A Project Report

on

SUSPICIOUS HUMAN ACTIVITY RECOGNITION FROM SURVEILLANCE VIDEOS USING DEEP LEARNING

Submitted in partial fulfillment to Jawaharlal Nehru Technological University

of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science and Engineering

by

M B VENKATA DINESH KUMAR REDDY	**(21FH1A0596)**
V.MAHESH	**(21FH1A05C8)**
S.VIJAY WINSTON	**(22FH5A0509)**
M.LAKSHMI KANTH	**(22FH5A0506)**

Under the esteemed guidance of

Mr.A.EMMANUEL RAJU, M.Tech.,
Assistant Professor.

Department of Computer Science & Engineering

Accredited by NBA

Dr. K.V. SUBBA REDDY INSTITUTE OF TECHNOLOGY

(Autonomous)

(Affiliated to JNTUA, Anantapur, Accredited by NAAC A+ & Approved by AICTE)

Kurnool - 518 218

(2024-25)

Dr. K.V. SUBBA REDDY INSTITUTE OF TECHNOLOGY

(Autonomous)

(Affiliated to JNTUA, Anantapur & Approved by AICTE)

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **"Suspicious human activity recognition from surveillance videos using deep learning"** being submitted by **M B VENKATA DINESH KUMAR REDDY (21FH1A0596),V.MAHESH (21FH1A05C8),S.VIJAY WINSTON (22FH5A0509), M.LAKSHMI KANTH (22FH5A0506)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** to the **Dr. K.V. Subba Reddy Institute of Technology**,Affiliated to JNTU Anantapur is a record of bonafide work carried out during the year 2024-
2025.

The results presented in this thesis have been verified and found to be satisfactory. The results embodied in this thesis report have not been submitted to any other University for the award of any other degree or diploma.

PROJECT GUIDE

Mr.A.EMMANUEL RAJU, M.Tech.
Assistant Professor

HOD

Dr. C. MD. Gulzar, M.Tech., Ph.D,
Associate Professor

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

ANANTAPUR

CERTIFICATE OF EXAMINERS

Dissertation work entitled "**Suspicious human activity recognition from surveillance videos using deep learning**" submitted by **M B VENKATA DINESH KUMAR REDDY (21FH1A0596), V.MAHESH (21FH1A05C8), S.VIJAY WINSTON (22FH5A0509) ,M.LAKSHMI KANTH (22FH5A0506)** is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiners:

1.

2.

ACKNOWLEDGEMENT

It is our privilege and pleasure to express our profound sense of respect, gratitude and indebtedness to our guide, **Mr.A.EMMANUEL RAJU, M.Tech.,** Assistant Professor, Department of Computer Science and Engineering, **Dr. K.V. Subba Reddy Institute of Technology**, for his indefatigable inspiration, guidance, cogent discussion and encouragement throughout this dissertation work.

We express our sincere gratitude to **Dr. C. MD. Gulzar,** M.Tech., Ph.D, Associate Professor & HOD, Department of CSE, **Dr. K.V. Subba Reddy Institute of Technology**, for his precious suggestions, motivation and co-operation for the successful completion of this project work.

We extend our sincere thanks to **Dr. J. Kanna Kumar,** Principal, **Dr. K.V. Subba Reddy Institute of Technology**, Dupadu(V), Kurnool, for his encouragement and constant help.

This Acknowledgement will be incomplete without mentioning our sincere gratefulness to our honorable Chairman **Dr. K.V. Subba Reddy Garu,** who has been observed posing valiance in abundance, forwarding our individuality to acknowledge our project work tendentiously.

Last but not least, we wish to acknowledge **our friends, family members and colleagues** for giving moral strength and helping us to complete this dissertation.

PROJECT TEAM:

M B VENKATA DINESH KUMAR REDDY	(21FH1A0596)
V.MAHESH	(21FH1A05C8)
S.VIJAY WINSTON	(22FH5A0509)
M.LAKSHMI KANTH	(22FH5A0506)

INDEX

ABSTRACT

Suspicious Human activity recognition (SHAR) is crucial for improving surveillance and security systems by recognizing and reducing possible hazards in different situations. Despite the abundance of research on the subject of SHAR, current methods frequently need to be revised with restricted levels of precision and efficiency. We aim to address the problem of inaccurate and inefficient activity recognition in surveillance systems through rigorous data collection, preparation, and model training. By leveraging Convolutional Neural Networks (CNNs) and deep learning architectures, including our time-distributed CNN and Conv3D models, it achieved improved accuracy rates of 90.14% and 88.23%.

The exponential growth of CCTV installations in public and private spaces has revolutionized surveillance practices but also introduced challenges in monitoring and analysis. Traditionally, security personnel manually review surveillance footage, which is labor-intensive, time-consuming, and prone to errors caused by fatigue and oversight. To address these limitations, this project proposes an automated system powered by Machine Learning (ML) and Deep Learning (DL) technologies to detect suspicious activities in real-time, significantly enhancing efficiency and accuracy. The system utilizes Convolutional Neural Networks (CNNs), a proven tool in computer vision, to analyze frames extracted from video feeds. These frames are preprocessed for noise reduction and quality optimization before feeding into the CNN model. The trained model identifies unusual patterns based on spatial and temporal dynamics, classifying activities as either “Normal” or “Suspicious.” Suspicious activities such as theft, aggression, or unauthorized movements trigger instant alerts, enabling security personnel to respond swiftly. The system also securely stores processed data for later analysis, supporting forensic investigations and improving threat prevention mechanisms.

Designed for scalability, the system can operate across diverse environments, including airports, malls, corporate offices, and residential areas. By automating labor-intensive tasks, it reduces human workload while improving detection reliability

and response times. The integration of multimodal analysis, combining video and audio input, further refines its accuracy. Predictive modeling techniques offer proactive threat identification, and edge computing ensures decentralized, real-time data processing directly at surveillance sites. With its advanced capabilities and adaptive design, this system positions itself as a vital tool in modern security frameworks, offering improved monitoring, reduced operational costs, and enhanced safety across large-scale installations.

Keywords : Suspicious human activity recognition (SHAR), deep learning, convolutional neural
network, multimedia data.

List of Tables

List of Figures

CHAPTER I

CHAPTER-01

INTRODUCTION

The widespread incorporation of many applications in modern society has significantly transformed many aspects of our lives, with visual systems emerging as essential instruments. One important area of study in this field is the detection of suspicious human behaviour using video surveillance, which involves classifying behaviours as either normal or abnormal [1]. The increasing frequency of disruptive incidents in public areas globally, ranging from banks to airports, highlights the urgent requirement for efficient security measures [2]. As a result, surveillance systems, mostly dependent on CCTV cameras, have grown quite common, producing large quantities of video data for examination. Nevertheless, the labour-intensive nature of manual monitoring makes it unfeasible, thus necessitating the development of automated detection systems [3].

Researchers are using breakthroughs in machine learning, artificial intelligence, and deep learning to improve surveillance systems. Their goal is to proactively identify and categorize suspicious activity [4]. The objective of this project is to implement deep learning models for the purpose of identifying and categorizing six primary activities: Running, Punching, Falling, Snatching, Kicking, and Shooting. This will enhance security measures and allow for prompt intervention [5]. Deep learning architectures, specifically CNNs, have emerged as strong tools for extracting essential capabilities from video data aimed toward facilitating efficient detection [4]. Yakkali et al. [7] suggested the utilization of digital image and video processing techniques to monitor item movement. They underscore the importance of training deep temporal models for accurate activity identification, as emphasized by Ma et al. [8]. Their emphasis lies in highlighting the importance of Recurrent Neural Networks (RNNs), mainly long short-term memory (LSTM) models, in comprehending the progression of activities and minimizing classification errors. Moreover, improvements in video representation learning, in particular in long term Temporal Convolutions (LTC), demonstrate promise in improving activity recognition [9]. However, there persists a need to enlarge the scope of detectable activities and improve overall performance metrics.

1.1 Purpose of the project

The purpose of the project, *Suspicious Human Activity Recognition From Surveillance Videos Using Deep Learning*, is to proactively enhance public safety by identifying and categorizing human activities captured in surveillance footage as either normal or abnormal. This initiative aims to improve the efficiency and accuracy of surveillance systems, which traditionally rely on manual monitoring and are prone to delays and errors.

Key objectives of the project include:

Implementing deep learning models like Convectional Neural Networks (CNN's) and advanced algorithms such as YOLOv5 for detecting and classifying activities.

Focusing on specific human actions, including **running, punching, falling, snatching, kicking, and shooting**, to distinguish suspicious activities in real-time.

Enabling prompt interventions by providing real-time alerts and detailed classifications of detected behaviors.

This project seeks to strengthen security systems across diverse environments such as airports, banks, and public spaces, addressing the limitations of traditional systems while contributing to technological advancements in surveillance.

1.2 Existing System

The **existing system** for Suspicious Human Activity Recognition (SHAR) relies on deep learning models, especially Convectional **Neural Networks (CNN's)**, for detecting potentially suspicious behaviors in surveillance scenarios. These systems extract significant features from video data to enhance security and monitoring accuracy. However, they come with specific challenges and limitations:

Disadvantages:

Limited Adaptability: Current models often struggle to perform well in diverse surveillance environments due to constraints in the datasets they were trained on.

Real-Time Performance Issues: They are not always effective in detecting suspicious activities promptly, limiting their usefulness for real-time intervention.

Dependency on Preexisting Patterns: These models rely heavily on learned patterns, which hampers their performance in handling novel or unpredictable scenarios.

Performance Overview of Popular Models:

Time Distributed CNN: Achieves an accuracy rate of **90.14%**. Hybrid

Model: Reaches **88.23%** accuracy.

Keras_GRU: Provides an accuracy of **84.51%**. Conv3D:

Attains an accuracy of **83.80%**.

