**Project Progress Report on**



**DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS**



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

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)**

**Submitted by:**

**SHRISTY CHAUDHARY** **2019498**

***Under the Guidance of***

**Mr. Priyank Pandey**

**Asst. Professor**

**Project Team ID: MP24ML010**

**Project Progress Report No: 2**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**2024-25**



**CANDIDATE’S DECLARATION**

I/We hereby certify that the work which is being presented in the project progress report entitled **“DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering **(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)** in the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun shall be carried out by the undersigned under the supervision of Mr. Priyank Pandey**, Asst. Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

SHRISTY CHAUDHARY 2019498

The above mentioned students shall be working under the supervision of the undersigned on the **“DEEP LEARNING BASED SMART SECURITY SURVEILLANCE SYSTEM FOR SAFETY ENHANCEMENTS”**

**Supervisor** **Head of the Department**

**Examination**

**Name of the Examiners: Signature with Date**

1. Dr. Mohd. Rehan

2. Mr. Piyush Agarwal

3. Mr. Vivek Tomar

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction and Problem Statement | **4-5** |
| Chapter 2 | Objectives | **6-6** |
| Chapter 3 | Project Work Carried Out | **7-11** |
| Chapter 4 | Future Work Plan | **12-12** |
| Chapter 5 | Weekly Task | **13-14** |
|  | References | **15-16** |

**Chapter 1**

**Introduction and Problem Statement**

In the following sections, a brief introduction and the problem statement for the work has been included.

* 1. **Introduction**

In today's world, security is paramount, and technology plays a crucial role in enhancing surveillance capabilities. The Smart Security Surveillance System leverages advanced machine learning techniques, specifically Mask R-CNN, to detect and segment objects in real-time from security camera feeds. This system aims to provide a comprehensive solution for monitoring and analyzing surveillance footage, offering enhanced security through automated detection and alert mechanisms. By integrating state-of-the-art computer vision models, the system can identify and track various objects, such as people and vehicles, in different environments, ensuring accurate and reliable surveillance.

The core of the Smart Security Surveillance System is its ability to process video feeds in real-time, using Mask R-CNN for precise object detection and segmentation. Mask R-CNN, a powerful convolutional neural network, enables the system to not only identify objects within a frame but also delineate their boundaries, providing a clear and detailed view of the monitored area. This capability is essential for distinguishing between different objects and understanding their interactions, which is crucial for detecting suspicious activities. The system's backend, developed using frameworks like Django or Flask, handles the video processing, alert generation, and data management, ensuring seamless integration and efficient performance.

The user interface, built with React.js, offers a responsive and interactive platform for monitoring live video feeds and reviewing detected events. Through real-time data transmission using WebSockets, users can receive instant alerts about potential security threats, allowing for timely responses. The system's ability to send notifications via various channels, such as email and SMS, ensures that users are always informed about critical events. By combining advanced machine learning models with robust web technologies, the Smart Security Surveillance System provides a reliable and efficient solution for modern surveillance needs, enhancing security and situational awareness in diverse environments.

**1.2** **Problem Statement**

The problem at hand revolves around the Traditional surveillance systems rely heavily on continuous manual monitoring, which is labor-intensive, prone to human error, and often ineffective in real-time threat detection. There is a critical need for an advanced, automated solution that can accurately detect and segment objects in live video feeds, generate timely alerts for suspicious activities, and provide detailed visual insights to enhance security operations. The Smart Security Surveillance System addresses these challenges by leveraging Mask R-CNN for real-time object detection and segmentation, combined with robust web technologies to deliver a scalable, efficient, and cost-effective surveillance solution.

**Chapter 2**

**Objectives**

The proposed work objectives are as follows:

1. To Implement a robust Mask R-CNN model to detect and segment objects in real-time from security camera feeds.
2. To Develop algorithms to identify and alert users about suspicious activities based on detected objects and predefined rules.
3. To Optimize the system for high-performance video processing to ensure smooth and efficient real-time surveillance.
4. To Create an intuitive web application interface for monitoring live feeds, reviewing alerts, and managing system settings.
5. To Implement a reliable data storage solution for saving video footage, detected objects, and alert logs.
6. To Incorporate machine learning techniques to detect anomalies in the video feed, enhancing the system's capability to identify unusual patterns or behaviors.
7. To Design the system to be scalable and flexible, allowing easy integration of additional cameras and sensors as needed.
8. To Develop a robust alert mechanism that notifies users through multiple channels (e.g., email, SMS, in-app notifications) in case of detected suspicious activities.
9. Implement comprehensive monitoring and logging to track system performance, detect issues, and ensure the system runs smoothly.
10. Ensure the system adheres to privacy and security standards, protecting sensitive data and complying with relevant regulations.

