few minutes.

Verify the Python Installation

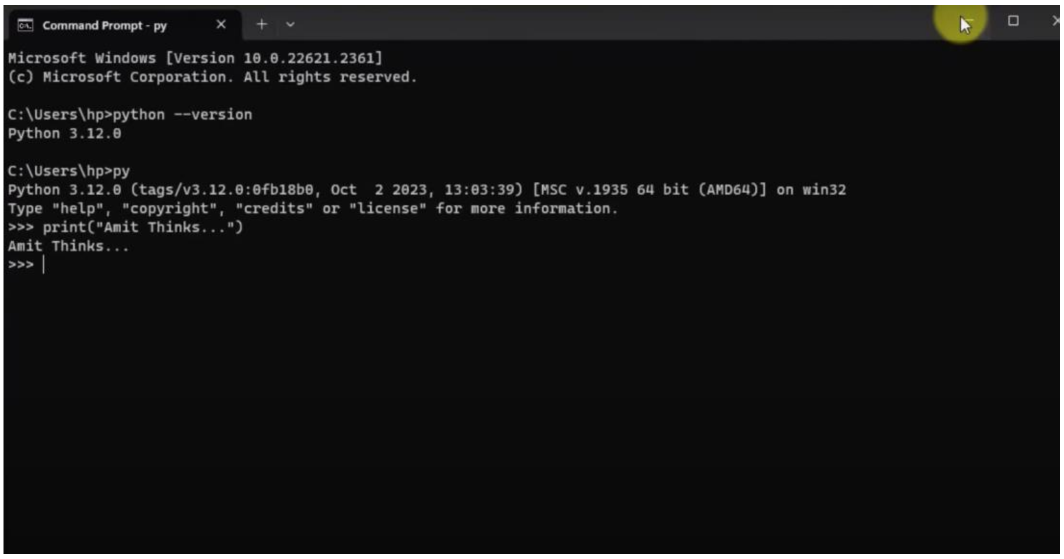
Step 1: Click on Start

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



```
Microsoft Windows [Version 10.0.22621.2361]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>python --version
Python 3.12.0

C:\Users\hp>py
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Amit Thinks...")
Amit Thinks...
>>> |
```

Fig.5.2.3

Step 5: You will get the answer as 3.12.0

Anaconda:

Anaconda is a free and open-source platform that supports both Python and R programming languages. It's widely used for scientific computing tasks such as data science, machine learning applications, large-scale data processing, and predictive analytics.goals to simplify package deal management and deployment. It is advanced and maintained with the aid of Anaconda, Inc.The Anaconda distribution includes tools and libraries for data science and works across Windows, Linux, and macOS. It uses a package manager called conda to handle software installations and updates. Originally part of Anaconda, conda became a standalone open-source tool because of its usefulness beyond just managing Python packages. There is also a small, bootstrap version of Anaconda known as Miniconda, which incorporates only conda, Python, the programs they relies upon on, and a small wide variety of other applications.

1. Download the Installer

•Choose your OS (Windows / Mac / Linux) — primarily based on what you're the use of.

•Click the inexperienced Download button to your device

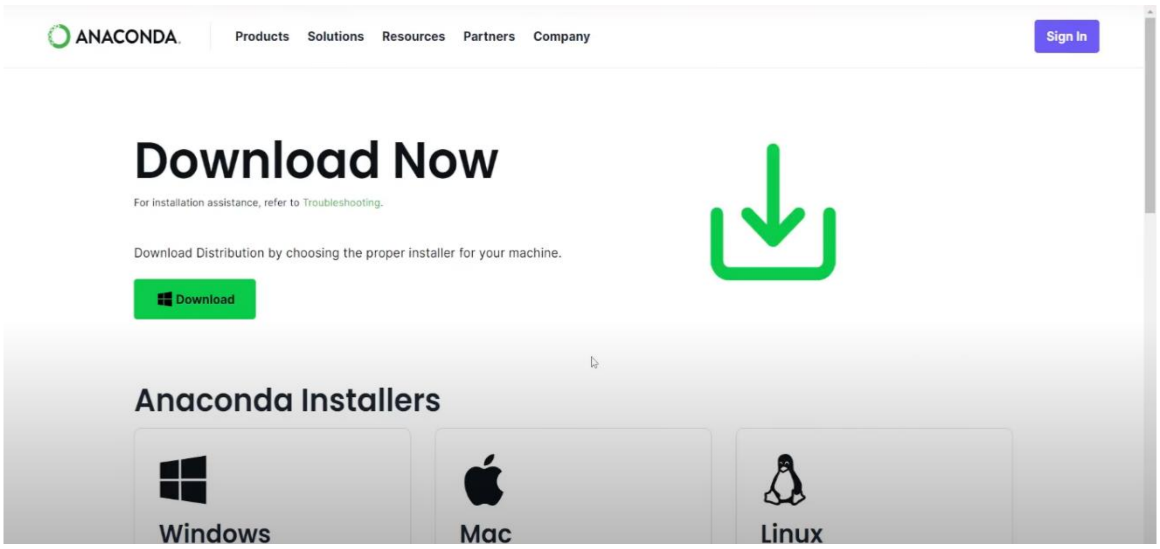
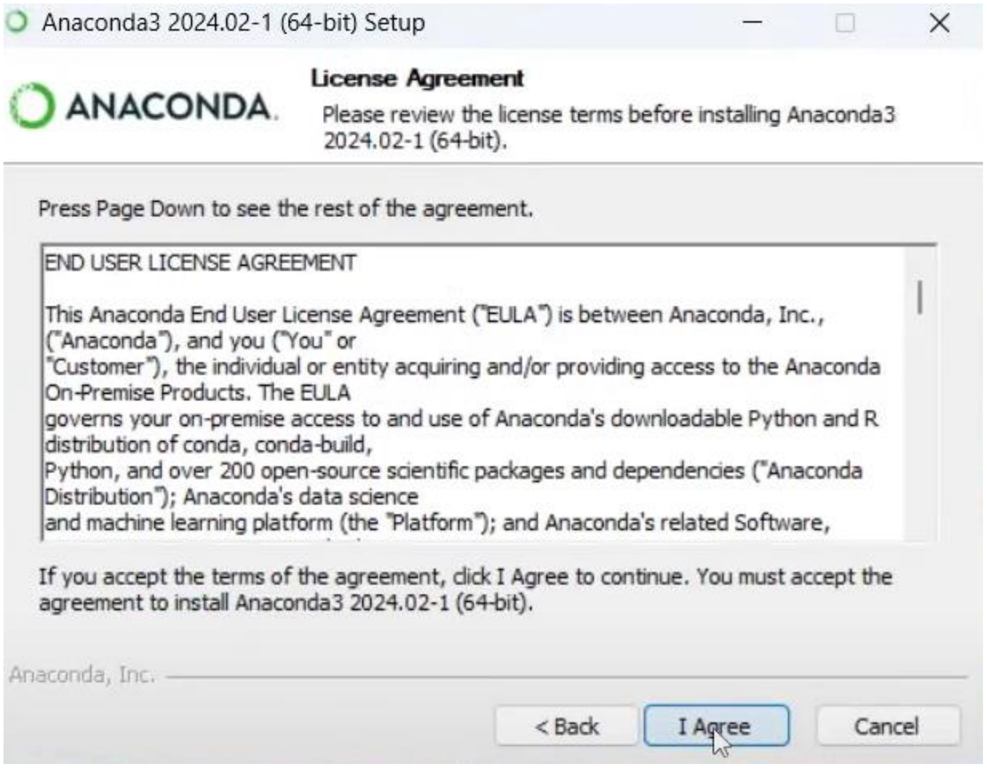