While these models provide reasonable accuracy, they are not without their shortcomings, particularly in handling dynamic and unexpected real-world situations

1.3 Proposed System

The **proposed system** for Suspicious Human Activity Recognition (SHAR) aims to enhance surveillance video analysis by integrating advanced deep learning models, specifically YOLOv5 (You Only Look Once), which is known for its real-time object detection capabilities. Here's how the proposed system improves upon the existing framework:

Features of the Proposed System:

Video Frame Conversion: Converts video into frames to extract meaningful data for analysis.

Normalization and Re-sizing: Ensures compatibility with YOLO models by normalizing data and re sizing images to standard dimensions (e.g., 414x414 or 440x440)

Enhanced Detection: YOLOv5 is adept at identifying objects and activities with greater precision in diverse surveillance contexts.

Advantages:

Improved adaptability to different environments, overcoming the limitations of existing systems.

Higher real-time detection accuracy, allowing for quicker and more reliable identification of suspicious behaviors.

Robust integration of spatial-temporal patterns for better understanding and classification of activities.

Promotes public safety through advanced surveillance technology.

CHAPTER II

CHAPTER- 02 REQUIREMENTS &

ANALYSIS LITERATURE SURVEY

While Human activity recognition has been a topic of considerable study in present literature, this section delves into the latest improvements in this domain. Cutting-edge studies in Human activity recognition predominantly revolve across the realms of machine learning and deep learning methodologies.

In the area of machine learning, Ghazal et al. [10] carried out a comparative study specializing in Human activity recognition with the use of 2d-skeletal facts. They utilized the OpenPose package to extract visual and motion attributes from 2d landmarks of human skeletal joints. The examine evaluated five supervised machine learning strategies, consisting of support Vector machine (SVM), Naive Bayes (NB), Linear Discriminant (LD), k-nearest neighbours (KNNs), and feed-forward backpropagation neural networks. The primary objective was to identify four awesome activity instructions: sitting, standing, walking, and falling, with the k-nearest neighbours (KNNs) exhibiting the most promising overall performance. In another study, Zhu et al. [11] introduced an online continuous Human action recognition (CHAR) approach based on skeletal records captured from Kinect depth sensors. Their approach employed a variable-length maximum Entropy Markov model (MEMM) for continuous hobby reputation without the need for earlier detection of activity begin and give-up factors. Additionally, a singular technique utilized bone information from a depth camera, leveraging machine learning to identify human actions appropriately [12].

In comparison to previous methodologies where every activity is identified by using a unique range of clusters distinct from activity instances, Hbali et al. [13] proposed a unified method based on skeletons to analyze the spatial temporal elements of human activity sequences. Their approach concerned using Minkowski and cosine distances to quantify the dissimilarity between joint data acquired from Microsoft Kinect. The model was trained and assessed using publicly available datasets such as MSR each day activity 3D and Microsoft MSR 3D motion. Leveraging the extremely Randomized Tree technique, the model showcased promising results in the improvement

of monitoring systems for the elderly. Considerably, this was achieved through the utilization of low-cost depth sensors and open-source libraries.

Karpathy et al. [14] underscored the effectiveness of CNNs in addressing challenges related to image identification. Their study focused on the classification of a diverse array of videos, leveraging a dataset comprising 1 million videos categorized into 487 various categories. Exploring numerous strategies to comprise local spatio-temporal information into CNNs, they proposed a multi-resolution architecture aimed at expediting the training process. Through the retraining of the top layers of the model using the UCF-one hundred and one action recognition dataset, researchers determined enormous enhancements within the model's generalization competencies, resulting in an incredible growth in accuracy from the baseline model's 43.9% to 43.3%. Feichtenhofer et al.
[15] investigated the current improvements in utilizing CNNs for detecting human activities in videos. Their focus changed to strategies that remember both the visible appearance and the actions of subjects. The study delved into leveraging CNN towers to harness spatio-temporal facts, highlighting the potential of merging spatial and temporal networks at a convolution layer without compromising performance. Introducing a modern CNN architecture for integrating video capabilities throughout both space and Time, the researchers established exceptional overall performance on broadly recognized evaluation datasets.

Authors	Year	Technology	Method	Accuracy
R. Teja, R. Nayer, and S. Indu	2018	Object Tracking	Suspicious activity identification during occlusion	85.67%
S. Ma, L. Sigal, and S. Sclaroff	2016	Activity Detection	Learning activity progression using LSTMs	76.09%
A. Manzi, P. Dario, and F. Cavallo	2017	Human Activity Recognition	Dynamic clustering of skeleton data	79.43%
Y. Hbali, S. Hbali, L. Ballihi, and M. Sadgal	2018	Skeleton-Based Human Activity Recognition	Method for elderly monitoring syatems	80.13%

Table-2.1 Literature Survey

Li et al. [14] proposed a recognition method using a CNN to enhance the accuracy of indoor human activity identification using geographical location data. Their state of-the-art system, comprising convolutional layers, fully connected layers, and max pooling, performed high-quality consequences via appropriately identifying six behaviours with a recognition rate of 84.7%, demonstrating the practicality of their method. Anishchenko [17] investigated the software of deep studying and switch learning techniques for fall detection by studying information captured from security cameras. Utilizing the CNN AlexNet architecture, the classifier becomes tailor-made to detect falls especially. The purpose was to enhance its efficiency by incorporating new heuristics that consider the temporal association of frames and the typical period of fall activities. Gul et al.

[18] delved into the utilization of the You Look Only Once (YOLO) network as the primary CNN model for real -time affected person surveillance geared toward spotting human actions. Through retraining the model across 32 epochs using categorized affected person behaviour snapshots, researchers achieved an impressive accuracy of 94.8% in action recognition, underscoring the capacity of their technique. In their research, Ullah et al. [19] brought an advanced anomaly detection framework leveraging a pre-trained CNN model for function extraction from video frames observed via processing with BD-LSTM. Their architecture verified promising results on the UCF-Crime dataset, illustrating the functionality for effective anomaly identity in surveillance networks.

2.1.1 Overview

Traditional surveillance systems depended on human observation, leading to inefficiencies. AI- powered solutions automate activity recognition, reducing human intervention. CNN's help extract spatial features from video frames, improving object detection accuracy. RNNs and LSTMs allow systems to track movement progression over time. YOLO models enable real-time detection, making security monitoring more effective. Modern architectures, such as Conv3D and Transformers, improve complex pattern recognition. HAR systems can now detect aggression, theft, and other abnormal behaviors in videos. AI integration speeds up response times by generating instant alerts for security personnel. Further research focuses on refining models to handle diverse surveillance conditions. Future advancements may integrate multi-modal approaches for more reliable detection.

2.1.2 Architecture

The system is divided into multiple functional modules, each playing a crucial role in ensuring **efficient processing, security, and real-time suspicious activity detection.**

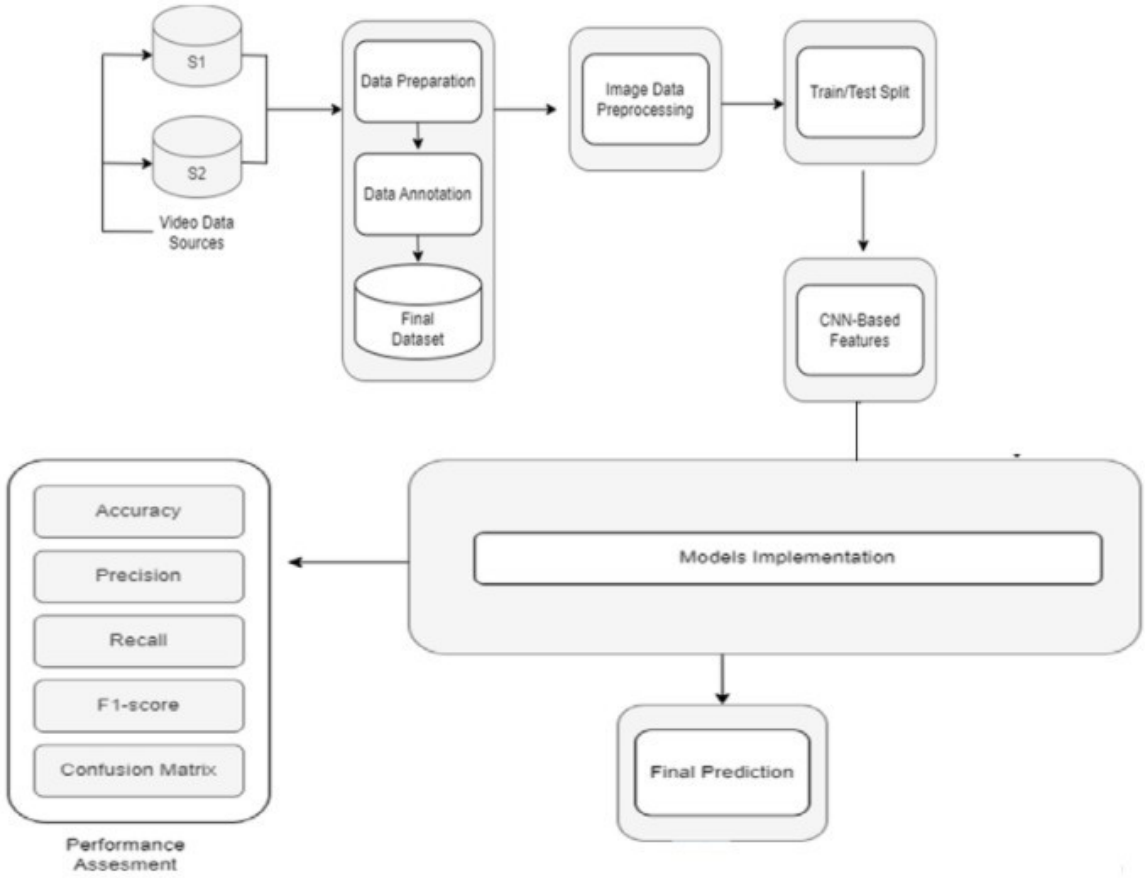

Fig-2.1.2.1 Architecture

2.2 Modules Description

1. Branching Program

- The **branching logic** determines how different activity sequences are classified.
- Uses **deep learning models** to categorize actions into **normal or suspicious behaviors.**

- Implements **decision trees and neural network layers** to refine classification accuracy.
- Improves **real-time response capabilities**, ensuring quicker detection of threats.
- Adapts dynamically to **environmental variations and different surveillance conditions**.