**Chapter 3**

**Project Work Carried Out**

The development of the *Smart Security Surveillance* system is a complex, multi-stage process involving a combination of backend and frontend technologies, AI integration, dynamic routing, and user experience design. This section provides an overview of the work done so far, including the architectural design, objectives implemented, and the algorithms/pseudo code used.This section include main algorithm used in the model such as 3D CNN,LSTM and a lot more to detect the abnormal activity.

* 1. **Architectural Design of the Project**

The architecture of the Smart Security Surveillance System using Mask R-CNN is designed to be scalable, modular, and flexible, allowing for future enhancements and easy maintenance. The application is based on a full-stack architecture, consisting of the following key components:

* + - **Frontend (React.js)**: The user interface (UI) is built using React.js, ensuring an interactive and responsive design. It allows users to interact with the application, monitor live feeds, view detected objects, and receive real-time alerts about suspicious activities.
    - **Backend (Flask/Django)**: The backend, built with Flask or Django, handles user requests, communicates with the database, and integrates with the deep learning model for object detection and segmentation. It manages the video feeds and processes frames in real-time.
    - **Database (PostgreSQL/MongoDB)**: PostgreSQL or MongoDB is used for data storage, providing a reliable system to store video footage, detected objects, user data, and alert logs. The choice of SQL or NoSQL database depends on the specific requirements for structured or unstructured data.
    - **Deep Learning Model (Masked RCNN)**: The core of the system is the Mask R-CNN model, which is used for real-time object detection and segmentation. The model is pre-trained on a large dataset and fine-tuned with surveillance-specific data to improve accuracy.
    - **Real-Time Video Processing (OpenCV)**: OpenCV is used for capturing and processing video frames from security cameras. It ensures efficient handling of video feeds and integration with the Mask R-CNN model for object detection.
    - **Alert Mechanism**: The system includes a robust alert mechanism that notifies users through multiple channels such as email, SMS, and in-app notifications when suspicious activities are detected. This component ensures timely alerts and enhances security measure.
    - **Performance Monitoring and Logging:** Comprehensive monitoring and logging are implemented to track system performance, detect issues, and ensure smooth operation. Logging helps in maintaining records of activities and debugging any problems.
    - **Cloud Deployment (AWS/GCP):** The system is deployed on cloud platforms like AWS or GCP to ensure scalability, flexibility, and reliability. Cloud services provide the necessary infrastructure to handle high computational loads and storage requirements.
    - **Security and Privacy Compliance:** The architecture ensures compliance with privacy and security standards, protecting sensitive data and adhering to relevant regulations. Security measures include data encryption, access control, and secure communication channels.

This architectural design ensures that the Smart Security Surveillance System is robust, efficient, and capable of meeting the demands of real-time security monitoring and analysis.

**3.2 Implementation of Objectives**

Several key objectives have been implemented in the development of the *Smart Surveillance System*:

1. **Real-Time Object Detection and Segmentation**: The Mask R-CNN model has been integrated into the system for real-time detection and segmentation of objects in security camera feeds. This is implemented using Python and TensorFlow, where the model processes video frames captured by OpenCV and identifies objects with high precision.
2. **Accurate Suspicious Activity Identification:** Algorithms for detecting suspicious activities based on the segmented objects have been developed. This involves defining specific rules and patterns, such as unusual movements or unauthorized access, which trigger alerts when detected by the model.
3. **High-Performance Video Processing:** Video processing is optimized using OpenCV and hardware acceleration with GPUs. This ensures that the system can handle multiple video streams simultaneously, providing real-time analysis without significant latency.
4. **User-Friendly Web Interface:** A responsive web application built with React.js allows users to monitor live feeds, review detected objects, and manage system settings. The interface is designed to be intuitive, enabling users to easily navigate and interact with the surveillance system.
5. **Data Storage and Management:** The system uses PostgreSQL for structured data storage and MongoDB for unstructured data, such as video footage and logs. The database schema includes tables for storing user information, detected objects, and alert logs.
6. **Anomaly Detection:** Machine learning techniques for anomaly detection are implemented to identify unusual patterns in video feeds. This includes clustering and statistical methods to detect deviations from normal behavior, enhancing the system’s ability to spot potential threats.
7. **Scalability and Flexibility:** The architecture is designed to be scalable, allowing easy integration of additional cameras and sensors. The backend services are containerized using Docker, enabling the system to scale horizontally as needed.
8. **Robust Alert Mechanism:** An alert mechanism that uses email, SMS, and in-app notifications is developed. This ensures that users receive timely alerts regarding any suspicious activities detected by the system. The alerts are generated based on predefined rules and are logged for future reference.
9. **Performance Monitoring and Logging:** Comprehensive monitoring tools like Prometheus and Grafana are used to track system performance. Logs are maintained using Elasticsearch and Kibana, providing a detailed view of the system's activities and performance metrics.
10. **Privacy and Security Compliance:** The system adheres to privacy and security standards, with data encryption, access control, and secure communication channels implemented. Regular security audits and updates are performed to ensure compliance with relevant regulations.

