A PROJECT REPORT ON

ANALYSIS OF VIDEO SURVEILLANCE IN BANK USING MACHINE LEARNING

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

STUDENT NAME	EXAM NO
NEHA SIRURMATH	B190244489
VAIBHAV WABLE	B190244514
VIDYASAGAR MORE	B190244373

DEPARTMENT OF COMPUTER ENGINEERING

Dr. D. Y. PATIL INSTITUTE OF TECHNOLOGY,

PIMPRI, PUNE

SAVITRIBAI PHULE PUNE UNIVERSITY
2022 -2023

CERTIFICATE

This is to certify that the project report entitles

ANALYSIS OF VIDEO SURVEILLANCE IN BANK USING MACHINE LEARNING

SUBMITTED BY

STUDENT NAME	EXAM NO
VAIBHAV WABLE	B190244514
NEHA SIRURMATH	B190244489
VIDYASAGAR MORE	B190244373

are a bonafide student of this institute and the work has been carried out by them under the supervision of **Prof. Jithina Jose** and is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

Mrs. Jithina Jose	**Mrs. Sunita Patil**	**Dr. Vinod Kimbahune**
Guide	Project Coodinator	HOD

Dr. Lalit Kumar Wadhwa

Principal,

Dr. D. Y. PATIL INSTITUTE OF TECHNOLOGY,
PIMPRI, PUNE

Place: Pune

Date :

ACKNOWLEDGE

We would like to express our deepest gratitude to all those who provided us with the possibility to complete this task. Special gratitude to our project guide, **Prof. Jithina Jose Mam**, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project. We also appreciate the guidance panel members gave, which helped us improve our presentation and thanks to their comments and advice. We are also thankful to our parents who provided their wishful support for our project completed successfully. And lastly, we thank our all friends and the people who are directly or indirectly related to our project work.

VAIBHAV WABLE

NEHA SIRURMATH

VIDYASAGAR MORE

ABSTRACT

As we know, theft, fights, and many other abnormal events occur in banks and are increasing daily. To tackle these problems many video surveillance systems are introduced in the market those are based on video surveillance monitored by humans and some are AI-based.

We want to develop an effective surveillance system using machine learning to detect anomalous activities and raise an alarm.

Video Surveillance is an activity of looking at some behaviors that need attention or some anomaly activity that is taking place or to observe a scene that is different from that of a normal one. Video Surveillance is a process where identification takes place in some areas where the chance of happening anomaly activity is high so that these cameras can view those areas.

Keywords: Surveillance, CCTV, video analytics, ethics, regulation, computer vision,cyber-physical system, and action modeling.

TABLES OF CONTENTS

	Appendix A: Problem statement feasibility assessment using, satisfiability analysis and NP Hard, NP-Complete or P type using modern algebra and relevant mathematical models. **Appendix B:** Details of paper publication: name of the conference/journal, comments of reviewers, certificate, paper. **Appendix C:** Plagiarism Report of project report	

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

Video Surveillance is the process of identifying activities that are different from that normal one or the one which is anomalous or which is improper in behavior. This is an automatic video anomaly detection process that reduces labor and waste time. Video Surveillance is very useful to identify abnormal events and maintain social control. In banks, Video Surveillance is used to provide a high level of security and solve financial problems banks will be under control and crimes will be minimized.

THE ANOMALY DETECTION METHODS

Supervised

Unsupervised

Semi-Supervised

Supervised are the ones where the training dataset is used to train the model. This Training set contains a normal dataset and an abnormal dataset. The model will learn from the provided dataset and classify the input video as abnormal or normal.

Unsupervised decreases the manual work in anomaly detection. In an unsupervised model is input unlabeled data and the model learns from unlabeled one with the help of algorithms model analyzes and clusters the unlabeled dataset.

Semi-Supervised is a combination of supervised and unsupervised ones, Semi-Supervised contains benefits of both methods and this method uses labeled data of small size and unlabeled data of large size.

The main goal of real-world anomaly detection is to provide a signal upon some abnormal activity taking place. Therefore, abnormal banking detection can be considered coarse-level video understanding that filters anomaly patterns from normal patterns.

1.2 MOTIVATION

Surveillance cameras are increasingly being used in public places e.g. streets, intersections, banks, shopping malls, etc. to increase public safety. One critical task in video surveillance is detecting anomalous events such as traffic accidents, crimes, or illegal activities. The goal of a practical anomaly detection system is to timely signal an activity that deviates from normal patterns and identify the time window of the occurring anomaly in bank sectors. Therefore, anomaly detection in bank sectors can be considered as coarse-level video understanding, which filters out anomalies from normal patterns.

1.3 PROBLEM DEFINITION

Bank is using video cameras for surveillance at many branches, ATMs, and digital lobbies. Getting video analytics of different parameters from the video recording will help the bank to resolve many operational issues at the branches. The bank wants to explore video analytics to understand customer sentiments, understand the patterns /behaviors/actions in certain branches for proactive surveillance, and provide better services to customers.

1.4 PROJECT SCOPE & LIMITATIONS

Project scope

1. Security spy cameras
2. Use in Banks, ATMs, Hospitals, and Government Buildings.
3. Use in Military related applications.
4. Personal use such as for private households.
5. Shopping malls, Cinema halls, jewelry shops, etc.
6. Analysis of human behavior for anomaly detection.

7 Prediction of the anomalies in the scene.

8. Crowd behavior analysis: anomaly detection can be used to monitor crowd behavior in public places such as stadiums, and shopping malls, to detect unusual crowd movements.

8. Traffic monitoring is used in surveillance to detect abnormal traffic patterns such as a sudden change in lanes

Limitations

It can be a bit costly

Video Surveillance footage may not always be of sufficient quality to identify perpetrators of crime.

Poor lighting, camera positioning, and camera resolution can all contribute to poor-quality footage that is not useful for identifying individuals or actions.

Footage can be edited,tempered with,or deleted,which can undermine its usefulness as evidence in a criminal investigation.

1.5 Methodologies of Problem solving

Importing Libraries And Loading Datasets

Firstly imported several libraries including OpenCV, TensorFlow, and MoviePy, and defines several functions to work with image and video data. then importing some basic Python libraries such as NumPy and datetime and sets up inline plotting with matplotlib. then importing several functions from TensorFlow and sci-kit-learn libraries.

We have created a Matplotlib figure that displays a random sample of two videos from the UCF50 dataset. The UCF50 is used for video clips of human behavior, and each clip is classified into one of 2 classes.
The code starts by getting a list of all class names in the UCF50 dataset. Then, it generates a random sample of two class names and selects a random video file from each class.
For each selected video file, the code reads the first frame of the video using OpenCV's VideoCapture function and converts the BGR frame to an RGB frame.
The code iterates through both selected videos and displays them side by side in the figure.

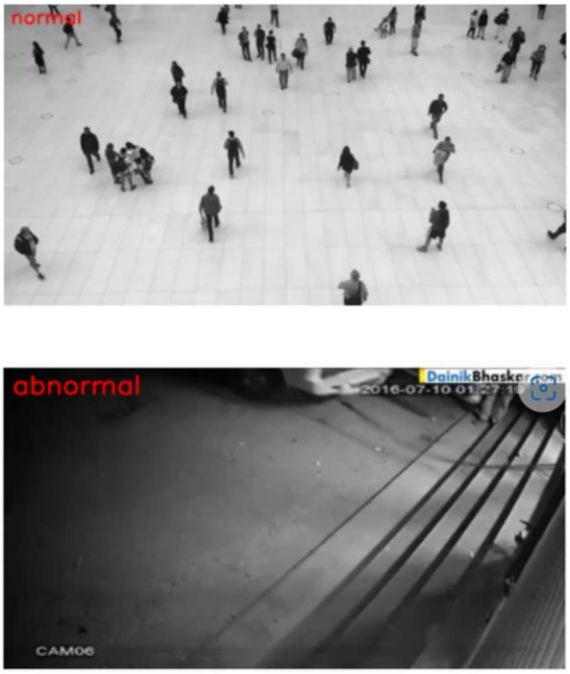


Figure no: 1.5 Normal and Abnormal detection

We have then set up some variables for later use in the code. Image_height and image_width are set to 64.max_images_per_class is set to 400, which is the maximum number of images that will be used per class for training the model.
dataset_directory is set to "UCF50".

classes_list is a list containing the two class names ("Abnormal" and "Normal") that will be used

for training the model and model_output_size is set to 2

Feature Extraction

We have created a Python function that extracts frames from a video file and returns a list of these frames. The function takes in the path of the video file as an argument. Next, the function uses OpenCV's VideoCapture function to read frames from the video file one by one. The frames are resized to a fixed size of image_height and image_width, which are set to 64 in the previous code block.

After resizing, and dividing it by 255 the frame is normalized so that the value lies between 0 and 1 pixel value. The function continues to read frames from the video file and repeats the above process until all frames have been read. Once all frames have been read, the function releases all resources and returns the frames_list.

Actual Working

We have created a function called create_dataset() which is used for extracting the features and labels from the videos of the UCF50 dataset.

Here's how the function works:

1. The function first initializes two empty lists called temp_features and features and one empty list called labels to store the features and labels values.
2. It then iterates through all the classes that are present in the classes_list.
3. For all classes, it gets the list of video files that are present in the specific class name directory.
4. It then iterates through all the files that are present in the files list.
5. It calls the frames_extraction() function to extract the frames from the video.
6. It then adds the frames to a temporary list called temp_features.
7. It adds randomly selected frames from temp_features to the features list and adds a fixed number of labels to the labels list for that particular class.
8. To reuse the temp_feature list we store all frames of the next class
9. Finally, it converts the features and label lists to numpy arrays and returns them.

The create_dataset() function takes a while to execute since it extracts frames from all the videos in the UCF50 dataset, and then creates a balanced dataset with a fixed number of frames from each class. Once the function has been completed, it returns the features and label arrays, which contain the extracted frames and corresponding class labels, respectively. We have used a one-hot-encoded vector to convert Keras's to_categorical method into labels.

The train_test_split function is used to split the dataset for training and testing purposes. input features and one_hot_encoded_labels are the output labels. The parameter called test_size is set to 0.2, which means data from 20% of the dataset is used for testing purposes while resting 80% of the data is used for training purposes. shuffle is set to True, which shuffles the data before splitting, and the random_state is set to seed_constant to ensure reproducibility.

Finally, we stored the training history in the model_training_history variable.

Layered Architecture of Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792

 conv2d_1 (Conv2D)           (None, 60, 60, 64)        36928

 batch_normalization (BatchN  (None, 60, 60, 64)       256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 30, 30, 64)       0
 )

 global_average_pooling2d (G  (None, 64)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               16640

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 57,154
Trainable params: 56,514
Non-trainable params: 640
_________________________________________________________________
Model Created Successfully!