ANACONDA
Products
Solutions
Resources
Partners
Company
Sign In
Download Now
For installation assistance, refer to Troubleshooting.
Download Distribution by choosing the proper installer for your machine.
Download
Anaconda Installers
Windows
Mac
Linux

Fig.5.2.4

2. Run the Installer

- Locate the downloaded record (it'll normally be to your Downloads folder).
- Double-click on the installer to star
- 

Anaconda3 2024.02-1 (64-bit) Setup

ANACONDA.

License Agreement

Please review the license terms before installing Anaconda3 2024.02-1 (64-bit).

Press Page Down to see the rest of the agreement.

END USER LICENSE AGREEMENT

This Anaconda End User License Agreement ("EULA") is between Anaconda, Inc., ("Anaconda"), and you ("You" or "Customer"), the individual or entity acquiring and/or providing access to the Anaconda On-Premise Products. The EULA governs your on-premise access to and use of Anaconda's downloadable Python and R distribution of conda, conda-build, Python, and over 200 open-source scientific packages and dependencies ("Anaconda Distribution"); Anaconda's data science and machine learning platform (the "Platform"); and Anaconda's related Software,

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2024.02-1 (64-bit).

Anaconda, Inc.

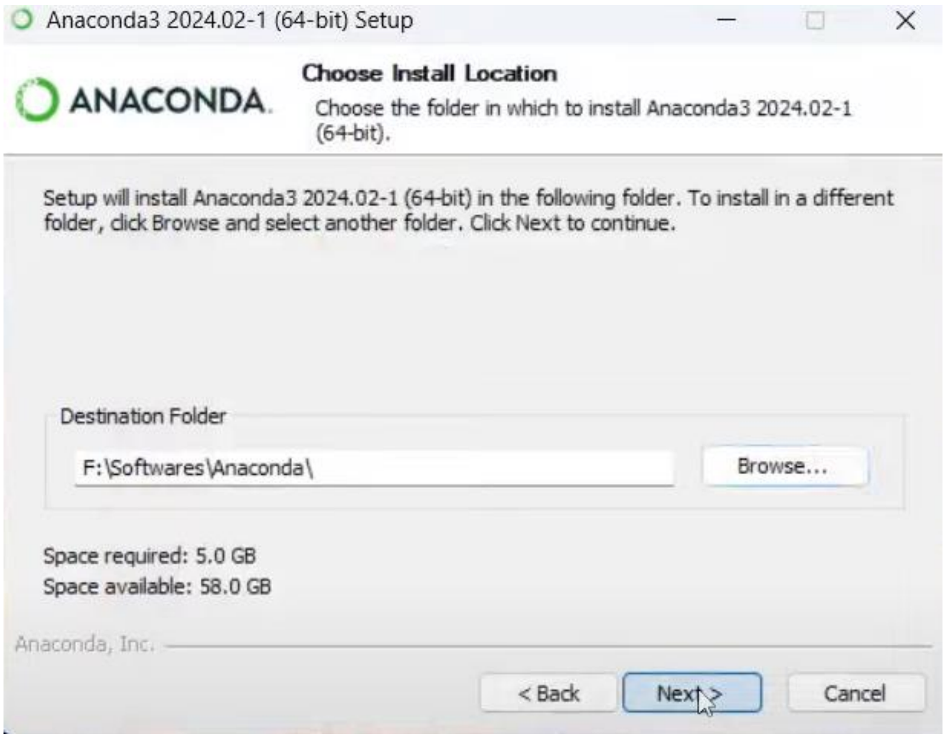
< Back | I Agree | Cancel

Fig. 5.2.5

3. Click Next.

- Accept the License Agreement.

- Choose Install for "Just Me" (encouraged) until you want to install for all users.


Anaconda3 2024.02-1 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2024.02-1 (64-bit).

Setup will install Anaconda3 2024.02-1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: F:\Softwares\Anaconda\

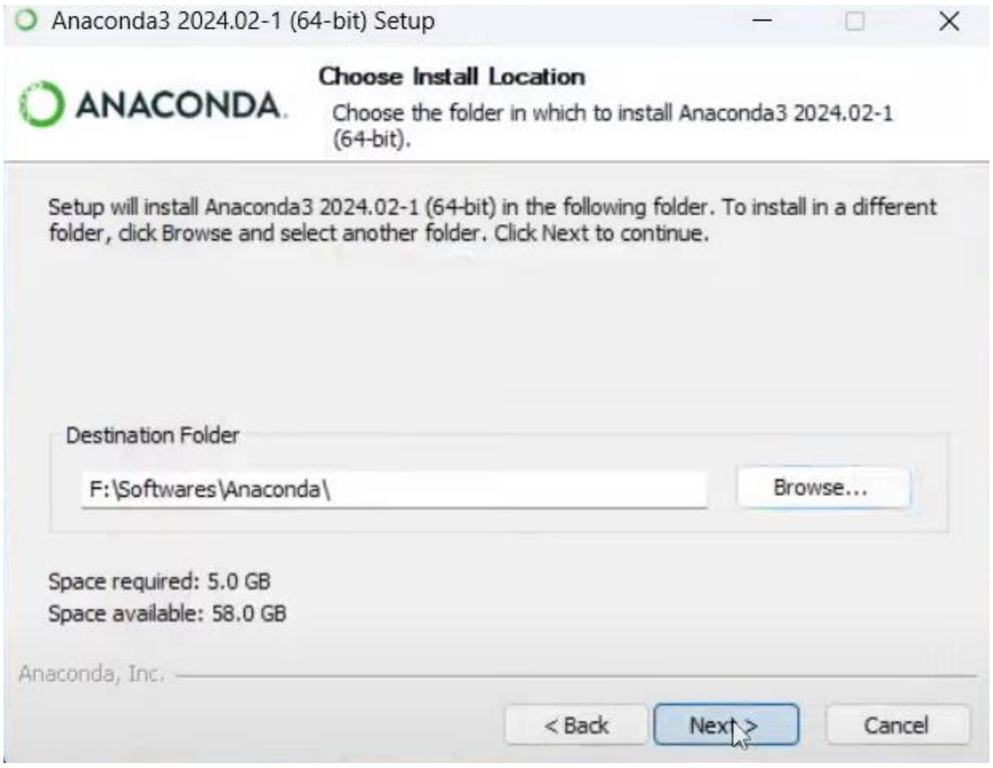
Space required: 5.0 GB
Space available: 58.0 GB

Anaconda, Inc.