2. Token Generation

- Generates **secure authentication tokens** for accessing surveillance data.
- Uses **cryptographic algorithms** to prevent unauthorized access.
- Ensures **role-based access control**, allowing different levels of permissions.
- Helps track **user interactions** within the surveillance system for auditing purposes.
- Protects stored surveillance records from **tampering or external security threats**.

3. Query Processing

- Enables **rapid retrieval** of surveillance records through an optimized query system.
- Uses **AI-driven indexing** to allow faster searches in large video datasets.
- Supports **event-based retrieval**, making forensic investigation more efficient.
- Implements **metadata tagging**, helping classify activities with improved precision.
- Ensures **real-time analysis** to flag unusual actions instantly for security teams.

4. Semi-Trusted Authority

- Manages **controlled access** to surveillance logs to prevent misuse.
- Implements **multi-layer authentication** for security personnel.
- Tracks **user activities** to prevent unauthorized modifications.
- Uses **secure encryption** to protect sensitive surveillance information.Balances **accessibility and privacy**, ensuring compliance with security policies.

2.3 Security Model

A security model protects surveillance systems from unauthorized access and data leaks. Homomorphic encryption ensures computations can be performed on encrypted data. Decryption outsourcing improves system efficiency while maintaining confidentiality. Proxy re-encryption facilitates secure access control across multiple users. AI-driven security frameworks adapt to emerging threats in surveillance environments. Automated access restriction enhances database security and prevents breaches. System audits track modifications to ensure integrity and compliance. Biometric authentication may further strengthen security protocols. Machine learning- driven anomaly detection prevents unauthorized data manipulation. Future security models will integrate blockchain for enhanced data protection. Surveillance systems remain resilient against cyber threats with continuous improvements.

2.3.1 Homomorphic Encryption

Homomorphic encryption allows computations on encrypted surveillance data. It prevents unauthorized exposure while enabling AI models to process video footage. Surveillance recordings remain protected, even during activity classification. AI-driven security ensures compliance with strict data privacy regulations. The encryption framework allows secure sharing of surveillance analytics. Computational efficiency is maintained despite encryption overhead. A layered approach strengthens cryptographic security against cyber threats. AI-enhanced encryption protocols improve real-time activity recognition accuracy. Homomorphic encryption is increasingly adopted for secure surveillance applications. Future implementations will optimize processing speed for encrypted video analysis.

2.3.2 Decryption Outsourcing

Decryption outsourcing enables secure access to encrypted surveillance records. It ensures computational efficiency by delegating decryption to external trusted entities. This model improves system scalability while maintaining strict confidentiality. AI-driven decryption protocols prevent unauthorized exposure of surveillance data. Advanced cryptographic techniques protect stored footage from cybersecurity threats. Secure remote access ensures surveillance operations remain

unaffected. Auditing mechanisms track access requests for compliance purposes. The system balances security with usability through controlled outsourcing. Future developments may integrate decentralized authentication frameworks. Encryption layers will continue refining security protocols for better threat prevention.

2.3.3 Key Private Proxy Re-Encryption (PRE)

Proxy re-encryption strengthens access control in surveillance environments. AI-based encryption techniques refine key management security protocols. Encrypted access ensures that surveillance records remain tamper -proof. Multi-layer security prevents unauthorized modifications to sensitive footage. Machine learning enhances predictive security for re-encryption frameworks. Proxy-based access management ensures seamless authentication across systems. Surveillance analytics remain protected with strict encryption policies. Continuous refinement improves re-encryption model efficiency. Future innovations will explore quantum-resistant cryptographic methods. Proxy re- encryption enhances surveillance cybersecurity in modern AI applications.

2.4 System Requirements

The **System Requirements** are crucial to ensure the project operates smoothly and efficiently across various surveillance environments.

2.4.1 Software Requirements

Operating System: Windows 7 Ultimate or newer for compatibility and efficiency. **Programming Language**: Python, versatile for both front-end and back-end development. **Framework**: Flask, a lightweight web framework to build the application interface.

Database: MySQL with WAMP Server for handling and organizing surveillance data securely.

Design Tools: HTML, CSS, and JavaScript to design an interactive and user-friendly interface.

AI Libraries: TensorFlow and PyTorch for implementing and training deep learning models like YOLOv5.

Image Processing Tools: OpenCV for preprocessing video frames.

Development Tools: Jupyter Notebook or VS Code for code development and debugging.

2.4.2 Hardware Requirements

Processor: Intel Pentium-IV or equivalent, with support for multi-core processing for deep learning tasks.

RAM: At least 4GB for basic operation, but 8GB or more is recommended for smoother execution.

Storage: 20GB minimum, preferably SSD, for storing large datasets and model files.

Graphics Card: GPU with CUDA support, such as NVIDIA GTX series, for accelerating deep learning model training.

Monitor: SVGA or higher-resolution display to review surveillance footage and system outputs.

Input Devices: A standard keyboard and mouse for user interaction.

Network Hardware: Stable internet connection for data transfer and updates.

Cooling System: Effective cooling for extended operations, especially during model training.

CHAPTER III

CHAPTER-3

SYSTEM DESIGN

3.1 INTRODUCTION

System design is a crucial step in creating a framework that translates theoretical requirements into practical, functional solutions. It ensures scalability, efficiency, and integration within real-time surveillance systems. The Suspicious Human Activity Recognition (SHAR) project focuses on implementing deep learning algorithms to identify human activities in video feeds effectively. The design aims to integrate layers for video prepossessing, feature extraction, classification, and secure reporting. Emphasis is placed on modular, flexible components that can adapt to diverse environments such as public spaces or restricted zones. By incorporating system design elements, SHAR ensures that real-time detections, actionable alerts, and secure data management are achieved.

3.2 UNIFIED MODELING LANGUAGE DIAGRAMS

Using the blue print, the UML method allows for a detailed description of the system architecture.When it comes to demonstrating huge, complex frameworks, UML offers a variety of design best practices that have proven to be powerful. Make object-oriented programming and product improvement process heavily dependent on UML. UML primarily communicates the programming project plan through graphical documentation.Project groups can convey all the more successfully, research elective plans, and approve the product's engineering configuration by utilizing the UML.

3.2.1 CLASS DIAGRAM

A class diagram, as used in programming, is a type of static design model in the Unified Modeling Language (UML) that describes classes, functions, operations (or techniques), and connections between classes and frameworks. Shows the type of information. .

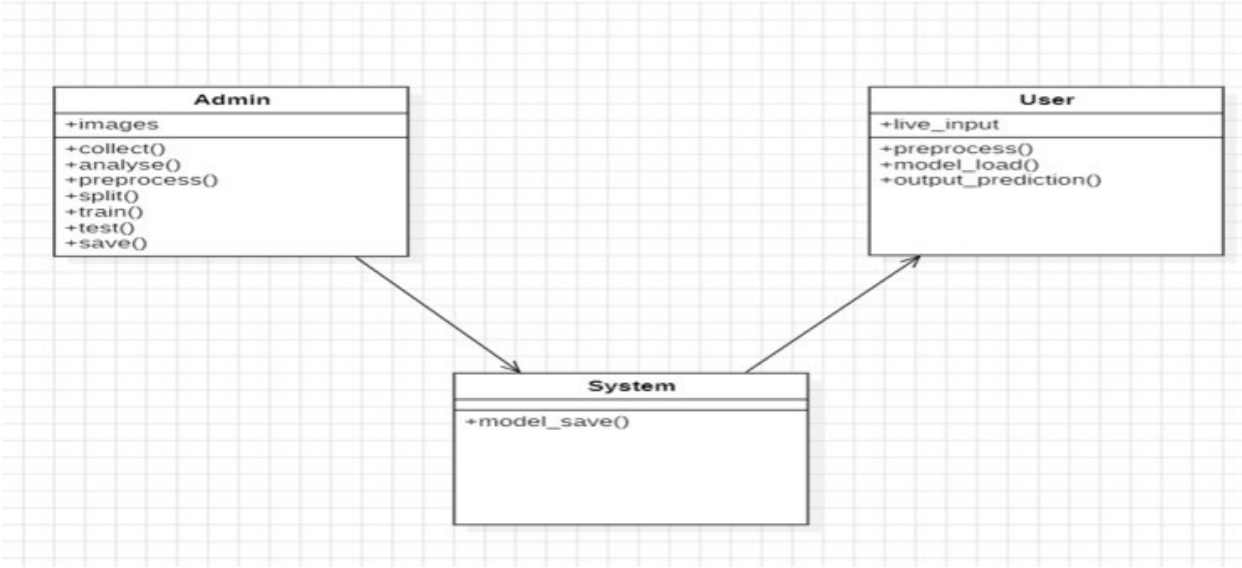

Fig-3.2.1.1 Class Diagram

3.2.2 USE CASE DIAGRAM

As per the Unified Modeling Language (UML), a use case chart is a particular kind of friendly outline made and described by use case research.Its objective is to give a graphical synopsis of the utility given by a system as far as liveliness, use cases (portrayal of its objectives), and any interdependencies between these use cases.The fundamental reason for a utilization case chart is to show which specialists get which capacities from the structure.It is possible to identify the occupations of actors within the framework.