<keras.engine.sequential.Sequential at 0x16b9aec6dc0>
```

Figure no:1.6 Model Creation

The model architecture includes two 2D convolutional layers with 64 filters each, and the model includes a batch normalization layer, a max pooling layer, a global average pooling layer, a dense layer with 256 units, and also another batch normalization layer, and finally, a dense output layer along an activation function called softmax activation. The model takes input images of size 64x64x3. The total no of parameters in the model is 255,811.

Training & Testing

We started the model training using the fit method, passing the training data as input. We have set the total number of epochs to 50, set batch size to 4, and validation split to 0.2. We also passed the early stopping callback to the callbacks parameter so that it can be used during the training.

Finally, we stored the training history in the model_training_history variable.

CHAPTER 2: LITERATURE SURVEY

2.1 Study of Research Paper

Abnormal event detection at 150 fps in Matlab

This IEEE paper [1] is based on the sparse Combination learning framework in which abnormal events are detected through an enormous number of videos. This light Combination learning approach learns sparse combinations and this increases the speed of testing with effectiveness. This process makes use of representation errors to build sparse combinations. those errors are upper-bound errors. This strategy is reliable and efficient Thus they have presented an abnormal event detection method via sparse combination learning and the model is very much faithful input data when compared with original sparse data and the model is verified by a large number of videos. the method is robust that distinguishes between abnormal patterns and normal patterns.

Anomaly detection and localization in crowded scenes

This IEEE paper [2] is based on joint detection and has proposed spatial anomalies. They proposed anomaly detection that spans spatial scale, time, and space too. They introduced some challenges. The first one is to make anomalies dependent on the scales and at multiple scales, the model is defined. The second one is that different models of normalcy are used for different tasks. The third one is that normalcy models are used for the crowded scene which consists of some independently moving objects. Spatial anomaly is a broad term for any kind of extraordinary disruption to normal space-time.

Learning temporal regularity in video sequences

This IEEE paper [3] approaches a problem that is difficult in perceiving meaningful activities in a very long video. They have learned a model which needs very limited supervision. They have proposed two methods which are built on autoencoders and no supervision is needed. In

the first process, They handcrafted spatiotemporal local features and they learned a fully connected autoencoder on them. In the second process, they have to build a fully connected feed-forward autoencoder for learning local features and classification.

Robust real-time unusual event detection using multiple fixed-location monitors

In this paper [4] they have presented an algorithm for abnormal events detection called multiple local monitors here alerts are generated by each local monitor and are attached to the final result. when an abnormal event is detected alerts are generated. The limitation of the algorithm is that it lacks sequence monitoring and is not suitable for large-scale video surveillance projects.

Learning spatiotemporal features with 3d convolutional networks

In this paper [5] they have addressed the problem of learning spatiotemporal features using 3D ConvNets.and they are trained on large-sized video datasets.

The video analysis system of intelligent surveillance based on Bayesian

In this paper, Intelligent Video surveillance is a field of image processing and computer vision that consist of a large number of challenges. applications and requirements determine the algorithm of the total system. Two different technologies provide a high level of security and applications cornerstone for the classification of targets and analysis of behavior and detection of abnormal events .these all are very important in an intelligent video surveillance system.

Theft detection using machine learning

In this paper [7], they have discussed the development of an anti-theft device that detects theft using the camera's motion and generates an alarm when suspicious activity is found. this device is a real-time system with easy to use interface. these systems capture images only when motion exceeds the threshold value. This also helps to save data space by not allowing unused data to enter.

Real-time human action recognition from motion capture data

In this paper [8], they have proposed an approach for human action recognition system .the temporal information helps to improve the classification of normal similar-looking actions this helps us in our paper for the classification of normal and abnormal videos. This representation, however, lacks important properties such as view- and scale invariance

A Simple AI-Powered Video Analytics Framework for Human Motion Imitation

In this paper [9], they have proposed a simple service-oriented video analytics framework. They have introduced a hybrid deep learning method for human recognition and motion. The proposed system can predict a chess player's moves in a video using a flexible and hybrid deep learning model and reproduce motion with the V-REP robot simulator. Here human motion is analyzed with AI.

Interpretable human action recognition in the compressed domain

In this paper [10], they have presented the LRP method to interpret and understand the predictions. LRP propagates classifier decisions and finds voxels in the video. They have demonstrated the localization of actions performed by identifying frames. For future work, they will make use of heatmaps as cues for recomputing the features and then again classify the videos .this is unsupervised preprocessing via LRP and the accuracy rate is high.

Sr no	IEEE Paper	Authors	Advantages	Future Work
1	https://ieeexplore.ieee.org/document/6751449	Cewu Lu , Jianping Shi ,jiaya jia.	they have presented an abnormal event detection method via sparse combination learning and the model is very much faithful input data when compared with original sparse data and the model is verified by a large number of videos. the method is robust that distinguishes between abnormal patterns and normal patterns	Their future work is to extend the sparse combination learning framework to other video applications.
2	https://www.researchgate.net/publication/239943156_Anomaly_Detection_and_Localization_in_Crowded_Scenes	Wei-Xin LI, Vijay Mahadevan	This IEEE paper is based on joint detection and has proposed spatial anomalies. They proposed anomaly detection that spans spatial scale, time, and space too. They have Used MDT models	Future work is to improve frame-level and pixel-level
3	https://ieeexplore.ieee.org/document/7780455	Mahmudul Hasan,Jonghyun Choi, Jan	This IEEE paper approaches a problem that is difficult in	learned models in several ways such as

		Neumann,Amit K. Roy-Chowdhury.	perceiving meaningful activities in a very long video. They have learned a model which needs very limited supervision. They have proposed two methods which are built on autoencoders and no supervision is needed	visualizing the regularity in frames and pixels and predicting a regular video of past and future given only a single image
4	https://ieeexplore.ieee.org/document/4407716	Amit Adam, Ehud Rivlin,Iian Shimshoni,Daviv Reinitz	In this paper they have presented an algorithm for abnormal events detection called multiple local monitors here alerts are generated by each local monitor and are attached to the final result. when an abnormal event is detected alerts are generated.	The limitation of the algorithm is that it lacks sequence monitoring and is not suitable for large-scale video surveillance projects.
6	https://ieeexplore.ieee.org/document/6182132	N Bird, S.Atev, Camaelli, R.Martin	The intelligent video surveillance system has good application prospects in the security field, the detection, tracking, and classification of its targets are key parts of the system	The whole monitoring system is also constrained by two major difficulties, which can easily deal with complex

				environments and explore a variety of changes,
7	https://sci-hub.se/10.1109/ICIP.2019.8803439	Jung Uk Kim Yong Man Ro	proposed a new object detection network considering the distinct difference in object classification and object localization.	proposed methods were effectively encoded each layer by considering two tasks properties.
8	https://sci-hub.se/10.1109/INVENTIVE.2016.7830065	Dr.Vijayalakshmi M.N M.Senthilvadivu	object detection by using threshold values based on intensity of pixels. Detecting objects with clouds, fire, smoke etc needs an appropriate method to get accurate results.	mprovement of an appropriate technique for detecting the objects along with clouds ,smoke,fire etc evidently is a great requirement.
9	https://sci-hub.se/10.1109/ETIICT.2017.7977025	Pallavi S. Bangare Sunil L. Bangare	proposed system is based on algorithms like Blurring, Thresholding, Detection of blob, Motion detection and	In future the system can be made more robust and accuracy with image

			conversion of RGB to HSV.	quality may be enhanced.
10	https://sci-hub.se/10.1109/COMPSC.2014.7032664	Anaswara S Mohan Resmi R	two methods are presented for detection and segmentation of moving objects in videos. First method is for object detection using background subtraction and second method for segmentation using two approaches i.e. thresholding and edge detection	Background Subtraction method is better compared to Thresholding technique.

CHAPTER 3: SOFTWARE REQUIREMENT AND SPECIFIC

3.1 Assumptions and Dependencies

Assumptions and dependencies are important considerations in the design and implementation of video surveillance systems.

Assumptions are the underlying beliefs or expectations that the system relies on in order to function properly. For example, a common assumption in video surveillance is that the cameras will be able to capture clear and detailed images of the area being monitored. Other assumptions might include the reliability of the hardware and software components, the availability of power and internet connectivity, and the cooperation of individuals being monitored.

Dependencies, on the other hand, are the external factors or systems that the video surveillance system relies on in order to operate effectively. For example, a video surveillance system may depend on other security measures such as access control systems, alarm systems, or security personnel to respond to incidents detected by the cameras. The system may also depend on network infrastructure and IT support to ensure that the cameras are properly connected, configured, and maintained.

Identifying and addressing assumptions and dependencies is critical to ensuring that video surveillance systems are effective and reliable. Failing to account for these factors can lead to system failures, inaccurate data, or even security breaches. Therefore, it is important to conduct a thorough analysis of the system and its requirements in order to identify and mitigate any potential risks associated with assumptions and dependencies.

3.2 FUNCTIONAL REQUIREMENT

Functional requirements for video surveillance analysis can vary depending on the specific use case and goals of the analysis. However, some common functional requirements for video surveillance analysis may include:

1. Real-time monitoring: The ability to monitor video feeds in real-time and detect events as they occur.

2. Object detection: The ability to detect and track objects in the video feed, such as people, vehicles, or other relevant objects.