4. Advanced Options

- Add Anaconda to PATH:
- It's typically NOT encouraged (leave it unchecked).
- Instead, Anaconda will set up a Start Menu shortcut and its personal terminal (Anaconda Prompt).
- Register Anaconda as your default Python:Register Anaconda as your default Python:

- You can check this if you want Anaconda's Python to be your default.



Choose the folder in which to install Anaconda3 2024.02-1 (64-bit).

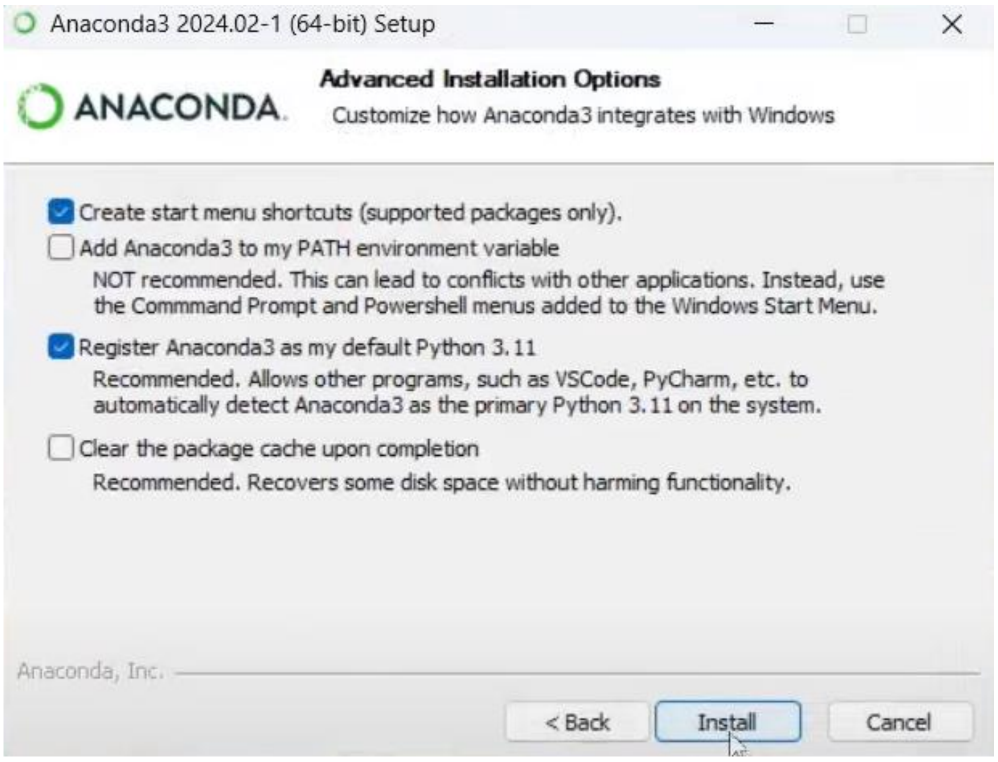
Setup will install Anaconda3 2024.02-1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: F:\Softwares\Anaconda\

Space required: 5.0 GB
Space available: 58.0 GB

5. Finish Installation

- Click Install, wait for it to complete.
- Then Finish.
- You would possibly see an choice to launch Anaconda Navigator or open the Anaconda Prompt — you can do this immediately in case you want!
- Test if Installation became Successful


Anaconda3 2024.02-1 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda3 integrates with Windows

- [✓] Create start menu shortcuts (supported packages only).
- [ ] Add Anaconda3 to my PATH environment variable
  NOT recommended. This can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the Windows Start Menu.
- [✓] Register Anaconda3 as my default Python 3.11
  Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Anaconda3 as the primary Python 3.11 on the system.
- [ ] Clear the package cache upon completion
  Recommended. Recovers some disk space without harming functionality.

Anaconda, Inc.

Fig.5.2.6

6. Test if Installation was Successful

Open Anaconda Navigator

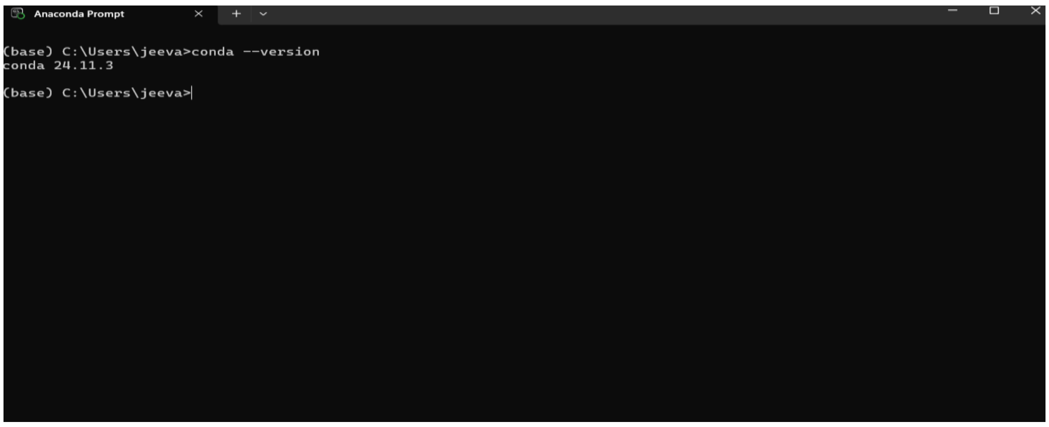
- Open Anaconda Prompt and sort:

css

CopyEdit

conda --model

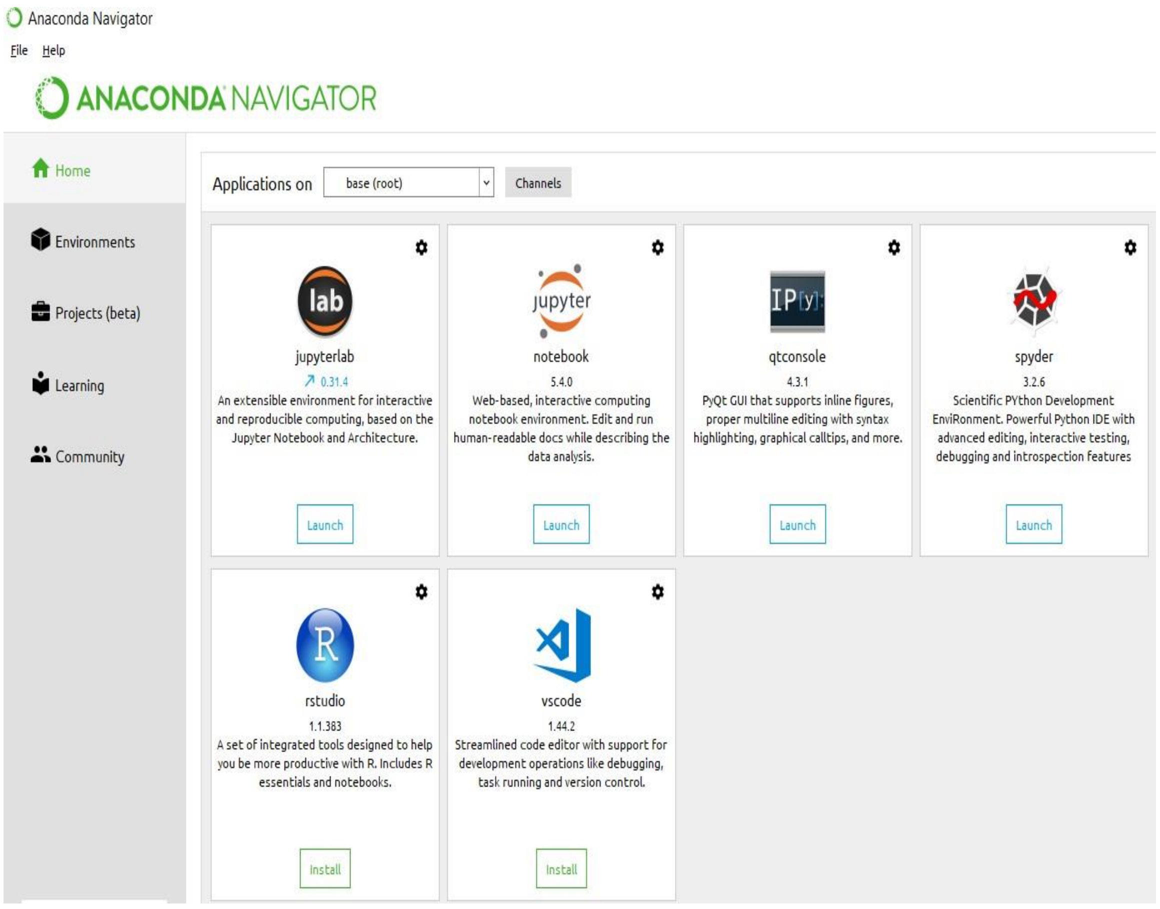
If it returns a model quantity, you're accurate to move



```
(base) C:\Users\jeeva>conda --version
conda 24.11.3

(base) C:\Users\jeeva>
```

Fig.5.2.7



jupyterlab
↗ 0.31.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

rstudio
1.1.383
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

vscode
1.44.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Fig:5.2.8

6. CODING

```
import cv2
import numpy as np
import argparse
import time
import winsound

parser = argparse.ArgumentParser()
parser.add_argument('--webcam', help="True/False", default=False)
parser.add_argument('--play_video', help="True/False", default=False)
parser.add_argument('--image', help="True/False", default=False)
parser.add_argument('--video_path', help="Path of video file", default="videos/fire1.mp4")
parser.add_argument('--image_path', help="Path of image to detect objects",
default="Images/gun.jpg")
parser.add_argument('--verbose', help="To print statements", default=True)
args = parser.parse_args()

# Load YOLO
def load_yolo():
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    classes = []
    with open("obj.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]

    layers_names = net.getLayerNames()
    output_layers = [layers_names[i - 1] for i in net.getUnconnectedOutLayers().flatten()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    return net, classes, colors, output_layers

def load_image(img_path):
    # Image loading
    img = cv2.imread(img_path)
    img = cv2.resize(img, None, fx=0.4, fy=0.4)
    height, width, channels = img.shape
    return img, height, width, channels

def start_webcam():
    cap = cv2.VideoCapture(0)
    return cap

def display_blob(blob):
    '''
    Three images each for RED, GREEN, BLUE channel
    '''
    for b in blob:
        for n, imgb in enumerate(b):
            cv2.imshow(str(n), imgb)

def detect_objects(img, net, outputLayers):
    blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=(320, 320), mean=(0, 0,
0), swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(outputLayers)
    return blob, outputs

def get_box_dimensions(outputs, height, width):
    boxes = []
    confs = []
    class_ids = []
    for output in outputs:
        for detect in output:
            scores = detect[5:]
            class_id = np.argmax(scores)
            conf = scores[class_id]
            if conf > 0.3:
                center_x = int(detect[0] * width)
                center_y = int(detect[1] * height)
                w = int(detect[2] * width)
                h = int(detect[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confs.append(float(conf))
                class_ids.append(class_id)
    return boxes, confs, class_ids

def play_alarm():
    frequency = 2500  # Set Frequency To 2500 Hertz
    duration = 1000  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)

def draw_labels(boxes, confs, colors, class_ids, classes, img):
    indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)
    font = cv2.FONT_HERSHEY_PLAIN
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            color = colors[i]
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            cv2.putText(img, label, (x, y - 5), font, 1, color, 1)

            # Trigger alarm if fire or gun is detected
            if label in ["Fire", "Gun", "Rifle"]:
```

```
            print(f"⚠ Alert: {label} detected! Triggering alarm...")
            play_alarm()

    img = cv2.resize(img, (800, 600))
    cv2.imshow("Image", img)

def image_detect(img_path):
    model, classes, colors, output_layers = load_yolo()
    image, height, width, channels = load_image(img_path)
    blob, outputs = detect_objects(image, model, output_layers)
    boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
    draw_labels(boxes, confs, colors, class_ids, classes, image)
    while True:
        key = cv2.waitKey(1)
        if key == 27:
            break

def webcam_detect():
    model, classes, colors, output_layers = load_yolo()
    cap = start_webcam()
    while True:
        _, frame = cap.read()
        height, width, channels = frame.shape
        blob, outputs = detect_objects(frame, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
        draw_labels(boxes, confs, colors, class_ids, classes, frame)
        key = cv2.waitKey(1)
        if key == 27:
            break
    cap.release()

def start_video(video_path):
    model, classes, colors, output_layers = load_yolo()
    cap = cv2.VideoCapture(video_path)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        height, width, channels = frame.shape
        blob, outputs = detect_objects(frame, model, output_layers)
        boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
        draw_labels(boxes, confs, colors, class_ids, classes, frame)

        key = cv2.waitKey(1)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()

if __name__ == '__main__':
    webcam = args.webcam
    video_play = args.play_video
    image = args.image
    if webcam:
        if args.verbose:
            print('---- Starting Web Cam object detection ----')
        webcam_detect()
    if video_play:
        video_path = args.video_path
        if args.verbose:
            print('Opening ' + video_path + " .... ")
        start_video(video_path)
    if image:
```

```
    image_path = args.image_path
    if args.verbose:
        print("Opening " + image_path + " .... ")
    image_detect(image_path)

cv2.destroyAllWindows()
```

7. SYSTEM TESTING

The purpose of sorting out is to discover errors. Testing is the machine of searching for out each capability fault or weak factor in a piece product. It presents a way to test the capability of components, sub-assemblies, assemblies and/or a completed product It is the process of exercise software program program with the reason of making sure that the Software device meets its necessities and user expectancies and does now not fail in an unacceptable manner. There are numerous forms of tests. Each check type addresses a particular checking out requirement.