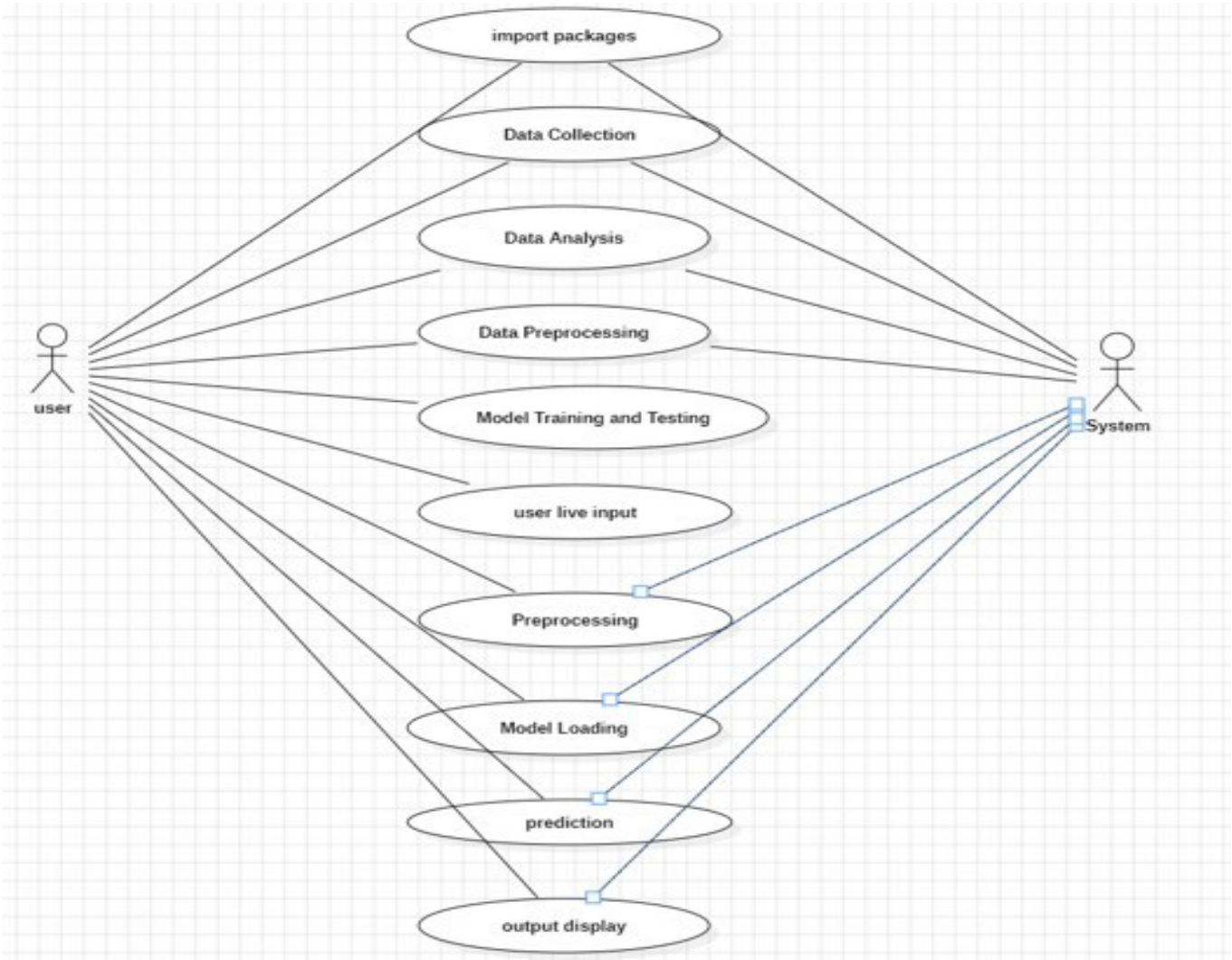

Fig-3.2.2 Use Case Diagram

3.2.3 SEQUENCE DIAGRAM

In the Unified Modeling Language (UML), a succession outline is a sort of cooperative arrangement that depicts the associations and gatherings wherein cycles work together.This is the development of the message grouping graph.Event chart, occasion situation and timing outline are various names of arrangement chart.

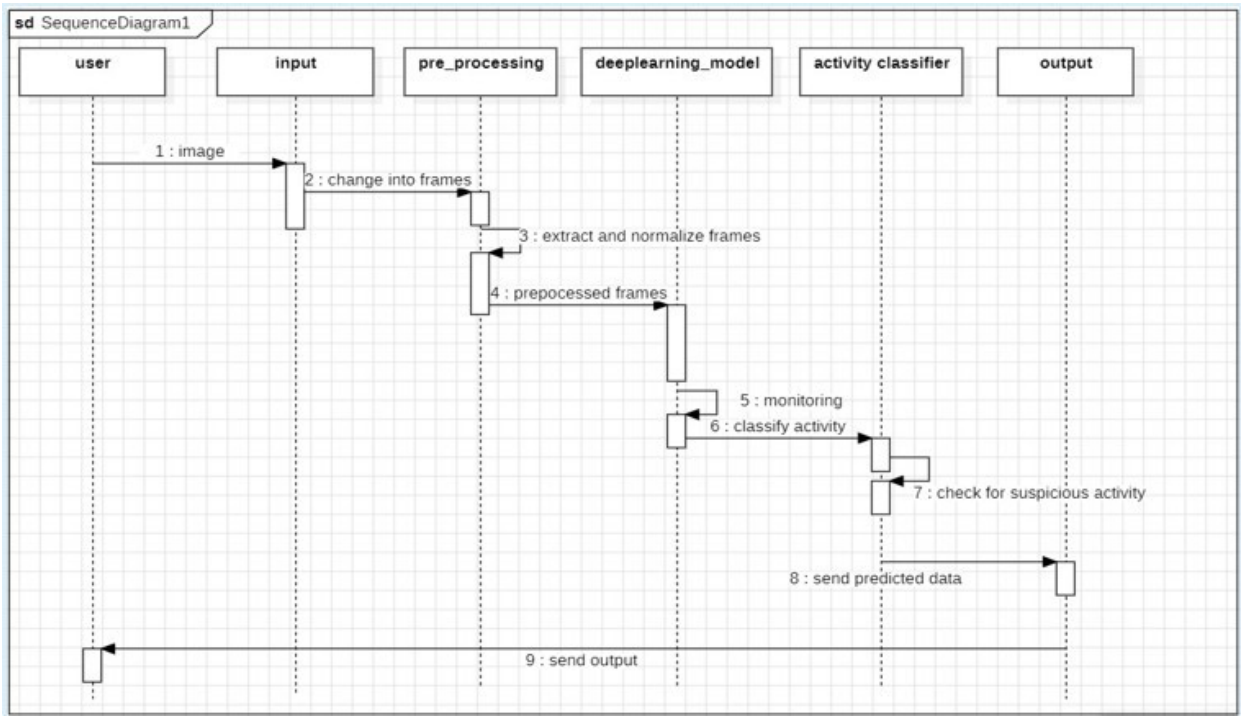

Fig-3.2.3 Sequence Diagram

3.2.4 ACTIVITY DIAGRAM

Activity diagrams are graphical work processes that help decision, iteration, and concurrency in consecutive exercises and activities.Activity graphs in the Unified Modeling Language can be used to make sense of subsequent functional and business workflows of parts of the framework. A stock chart shows the overall flow of control.

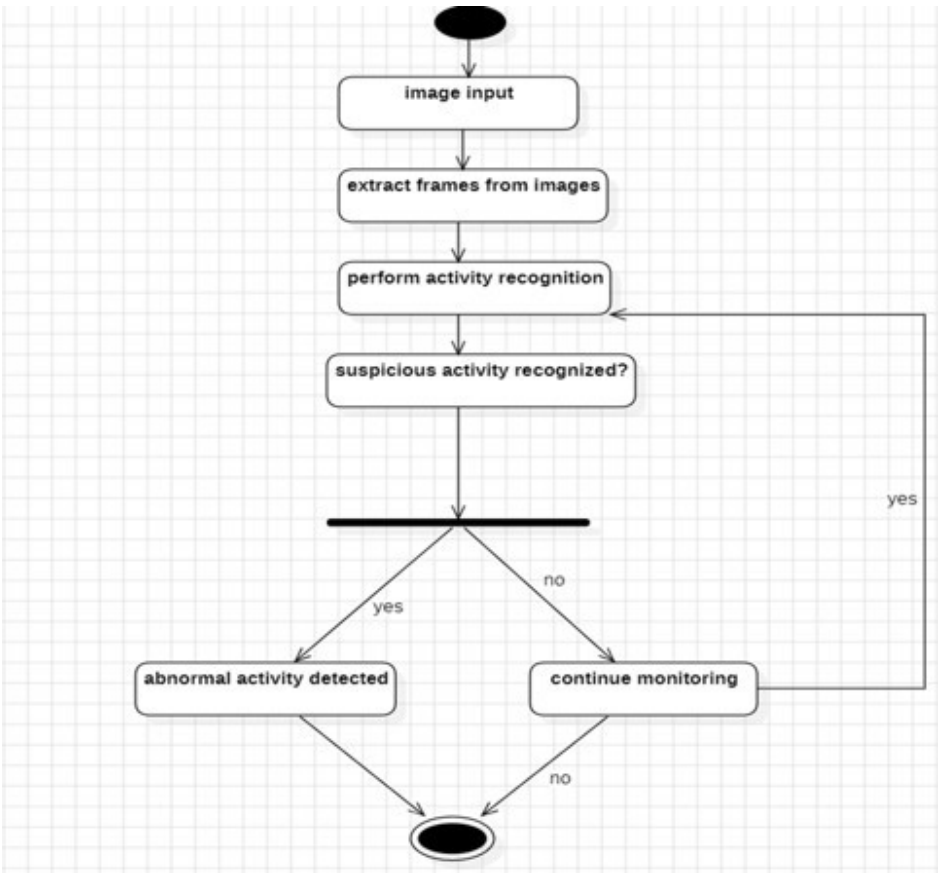

Fig-3.2.4.1 Activity Diagram

CHAPTER IV

CHAPTER -04

IMPLEMENTATION

The process initiates with the identification and collection of **Video Data Sources** (denoted as S1 and S2), which are repositories containing raw video inputs. These sources may vary in format, quality, and content. To ensure the usability of this data, a rigorous **Data Preparation** process is employed. This phase focuses on cleaning, filtering, and organizing the raw video data to remove inconsistencies, noise, or redundant information. Once prepared, the data undergoes **Data Annotation**, a critical step where domain experts or automated tools label and classify specific sections of the video data. Annotation ensures that the dataset is contextualized and ready for machine learning tasks. This curated and annotated dataset, referred to as the **Final Dataset**, serves as the foundational input for subsequent model training and evaluation. Proper preparation and annotation ensure data reliability, relevance, and representativeness for achieving robust predictions.