3. Face recognition: The ability to recognize and identify individuals in the video feed using facial recognition technology.

4. Activity recognition: The ability to recognize and classify different types of activities in the video feed, such as walking, running, loitering, or fighting.

5. Event detection: The ability to detect specific events, such as the presence of an unauthorized person in a restricted area, a fire, or an act of violence.

6. Alerts and notifications: The ability to send alerts and notifications to security personnel or other relevant stakeholders in real-time when an event is detected.

7. Analytics and reporting: The ability to analyze video data over time to identify patterns and trends, and generate reports for operational or investigative purposes.

8. Integration with other systems: The ability to integrate video surveillance analysis with other security systems, such as access control, intrusion detection, or alarm systems.

3.3 EXTERNAL INTERFACE REQUIREMENT

3.3.1 User Interface

Application Based Anomaly analysis

3.3.2 Hardware Interfaces:

RAM: 8 GB

As we are using Machine Learning Algorithm and Various High-Level Libraries Laptop The AM minimum required is 8 GB.

Hard Disk: 40 GB

A data set of CT Scan images is to be used hence a minimum of 40 Gof B Hard Disk memory is required.

Processor: Intel i5 Processor

Pycharm IDE that Integrated Development Environment is to be used and data loading should be fast hence Fast Processor is required

IDE: Pycharm

Best Integrated Development Environment as it gives possible suggestions at the time of typing code snippets that make typing feasible and fast.

Coding Language: Python Version 3.5

Highly specified Programming Language for Machine Learning because of the availability of High-Performance Libraries.

Operating System: Windows 10

Latest Operating System that supports all types of installation and development Environment

3.3.3 Software Interfaces

Operating System: Windows 10
IDE: Pycharm,S spyder
Programming Language: Python

3.4 NON FUNCTIONAL REQUIREMENT

3.4.1 PerformanceRequirements

The performance of the functions and every module must be well. The overall performance of the software will enable the users to work efficiently. The performance of the encryption of data should be fast. Performance of the providing virtual environment should be fastSafety Requirement•The application is designed in modules where errors can be detected and easily. This makes it easier to install and update new functionality if required.

3.4.2 Safety Requirement

The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required.

3.4.3 Software Quality Attributes

Our software has many quality attributes that are given below:-

Adaptability: This software is adaptable by all users.

Availability: This software is freely available to all users. The availability of the software is easy for everyone.

Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.

Reliability: The performance of the software is better which will increase the reliability of the Software. User Friendliness: Since the software is a GUI application; the output generated is much user friendly in its behavior.

Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.

Security: Users are authenticated using many security phases to provide reliable security.

Testability: The software will be tested considering all the aspects

3.5 SYSTEM REQUIREMENTS

3.5.1 Database Requirements

Video surveillance analysis requires a database that can store and manage the vast amount of video data generated by cameras. database requirements for video surveillance analysis include:

1. Scalability: The ability to scale the database to accommodate large amounts of video data over time.
2. Performance: The database should be able to handle the high volume of read and write operations required for video surveillance analysis.
3. Security: The database should provide robust security features, including encryption, access control, and authentication, to ensure the privacy and integrity of the video data.
4. Integration: The database should be able to integrate with other security systems, such as access control or alarm systems.

5. Search and retrieval: The ability to search and retrieve specific video clips based on criteria such as date, time, location, or object detected.
6. Metadata management: The ability to store and manage metadata associated with the video data, such as camera location, time stamp, and object recognition data.
7. Analytics support: The database should support advanced analytics, such as machine learning or artificial intelligence, to extract insights and detect patterns from video data.
8. Backup and recovery: The database should provide robust backup and recovery features to ensure that the video data is protected and can be restored in the event of a failure or disaster.

3.5.2 Software Requirements

Software requirements for video surveillance analysis are essential to enable the analysis of the video data captured by surveillance cameras. software requirements for video surveillance analysis include:
1. Video management software (VMS): A VMS is required to manage and control the video data from surveillance cameras. It should be able to handle video data from multiple cameras and provide a unified interface for monitoring and control.
2. Video analytics software: Video analytics software is used to analyze the video data and detect events, objects, or anomalies. It can be used for real-time monitoring or for post-event analysis.
3. Object recognition software: Object recognition software is used to detect and identify objects in the video data, such as people, vehicles, or other relevant objects.
4. Face recognition software: Face recognition software is used to recognize and identify individuals in the video data using facial recognition technology.
5. Activity recognition software: Activity recognition software is used to recognize and classify different types of activities in the video data, such as walking, running, loitering, or fighting.
6. Alarm management software: Alarm management software is used to manage and respond to alarms generated by the video analytics software.
7. Data visualization software: Data visualization software is used to present the video data and analysis results in a visual format, such as graphs, charts, or maps.

8. Integration software: Integration software is used to integrate video surveillance analysis with other security systems, such as access control or intrusion detection systems. It can be used to trigger actions or generate alerts based on the analysis results.

10. User interface: A user interface is required to provide an intuitive and user-friendly way to access and control the video surveillance analysis software. The user interface should be customizable and able to support multiple languages.

3.5.3 Hardware Requirements

The hardware requirements for video surveillance analysis using machine learning depend on the complexity of the algorithms and the size of the data sets. Generally, the more complex the analysis, the more powerful hardware is needed. Some common hardware requirements for machine learning-based video surveillance analysis include:

1. Graphics Processing Units (GPUs): GPUs are highly efficient at processing large amounts of data in parallel, making them an ideal choice for machine learning tasks. They can dramatically reduce the time required to train models and analyze data.

2. Central Processing Units (CPUs): CPUs are the primary component of a computer and can be used for general-purpose computing tasks, including machine learning. They are slower than GPUs but can be used for smaller data sets or less complex analysis.

3. Random Access Memory (RAM): RAM is used to store data and algorithms temporarily while the analysis is performed. More RAM is required for larger data sets and more complex analysis.

4. Solid State Drives (SSDs): SSDs are faster than traditional hard disk drives and are used tostore large data sets and models for analysis.

5. Cameras: High-quality cameras are required to capture video data for analysis. The cameras should be capable of recording high-resolution video and support features such as zoom and pan.

6. Network Infrastructure: A high-speed and reliable network infrastructure is required to transfer large amounts of video data between cameras, servers, and other hardware components.

7. Power Supply: A stable and reliable power supply is required to power the hardware components and ensure that the analysis is not interrupted due to power outages or fluctuations.

8. Cooling: Machine learning tasks generate a lot of heat, and proper cooling is essential to prevent damage to the hardware components and ensure that the analysis is performed at peak performance.

CHAPTER 4:SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

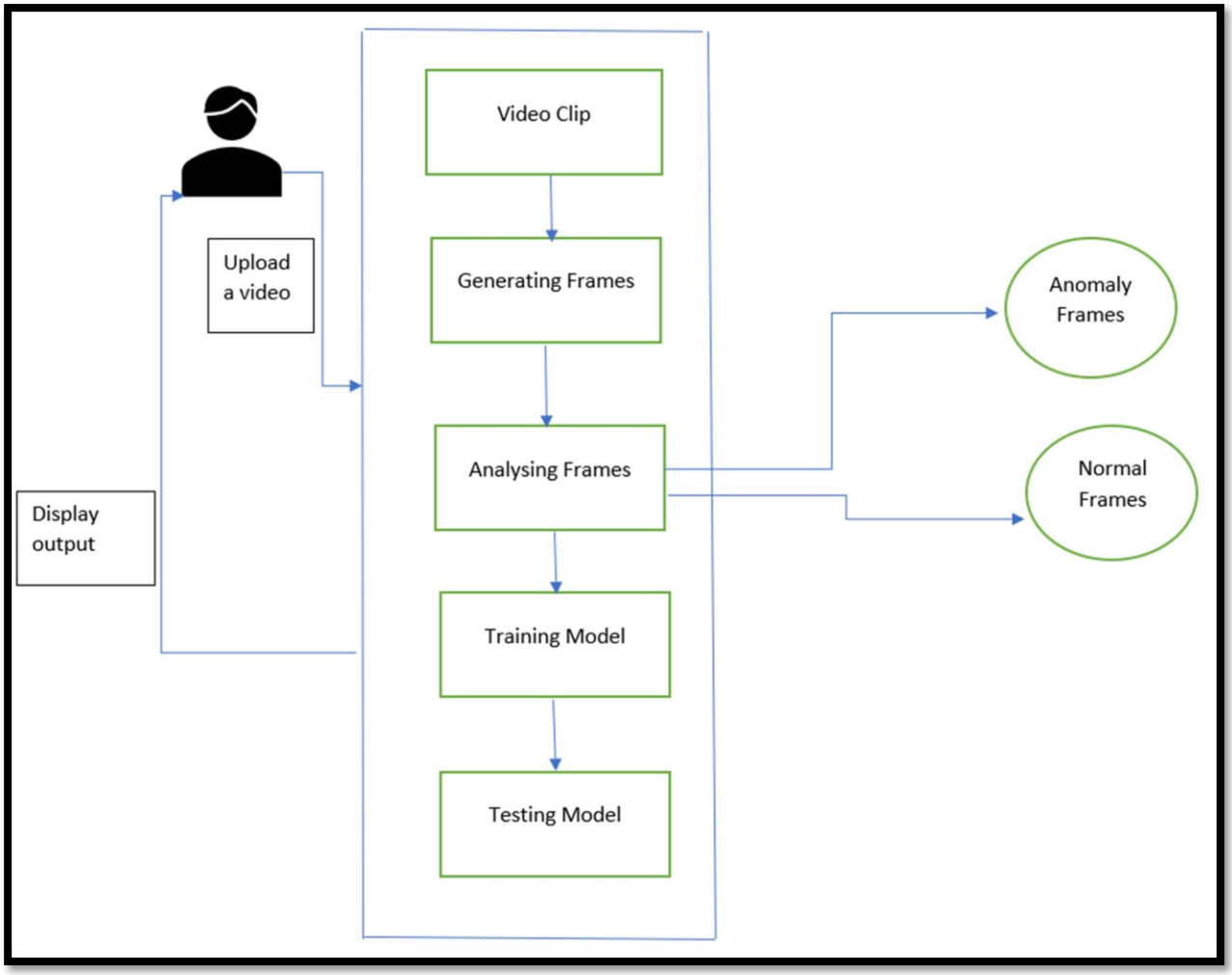


Figure no4.1: System Architecture

4.2 Mathematical Model

ADMIN:

Admin is a user of system .who has to upload a video from the repository output will be displayed whether video clip is normal or abnormal if abnormal then it gives alert messages as an output .

4.3 DATA FLOW DIAGRAM

In Data Flow Diagram, we Show the flow of data in our system in DFD0 we show that base DFD in which the rectangle presents input, as well as output and circle, shows our system, In DFD1 we soothe w actual input and actual output othe f system and output is rumor detected likewise in DFD 2 we present.

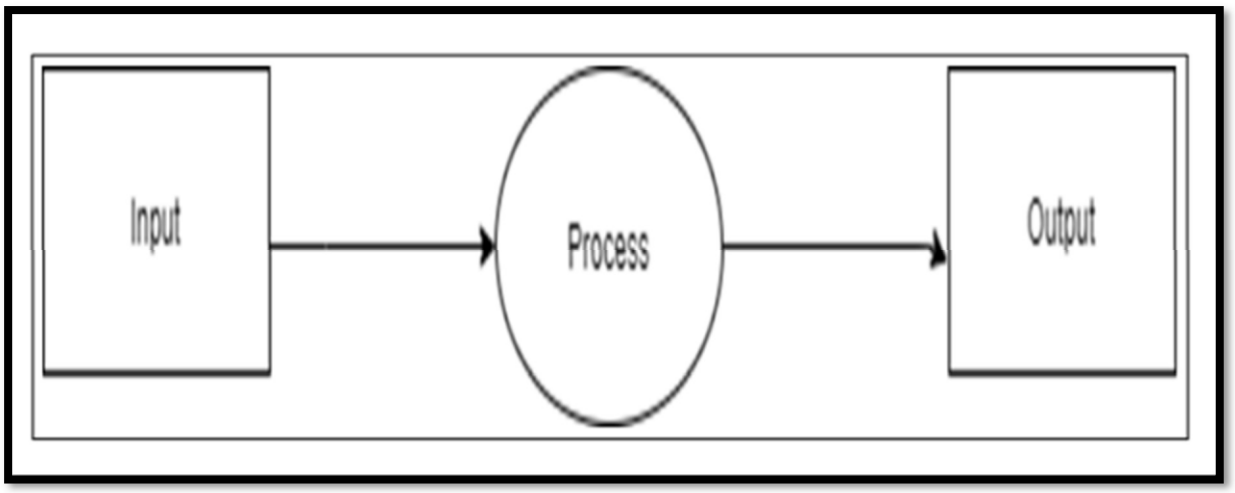


Figure no 4.3.1 Data Flow(0) Diagram

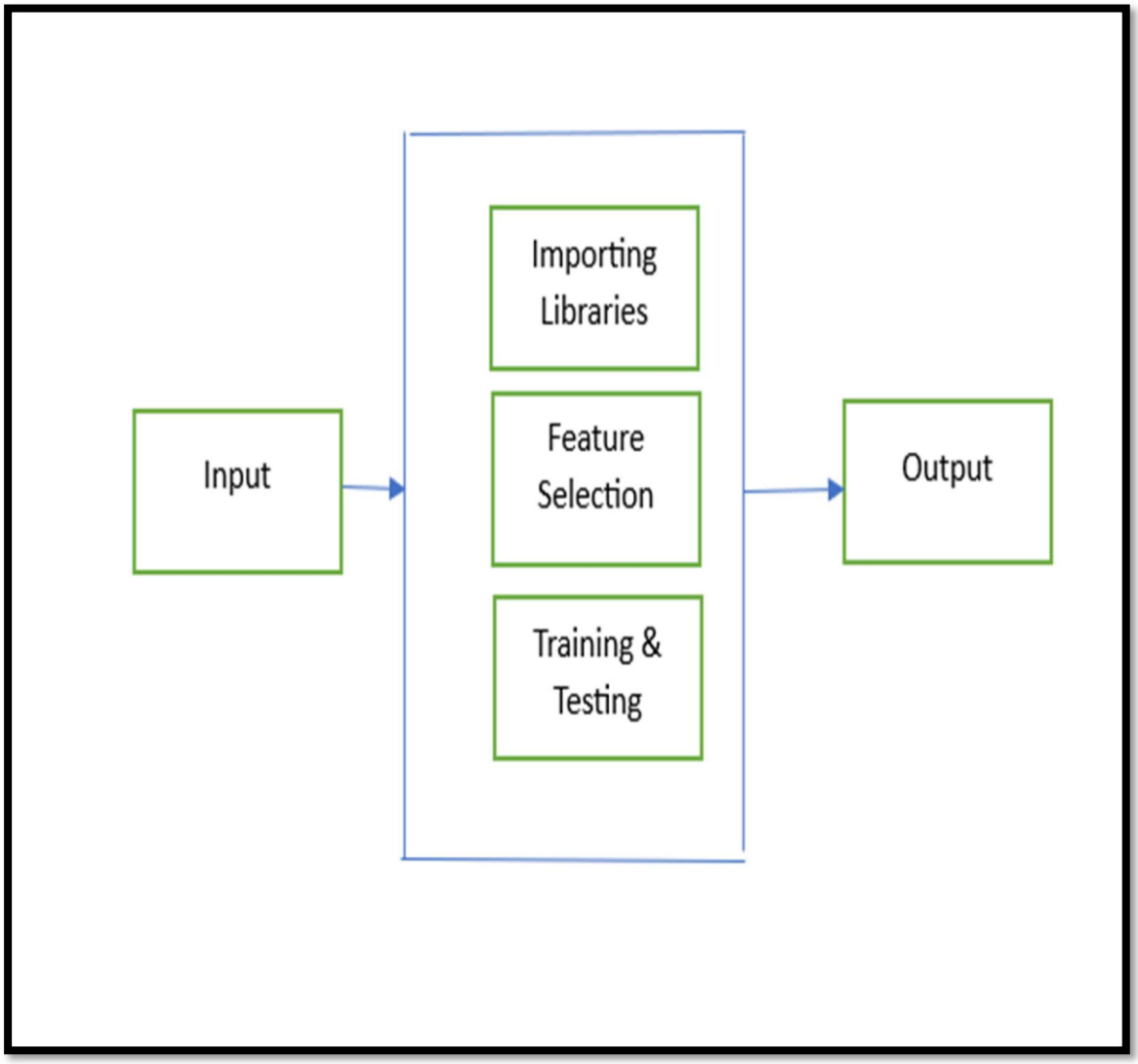


Figure no 4.3.2 Data Flow(1) Diagram

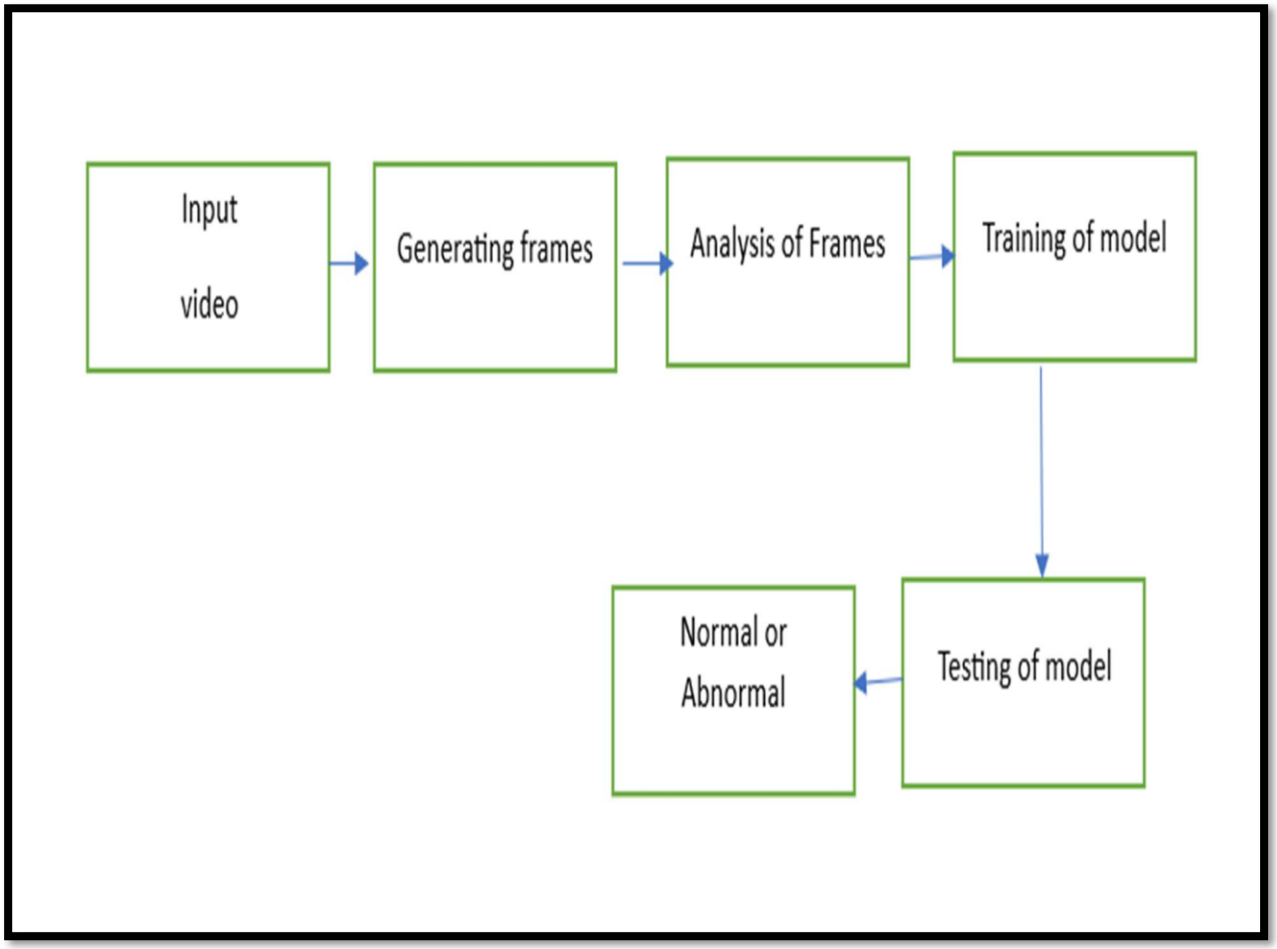


Figure no 4.3.3 Data Flow(2) Diagram

4.4 UML DIAGRAMS

Unified Modeling Language is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct,t and document the artifacts of a software-intensive system.UML is process independent, although optimally it should be used in a process that is use case driven,architecture-centric, iterative, and incremental. The Number of UML diagrams are available.

Class Diagram.

Use case Diagram.

Activity Diagram.

Sequence Diagram.

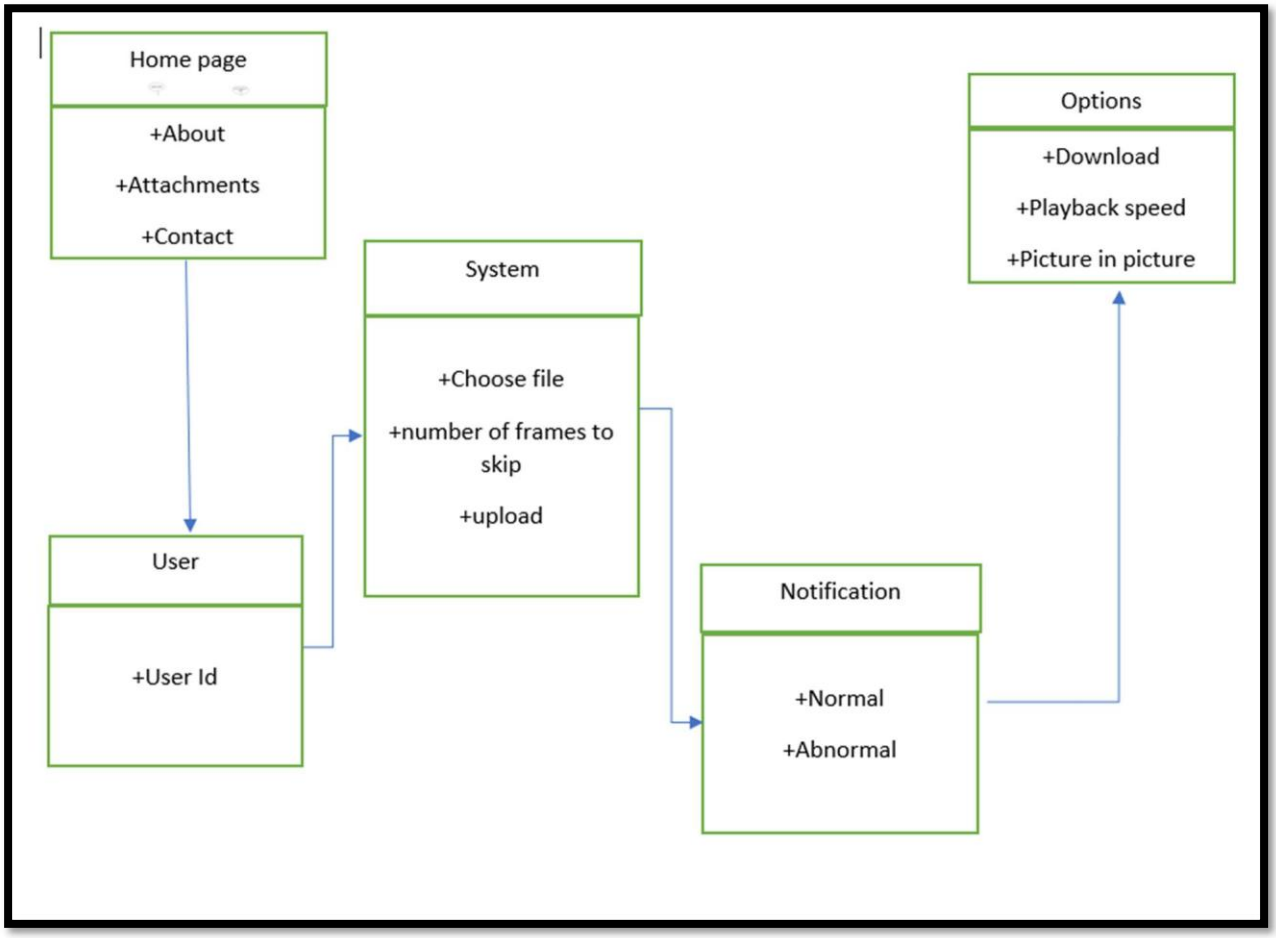


Figure no 4.4.1: Class Diagram

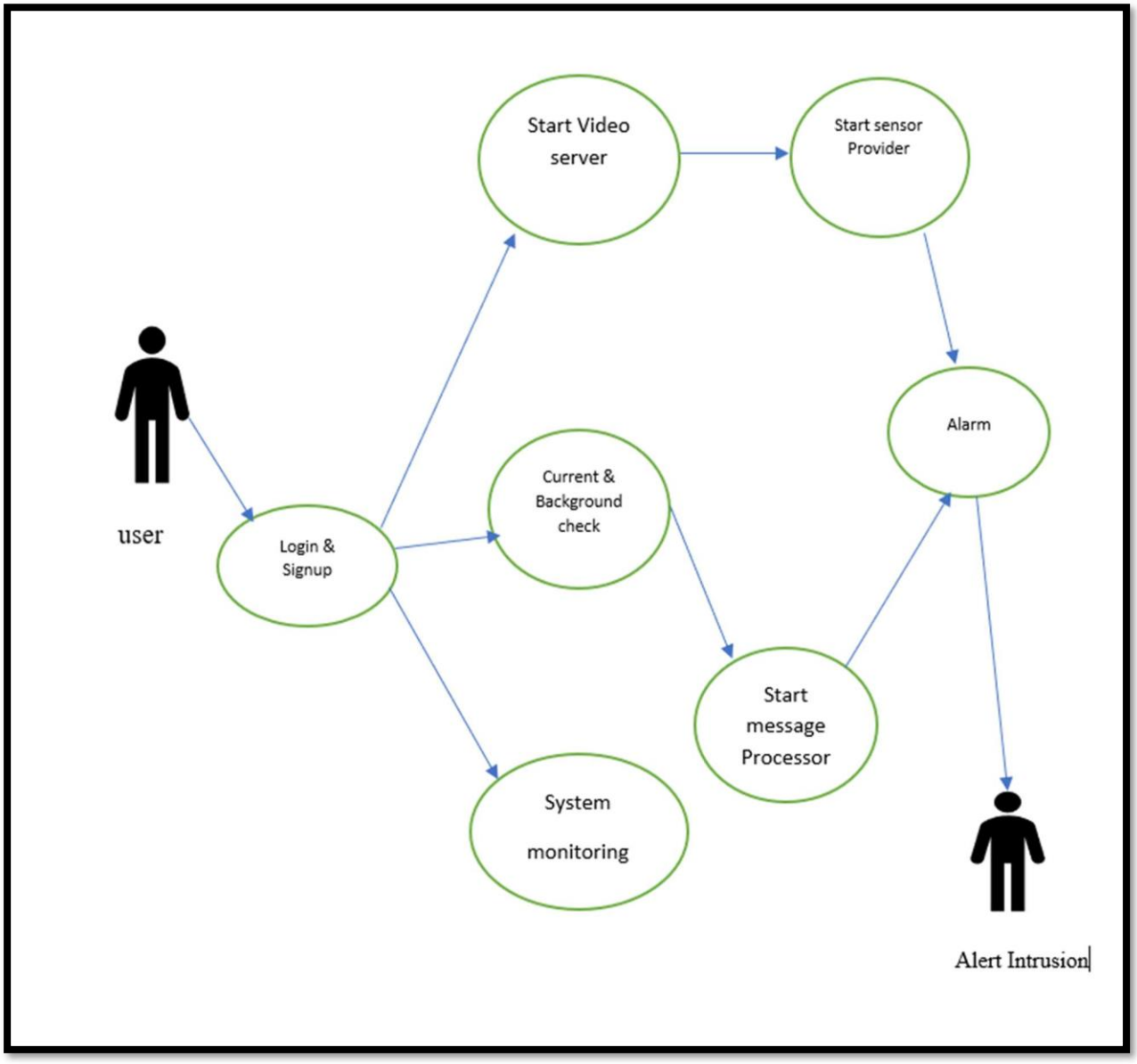


Figure no 4.4.2:UML Diagram

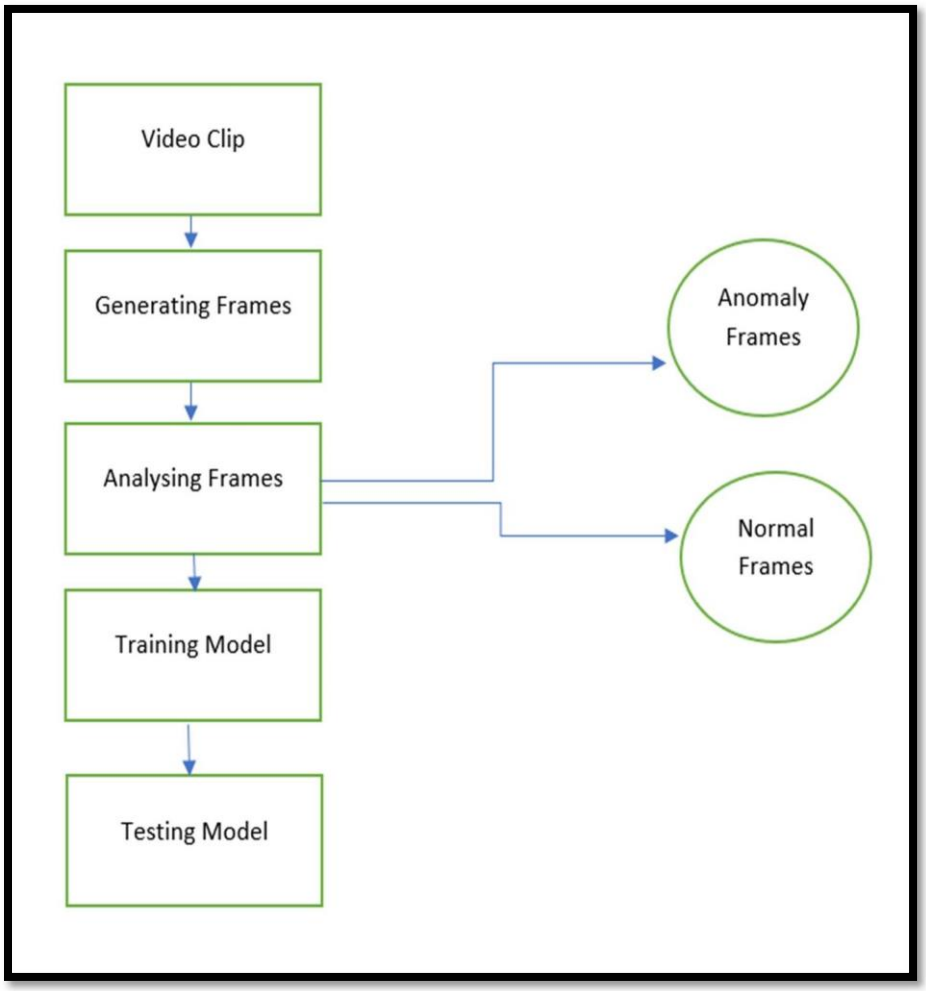


Figure No 4.4.3:Activity flow Diagram

4.4.4 ENTITY RELATIONSHIP DIAGRAM

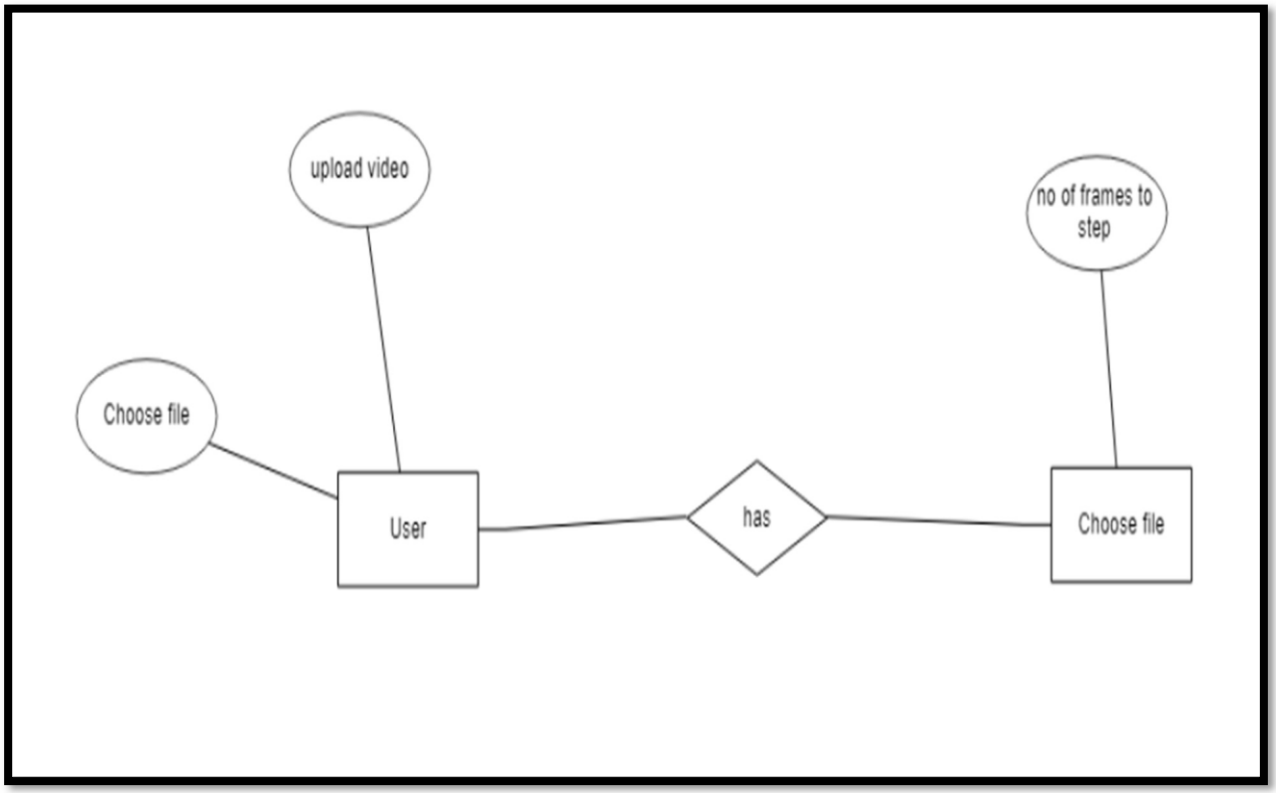


Figure no 4.4.4 Entity Relationship Model

CHAPTER 5 :PROJECT PLAN

5.1 PROJECT ESTIMATE

5.1.1 Reconciled Estimates

Reconciled estimation is a statistical method used to combine multiple sets of data or forecasts into a single estimate or forecast that minimizes the overall error. This method involves using a statistical model to find the optimal combination of the data or forecasts that minimizes the difference between the combined estimate and the actual value. Reconciled estimation is commonly used in fields such as finance, economics, and meteorology to improve the accuracy of predictions and reduce uncertainty.

5.1.2

5.1.2 Project resources

The resources required for the analysis of a video surveillance system project can vary depending on the scope and complexity of the project. However, some common resources that may be required are:

1. Personnel: The project may require a team of experts with skills in areas such as video analysis, machine learning, computer vision, and data analysis.
2. Hardware: The project may require high-end computers or servers with specialized processing capabilities to analyze the video data.
3. Software: Specialized software may be required for video processing, machine learning, and data analysis.