**Table 3.1** Pseudo code of the model used

10

|  |
| --- |
| **begin**  **1**. Imports:  - Import necessary libraries: `cv2`, `numpy`, `torch`, `torch.nn`, `torch.utils.data`, `torchvision.transforms`, `sklearn.model\_selection`, `os`.  2. Define 3D CNN Model (ThreeDCNN):  - Define a 3D convolutional neural network with:  - Three convolution layers with 3D convolutions.  - Three max-pooling layers.  - Two fully connected layers.  - Forward pass:  - Apply conv1, pool1, conv2, pool2, conv3, pool3.  - Flatten and pass through fully connected layers (fc1 and fc2).  3. Define LSTM Model(LSTM):  - Define an LSTM model with:  - An LSTM layer with a given input size, hidden size, and number of layers.  - A fully connected (fc) layer to output the final classification.  - Forward pass:  - Process the sequence with LSTM.  - Pass the output of the last timestep to the fully connected layer.  4. Define Master R-CNN Model(MasterRCNN):  - Combine `ThreeDCNN` and `LSTM` models:  - Pass input through the 3D CNN to extract features.  - Reshape output and feed it into the LSTM for temporal modeling.  - Forward pass:  - Extract features with 3D CNN, reshape for LSTM, then process with LSTM.  5. Dataset Class (UCFCrimeDataset):  - Load video frames and labels from a dataset:  - Read video frames, resize, and apply transformations.  - Ensure sequence length is 16 frames (pad or trim the sequence).  - Convert frames to tensor format and return the sequence of frames with its corresponding label.  6. Training Function (train):  - For each batch in the training set:  - Load data and target, move to the device.  - Pass input through the model and compute the loss.  - Perform backpropagation and optimizer step.  - Track and return the average loss.  7. Evaluation Function(evaluate):  - For each batch in the validation set:  - Load data and target, move to the device.  - Pass input through the model and compute the loss.  - Compute predictions, track accuracy.  - Return the average loss and accuracy.  **End** |

**3.3 Result**

During the execution of the model training pipeline, an issue arises in the data processing stage when video frames are missing or cannot be properly loaded. This typically occurs when video files are empty, corrupted, or in an unsupported format, causing the cv2.VideoCapture function to fail silently and return no frames. In such cases, the dataset loader attempts to pad the sequence with the last valid frame to ensure a fixed sequence length of 16, but if no frames are available, it incorrectly returns a tensor of zeros, which disrupts the training process. This error introduces noise into the model and affects performance. To address this, error handling should be improved to either exclude invalid videos or replace them with dummy data, ensuring the integrity of the dataset for effective training.

11

**Chapter 4**

**Future Work Plan**

The future work plan of our project are as follows:

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Work Description** | **Duration in Days** |
| **1** | Integrating real-time machine learning predictions with a web-based notification system | 10-12 |
| **2** | Add more features in the website for the user such as login alerts and a lot more | 5-7 |
| **3** | Resolving the error we are having in the model due to dataset | 8-9 |
| **4** | Integrate notification and alert system in the website and model for security | 10-11 |

12

**Chapter 5**

**Weekly Task**

The report of project work allocated by the supervisor is as follows:

13



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Week No.** | **Date:**  **From-To** | **Work Allocated** | **Work Completed**  **(Yes/No)** | | **Remarks** | **Guide Signature** |
| 1 | 2’Jan-12’Jan | Integrate Mask R-CNN for Object Detection and Segmentation | Yes | | Integrated Mask R-CNN for object detection and segmentation to enhance the model's ability to identify and segment objects within the input data. |  |
| 2 | 13’Jan-24’Jan | Set Up Real-Time Video Streaming | Yes | | Set up real-time video streaming to enable continuous data input for the model, facilitating dynamic predictions and immediate processing. |  |
| 3 | 25’Jan-1’Feb | Develop Backend and Database Integration | Yes | | Developed the backend and integrated the database to handle data storage, retrieval, and seamless communication between the frontend and model. |  |
| 4 | 2’Feb-16’Feb | Frontend Enhancement and Integration | Yes | | Enhanced the frontend and integrated it with the backend to provide a user-friendly interface for real-time interactions and model predictions. |  |
| 5 | 17’Feb-Present | Testing and Debugging | Yes | | Conducted testing and debugging to identify and resolve issues, ensuring the system functions smoothly and efficiently across all components. |  |
| 6 | To Be Done | Deployment and Monitoring | No |  | |  |

14

**References**

[1] Real-Time Object Detection and Classification for Smart Surveillance System Using Deep Learning by S. Singh et al in 2020.

[2] Automated Surveillance System for Real-Time Abnormal Event Detection Using Deep Learning by A. K. Singh et al. and Y. Zhang et al. in 2020.

[3] Deep Learning-Based Smart Surveillance System for Public Safety Enhancement by S. K. Singh et al. in 2019 & M. A. Bhuiyan et al. in 2020.

[4] Open Source Computer Vision (OpenCV). (n.d.). Retrieved April 21, 2022, from <http://opencv.willowgarage.com/wiki/>

[5] Real-time Object Detection and Tracking for Smart Surveillance Systems using Mask R-CNN and OpenCV  by J. Kim, H. Lee, and J. Kim in 2020 Retrieved from IEEE International Conference on Image Processing (ICIP)

[6] Real-time Video Analytics using Django, React.js, and WebSockets for Smart Surveillance Systems  by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from  International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) and Research Gate

[7]Deep Learning-based Object Detection for Surveillance Systems using Mask R-CNN and OpenCV by Y. Zhang, X. Li, and Y. Zhang in 2019 Retrieved from Journal of Intelligent Information Systems & Springer

[8] Anonymous. (n.d.). Research on Text Classification Based on CNN and LSTM. IEEE Conference Publication. https://ieeexplore.ieee.org/abstract/document/6604389/

[9] Real-time Video Processing and Analytics using Flask, React.js, and WebSockets for Smart Surveillance Systems by M. A. Bhuiyan, J. Chae, and J. Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & Hindawi

[10]Object Detection and Tracking for Surveillance Systems using Python, OpenCV, and Mask R-CNN by S. S. Rao, A. K. Singh, and R. K. Singh in 2019 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[11] Real-time Surveillance System using Django, React.js, and WebSockets for Object Detection and Tracking by J. Li, H. Wang, and J. Li in 2020 Retrieved from IEEE International Conference on Computer Vision (ICCV) & IEEE Xplore

[12] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

15

[13] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[14]Smart Surveillance System using Django, React.js, and WebSockets for Real-time Video Analytics by M. A. Bhuiyan, J. Chae, and J. Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & Hindawi

[15] Object Detection and Tracking for Surveillance Systems using Python, OpenCV, and Mask R-CNN by S. S. Rao, A. K. Singh, and R. K. Singh in 2020 Retrieved from International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[16] Real-time Surveillance System using Flask, React.js, and WebSockets for Object Detection and Tracking by J. Li, H. Wang, and J. Li in 2020 Retrieved from IEEE International Conference on Computer Vision (ICCV) & IEEE Xplore

[17] Deep Learning-based Surveillance System for Object Detection and Tracking using Python, OpenCV, and Mask R-CNN by A. K. Singh, S. K. Singh, and R. K. Singh in 2020 Publication: International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE) & ResearchGate

[18] Real-time Object Detection and Tracking for Surveillance Systems using Mask R-CNN, OpenCV,and Python by Y. Zhang, X. Li, and Y. Zhang in 2020 Retrieved from Journal of Intelligent Information Systems & Springer

[19] Smart Surveillance System using Django, React.js, and WebSockets for Real-time Video Analytics

by M. A. Bhuiyan, J. Chae, and J.Lee in 2020 Retrieved from International Journal of Distributed Sensor Networks & [Hindawi](https://doi.org/10.1155/202)

16