Following data preparation, the annotated dataset enters the **Image Data Preprocessing** stage. This step focuses on extracting frames or image samples from the video data and performing transformations to prepare them for analysis. Key preprocessing tasks include resizing frames to a uniform resolution, normalizing pixel values to maintain consistency, and applying enhancement techniques to improve the quality of images. This ensures that the data is standardized and free of distortions. Once preprocessed, the dataset is split into **Train/Test Splits**, creating subsets for training and evaluating the models. An essential part of this methodology involves **CNN-Based Feature Extraction**, where Convolutional Neural Networks (CNNs) are utilized to learn and extract deep, meaningful features from the preprocessed images. These features capture intricate patterns and visual structures critical to solving the prediction task, making them a robust input for model implementation.

The pipeline culminates in the Models Implementation stage, where the extracted CNN-based features serve as inputs for training sophisticated machine learning or deep learning models. These

models leverage the deep features to build a predictive framework capable of handling complex datasets. The predictions generated by these models are consolidated into a comprehensive Final Prediction output. To ensure the reliability and effectiveness of the model, a detailed Performance Assessment is conducted. This evaluation involves calculating metrics such as Accuracy, Precision, Recall, F1-Score, and generating a Confusion Matrix. These metrics provide a quantitative understanding of the model's strengths and weaknesses, enabling targeted refinements if necessary. This systematic approach, from data acquisition to performance assessment, ensures the development of a highly effective and accurate predictive system.

4.1 MODULES

Load Data Data
collection
Data pre-processing
Feature Selection
Feature Extraction Deep
Learning

4.1.1 LOAD DATA:

With Pandas, you can import data straight into a dataframe from a variety of data sources. These can be a variety of widely used databases, including MySQL, PostgreSQL, and Google BigQuery, as well as static files like CSV, TSV, fixed width files, Microsoft Excel, JSON, SAS, and SPSS files. Data can also be directly scraped from websites and entered into Pandas dataframes.

4.1.2 DATA COLLECTION :

Gathering data from various sources both online and offline by scraping, capturing, and loading it is known as data collection. The most challenging aspect of a machine learning project can be creating or gathering large amounts of data, especially when working at scale.In order to use data analysis to identify recurrent patterns, data collection enables you to record previous events. Using

machine learning algorithms, you can create predictive models based on those patterns that identify trends and project future changes.Since predictive models can only be as good as the data they are built on, effective data collection techniques are essential to creating models that perform well. The information in the data must be accurate and pertinent to the task at hand. For instance, a loan default model might eventually profit from gas prices but not from changes in tiger populations.

4.1.3 DATA PRE-PROCESSING :

Data preprocessing is the most commonly used method to prepare raw data for machine learning models.This is the first important step in developing a machine learning model.It's not always easy to find clear and well-organized data when starting a machine learning project.Therefore, data cleaning and formatting are essential tasks for any project that involves data.This is where data preprocessing comes into play.

True information frequently contains errors, missing qualities, and might be in a configuration unsatisfactory for direct utilization of machine learning models.Cleaning and getting ready information for machine learning models are significant stages in the information preprocessing process, assisting with working on the precision and proficiency of machine learning models.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

4.1.4 FEATURE SELECTION:

The objective of feature selection strategies in deep learning is to find the ideal arrangement of highlights that work on the advancement of exact models for the qualities being studied.There are various sorts of deep learning highlight determination procedures that can be utilized, including supervised techniques, which are utilized on marked information to distinguish important elements that will further develop performance.performance of administered models like relapse and grouping (e.g.SVM, choice tree, linear,Regression).On the other hand, unsupervised techniques can be utilized on unlabeled information, like principal component analysis, hierarchical clustering, and K-Means clustering.These procedures are ordered into filtering, packing, embedding, and combination techniques in light of their methodology.

4.1.5 FEATURE EXTRACTION:

The dimensionality decrease process, what separates and lessens an underlying arrangement of crude information into additional sensible gatherings, include feature extraction. It will in this manner be easier to process when you need to. These enormous informational collections' overflow of factors is by a long shot their most huge component. Handling these factors takes a ton of computer power. Accordingly, feature extraction productively lessens how much information by picking and joining factors into elements to assist with removing the best component from those huge informational indexes. These qualities are easy to deal with while precisely and innovatively portraying the genuine informational collection.They serve as a general description of the image's color statistics.

4.1.4 DEEP LEARNING

Deep learning is a part of machine learning that has practical experience in demonstrating and taking care of muddled issues with artificial neural networks.It draws inspiration from the composition and operations of the human brain, particularly from the networked layers of neurons.

Because deep learning can automatically learn hierarchical representations from data, it has become very popular and successful in many different fields.

Here are some essential details regarding deep learning:

1. Neural Networks: Neural networks, which are made up of layers of connected nodes or artificial neurons, are the foundation of deep learning. The term "deep" in deep learning describes these networks' depth, which indicates that they have several layers (deep architectures).

2. Deep Neural Networks (DNNs): Training deep neural networks, which can have several layers (deep networks) or just a few (shallow networks), is a common step in deep learning. The network can extract complex features and representations from the input data thanks to the depth.

3. Representation Learning: Automatic feature extraction and representation learning are two areas in which deep learning shines. During training, the model learns to extract pertinent features from the data rather than manually engineering features.

4. Backpropagation Training: Backpropagation is a technique used in the training of deep neural networks. The process entails making iterative adjustments to the weights of connections among neurons in order to reduce the discrepancy between the target and the predicted output. Usually, stochastic gradient descent and other optimization algorithms are used for this.

5. Architectures: Different deep learning structures are open, including Transformers for regular language handling, Recurrent Neural Networks (RNNs) for consecutive information, and Convolutional Neural Networks (CNNs) for picture related tasks.These designs are made to deal with specific tasks and data types.

Two well-liked deep learning architectures, ResNet (Residual Network) and DenseNet (Densely Connected Convolutional Network), were created expressly to solve training difficulties for extremely deep neural networks. Let's examine each in brief:

Algorithms Used

The YOLO (You Only Look Once) family of object detection algorithms is renowned for its real- time performance and accuracy in detecting objects in images and videos. Each subsequent version introduces enhancements to architecture, feature extraction, and training strategies, improving upon its predecessors. Below is a detailed explanation of the working mechanisms for YOLOv5,YOLOv8, YOLOv9, and YOLOv11.

YOLO V5 Algorithm

1. **Backbone:** Model Backbone is mostly used to extract key features from an input image. CSP(Cross Stage Partial Networks) are used as a backbone in YOLO v5 to extract rich in useful characteristics from an input image.

2. **Neck:** The Model Neck is mostly used to create feature pyramids. Feature pyramids aid models in generalizing successfully when it comes to object scaling. It aids in the identification of the same object in various sizes and scales.

Feature pyramids are quite beneficial in assisting models to perform effectively on previously unseen data. Other models, such as FPN, BiFPN, and PANet, use various sorts of feature pyramid approaches. PANet is used as a neck in YOLO v5 to get feature pyramids.

3. **Head:** The model Head is mostly responsible for the final detection step. It uses anchor boxes to construct final output vectors with class probabilities, objectness scores, and bounding boxes.

The head of the YOLO v5 model is the same as in the previous YOLO V3 and V4 editions.

Advantages & Disadvantages of Yolo v5

- It is about 88% smaller than YOLOv4 (27 MB vs 244 MB)
- It is about 180% faster than YOLOv4 (140 FPS vs 50 FPS)
- It is roughly as accurate as YOLOv4 on the same task (0.895 mAP vs 0.892 mAP)

- But the main problem is that for YOLOv5 there is no official paper was released like other YOLO versions. Also, YOLO v5 is still under development and we receive frequent updates from ultralytics, developers may update some settings in the future.

YOLOv8: Improvements and Mechanism

YOLOv8 builds on YOLOv7 by improving computational efficiency, accuracy, and generalization. It introduces architectural refinements and focuses on optimal use of computational resources.

1. **Key Features**:
    - **Decoupled Head**: YOLOv8 separates classification and regression tasks in its head, improving the accuracy of object localization and label prediction.
    - **Dynamic Anchor Free Mechanism**: YOLOv8 uses a dynamic anchor-free mechanism, which eliminates reliance on predefined anchor boxes. This reduces computational complexity and improves detection for objects of varying scales.
    - **Scaled-Down Neural Blocks**: Efficient convolutional modules are used to reduce latency and improve throughput.
2. **Workflow**:
    - The input image is divided into an SxS grid, and each cell is responsible for detecting objects within its region.
    - Feature extraction happens via a deep CNN backbone, where the extracted features are passed to the head for detection and classification.
    - The decoupled head processes bounding box regression and class probabilities independently, yielding precise outputs.
3. **Performance Enhancements**:
    - Optimized for real-time applications by reducing model size without sacrificing accuracy.
    - Improved generalization on diverse datasets, making it suitable for various domains, including animal detection.

MODEL SELECTION IN Deep Learning:

The process of choosing the optimal algorithm and model architecture for a given task or data set is called model selection in deep learning.It involves comparing and evaluating multiple models to determine which model best fits the data and provides the best results.When choosing a model, considerations such as model complexity, data processing capabilities, and ability to generalize to new examples are taken into account.Indicators like accuracy and mean squared error are used along with techniques like grid search and cross-validation to assess and compare models. The goal of model selection is to identify a model that strikes a balance between performance and complexity in order to generate solid predictions and strong generalization capabilities.