4. Data: The project will require large amounts of video data to analyze and train the machine learning models.
5. Communication and networking: The project may require a network infrastructure to connect the cameras, servers, and other hardware.
6. Training and documentation: The project may require training materials and documentation for the personnel involved in the project.

7. Budget: The project may require a budget to cover the costs of personnel, hardware, software, and other expenses.
8. Time: The project may require a significant amount of time to complete, depending on the complexity of the system and the amount of data to be analyzed.

5.2 RISK MANAGEMENT

5.2.1 Risk Identification

The identification of risks in the analysis of video surveillance systems involves identifying potential threats and vulnerabilities that could negatively impact the system's performance, security, or reliability.

Other potential risk areas could include issues related to the accuracy or reliability of video analytics or machine learning algorithms, the potential for false alarms or missed events, or the impact of environmental factors such as lighting or weather conditions.

To identify risks in the analysis of video surveillance systems, a thorough risk assessment should be conducted, involving input from relevant stakeholders, such as IT professionals, security experts, and end-users. This assessment should consider the likelihood and potential impact of each risk, as well as potential mitigation strategies to reduce or eliminate the risk.

5.2.2 Risk Analysis

Risk analysis for identifying abnormal and normal video in a video surveillance system involves identifying potential risks that could affect the accuracy and reliability of the system in detecting abnormal or normal events.

1. False positives and false negatives: One of the main risks in video surveillance is the potential for false positives, where the system detects an event as abnormal when it is not, or false negatives, where the system fails to detect an abnormal event
2. Environmental factors: Environmental factors such as lighting, weather, and camera placement can also affect the accuracy of video surveillance systems.
3. Security risks: The risk of security breaches and unauthorized access to the video surveillance system is also a concern, as these risks can compromise the accuracy and reliability of the system.

5.2.3 Overview of Risk Mitigation,Monitoring,Management

Risk mitigation, monitoring, and management are critical processes that organizations undertake to identify, assess, and control potential risks that can negatively impact their business operations. Here's an overview of each process:

1. Risk mitigation: Risk mitigation is the process of identifying potential risks and taking proactive steps to minimize their likelihood and impact. This involves implementing risk control measures to reduce the risk, transferring the risk to a third party, or accepting the risk and managing it.

2. Risk monitoring: Risk monitoring involves keeping track of identified risks, evaluating their effectiveness of mitigation measures and identifying any new risks that may arise. This includes regular assessments, periodic reviews, and continuous monitoring to ensure that the risk mitigation measures are working as intended.

3. Risk management: Risk management is a comprehensive process that includes all aspects of identifying, assessing, and mitigating risks. It involves developing a risk management plan, identifying the appropriate risk management techniques to be used, and implementing those techniques. The goal of risk management is to minimize the probability and impact of risks on business operations, assets, and people.

Effective risk mitigation, monitoring, and management require a systematic approach that involves continuous evaluation and improvement. It involves a range of activities, including risk identification, analysis, and prioritization, as well as the development and implementation of risk mitigation plans, risk monitoring, and reporting mechanisms. These activities help organizations to identify, assess, and control risks, thereby reducing the likelihood of negative impacts on their business operations.

CHAPTER 6: PROJECT IMPLEMENTATION

6.1 OVERVIEW OF PROJECT MODULES

Python is an interpreted, high-level, and general-purpose programming language.
Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.
The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release."[30] No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his ”permanent vacation” from his responsibilities as Python’s Benevolent Dictator For Life, a title the Python community bestowed College Short Form Name, Department of Computer Engineering 32 upon him to reflect his long-term commitment as the project’s chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing, and Van Rossum to a five-member ”Steering Council” to lead the project

Anaconda: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between condo and the pip package manager is in how to package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages[citation needed]. It will install a package and any of its dependencies regardless of the state of the existing installation[citation needed]. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working after having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, condo analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user College Short Form Name, Department of Computer Engineering 34 may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open-source packages can be individually installed from the Anaconda reposstory, Anaconda Cloud (anaconda.org), or the user's private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit, and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and the condo will keep track of what it has installed itself and what pip has installed. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI, or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

6.2 TOOLS AND TECHNOLOGIES USED

6.2.1 HARDWARE REQUIREMENTS

System Processors: Core2Duo

Speed: 2.4 GHz

Hard Disk: 150 GB

6.2.2 SOFTWARE REQUIREMENTS

Operating system: 32bit Windows 7 and on words

Framework: Anaconda Navigator

IDE: Spyder

Database: DBSQLITE

Front-end: Tkinter

Back-end: Python

6.2.3 INSTALLATION AND UNINSTALLATION

We need to install the following software to execute our project
Anaconda Navigator

DBSQLITE

Windows 32-bit Windows 7 and on words OS.

6.3 ALGORITHM DETAILS

6.3.1 CNN(Convolutional Neural Network)

The CNN model architecture is a Convolutional Neural Network (CNN) consisting of several layers. The first layer is a Conv2D layer with 64 filters and a kernel size of 3x3. The second layer is also a Conv2D layer with 64 filters and a kernel size of 3x3. The third layer is a BatchNormalization layer, MaxPooling2D layer, GlobalAveragePooling2D layer, Dense layer with 256 neurons and the ReLU function, BatchNormalization layer, and Dense layer with some neurons equal to the number of classes (2 in this case).

In deep learning, Convolution Neural network(CNN) is a class of artificial neural networks(ANN) is commonly used for analyze visual imagery purpose.it is particularly used for finding patterns in images to recognize objects, faces and scenes. CNN is inspired by the human brain which contains nodes and each node is connected to all other nodes.it takes the input from the convolution layer and pass the information to all the layers and with the help of bias and weights the image is detected.by using the sigmoid the image is detected A convolution neural network has tens or hundreds of layers to detect the different features of an image.

A simple CNN consists of an input layer, followed by a stack of a convolutional layer with a certain activation function (CL) and a pooling layer (PL), the fully connected layer, and a final classification activation layer. The convolution layer and pooling layer are used for feature extraction and the layers fully connected layer and output layer is used for classification

purpose. The convolution layer takes the input as input and produce feature maps. And the pooling layer is reduced the size of feature maps .so the usage of memory is reduced by reducing the size of images and it is also avoid the over fitting.

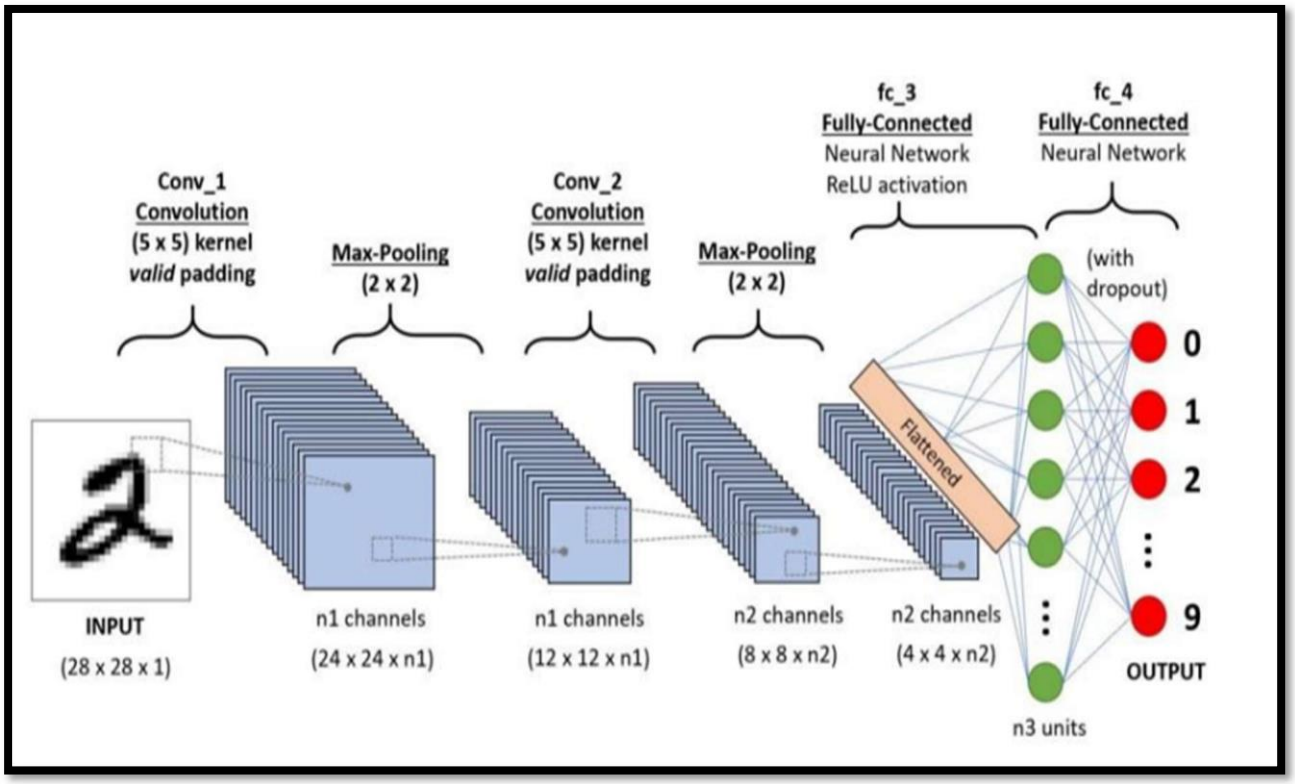


Figure No 6.3.1 CNN(Convolutional Neural Network)

CHAPTER 7: SOFTWARE TESTING

7.1 TYPES OF TESTINGS

software testing is a process of analyzing an application's functionality as per the customer's prerequisite.

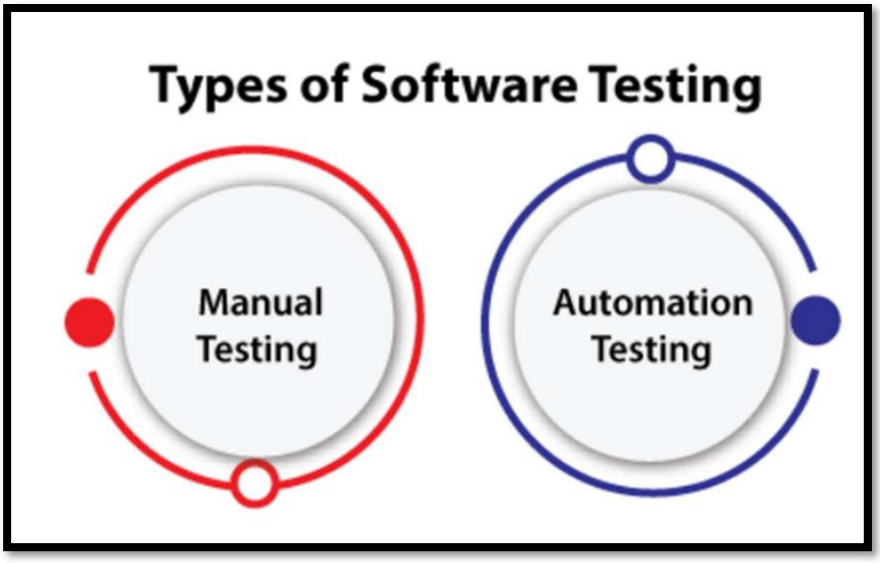


Figure No 7.1: Types of Software Testing

Classification of Manual Testing

In software testing, manual testing can be further classified into **three different types of testing**, which are as follows:

- **White Box Testing**
- **Black Box Testing**
- **Grey Box Testing**

In white-box testing, the developer will inspect every line of code before handing it over to the testing team or the concerned test engineers. we can say that the **developer** will execute the complete white-box testing for the particular software and send the specific application to the

testing team. The purpose of implementing white box testing is to emphasize the flow of inputs and outputs over the software and enhance the security of an application.

In Black Box testing, the test engineer will analyze the software against requirements, identify the defects or bug, and sends it back to the development team. Then, the developers will fix those defects, do one round of White box testing, and send it to the testing team. Here, fixing the bugs means the defect is resolved, and the particular feature is working according to the given requirement. The main objective of implementing black box testing is to specify the business needs or the customer's requirements. In other words, we can say that black box testing is a process of checking the functionality of an application as per the customer's requirement. The source code is not visible in this testing; that's why it is known as **black-box testing**.

7.2 Test case & Test Result

In machine learning, a test case is a set of input data used to evaluate the performance of a trained model. The input data is usually selected to represent a range of scenarios that the model may encounter in real-world use. The test case includes both the input data and the expected output or label for that data. The expected output is typically known in advance and used to measure the accuracy of the model's predictions.

The test case is fed into the model, which generates an output or prediction for each input data point. The model's predictions are then compared to the expected output for each input data point to calculate the accuracy of the model. The accuracy is typically measured using metrics such as precision, recall, F1-score, and confusion matrix.

Test results are the outcomes of running the test cases on the trained machine learning model. These results provide information on the model's performance, accuracy, and limitations. The test results can be used to evaluate the model's overall effectiveness and identify areas for improvement.

CHAPTER 8: RESULTS

8.1 OUTCOMES

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792

 conv2d_1 (Conv2D)           (None, 60, 60, 64)        36928

 batch_normalization (BatchN  (None, 60, 60, 64)       256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 30, 30, 64)       0
 )

 global_average_pooling2d (G  (None, 64)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               16640

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 57,154
Trainable params: 56,514
Non-trainable params: 640
_________________________________________________________________
Model Created Successfully!

<keras.engine.sequential.Sequential at 0x16b9aec6dc0>
```