TYPES OF TESTING

UNIT FINDING OUT

Unit checking out involves the layout of test instances that validate that the inner software not unusual feel is functioning well, and that utility inputs produce legitimate outputs. All choice branches and inner code go with the flow ought to be confirmed. It is the trying out of individual software devices of the software .It's miles performed after the finishing touch of an man or woman unit earlier than integration. This is a structural trying out, that is predicated on know-how of its manufacturing and is invasive. Unit checks perform fundamental assessments at factor stage and test a particular business method, software, and/or gadget configuration. Unit tests ensure that each unique course of a commercial enterprise procedure performs accurately to the documented specifications and contains truely described inputs and predicted consequences.

INTEGRATION CHECKING OUT

Integration checks are designed to check included software program program components to decide inside the occasion that they run as one program. Testing is event pushed and is more concerned with the primary outcome of monitors or fields. Integration exams display that even though the additives had been in my view pride, as shown with the resource of successfully unit by checking the mixture of additives, we ensure it remains accurate and consistent. Integration testing is primarily aimed at uncovering issues in the interaction between components.

FUNCTIONAL TESTING

Functional testing ensures that a device's functions paintings exactly as mentioned in business and technical requirements, consumer manuals, and documentation. It makes a speciality of the subsequent key aspects:

- Valid Inputs: Verifying that every one correct inputs are well customary.
- Invalid Inputs: Ensuring that invalid or wrong inputs are rejected as predicted.
- Functionality: Checking that every required characteristic operates accurately.
- Outputs: Confirming that an appropriate outputs are produced from given inputs.
- Systems and Procedures: Making sure that any connected systems or approaches are precipitated correctly.

Test making plans for practical checking out revolves round figuring out device necessities, most important capabilities, and crucial test instances. Testers also ensure whole coverage by evaluating workflows, records fields, processes, and sequences. As checking out maintains, new cases may be brought to enhance the effectiveness and thoroughness of the assessments.

SYSTEM TESTING

System trying out is conducted to confirm that the absolutely integrated software machine meets the specified necessities. This segment exams the configuration to make sure predictable and consistent outcomes. For instance, a configuration-based totally device integration test validates that a couple of system components paintings collectively consistent with deliberate workflows and process designs.

WHITE BOX TESTING

White box testing involves examining the internal logic, structure, and code of a software application to ensure that it functions as intended. Testers have whole visibility into the gadget's supply code and structure. This method lets in exam of logic paths, loops, and internal data processing, specializing in areas that external checks can't access. Test instances are developed based totally at the software's internal layout and logic.

Performance Testing

Performance testing is carried out to assess how fast and efficiently a system operates, ensuring it delivers results within a predefined time frame as per the performance requirements. In our current setup, which has a 4GB RAM capacity, the system performs well while streaming live data. However, performance starts to degrade after handling approximately 1300 tweets on average, indicating the system's threshold under continuous data load.

Module Testing

Module testing involves evaluating each individual module separately to identify and correct errors without impacting other parts of the system. When a module fails to perform its intended function, it is refined until the expected results are achieved. This bottom-up approach begins with testing the smallest, most basic modules, then progresses to higher levels. For example, the job classification module is tested on its own by running different jobs and measuring their execution times. The outcomes are then compared with manually calculated results to verify accuracy. Similarly, other modules like resource classification and job scheduling are independently tested, and their outputs help in optimizing process efficiency by reducing waiting time.

.

8. OUTPUT SCREENS

FIRE DETECTION:

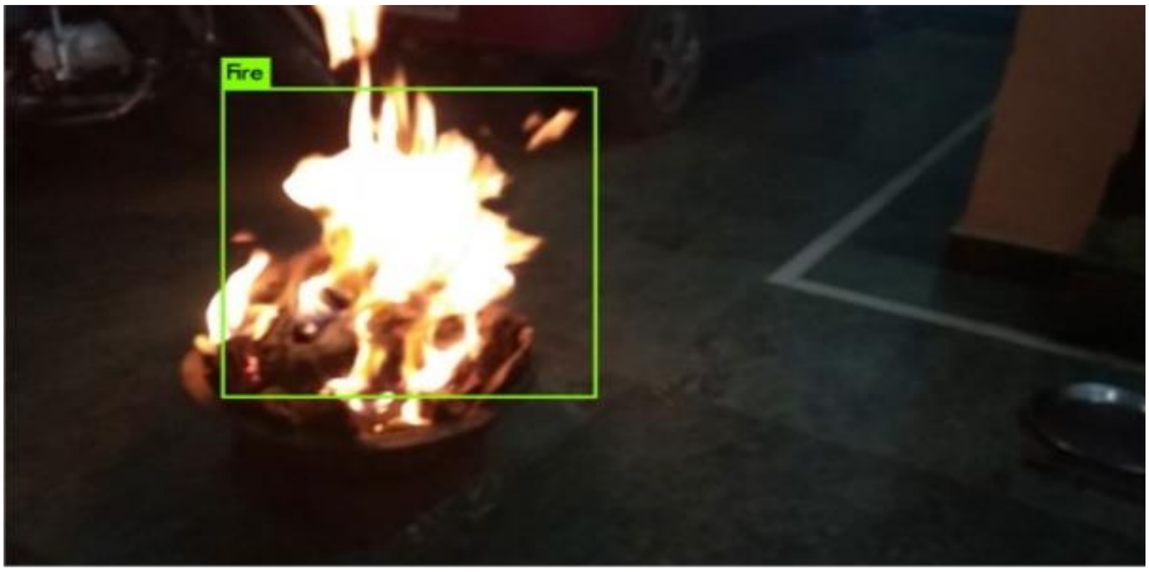


Fig 8.1

The photo above demonstrates the effectiveness of the real-time fireplace detection device in identifying even small-scale flames, which include a matchstick. The device efficiently detects the hearth and marks it with a inexperienced bounding box classified “Fire,” showcasing the version's sensitivity and precision. This proves the robustness of the deep mastering version, which has been trained on a numerous dataset of fireplace snap shots, permitting it to as it should be stumble on fire regardless of its size or depth. Such accuracy is crucial for surveillance structures, in which early detection of even the smallest spark can prevent large disasters. The capacity to perform effectively in an indoor surroundings and detect fireplace from normal objects enhances the reliability and applicability of the machine across numerous settings. This reinforces the utility of the challenge in building intelligent surveillance structures capable of safeguarding lives and assets. The result also demonstrates the model’s robustness in differentiating between history and foreground factors, ensuring that false positives are minimized even in complex visible scenes. Integrating this level of special detection into closed-circuit digicam (CCTV) systems complements proactive safety measures via enabling early signals and instant responses.

