VISIONGUARD CCTV

Summer Intern Project Report Submitted in complete Fulfillment of the Requirements for the Award of the Degree of

Bachelor of Technology

in

Computer Science and Engineering

By

D. Lavanya (N200635)

Under the supervision of

Dr.Sadu Chiranjeevi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge Technologies – Nuzvid

Nuzvid, Krishna District, Andhra Pradesh – 521202. July 2025

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

(A.P. Government Act 18 of 2008) RGUKT-NUZVID, Krishna Dist - 521202

Tel: 08656-235557 / 235150

CERTIFICATE OF COMPLETION

This is to certify that the work entitled **"VISIONGUARD CCTV"** is the bonafide work of **D. Lavanya (N200635)**, carried out as part of a **Summer Internship Project Private**, under the guidance and supervision of **Dr.Sadu Chiranjeevi**. The internship was undertaken during the **academic session December 2024 – April 2025**, as a part of the Bachelor of Technology program in the Department of Computer Science and Engineering under RGUKT IIIT Nuzvid.

Dr. Sadu Chiranjeevi
Assistant Professor
Department of CSE
RGUKT-Nuzvid

Mr. Uday Kumar Ambati
Assistant Professor
HOD, Department of CSE
RGUKT-Nuzvid

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
(A.P. Government Act 18 of 2008) RGUKT-NUZVID, Krishna Dist - 521202
Tel: 08656-235557 / 235150

CERTIFICATE OF EXAMINATION

This is to certify that the work entitled, **"VISIONGUARD CCTV"** is the bonafide work of D. Lavanya (N200635) carried out as part of a **Summer Internship Project**, under the supervision and guidance of **Dr.Sadu Chiranjeevi**. This document is hereby approved for the purpose for which it has been submitted.. This approval does not necessarily endorse or accept every statements made, opinion expressed or conclusion drawn, as recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

Dr. Sadu Chiranjeevi,
Assistant Professor,
Department of CSE,
RGUKT Nuzvid.

Project Examiner
1.
2.
RGUKT Nuzvid.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
(A.P. Government Act 18 of 2008) RGUKT-NUZVID, Krishna Dist - 521202
Tel: 08656-235557 / 235150

DECLARATION

I, **D. Lavanya (N200635)**, hereby solemnly declare that the internship report entitled **"VISIONGUARD CCTV"**, completed during my **summer internship**, under the supervision of **Dr.Sadu Chiranjeevi**, is my original work. This report is submitted in **complete fulfillment of the requirements** for the summer internship as part of the **Bachelor of Technology** program in **Computer Science and Engineering** at **RGUKT–Nuzvid**, during the academic session **December 2024 to April 2025**.

We also declare that this project is a result of my own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 15 - 08 - 2025
Place: NUZVID

ACKNOWLEDGEMENT

I would like to express our deepest gratitude to our advisor, **Dr. Sadu Chiranjeevi Sir**, for his patient guidance, invaluable insights, and unwavering support throughout this research journey. His mentorship significantly shaped my understanding and development in the project on **"VISIONGUARD CCTV."**

Special thanks to **Prof. Uday Kumar Ambati (HOD, Dept. of CSE)** and all faculty members of **RGUKT–IIIT Nuzvid**, for providing academic mentorship, motivation, and support that significantly contributed to the successful completion of this project.

ABSTRACT

With the increasing need for intelligent and efficient surveillance, traditional CCTV systems—recording continuously regardless of activity—consume significant storage and require extensive review time. This project presents the design and implementation of **VisionGuard CCTV**, an automated surveillance system that classifies detected objects and triggers recording only when relevant categories (such as humans, vehicles, or animals) are present.

The system is developed using **TensorFlow/Keras** and trained on the **CIFAR-10 dataset**, which contains 60,000 labeled images across ten categories, including cars, ships, birds, cats, deer, dogs, frogs, horses, trucks, and airplanes. A **Convolutional Neural Network (CNN)** architecture is implemented for accurate image classification, leveraging multiple convolutional, pooling, and dense layers for feature extraction and decision-making. The trained model is evaluated on a dedicated test set to ensure high accuracy and robust performance.

For practical deployment, the model can process either live video feed from a webcam or static image uploads. The live monitoring mode enables real-time frame-by-frame classification, triggering recording and logging only when the detected class matches the configured target categories. This event-driven approach significantly reduces storage requirements while ensuring that critical events are captured..

The project demonstrates the effectiveness of deep learning for real-world surveillance automation, offering a scalable, resource-efficient, and customizable solution adaptable to various security environments.

TABLE OF CONTENTS

CHAPTER-1

1.1 INTRODUCTION

In today's security-driven world, surveillance cameras have become essential for monitoring and safeguarding environments such as industries, offices, public spaces, and residential areas. However, traditional Closed-Circuit Television (CCTV) systems record continuously, leading to excessive storage consumption and making it difficult to review hours of irrelevant footage.