Figure No 8.1.1 Layered Structure of the proposed model

```
model_evaluation_history = model.evaluate(features_test, labels_test)

5/5 [==============================] - 1s 94ms/step - loss: 0.1243 - accuracy: 0.9812
```

Figure 8.1.2: model evaluation

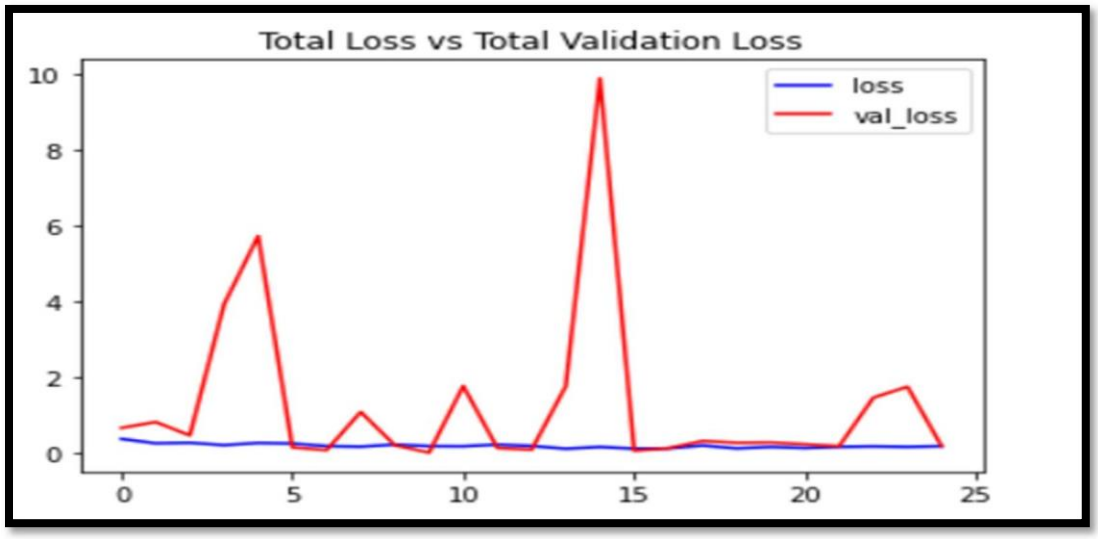


Figure No 8.1.3: Total validation loss

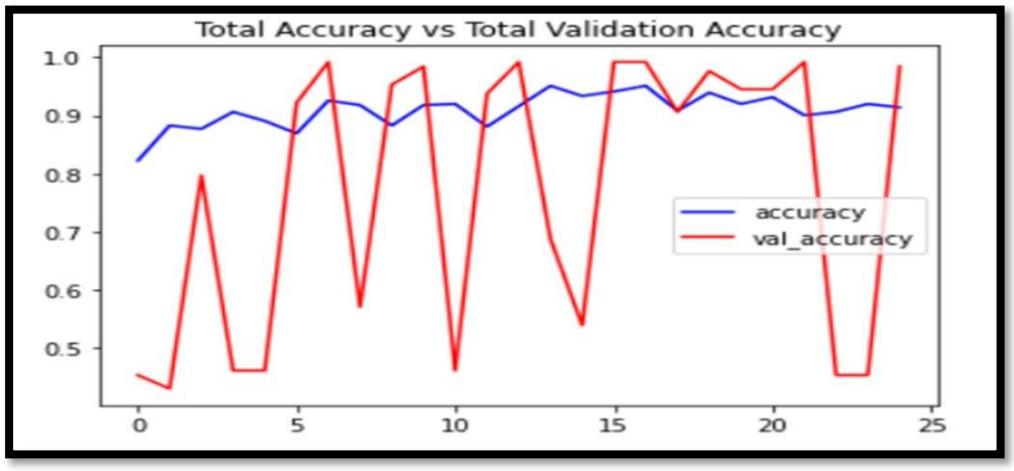


Figure 8.1.4: Total Accuracy vs Total Validation Accuracy.

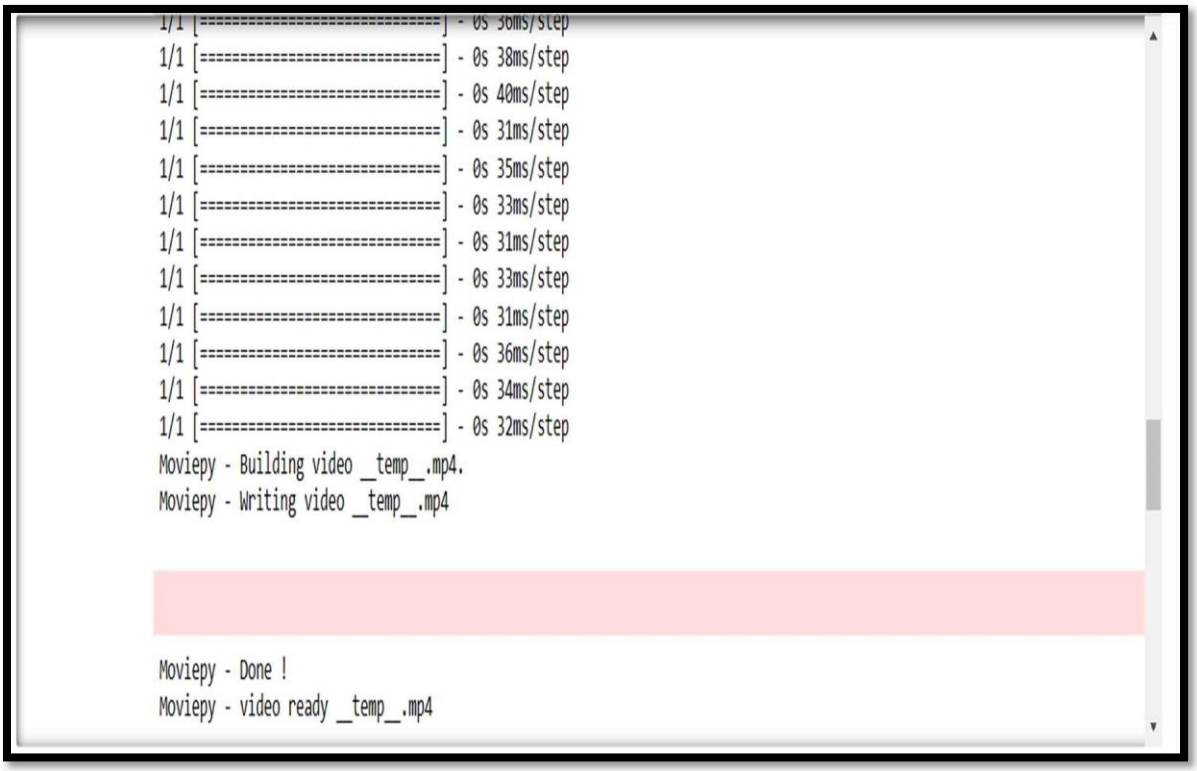


Figure 8.1.5: the actual video successfully uploaded. loading video DSA gfg intro actual video getting sub clip from it and showing a final clip

8.2 OUTPUTS SCREENSHOTS

Figure 8.2:Output that shows the detection of abnormal activity.