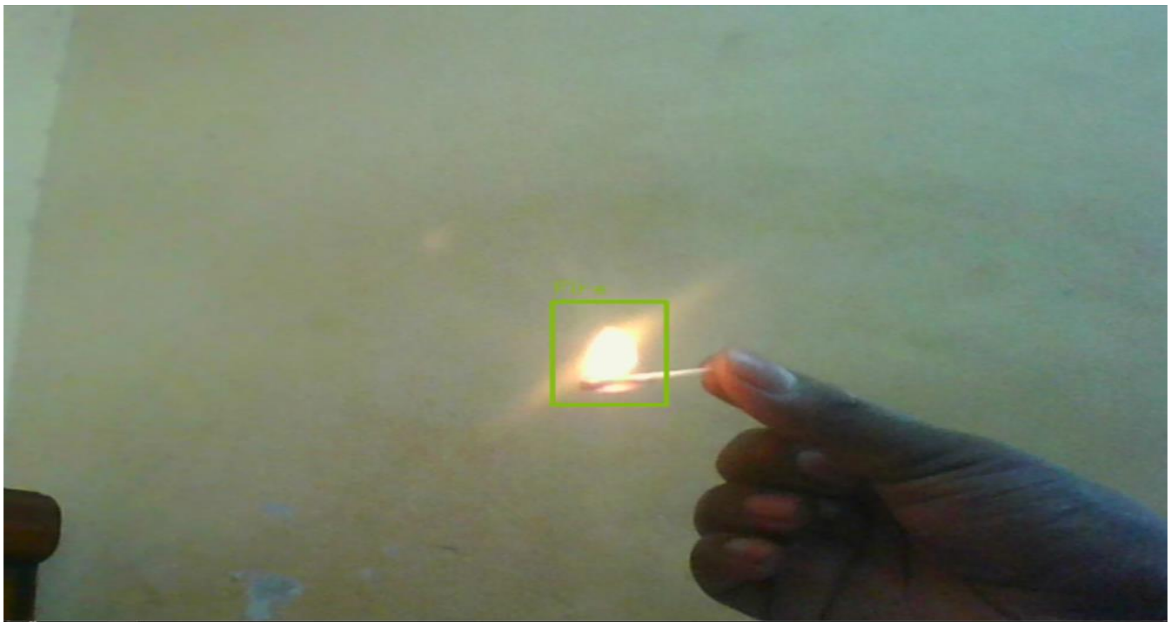


Fig 8.2

The system's flexibility and effectiveness make it a practical and scalable solution for fire prevention and management.. When combined with gun detection capabilities, the system represents a comprehensive safety surveillance solution, contributing significantly to the protection of both life and property in diverse environments.

GUN DETECTION:

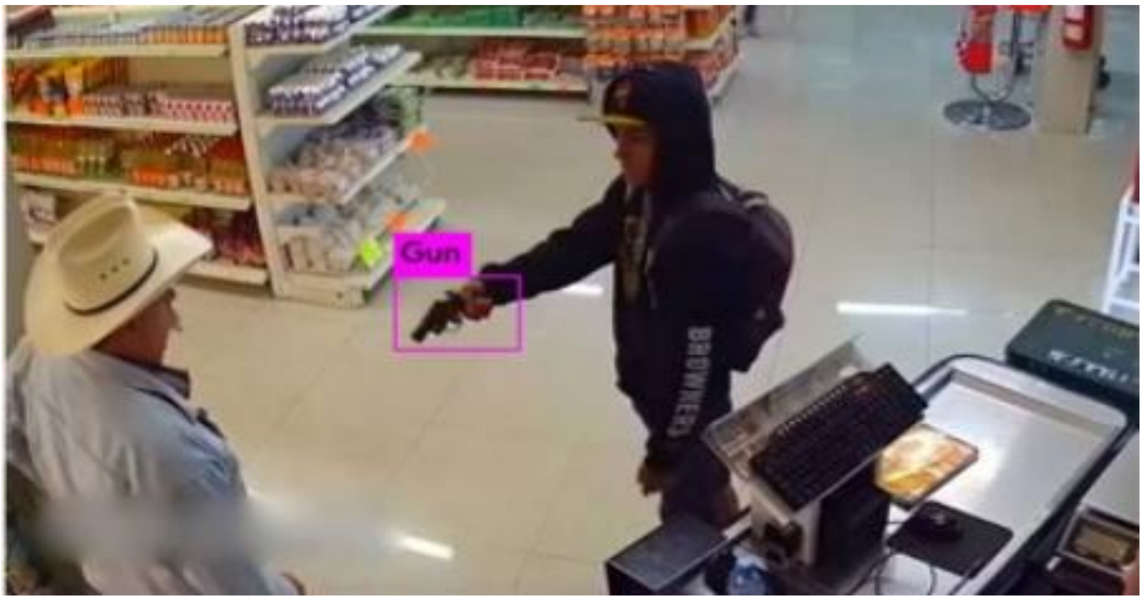


Fig 8.3

The system’s flexibility and effectiveness make it a realistic and scalable solution for fire prevention and management.. When combined with gun detection skills, the system represents a complete safety surveillance solution, contributing drastically to the safety of both lifestyles and belongings in diverse environments.The above snap shots illustrate the a hit implementation of real-time gun detection the use of deep learning inside a surveillance surroundings. In each eventualities, the version has appropriately recognized the presence of a gun and marked it with a really categorised bounding container in red, indicating “Gun.” These frames are captured from protection digital camera footage, where individuals are engaged in potentially dangerous or criminal interest, consisting of tried theft or assault. The effectiveness of the version in detecting firearms even under various lighting fixtures situations, angles, and partial occlusion highlights its robustness and adaptableness.
his machine, when included into closed-circuit cameras (CCTV), can play a essential role in improving safety throughout public places like retail stores, banks, and different high-chance regions. By triggering immediate alerts upon detection of a firearm, the model permits government or safety employees to respond swiftly, doubtlessly stopping damage and saving lives. Furthermore, its potential to analyze frames in real-time guarantees continuous monitoring and reduces reliance on manual statement. This smart surveillance answer, when blended with the earlier confirmed fireplace detection model, affords a complete protection framework capable of addressing each environmental and human threats efficiently.Addressing each environmental and human threats correctly.