4.2 TECHNOLOGIES

4.2.1 PYTHON

4.2.1 Flask

4.2.1 Python

Python is a popular programming language known for its ease of interpretation and numerous options for creating Graphical User Interfaces (GUIs). Among the various GUI technologies available, Flask is the most commonly used and serves as the standard interface for Python's Tk GUI toolkit.

The least demanding and quickest method for utilizing Flask results to make GUI applications is with Python. Utilizing Flask to make a GUI is a simple undertaking. Python is a generally utilized, flexible, and normal programming language.

It is phenomenal as a first language since it is succinet and easy to comprehend and furthermore great to use in any developer's heap since it tends to be used from improvement of the web to programming. It's fundamental, simple to-utilize sentence structure, making it the best language to initially learn PC programming.

Most implementations of Python (including C and Python), include a read- eval-print (REPL) loop that enables the user to act as a command-line interpreter that results in sequence and instantaneous intake of instructions. Other shells like as IDLE and Python provide extra features such as auto- completion, session retention and highlighting of syntax.

Interactive mode programming

This prompt appears when the interpreter is invoked without a script file passed as an argument.

– $ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type “help“, “copyright“, “credits“ or “license“ for more information Type

the following text at the Python prompt and press the Enter –

>>> print “Hello, Python!“ If you are running new version of Python, then you would need to use print statement with parenthesis as in print (“Hello, Python!“);. However in Python version 2.4.3, this produces the following result

– Hello,

Script mode programming

The script is executed when the interpreter is invoked with a script parameter, and it runs continuously until it is completed. The interpreter stops working when the script is done.

Let‘s write a script that runs a basic Python program. Python files have the.py extension. Enter the source code in a test.py file:

Live Demo print \“Hello, Python!\“ Let‘s say you have the Python interpreter configured in the PATH variable. Now try running this program as follows:

$ python test.py This produces the following output:

Hello, Python! Let‘s try another way to run a Python script. Here is the modified test.py file –

Live Demo

#!/usr/bin/python print “Hello, Python!“

Let‘s say you have a Python interpreter available in the /usr/bin directory. Now try running this program as follows:

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result –

Hello, Python!

4.2.2 Flask web framework

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.Unlike the Django framework, Flask is very Pythonic. It's easy to get started with Flask, because it doesn't have a huge learning curve.

On top of that it's very explicit, which increases readability.

It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre- existing third- party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object- relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools

4.3 CODING

```
import argparse
import io import
os
from PIL import Image
import cv2
import numpy as np
from torchvision.models import detection
import sqlite3
import torch
from torchvision import models
from flask import Flask, render_template, request, redirect, Response
```

```
import pathlib
temp = pathlib.PosixPath pathlib.PosixPath =
pathlib.WindowsPath app = Flask(_____name
_____)
model = torch.hub.load("ultralytics/yolov5", "custom", path = "best.pt", force_reload=True)
model.eval()
model.conf = 0.5
model.iou = 0.45
from io import BytesIO
def gen():
    """
    The function takes in a video stream from the webcam, runs it through the model, and
returns the
    output of the model as a video stream """
    cap=cv2.VideoCapture('Training VIdeos/4th video.mp4') while(cap.isOpened()):
        success, frame = cap.read() if
        success == True:
            ret,buffer=cv2.imencode('.jpg',frame)
            frame=buffer.tobytes()
            img = Image.open(io.BytesIO(frame)) results =
            model(img, size=415) results.print()
            img = np.squeeze(results.render())
            img_BGR = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
```

```
        else:
            break
        frame = cv2.imencode('.jpg', img_BGR)[1].tobytes()
        yield(b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video') def
video():
    """
    It returns a response object that contains a generator function that yields a sequence
of images
    :return: A response object with the gen() function as the body. """
    return Response(gen(),
                            mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route("/predict", methods=["GET", "POST"]) def
predict():
    """
    The function takes in an image, runs it through the model, and then saves the output
image to a
    static folder
    :return: The image is being returned. """
    if request.method == "POST": if
        "file" not in request.files:
            return redirect(request.url) file
        = request.files["file"]
```

```
        if not file: return
        img_bytes = file.read()
        img = Image.open(io.BytesIO(img_bytes)) results =
        model(img, size=415) results.render()
        for img in results.render():
            img_base44 = Image.fromarray(img) img_base44.save("static/image0.jpg",
            format="JPEG")
        return redirect("static/image0.jpg")
    return render_template("index.html")
@app.route('/')
@app.route("/index") def
index():
    return render_template("index.html") if _
name_____== "____main___":
    app.run(port=5000)
```

CHAPTER V

CHAPTER -05 TESTING

The reason for testing is to recognize errors.Inspection plans to distinguish any likely deformities or weaknesses in a thing of work, consequently giving a method for assessing the exhibition of individual parts, subassemblies, congregations and items Final.This is a key strategy for assessing programming to guarantee that it meets client assumptions and necessities without unsuitable degrees of error.There are a wide range of kinds of testing, each gathering a particular testing prerequisite.

5.1 TYPES OF TESTS

5.1.1 Unit testing

The most common approach to test planning for unit testing ensures that a program's internal logic works correctly and that the program's inputs produce valid results.Validation must be performed on all internal code paths and decision branches.This includes testing different programming parts of the application.Before integration, it was finished by the decision of a solitary unit.This is an essentially obtrusive test that relies upon understanding its structure.Unit trying assesses a particular application, framework design, or business process at the part level.Unit testing guarantees that every individual way in the business cycle has clear data sources and results and that it works accurately as per archived determinations.

5.1.2 Integration testing

Integration testing is performed to evaluate how well integrated software components function as a unit.These tests are event-based and mainly focus on the primary outcome of the interaction between fields or screens.They show that while unit testing can effectively approve individual parts, the blend of parts stays exact and reliable.The objective of joining testing is to distinguish any potential issues that might emerge from integrating various parts.

5.1.3 Functional test

Functional testing gives deliberate check that the usefulness under test is open as indicated by specialized and business necessities, framework documentation, and client manuals.

Functional testing centers around the accompanying regions:

Valid input: You must accept the recognized classes of valid input. Invalid input:

Some input classes should be ignored.

Features: You must use the designated features. Output:

Identified application exits must be exercised.

Systems/Procedures: It is necessary to use additional systems or procedures.

Functional testing focuses on specific requirements, key components, and specialized tests. Additionally, testing should consider data fields, predefined processes, sequential steps, and system scope involved in defining business process flows.Additional tests are identified and the effectiveness of existing tests evaluated before completing functional testing.Framework auditing verifies that all prerequisites are met overall by the integrated programming framework.It checks the configuration to ensure reliable results.Configuration-oriented system integration testing is a description of framework testing.Framework examination emphasizes predetermined process connectivity and coordination goals and depends on process flows and representations.

5.1.4 White Box Testing

White box testing is a type of programming testing in which the analyst is aware of the inner workings, design and language of the program or in any case, what it is supposed to do.There's a reason.Used to test districts, they are beyond the scope of the black box level.

5.1.5 Black Box Testing

Performing black box testing in programming implies testing a module without knowing its interior tasks, plan, or language Like different kinds of testing, black box testing must likewise be composed in light of a dependable source, for example, a detail or necessities report.

This sort of testing regards the product as a "black box" that can't be seen.Inputs are created and yields are seen regardless of the activity of the product.

5.1.6 Unit Testing

Although coding and unit testing are often performed in separate phases, unit testing is often performed along with coding in the combined code and unit testing phases of the software development cycle.

5.2 Test strategy and approach

Functional testing will be carefully designed and field testing will be performed manually.

Test objectives

- All field entries must be operational.

- Pages must be accessed through the specified link.

- No lag should be experienced in the entry screen, messages, and responses.

Features to be tested

1. Verify that all entries adhere to the correct format.
2. Eliminate any duplicate entries.
3. Ensure that each link guides users to the correct page.

Integration Testing

Programming integration testing includes steadily coordinating at least two composed programming parts into a solitary stage to distinguish interface blunders that might cause mistakes. The objective of integration testing is to guarantee that product applications, framework parts, or significantly more elevated level endeavor programming applications work flawlessly together.

Acceptance Testing

Client acceptance is critical during the testing period of any errand, as it includes broad cooperation with end clients to guarantee that the framework meets useful particulars.

Test Results:

All recently mentioned test cases were successful with no deficiencies identified.