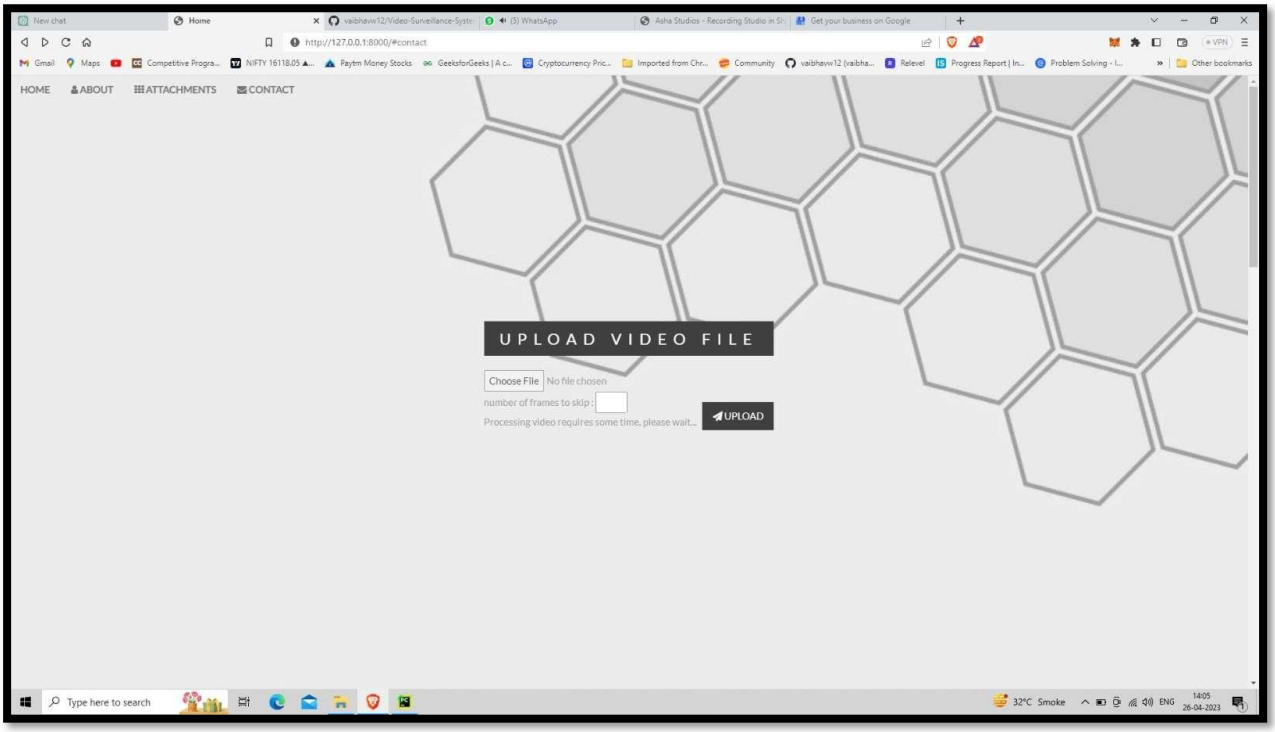


Figure no 8.2 Home page

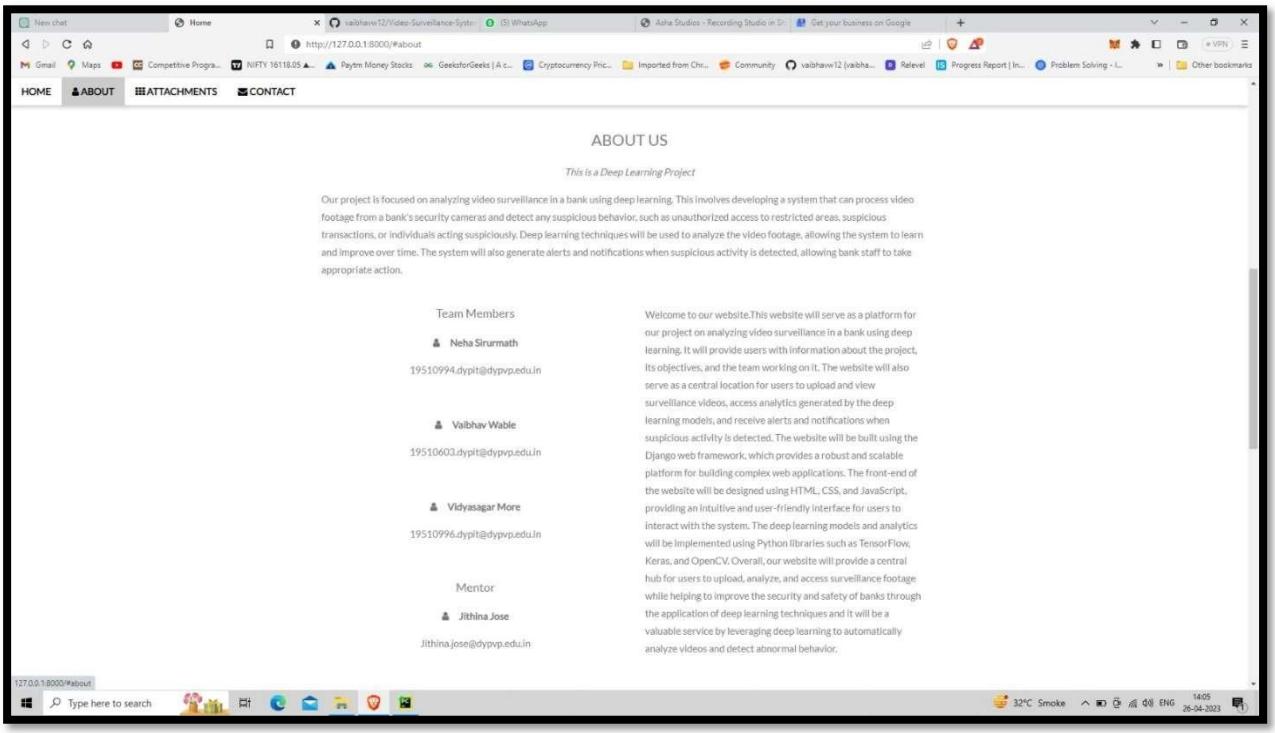


Figure no 8.3 About Us section

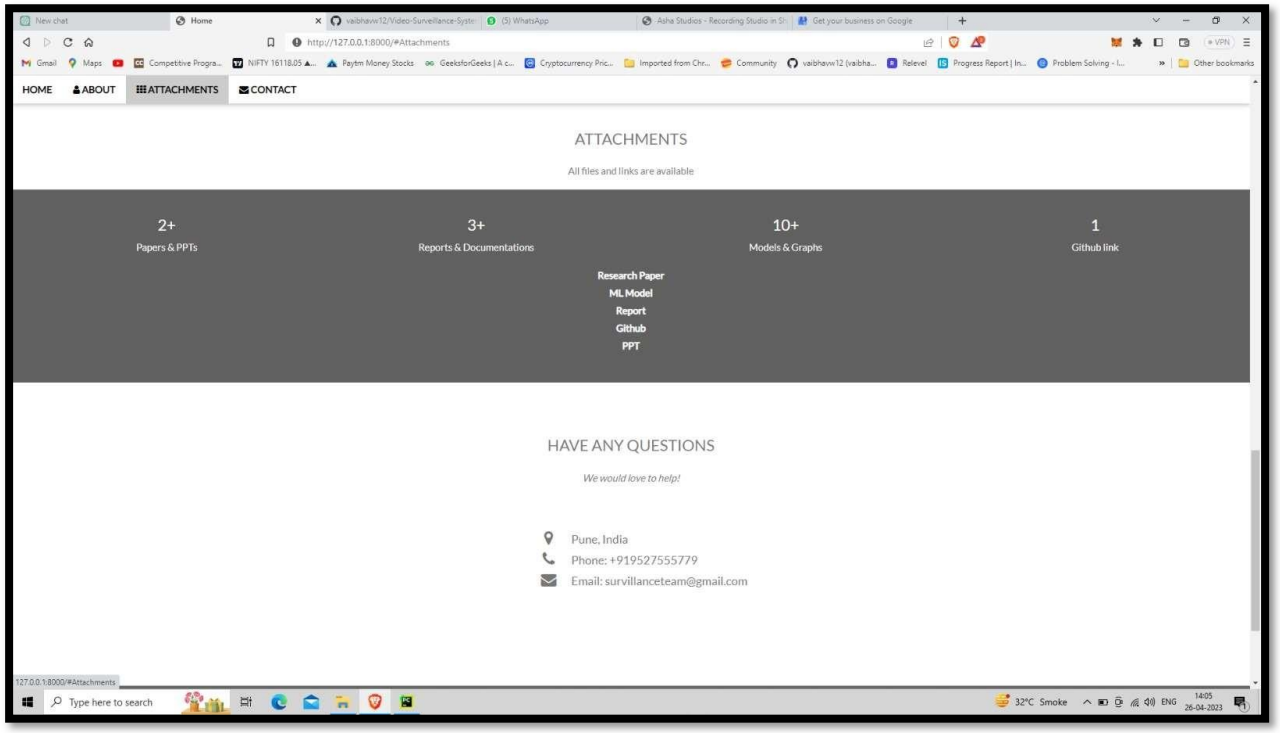


Figure no 8.4 Attachments page

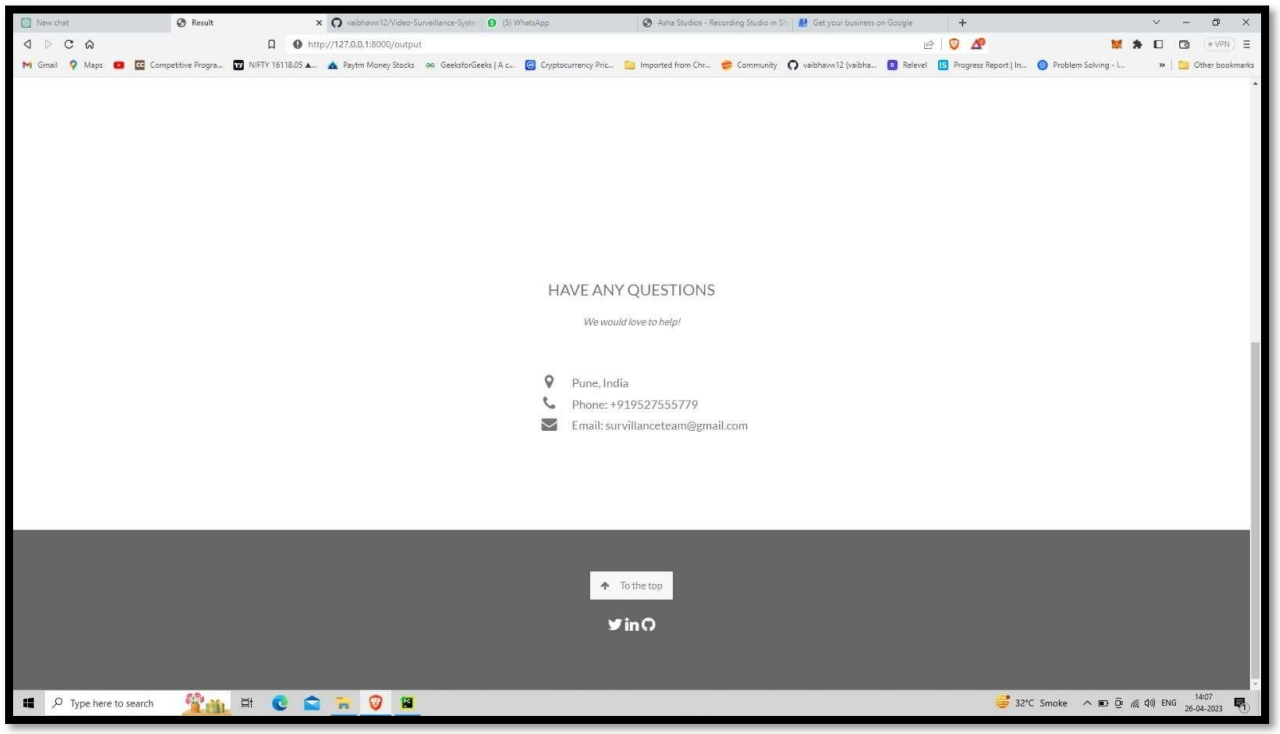


Figure no 8.5 Query page

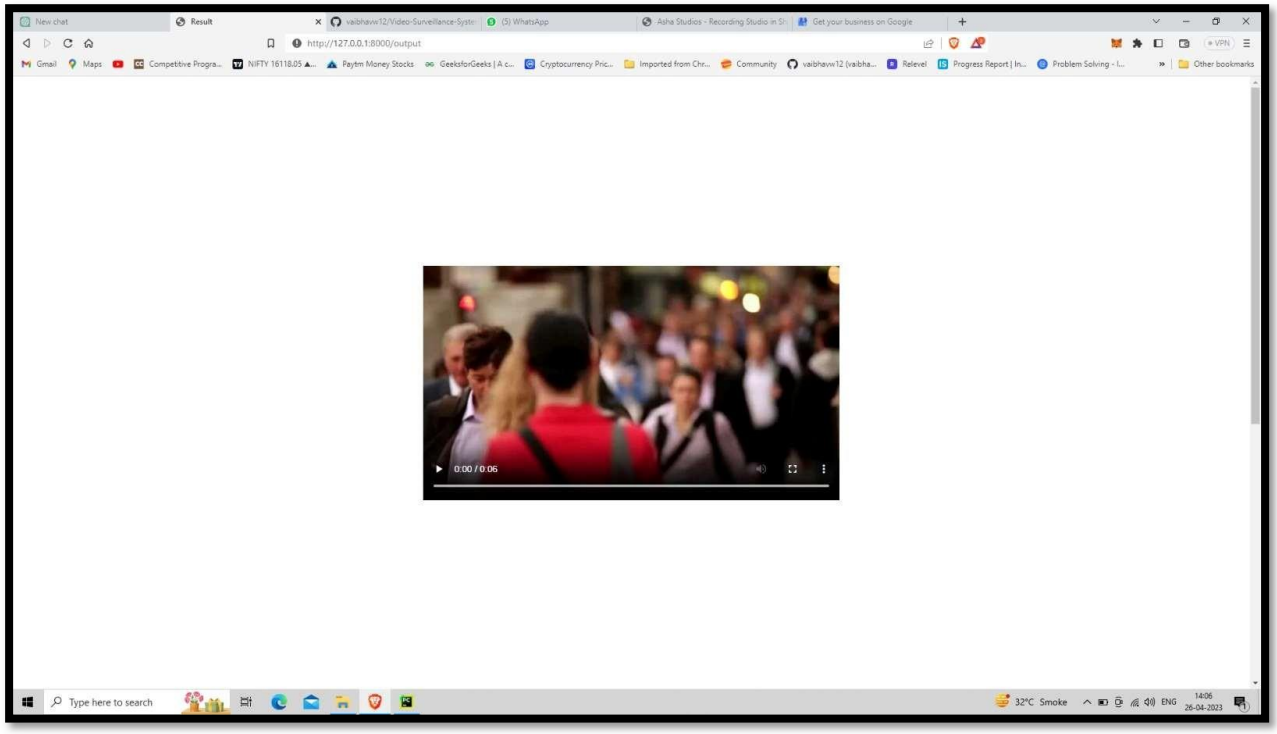


Figure no 8.6 Normal Output

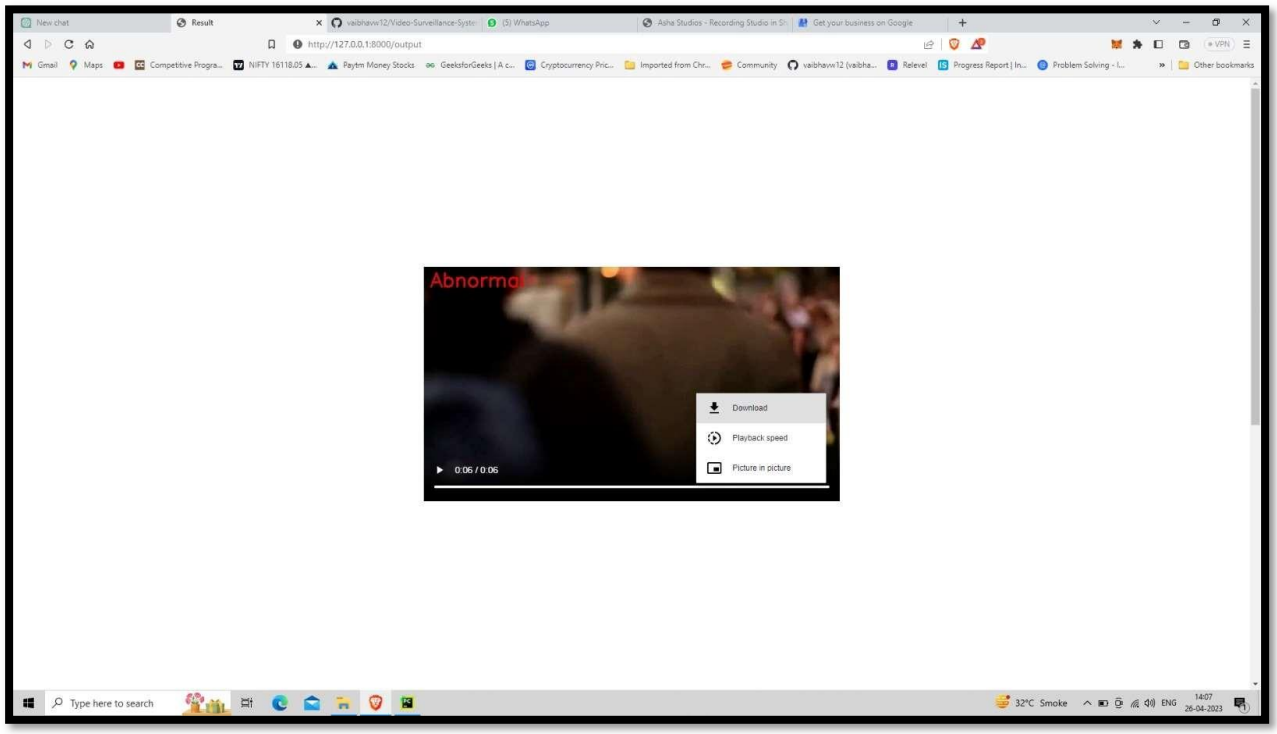


Figure no 8.7 Abnormal Output

CHAPTER 9: CONCLUSION

9.1 CONCLUSION

The research work would focus on designing and developing a video surveillance system to solve security problems which will help reduce abnormal events in banks.

The system will capture the abnormal task from the frames of videos which is given as input to the model and generates an alarm. After successfully implementing the project it can be used in banks, home security systems, museums, and streets at midnight.

9.2 FUTURE WORK

one can categorize Video Surveillance Systems based on the type of imaging modality acquired, producing categories like “one camera systems”, “many camera systems”, “fixed camera systems”, “moving camera systems” and “hybrid camera systems”.

Video Surveillance Systems can be categorized based on the architecture a system is built on, such as stand-alone systems, cloud-aware systems, and distributed systems.

9.3 APPLICATIONS

1. Security spy cameras
2. Use in Banks, ATMs, Hospitals, and Government Buildings.
3. Use in Military related applications.
4. Personal use such as for private households.
5. Shopping malls, Cinema halls, jewelry shops, etc.
6. Analysis of human behavior for anomaly detection.

7 Prediction of the anomalies in the scene.

Appendix A: Problem statement feasibility assessment using, satisfiability analysis and NP Hard, NP-Complete or P type using modern algebra and relevant mathematical models.

Feasibility assessment using satisfiability analysis is a method of determining whether a given problem can be solved by identifying whether a set of constraints or conditions can be satisfied. Satisfiability analysis is a branch of mathematical logic that deals with determining whether a propositional formula can be satisfied by assigning Boolean values (true or false) to its variables.

To perform feasibility assessment using satisfiability analysis, you would first need to convert the problem constraints or conditions into a propositional formula. This can be done by assigning Boolean variables to the different parts of the problem and expressing the constraints as logical expressions involving these variables.