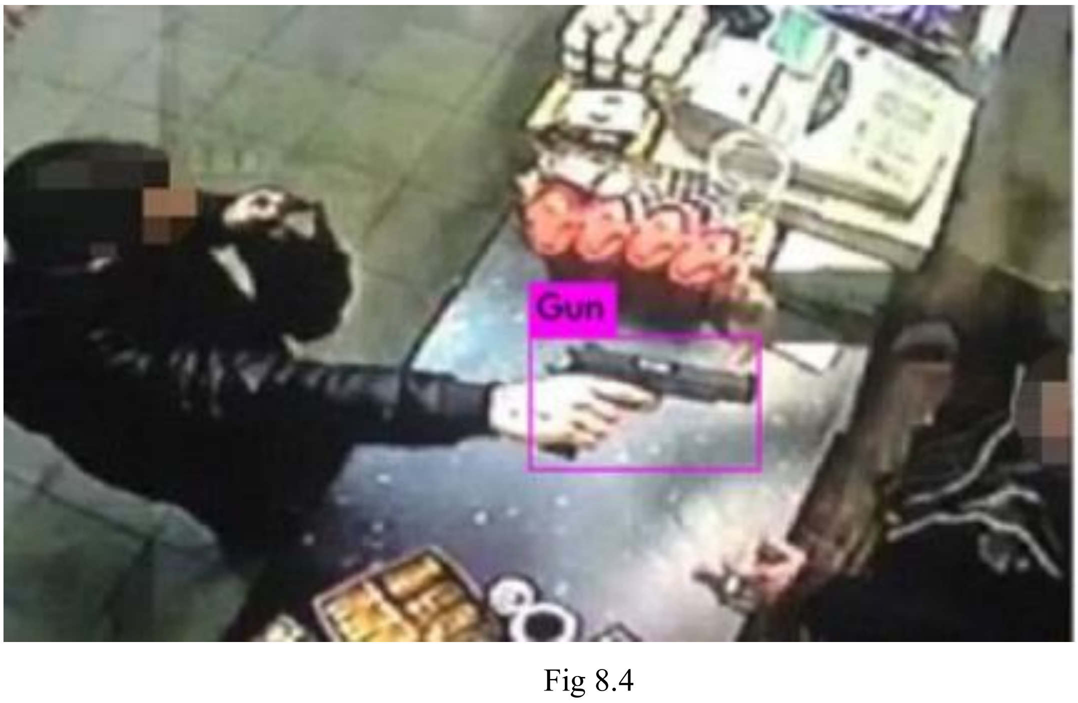


Fig 8.4

Gun detection in actual-time is a essential function in cutting-edge surveillance systems, particularly in public regions liable to safety risks such as shops, banks, malls, or transportation hubs. Early detection of a firearm can trigger computerized alerts to law enforcement or on-site protection teams, helping to de-improve probably violent conditions earlier than they lead to harm. Additionally, the presence of such shrewd surveillance era can serve as a deterrent to crook pastime, enhancing average safety.

When combined with hearth detection, as shown in preceding pictures, this gadget forms a dual-chance tracking answer able to responding to both guy-made and environmental dangers. This holistic method significantly enhances situational consciousness and allows brief, informed selection-making in emergency situations. Overall, integrating actual-time gun and hearth detection the use of deep getting to know into CCTV structures marks a tremendous development in clever surveillance and public protection infrastructure.

RIFILE DETECTION:

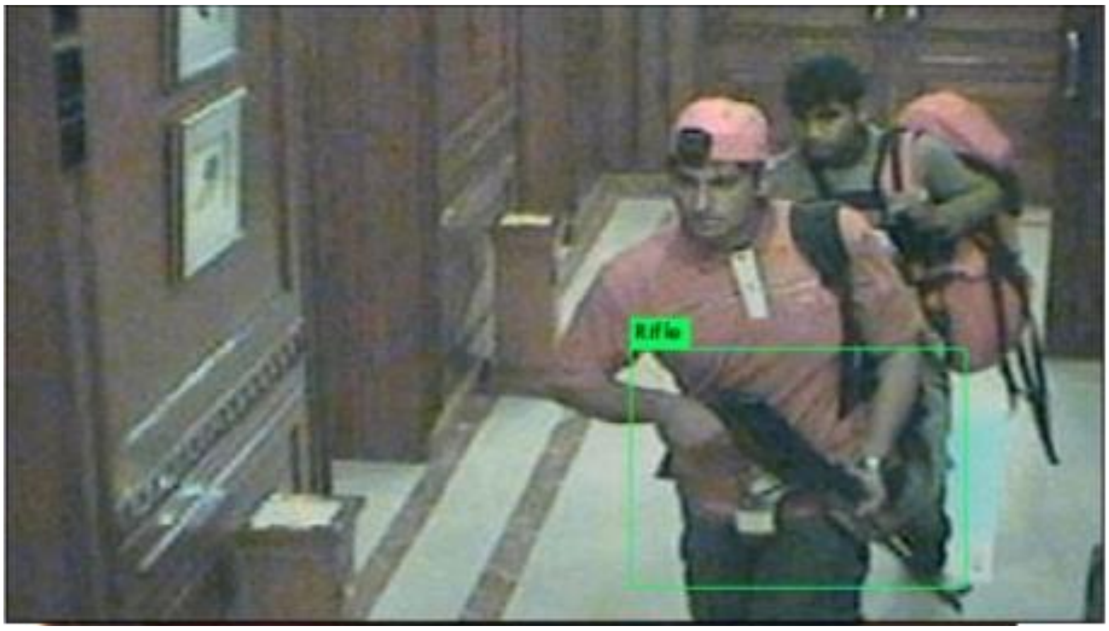


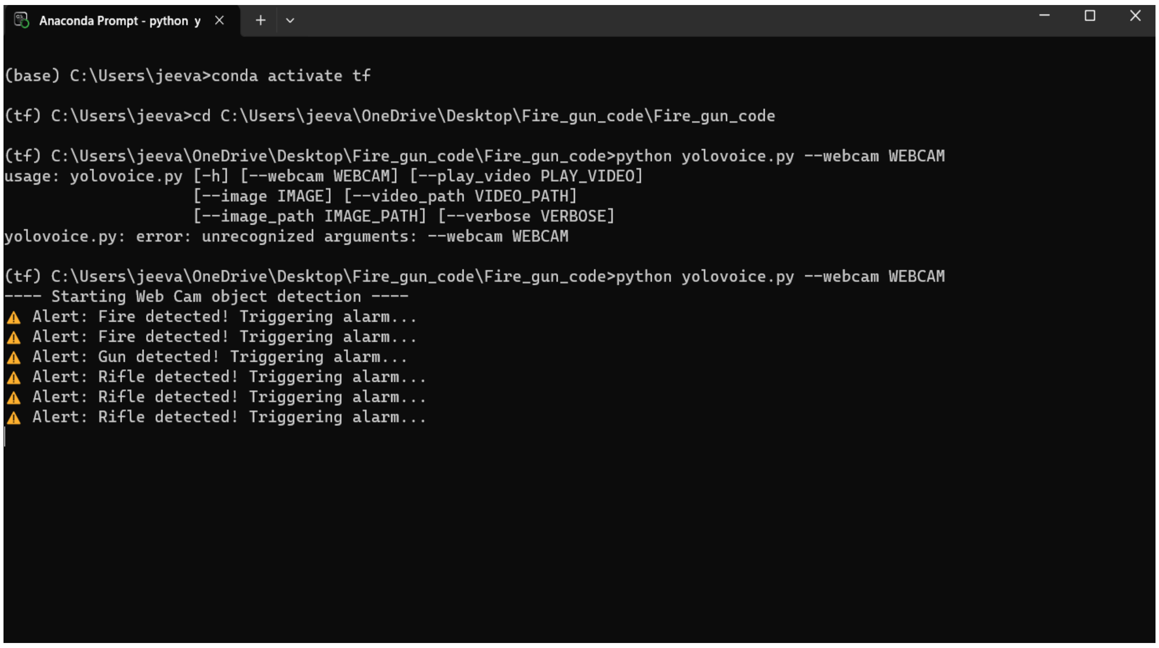
Fig 8.5

This photograph illustrates a actual-time rifle detection state of affairs captured thru CCTV pictures and processed by a deep studying surveillance device. The gadget correctly identifies a person sporting a rifle and marks it with a bounding container classified “Rifle” in green. The individuals appear like moving thru a corridor, in all likelihood within a public or confined facility, which makes the want for automatic weapon detection even extra vital.