TEST CASES:

S. No	Test Cases	Expected Results	Results	Remarks
1	Import packages	System should load the packages	Pass/Fail	Ensure that all the required packages are loaded
2	yolov5	System should Clone with yolov5	Pass/Fail	Check the accessibility of the algorithm
3	yolov8	System should Clone with yolov8	Pass/Fail	Check the accessibility of the algorithm
4	Data.yaml	Connection establishment	Pass/Fail	Data categories initialization

s.no	Testcase	Expected result	Result	Remarks
5	Image input	image loading	Pass/Fail	Ensure that the image is in the correct format
6	Upload	Uploading the image to the model	Pass/Fail	ensure the extension of the image and upload successfully
7	Best.pt	mAP value should be greater than 0.5	Pass/Fail	Checking the mAP value
8	Prediction	Precit the output	Pass/Fail	Ensure that the predicted output is correct

Table 5.2.1 Test Result

5.3 ScreenShots

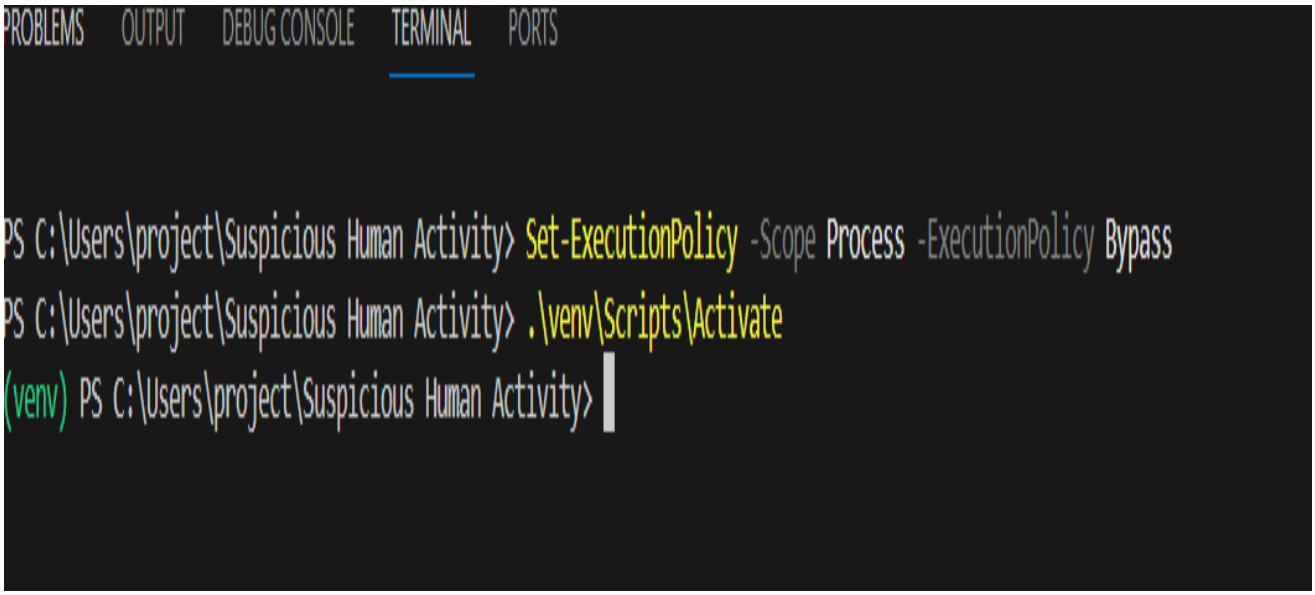

Fig-5.3.1 Terminal screen

Fig-5.3.2 Terminal screen with url link

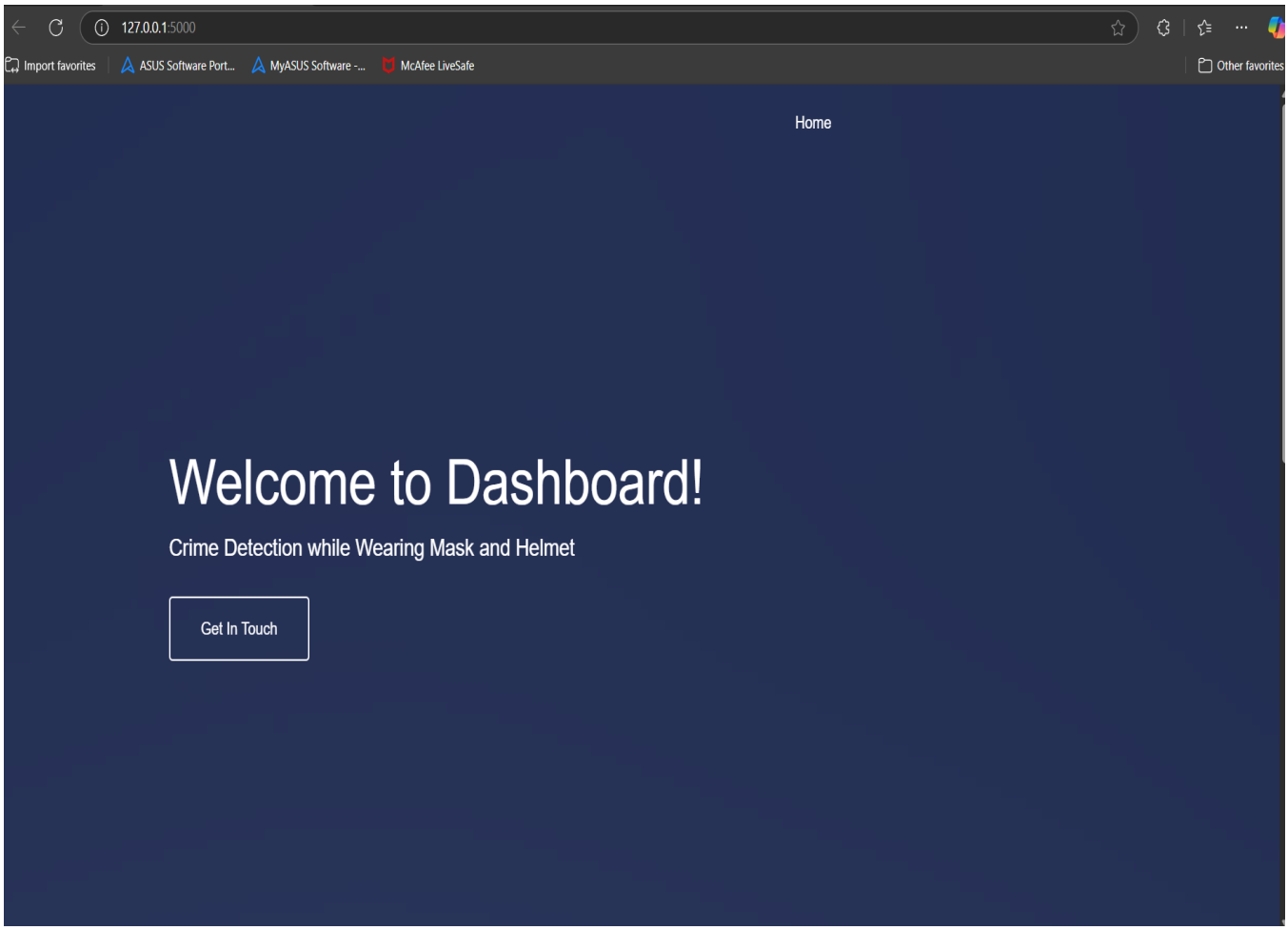

Fig-5.3.3 web page 1

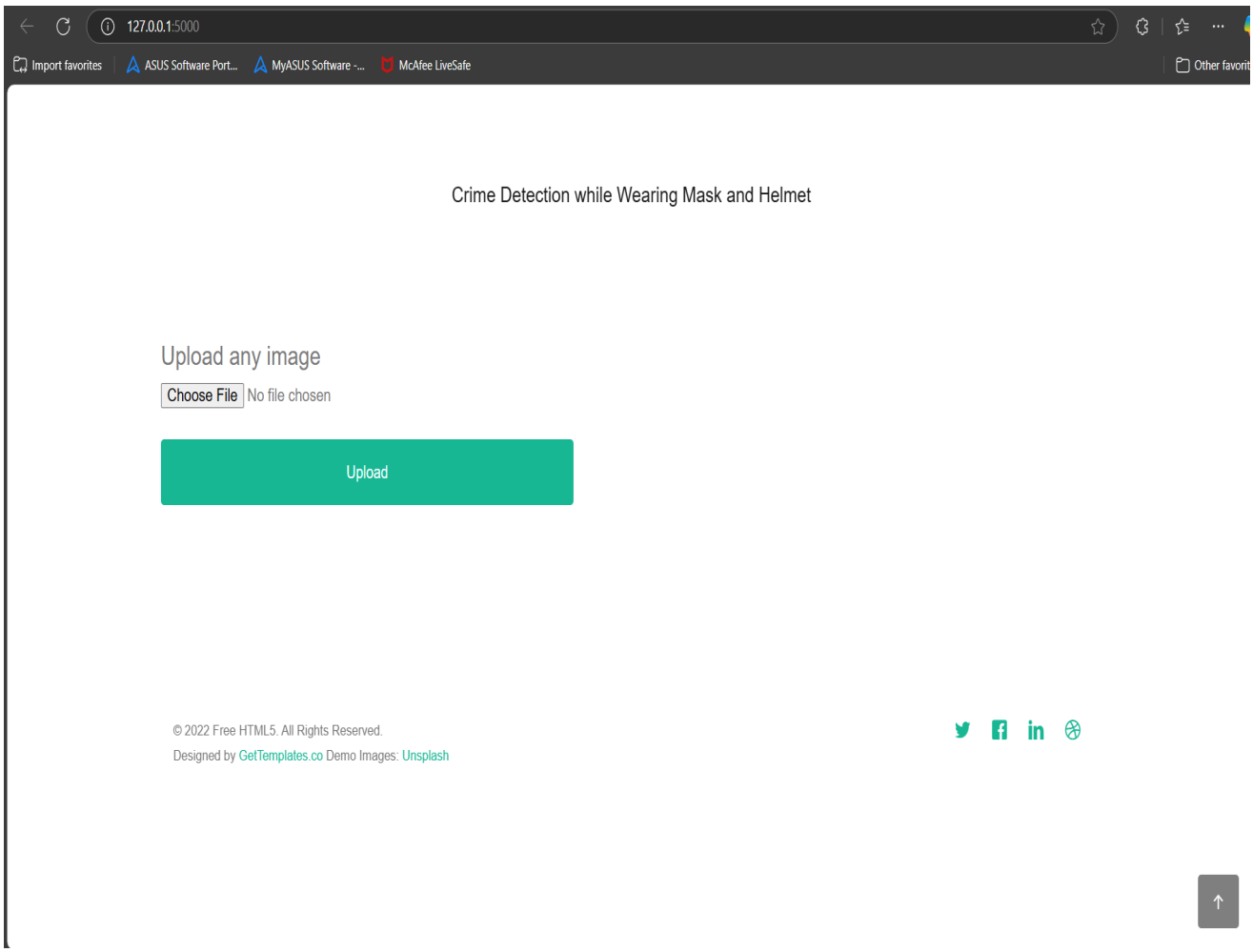

Fig-5.3.4 Web page-2

5.4 Screenshots

Fig-5.3.5 Result

Fig-5.3.6 Result

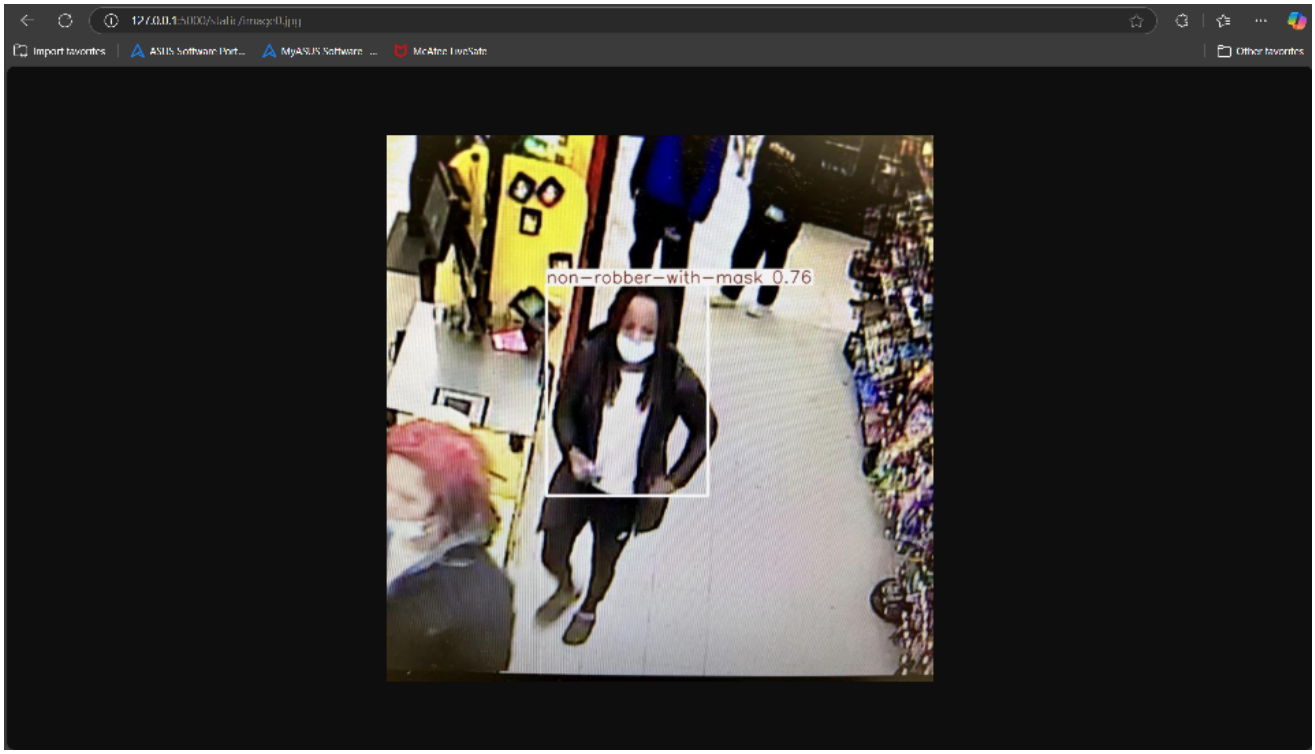

Fig-5.3.7 Result

Fig-5.3.8 Result

CHAPTER VI

CHAPTER-6 CONCLUSION

Utilizing the YOLOv5 algorithm for Suspicious Human Activity Recognition (SHAR) offers a highly efficient and accurate approach to real-time monitoring and threat detection. YOLOv5's lightweight architecture ensures rapid processing of video streams, enabling precise detection of suspicious activities with minimal latency. Its ability to recognize multiple objects and actions simultaneously makes it highly scalable for diverse applications, from public surveillance to restricted areas. Additionally, YOLOv5's support for continuous learning enhances adaptability to evolving behavioral patterns, while its compatibility with edge devices enables cost-effective and seamless integration into existing security systems. This combination of speed, scalability, and accuracy makes YOLOv5 a transformative tool for advancing SHAR solutions.

FUTURE ENHANCEMENT:

Action Recognition: Future versions of YOLO models can be enhanced to recognize not only static objects but also human actions or activities. For example:

Recognizing aggressive behavior (punching, shoving).

Identifying abnormal walking patterns (e.g., running, limping, staggering).

Recognizing interactions with objects (placing a suspicious object in a bag, tampering with vehicles).

Contextual Awareness: Beyond just detecting objects and actions, future YOLO models could be trained to understand context. For example, it could distinguish between a person walking normally and a person running in a restricted zone (e.g., an airport or secured building).

Fine-Grained Object Detection: Enhancing YOLOv5 and YOLOv8 to detect finer details of suspicious objects, such as distinguishing between different types of weapons (knives, or even makeshift weapons), or identifying abnormal items like bags or packages in crowded areas.

Tiny and Distant Object Detection: YOLOv5 and YOLOv8 could be further optimized to detect smaller or distant objects. For example, identifying a small object like a weapon hidden in someone's clothing or detecting people in crowded, far-away scenes.

CHAPTER VII

CHAPTER-7

BIBLIOGRAPHY

[1] K. Rezaee, S. M. Rezakhani, M. R. Khosravi, and M. K. Moghimi, "A survey on deep learning- based real-time crowd anomaly detection for secure distributed video surveillance," *Pers. Ubiquitous Comput.*, vol. 28, no. 1, pp. 135–151, Feb. 2024.

[2] M. Perez, A. C. Kot, and A. Rocha, "Detection of real-world fights in surveillance videos," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2442–2444.

[3] C. V. Amrutha, C. Jyotsna, and J. Amudha, "Deep learning approach for suspicious activity detection from surveillance video," in *Proc. 2nd Int. Conf. Innov. Mech. Ind. Appl. (ICIMIA)*, Mar. 2020, pp. 335–339.