Once the problem has been converted into a propositional formula, you can use satisfiability analysis to determine whether the formula is satisfiable, i.e., whether there exists a set of variable assignments that satisfies all the constraints. If the formula is satisfiable, then the problem is feasible; otherwise, it is infeasible.

Feasibility assessment using NP-hard is a method of determining whether a given problem is computationally feasible or infeasible. NP-hard problems are a class of problems in computer science that are at least as hard as the hardest problems in NP, the class of problems that can be solved in polynomial time by a non-deterministic Turing machine.

To perform feasibility assessment using NP-hard, you would first need to determine whether the problem can be reduced to an NP-hard problem. This involves finding a transformation from the original problem to an NP-hard problem in polynomial time.

If the problem can be reduced to an NP-hard problem, then it is likely to be computationally infeasible in practice. This is because there is no known polynomial-time algorithm that can solve NP-hard problems, and the best-known algorithms for these problems have exponential time complexity.

Therefore, the feasibility assessment using NP-hard can help in deciding whether to pursue the development of an exact algorithm or to settle for an approximate or heuristic solution. Approximate or heuristic methods are often used for NP-hard problems, which can provide suboptimal but reasonably good solutions in a reasonable amount of time.

Examples of NP-hard problems include the traveling salesman problem, the knapsack problem, and the graph coloring problem. These problems have a wide range of practical applications in areas such as logistics, finance, and computer networks.

Therefore, the feasibility assessment using NP-complete can help in deciding whether to pursue the development of an exact algorithm or to settle for an approximate or heuristic solution. Approximate or heuristic methods are often used for NP-complete problems, which can provide suboptimal but reasonably good solutions in a reasonable amount of time.

APPENDIX B:

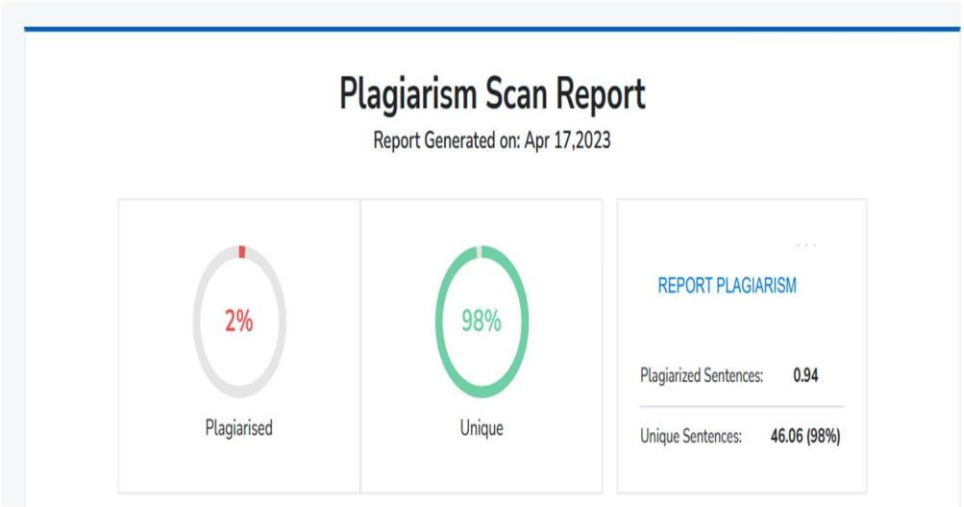
Details of paper publication:

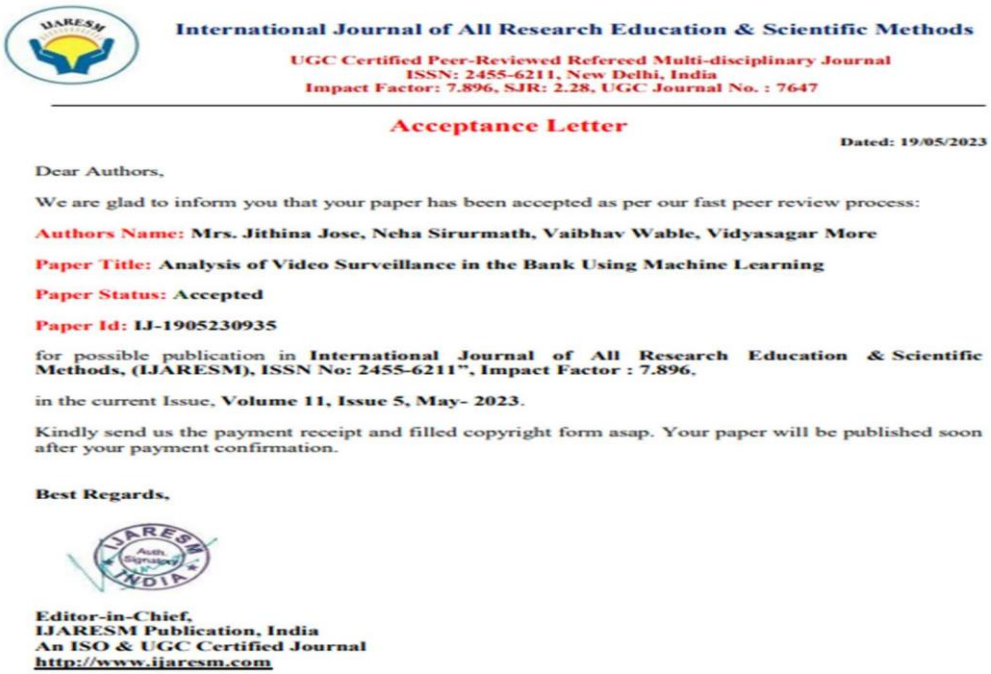
International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, IMPACT FACTOR**: 7.429, SJR: 2.28** is a scholarly online, UGC certified journal, open access, peer-reviewed and fully refereed journal, multi-disciplinary monthly journal focusing on theories, researches, scientific methods and applications in all research areas. It is an international scientific journal that aims to promote research in all the research fields like Engineering, Science, Technology, Education, Management, Medical Sciences, Dental Sciences, Agricultural Sciences, Social sciences, Health Care, Arts & Humanities and many more. IJARESM Publication is **indexed** in Google Scholar**, SJR, Research Gate, Thomson Reuters Researcherid** and also indexed in **UGC Approved List of Journals**.

UGC Journal Details	
Name of the Journal :	International Journal of all research education & scientific methods
ISSN Number :	24556211
e-ISSN Number :	24556211
Source:	UNIV
Subject:	Engineering(all);Management of Technology and Innovation;Management Science and Operations Research
Publisher:	IJARESM Publication
Country of Publication:	India
Broad Subject Category:	Multidisciplinary

name of the conference/journal: **International Journal of All Research Education and Scientific Methods (IJARESM)**

Appendix C: Plagiarism Report of project report.





International Journal of All Research Education & Scientific Methods

UGC Certified Peer-Reviewed Refereed Multi-disciplinary Journal
ISSN: 2455-6211, New Delhi, India
Impact Factor: 7.896, SJR: 2.28, UGC Journal No. : 7647

Acceptance Letter

Dated: 19/05/2023

Dear Authors,

We are glad to inform you that your paper has been accepted as per our fast peer review process:

Authors Name: Mrs. Jithina Jose, Neha Sirurmath, Vaibhav Wable, Vidyasagar More

Paper Title: Analysis of Video Surveillance in the Bank Using Machine Learning

Paper Status: Accepted

Paper Id: IJ-1905230935

for possible publication in **International Journal of All Research Education & Scientific Methods, (IJARESM), ISSN No: 2455-6211", Impact Factor : 7.896,**

in the current Issue, **Volume 11, Issue 5, May- 2023.**

Kindly send us the payment receipt and filled copyright form asap. Your paper will be published soon after your payment confirmation.

Best Regards,

Editor-in-Chief,
IJARESM Publication, India
An ISO & UGC Certified Journal
http://www.ijaresm.com

Copyright Form

Fill this form in same Word file or by Pen and send back to us at: editor.ijaresm@gmail.com

To,
The Editor-in-Chief,
IJARESM Publication,
New Delhi, India

Paper Title: Analysis of Video Surveillance in the Bank Using Machine Learning

Paper Id: IJ-1905230935

Author's Name(s): Vaibhav Wable, Neha Sirurmath, Vidyasagar More

Corresponding Author's full postal address: 16 silver oak D 2 sukhwani campus vallabh nagar pimpri pune 411018

Author's Email (s): vaibhavwable2000@gmail.com

Author's Contact Number(s): 9527555779

Name's / Signature's: ***Date:*** 21/05/2023

Vaibhav Wable

Neha Sirurmath

Vidyasagar More

Analysis of Video Surveillance in the Bank Using Machine Learning

[1]Vidyasagar More
Dr.D.Y.Patil Institute of Technology, Pimpri, Pune
19510996.dypit@dypvp.edu.in
[2]Vaibhav Wable
Dr.D.Y.Patil Institute of Technology, Pimpri, Pune
19510603.dypit@dypvp.edu.in
[3]Neha Sirurmath
Dr.D.Y.Patil Institute of Technology, Pimpri, Pune
19510994.dypit@dypvp.edu.in
Mrs. Jithina Jose Assistant Professor ,
Computer Engineering Department
Dr. D.Y.Patil Institute of Technology, Pimpri, Pune.
Jithina.jose@dypvp.edu.in

……..********……..

Abstract- As we know, theft, fights, and many other abnormal events occur in banks and are increasing daily. To tackle these problems many video surveillance systems are introduced in the market those are based on video surveillance monitored by humans and some are AI-based.

We want to develop an effective surveillance system using machine learning to detect anomalous activities and raise an alarm.

Video Surveillance is an activity of looking at some behaviors that need attention or some anomaly activity that is taking place or to observe a scene that is different from that of a normal one. Video Surveillance is a process where identification takes place in some areas where the chance of happening anomaly activity is high so that these cameras can view those areas.

Keywords: *Surveillance, CCTV, video analytics, ethics, regulation, computer vision,cyber-physical system, and action modeling.*

1. INTRODUCTION

Problem Statement

Bank is using video cameras for surveillance at many branches, ATMs, and digital lobbies. Getting video analytics of different parameters from the video recording will help the bank to resolve many operational issues at the branches. The bank wants to explore video analytics to understand customer sentiments, understand the patterns /behaviors/actions in certain branches for proactive surveillance, and provide better services to customers.

Video Surveillance is the process of identifying activities that are different from that normal one or the one which is anomalous or which is improper in behavior.

This is an automatic video anomaly detection process that reduces labor and waste time. Video Surveillance is very useful to identify abnormal events and maintain social control. In banks, Video Surveillance is used to provide a high level of security and solve financial problems banks will be under control and crimes will be minimized.

The Anomaly detection methods

Supervised

Unsupervised

Semi-Supervised

Supervised are the ones where the training dataset is used to train the model. This Training set contains a normal dataset and an abnormal dataset. The model will learn from the provided dataset and classify the input video as abnormal or normal.

Unsupervised decreases the manual work in anomaly detection. In an unsupervised model is input unlabeled data and the model learns from unlabeled one with the help of algorithms model analyzes and clusters the unlabeled dataset.

Semi-Supervised is a combination of supervised and unsupervised ones, Semi-Supervised contains benefits of both methods and this method uses labeled data of small size and unlabeled data of large size.

The main goal of real-world anomaly detection is to provide a signal upon some abnormal activity taking place.

Therefore, abnormal banking detection can be considered coarse-level video understanding that filters anomaly patterns from normal patterns.

1.1 LITERATURE REVIEW

This IEEE paper [1] presents an abnormal event detection method based on the sparse Combination learning framework. It uses representation errors to build sparse combinations and is reliable and efficient. The model is faithful to original sparse data and is verified by a large number of videos. The method is robust and distinguishes between abnormal patterns and normal patterns

This IEEE paper [2] proposes spatial anomaly detection that spans spatial scale, time, and space. It introduces challenges such as making anomalies dependent on scales, using different models of normalcy for different tasks, and using crowded scenes. Spatial anomaly is a broad term for any kind of extraordinary disruption to normal space-time.

This IEEE paper [3] proposes two methods to solve the problem of perceiving meaningful activities in a long video. The first involves handcrafted spatiotemporal local features and the second involves building a fully connected feed-forward autoencoder for learning local features and classification.

In this paper [4] Multiple local monitors generate alerts when an abnormal event is detected, but lack sequence monitoring and are not suitable for large-scale video surveillance projects.

In this paper [5] they have addressed the problem of learning spatiotemporal features using 3D ConvNets.and they are trained on large-sized video datasets.

[6] Intelligent Video Surveillance is a field of image processing and computer vision that requires a high level of security and applications for classification of targets and analysis of behavior and detection of abnormal events.

In this paper [7], An anti-theft device detects theft using motion and generates an alarm, capturing images only when motion exceeds threshold value to save data space.

In this paper [8], This paper proposes an approach to improve the classification of normal and abnormal videos, but lacks important properties such as view- and scale invariance.

In this paper [9], The proposed system can predict chess player's moves in a video using a flexible and hybrid deep learning model and reproduce motion with AI.

In this paper [10], LRP method propagates classifier decisions and finds voxels, providing unsupervised preprocessing with high accuracy rate.

1.2 METHODOLOGY

We have used CNN ie Convolutional neural network to detect the abnormal activities from the videos.

A. CNN Model Theory

The model architecture is a Convolutional Neural Network (CNN) consisting of several layers. The first layer is a Conv2D layer with 64 filters and a kernel size of 3x3. The activation function used in this layer is the Rectified Linear Unit (ReLU) function. The input shape of the layer is the dimensions of the image (height, width, and some channels) which are 64x64x3. The second layer is also a Conv2D layer with 64 filters and a kernel size of 3x3. The activation function used in this layer is also the ReLU function. Then, there is a Batch Normalization layer to normalize the output of the previous layer. This is followed by a MaxPooling2D layer with a pool size of 2x2 to reduce the dimensions of the output. Then, there is a GlobalAveragePooling2D layer to reduce the dimensions further. The output of this layer is then fed to a Dense layer with 256 neurons and the activation function used is the ReLU function. This is followed by another Batch Normalization layer. Finally, there is another Dense layer with some neurons equal to the number of classes (2 in this case) and the activation function used is the Softmax function

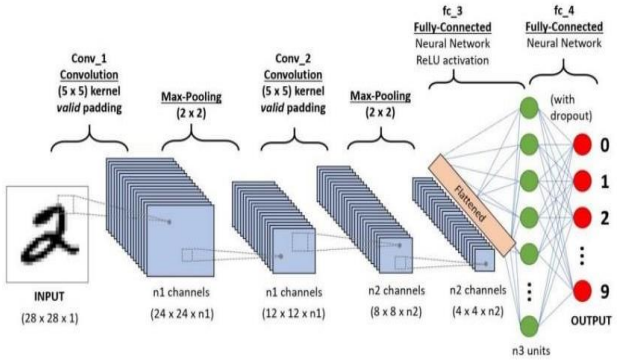
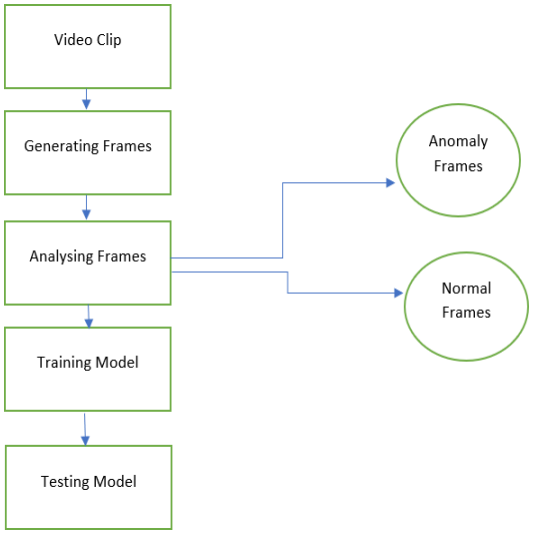


Figure 1:CNN(Convolution neural network)

Figure 2;Data Flow Diagram



B.Comparison

The existing system used deep MIL ie Multiple Instance Learning it is a variation of supervised learning where a single class label is assigned to a bag of instances and they have labeled positive and negative examples and the classifier is learned using function.