In excessive-stakes environments—together with airports, resorts, colleges, or government homes—the potential to detect rifles and other long-range guns in actual time can play a pivotal position in stopping large-scale violence. Unlike handguns, rifles are often more hid or carried in non-conventional postures, making manual tracking unreliable and slow. The use of advanced item detection fashions skilled on diverse datasets enables conquer these barriers with the aid of mastering visible functions and styles related to weapons, no matter lighting, pose, or partial occlusion.

This detection supports instantaneous hazard evaluation and can be connected with alerting systems to notify government with out human postpone. As a part of an incorporated clever surveillance answer, combining gun, rifle, and fireplace detection ensures a complete and wise protection machine capable of responding to more than one kinds of threats in actual time—thereby enhancing safety and situational control across sensitive and crowded areas.

ALERT TRIGGERING:



```
(base) C:\Users\jeeva>conda activate tf

(tf) C:\Users\jeeva>cd C:\Users\jeeva\OneDrive\Desktop\Fire_gun_code\Fire_gun_code

(tf) C:\Users\jeeva\OneDrive\Desktop\Fire_gun_code\Fire_gun_code>python yolovoice.py --webcam WEBCAM
usage: yolovoice.py [-h] [--webcam WEBCAM] [--play_video PLAY_VIDEO]
                    [--image IMAGE] [--video_path VIDEO_PATH]
                    [--image_path IMAGE_PATH] [--verbose VERBOSE]
yolovoice.py: error: unrecognized arguments: --webcam WEBCAM

(tf) C:\Users\jeeva\OneDrive\Desktop\Fire_gun_code\Fire_gun_code>python yolovoice.py --webcam WEBCAM
---- Starting Web Cam object detection ----
⚠ Alert: Fire detected! Triggering alarm...
⚠ Alert: Fire detected! Triggering alarm...
⚠ Alert: Gun detected! Triggering alarm...
⚠ Alert: Rifle detected! Triggering alarm...
⚠ Alert: Rifle detected! Triggering alarm...
⚠ Alert: Rifle detected! Triggering alarm...
```

Fig 8.6

This screenshot captures the real-time execution of a hearth and weapon detection device using YOLO (You Only Look Once) item detection in Python, operated thru Anaconda Prompt. The terminal indicates the technique where the script yolovoice.Py is being run with the --webcam flag to allow live detection from the gadget's digicam.

Initially, there's a controversy mistakes due to a likely case-sensitivity or wrong utilization within the command --webcam WEBCAM. After correcting it (most possibly just the usage of --webcam or every other valid parameter), the device starts offevolved efficiently and starts real-time surveillance.

The output displays more than one caution messages:

- ◈ Alert: Fire detected! Triggering alarm...
- ◈ Alert: Gun detected! Triggering alarm...
- ◈ Alert: Rifle detected! Triggering alarm...

These indicators imply that the gadget has effectively detected a couple of kinds of threats thru the webcam feed. Each detection triggers an alert mechanism, showcasing that the detection good judgment and the corresponding response gadget (like sounding an alarm or sending a notification) are functioning as meant.

This demo proves the effectiveness of your wise surveillance system, which combines deep studying with actual-time video processing to discover potentially unsafe objects like fire, guns, and rifles.

9. CONCLUSION

In conclusion, **"Real-Time Fire and Gun Detection Using Deep Learning for Enhanced Surveillance,"** presents a practical and efficient solution using transfer learning techniques with the InceptionResNetV2 architecture. By training the model on carefully curated datasets of fire, smoke, and firearms, the system effectively detects both fire incidents and gun threats in real time from CCTV footage.

The implementation is cost-effective, easy to deploy, and scalable across a wide range of environments, from forests to urban areas. The ability to detect threats early can significantly reduce environmental damage, protect wildlife, and enhance public safety by supporting faster emergency response.

The proposed system serves as a strong foundation for future improvements. Upcoming developments may include the integration of real-time GPS-based alert systems, model training with more diverse datasets, and the use of satellite imagery to expand detection over larger areas.

This project demonstrates the potential of deep learning in transforming traditional surveillance into intelligent safety systems, making it a vital tool for disaster prevention and public security enhancement.

10. FUTURE ENHANCEMENT

In future advancements, image processing techniques such as **image segmentation** can be employed to enhance the accuracy of fire and smoke detection systems. Through segmentation, each frame of a video can be analyzed individually, where **masking techniques** will be used to identify and isolate regions indicating the presence of fire or smoke. This enables the system to make real-time predictions **frame by frame**, displaying the output based on predefined fire status levels—ranging from no fire to severe fire conditions. Such detailed analysis allows for a more responsive and reliable monitoring system, especially in critical environments prone to fire hazards.

Moreover, the system can be further developed to incorporate automated **alert mechanisms**. Once fire or smoke is detected, the system can immediately **send notifications via email** or other communication channels to the respective **emergency response departments**. These alerts can include vital information such as the **exact location coordinates**, time of detection, and severity level, ensuring a swift response. Incorporating GPS data can greatly enhance the accuracy of alerts. Implementing these features not only aids in early fire prevention and control but also contributes to reducing environmental damage, preserving natural habitats, and maintaining ecological balance.

11. BIBLIOGRAPHY

[1] M. Byra, Discriminant analysis of neural style representations for breast lesion classification in ultrasound, Biocyber. Biomed. Engin. 38(3) (2018) 684–690.

[2] J. Z. Cheng, D. Ni, Y.H. Chou, J. Qin, C.M. Tiu, Y.C. Chang, C.S. Huang, D. Shen and C.M. Chen, Computeraided diagnosis with deep learning architecture: Applications to breast lesions in US images and pulmonary ndules in CT scans, Scientific Rep. 6(1) (2016) 1–13.

[3] F. Cui, Deployment and integration of smart sensors with IoT devices detecting fire disasters in huge environment, Computer Commun. 150 (2020) 818–827.

[4] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recogn. (2016) 770–778.

[5] P. Jain, S.C.P. Coogan, S.G. Subramanian, M. Crowley, S. Taylor and M.D. Flannigan, A review of machine learning applications in wildfire science and management, In Environmental Reviews, 28(4) (2020) 478–505.

[6] Z. Jiao, Y. Zhang, L. Mu, J. Xin, S. Jiao, H. Liu and D. Liu, A YOLOv3-based Learning Strategy for Real-time UAV-based Fire Detection, Proc. 32nd Chinese Control Decision Conf., IEEE, (2020) 4963–4967.

[7] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu and D. Liu, A Deep learning based fire detection approach using uav and yolov3, 1st Int. Conf. Indust. Artif. Intell. (2019) 1–5.

[8] A. Krizhevsky, I. Sutskever and G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM, 60(6) (2017) 84–90. [9] S.B. Kukuk and Z.H. Kilimci, Comprehensive analysis of fire detection using deep learning models and conventional machine learning algorithms, Int. J. Comput. Experimental Sci. Engin. 7(2) (2021) 84–94.