[4] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4479–4488.

[5] J. Wei, J. Zhao, Y. Zhao, and Z. Zhao, "Unsupervised anomaly detection for traffic surveillance based on background modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 129–134.

[4] A. Waheed, M. Goyal, D. Gupta, A. Khanna, A. E. Hassanien, and H. M. Pandey, "An optimized dense convolutional neural network model for disease recognition and classification in corn leaf," *Comput. Electron. Agricult.*, vol. 175, Aug. 2020, Art. no. 105454.

[7] R. Teja, R. Nayar, and S. Indu, "Object tracking and suspicious activity identification during occlusion," *Int. J. Comput. Appl.*, vol. 179, no. 11, pp. 29–34, Jan. 2018.

[8] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in LSTMs for activity detection and early detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 1942–1950.

[9] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 1510–1517, Jun. 2018.

[10] S. Ghazal, U. S. Khan, M. Mubasher Saleem, N. Rashid, and J. Iqbal, "Human activity recognition using 2D skeleton data and supervised machine learning," *IET Image Process.*, vol. 13, no. 13, pp. 2572–2578, Nov. 2019.

[11] G. Zhu, L. Zhang, P. Shen, and J. Song, "An online continuous human action recognition algorithm based on the Kinect sensor," *Sensors*, vol. 14, no. 2, p. 141, Jan. 2014.

[12] A. Manzi, P. Dario, and F. Cavallo, "A human activity recognition system based on dynamic clustering of skeleton data," *Sensors*, vol. 17, no. 5, p. 1100, May 2017.

[13] Y. Hbali, S. Hbali, L. Ballihi, and M. Sadgal, "Skeleton-based human activity recognition for elderly monitoring systems," *IET Comput. Vis.*, vol. 12, no. 1, pp. 14–24, Feb. 2018.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.

[15] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014,
pp. 1933–1941.

[14] J. Li, R. Wu, J. Zhao, and Y. Ma, "Convolutional neural networks (CNN) for indoor human activity recognition using ubisense system," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, May 2017, pp. 2048–2072.

[17] L. Anishchenko, "Machine learning in video surveillance for fall detection," in *Proc. Ural Symp. Biomed. Eng., Radioelectron. Inf. Technol. (USBEREIT)*, May 2018, pp. 99–102.

[18] M. A. Gul, M. H. Yousaf, S. Nawaz, Z. Ur Rehman, and H. Kim, "Patient monitoring by abnormal human activity recognition based on CNN architecture," *Electronics*, vol. 9, no. 12, p. 1993, Nov. 2020.

[19] W. Ullah, A. Ullah, I. U. Haq, K. Muhammad, M. Sajjad, and S. W. Baik, "CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks," *Multimedia Tools Appl.*, vol. 80, no. 11, pp. 14979–14995, May 2021.

[20] U. M. Butt, S. Letchmunan, F. Hafinaz, S. Zia, and A. Baqir, "Detecting video surveillance using VGG19 convolutional neural networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, 2020.

[21] Q.-U.-A. Arshad, M. Raza, W. Z. Khan, A. Siddiqa, A. Muiz, M. A. Khan, U. Tariq, T. Kim, and J.-H. Cha, "Anomalous situations recognition in surveillance images using deep learning," *Comput., Mater. Continua*, vol. 74, no. 1, pp. 1103–1125, 2023.

[22] R. Vrskova, R. Hudec, P. Kamencay, and P. Sykora, "A new approach for abnormal human activities recognition based on ConvLSTM architecture," *Sensors*, vol. 22, no. 8, p. 2944, Apr. 2022.

[23] M. Qasim Gandapur and E. Verdu, "ConvGRU-CNN: Spatiotemporal deep learning for real- world anomaly detection in video surveillance system," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 8, no. 4, p. 88, 2023.

[24] R. Rajeswari, "Anomalous human activity recognition from video sequences using brisk features and convolutional neural networks," *Galaxy Int. Interdiscipl. Res. J.*, vol. 10, no. 2, pp. 249–280, 2022.

[25] I. U. Khan, S. Afzal, and J. W. Lee, "Human activity recognition via hybrid deep learning based model," *Sensors*, vol. 22, no. 1, p. 323, Jan. 2022.