While CNN mainly used image classification.CNN is designed to learn spatial hierarchies of features from images using layers.CNN is a Supervised learning technique where the model is trained using the label.

The main difference is CNN & Deep MIL.CNN processes individual images while Deep MIL is designed to process bags of images. Both algorithms have specific characteristics of anomaly and data available for training.

If abnormal activities are well-defined and have distinct features then the CNN approach is appropriate. CNN is effective in learning visual patterns and identifying abnormal events based on visual characteristics.

Hence for video surveillance in a bank, the CNN algorithm is best when compared to Deep MIL.

PROPOSED SYSTEM

Importing Libraries And Loading Datasets

Firstly imported several libraries including OpenCV, TensorFlow, and MoviePy, and defines several functions to work with image and video data. then importing some basic Python libraries such as NumPy and datetime and sets up inline plotting with matplotlib.

Then importing several functions from TensorFlow and sci-kit-learn libraries. importing the train_test_split function from sci-kit-learn for splitting data into training and testing sets, and several functions from TensorFlow including layers, models, and callbacks for building and training neural networks. It also imports the to_categorical function for converting labels into one-hot encoded vectors and the plot_model function
for visualizing the architecture of a model.

We have set the random seed for NumPy, random, and TensorFlow libraries. The constant seed_constant is used as the seed value, and by setting the seed value, the random number generation in these libraries will be reproducible, meaning that the same sequence of random numbers will be generated every time the code is run with the same seed value. This is useful for debugging and ensuring consistent results.

We have created a Matplotlib figure that displays a random sample of two videos from the UCF50 dataset. The UCF50 dataset contains video clips of human actions, and each clip is classified into one of 50 classes.
The code starts by getting a list of all class names in the UCF50 dataset. Then, it generates a random sample of two class names and selects a random video file from each class.
For each selected video file, the code reads the first frame of the video using OpenCV's VideoCapture function, converts the BGR frame to an RGB frame, adds the class name text on top of the frame using OpenCV's putText function, and displays the frame on a subplot of the Matplotlib figure.
The code iterates through both selected videos and displays them side by side in the figure. The resulting figure has a total of 8 subplots, with each subplot showing a different frame from one of the two selected videos.

Figure 1 shows normal and abnormal videos.

We have then set up some variables for later use in the code.

image_height and image_width are set to 64, which is the size of the input images that will be used to train a deep-learning model.
max_images_per_class is set to 400, which is the maximum number of images that will be used per class for training the model.
dataset_directory is set to "UCF50", which is the directory containing the UCF50 dataset.

classes_list is a list containing the two class names ("Abnormal" and "Normal") that will be used for training the model.
model_output_size is set to 2, which is the number of output classes for the model. This is equal to the length of classes_list

A.Feature Extraction

We have created a Python function that extracts frames from a video file and returns a list of these frames. The function takes in the path of the video file as an argument. The function starts by initializing an empty list called frames_list, which will store the video frames.

Next, the function uses OpenCV's VideoCapture function to read frames from the video file one by one. The frames are resized to a fixed size of image_height and image_width, which are set to 64 in the previous code block.

After resizing, the frame is normalized by dividing it by 255 so that each pixel value lies between 0 and 1. The normalized frame is then appended to the frames_list. The function continues to read frames from the video file and repeats the above process until all frames have been read.

Once all frames have been read, the function releases all resources and returns the frames_list.

A. Actual Working

We have created a function called create_dataset() which is used to extract features and labels from the videos of the UCF50 dataset.

Here's how the function works:

1. The function first initializes two empty lists called temp_features and features and one empty list called labels to store the features and labels values.
2. It then iterates through all the classes mentioned in the classes_list.
3. each class, it gets the list of video files present in the specific class name directory.
4. It then iterates through all the files present in the files list.
5. For each video file path, it calls the frames_extraction() function to extract the frames from the video.
6. It then adds the frames to a temporary list called temp_features.
7. It adds randomly selected frames from temp_features to the features list and adds a fixed number of labels to the labels list for that particular class.
8. It then clears the temp_features list so it can be reused to store all frames of the next class.

Finally, it converts the features and label lists to numpy arrays and returns them.

The create_dataset() function takes a while to execute since it extracts frames from all the videos in the UCF50 dataset, and then creates a balanced dataset with a fixed number of frames from each class. Depending on the number of videos in the dataset, this process can take several minutes. Once the function has been completed, it returns the features and label arrays, which contain the extracted frames and corresponding class labels, respectively.

We have used Keras's to_categorical method to convert labels into one-hot-encoded vectors. One-hot encoding is a process of converting categorical data into a format that can be easily understood and processed by machine learning algorithms. In this case, the to_categorical method is applied to the labels array, which contains the class labels for each image in the dataset. The resulting one_hot_encoded_labels variable will contain the one-hot-encoded vectors for each class label.

The train_test_split function from Scikit-learn is used to split the dataset into training and testing sets. features are the input features and one_hot_encoded_labels are the output labels. The test_size parameter is set to 0.2, which means that 20% of the data is used for testing and the

remaining 80% is used for training. shuffle is set to True, which shuffles the data before splitting, and the random_state is set to seed_constant to ensure reproducibility. The split data is returned as features_train, features_test, labels_train, and labels_test.

Finally, we stored the training history in the model_training_history variable.

B.Layered Architecture of Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792

 conv2d_1 (Conv2D)           (None, 60, 60, 64)        36928

 batch_normalization (BatchN  (None, 60, 60, 64)       256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 30, 30, 64)       0
 )

 global_average_pooling2d (G  (None, 64)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               16640

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 57,154
Trainable params: 56,514
Non-trainable params: 640
_________________________________________________________________
Model Created Successfully!

<keras.engine.sequential.Sequential at 0x16b9aec6dc0>
```

Figure 2: layered structure of the proposed model

The model architecture includes two 2D convolutional layers with 64 filters each, a batch normalization layer, a max pooling layer, a global average pooling layer, a dense layer with 256 units, another batch normalization layer, and finally a dense output layer with softmax activation. The model takes input images of size 64x64x3. The total number of parameters in the model is 255,811.

C.Training & Testing

we added an early stopping callback to the model to prevent overfitting. The EarlyStopping callback monitors the validation loss, and if it does not improve for a certain number of epochs (determined by the patience parameter), the training stops early. After adding the callback, we compiled the model using categorical cross-entropy loss as the loss function, Adam optimizer as the optimizer, and accuracy as the metric.

We then started the model training using the fit method, passing the training data (features_train and labels_train) as input. We set the number of epochs to 50, batch size to 4, and validation split to 0.2 (20% of the training data is used for validation). We also passed the early stopping callback to the callbacks parameter so that it can be used during the training.

Finally, we stored the training history in the model_training_history variable.

I.RESULTS

evaluating the trained model on the testing dataset to obtain the overall accuracy of the model. The model. evaluate() method returns a list of two values: the testing loss and the testing accuracy. The first argument of the method is the testing features (i.e., features_test) and the second argument is the testing labels (i.e., labels_test). Note that the accuracy obtained on the testing dataset provides a metric of how well the model performs on unseen data and is a good way to ensure that the model is not overfitting to the training dataset.

```
model_evaluation_history = model.evaluate(features_test, labels_test)

5/5 [==============================] - 1s 94ms/step - loss: 0.1243 - accuracy: 0.9812
```

Figure 3: model evaluation

We have saved the trained model to a file with a name that includes the current date and time, as well as the model's evaluation loss and accuracy.

The model. save() method saves the model to a file in the Hierarchical Data Format (HDF5) format with the .h5 extension. This allows you to load the model later and use it for making predictions on new data. This will show a graph figure with the training and validation loss on the y-axis and the number of epochs on the x-axis. The blue line represents the training loss and the red line represents the validation loss.

The graph shows how the loss decreases over time as the model trains. The goal is to have both lines decrease, but not to overfit the training data by having the validation loss increase while the training loss decreases.

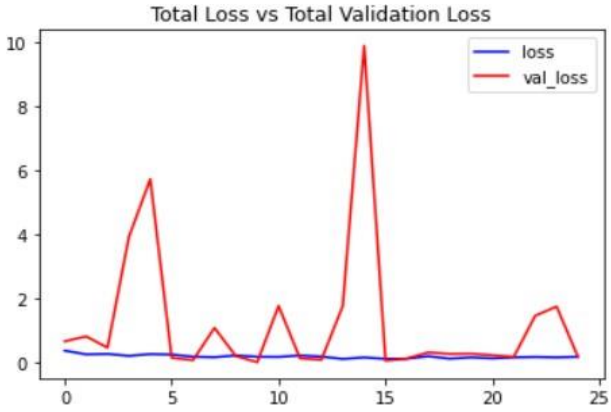


Figure 4: Total validation loss

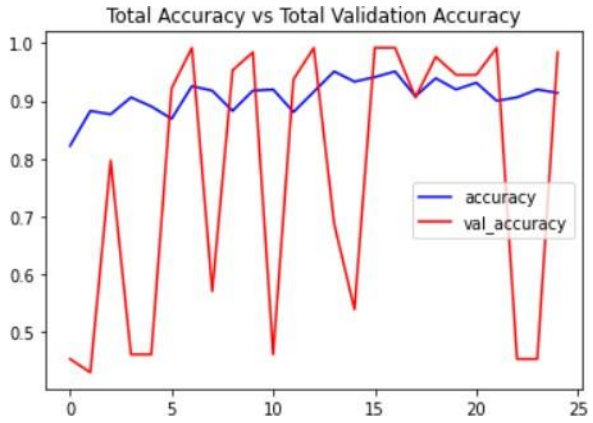


Figure 5: Total Accuracy vs Total Validation Accuracy.

In this graph, the blue line represents accuracy and the red color line represents val_accuracy.

We have created a function called predict_on_live_video() and initialized a Deque Object with a fixed size which will be used to implement moving/rolling average functionality and the Video File is read using the video capture Object. Writing the Overlayed Video Files Using the Video Writer Object. the frames of videos are read and written. these frames are Resized to fixed Dimensions. Normalize the resized frame by dividing it by 255 so that each pixel value then lies between 0 and 1. Passing the Image Normalized Frame to the model and receiving Predicted Probabilities. Appending predicted label probabilities to the deque object. we Assure you that the Deque is filled before starting the averaging process and then we convert Predicted Labels Probabilities Deque into a Numpy array. Then we Calculate the Average of Predicted Labels Probabilities Column Wise and Convert the predicted probabilities into labels by returning the index of the maximum value. Accessing The Class Name using a predicted label. Overlaying Class Name Text On top of the Frame.

We have then loaded the input video mp3 which is an anomalous one and then Called the function predict_on_live_video() method to start the Prediction.

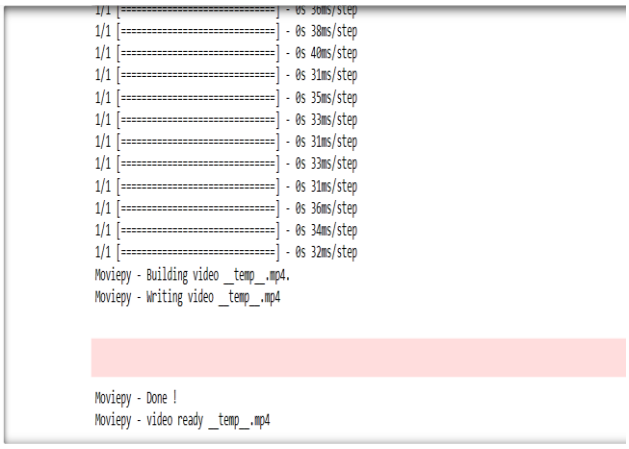


Figure 6: the actual video successfully uploaded.

loading video DSA gfg intro actual video getting sub clip from it and showing a final clip

Figure 7: Output that shows the detection of abnormal activity**.**

Here is the output, these are the frames of an abnormal video that detect abnormal activities by displaying abnormal on top with red color.

1ACKNOWLEDGE

We would like to express our deepest gratitude to all those who provided us with the possibility to complete this task. Special gratitude to our project guide, **Prof. Jithina Jose Mam**, whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project. We also appreciate the guidance panel members gave, which helped us improve our presentation, and thanks to their comments and advice. We are also thankful to our parents who provided their wishful support for our project completed successfully. And lastly, we thank our all friends and the people who are directly or indirectly related to our project work.

1.1CONCLUSION

The research work would focus on designing and developing a video surveillance system to solve security problems which will help reduce abnormal events in banks.

The system will capture the abnormal task from the frames of videos which is given as input to the model and generates an alarm. After successfully implementing the project it can be used in banks, home security systems, museums, and streets at midnight.

REFERENCES

[1] Lu, Cewu, Jianping Shi, and Jiaya Jia. "Abnormal event detection at 150 fps in Matlab." In *Proceedings of the IEEE international conference on computer vision*, pp. 2720-2727. 2013.

[2] Li, W., Mahadevan, V. and Vasconcelos, N., 2013. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, *36*(1), pp.18-32.

[3] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., & Davis, L. S. (2016). Learning temporal regularity in video sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 733-742).

[4] Adam, Amit, Ehud Rivlin, Ilan Shimshoni, and Daviv Reinitz. "Robust real-time unusual event detection using multiple fixed-location monitors." *IEEE Transactions on pattern analysis and machine intelligence* 30, no. 3 (2008): 555-560.

[5] Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." In *Proceedings of the IEEE international conference on computer vision*, pp. 4489-4497. 2015.

[6] Genshan, Zhang, Wu Qiuhong, Geng Kai, and Han Guodong. "Video analysis system of intelligent surveillance based on Bayesian." In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, vol. 2, pp. 1008-1011. IEEE, 2011.

[7] Kushwaha, Ajay, Ankita Mishra, Komal Kamble, Rutuja Janbhare, and Amruta Pokhare. "Theft detection using machine learning." *IOSR Journal of Engineering (IOSRJEN)* 8 (2018): 67-71.

[8] Vantigodi, S., & Babu, R. V. (2013, December). Real-time human action recognition from motion capture data. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)* (pp. 1-4). IEEE.

[9] Tanberk, S., Tükel, D. B., & Uysal, M. (2020, October). A Simple AI-Powered Video Analytics Framework for Human Motion Imitation. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE.

[10] Srinivasan, Vignesh, et al. "Interpretable human action recognition in the compressed domain." *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

REFERENCES

[1] Lu, Cewu, Jianping Shi, and Jiaya Jia. "Abnormal event detection at 150 fps in Matlab." In *Proceedings of the IEEE international conference on computer vision*, pp. 2720-2727. 2013.

[2] Li, W., Mahadevan, V. and Vasconcelos, N., 2013. Anomaly detection and localization in crowded scenes. *IEEE Transactions on pattern analysis and machine intelligence*, *36*(1), pp.18-32.

[3] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., & Davis, L. S. (2016). Learning temporal regularity in video sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 733-742).

[4] Adam, Amit, Ehud Rivlin, Ilan Shimshoni, and Daviv Reinitz. "Robust real-time unusual event detection using multiple fixed-location monitors." *IEEE Transactions on pattern analysis and machine intelligence* 30, no. 3 (2008): 555-560.

[5] Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." In *Proceedings of the IEEE international conference on computer vision*, pp. 4489-4497. 2015.

[6] Genshan, Zhang, Wu Qiuhong, Geng Kai, and Han Guodong. "Video analysis system of intelligent surveillance based on Bayesian." In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, vol. 2, pp. 1008-1011. IEEE, 2011.

[7] Kushwaha, Ajay, Ankita Mishra, Komal Kamble, Rutuja Janbhare, and Amruta Pokhare. "Theft detection using machine learning." *IOSR Journal of Engineering (IOSRJEN)* 8 (2018): 67-71.

[8] Vantigodi, S., & Babu, R. V. (2013, December). Real-time human action recognition from motion capture data. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)* (pp. 1-4). IEEE.

[9] Tanberk, S., Tükel, D. B., & Uysal, M. (2020, October). A Simple AI-Powered Video Analytics Framework for Human Motion Imitation. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE.

[10] Srinivasan, Vignesh, et al. "Interpretable human action recognition in the compressed domain." *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.