VisionGuard CCTV addresses this problem by integrating deep learning-based object classification to make CCTV recording intelligent and event-driven. Using a Convolutional Neural Network (CNN) trained on the CIFAR-10 dataset (cars, trucks, animals, airplanes, ships, etc.), the system identifies objects of interest in real time. If an object belongs to a specified target category (e.g., human, vehicle, animal), recording is triggered automatically; otherwise, the camera remains idle.

The project supports both live monitoring via webcam and manual image upload for testing. By combining real-time detection, selective recording, and efficient storage usage, VisionGuard CCTV offers a scalable solution for modern security needs.

1.2 OBJECTIVES

- **Automated Object Classification:** Implement a deep learning model to classify objects into predefined categories.
- **Event-Based Recording:** Record video only when relevant categories (e.g., person, car, animal) are detected.
- **Dual Operation Modes:** Support both live monitoring and static image upload for testing.
- **Efficient Storage Usage:** Reduce storage requirements by avoiding unnecessary continuous recording.

- **Scalable Architecture:** Allow easy integration into existing CCTV setups.
- **User-Friendly Interface:** Provide a simple UI for setting detection thresholds and viewing logs.

1.3 MOTIVATION

In industries and organizations, large-scale CCTV networks often generate terabytes of footage daily, most of which contains no significant events. This not only strains storage infrastructure but also increases the time required for security personnel to review recordings.

Recent advancements in **computer vision** and **deep learning** enable real-time classification of objects in video streams, making it possible to record only what matters. This project was motivated by the need to:

- Minimize redundant data storage.
- Improve operational efficiency of surveillance systems.
- Reduce manual review time for security staff.
- Apply AI technology for proactive monitoring.

By using a trained CNN model and real-time monitoring, VisionGuard CCTV transforms passive surveillance into an intelligent, cost-effective security tool.

1.4 LANGUAGES USED

- **Python:** Core programming language for model training, image processing, and application logic.
- **TensorFlow / Keras:** For building, training, and evaluating the CNN model.
- **OpenCV:** For accessing the webcam, processing frames, and rendering detection results.
- **Streamlit:** For creating a user-friendly dashboard to control detection modes and review logs.
- **NumPy & Pandas:** For data manipulation and preprocessing.
- **Matplotlib / Seaborn:** For model performance visualization.

2.SYSTEM REQUIREMENTS

2.1 Hardware Requirements

Minimum Requirements:

- Processor: Dual-core CPU (Intel i3 / AMD equivalent)
- RAM: 4 GB
- Storage: 10 GB free disk space
- Webcam: Integrated or external USB webcam
- Display: 1280×720 resolution

2.2 Software Requirements

- **Operating System:** Windows 10/11 (64-bit) or Linux (Ubuntu 20.04+)
- **Python:** Version 3.8+
- **Libraries & Frameworks:** TensorFlow / Keras
    - OpenCV
    - Streamlit
    - NumPy, Pandas
    - Matplotlib, Seaborn
- **IDE:** VS Code / PyCharm
- **Web Browser:** Google Chrome / Edge (for UI access)
- **Dataset:** CIFAR-10 (for model training)

2.3 Functional Requirements

1. **Data Loading & Training**
    - Load CIFAR-10 dataset.
    - Train CNN model with 60,000 labeled images.
    - Save trained model for later use.
2. **Live Monitoring**
    - Access webcam feed.
    - Classify each frame in real-time.
    - Trigger video recording when a target object is detected.
3. **Image Upload**

- Allow users to upload an image for classification.
- Display predicted class and confidence score.

4. **Logging & Reporting**
   - Maintain detection logs with timestamp,class name, and confidence.
   - Provide UI to view logs.

2.4 Non-Functional Requirements

These define system qualities and performance standards:

1. **Performance:** Real-time detection at a minimum of 15 FPS.
2. **Scalability:** Ability to add new classes or retrain with different datasets.
3. **Usability:** Simple and intuitive UI for non-technical users.
4. **Reliability:** Consistent performance over long monitoring sessions.
5. **Security:** Restrict access to dashboard controls in production environments.
6. **Portability:** Runs on Windows and Linux systems without major code changes.

3.SYSTEM DESIGN

3.1 Environment Setup

The development environment for VisionGuard CCTV is configured to support **deep learning model training** and **real-time image classification**. The system uses Python along with key machine learning and image processing libraries.

1. Development Tools:

- **VS Code –** Primary code editor for Python scripts.
- **Anaconda / Virtualenv –** For creating isolated Python environments.
- **Streamlit –** For interactive web-based dashboard.
- **Jupyter Notebook –** For model experimentation and dataset exploration.

2. Backend Setup:

- **Python 3.8+** for core programming.
- **TensorFlow / Keras** for CNN model creation and training.
- **NumPy, Pandas** for dataset preprocessing.
- **Matplotlib, Seaborn** for model accuracy/loss visualization.

3. Frontend Setup:

- **Streamlit UI** for user interaction.
- **OpenCV** integration for live video feed and frame capture.

4.Project Structure:

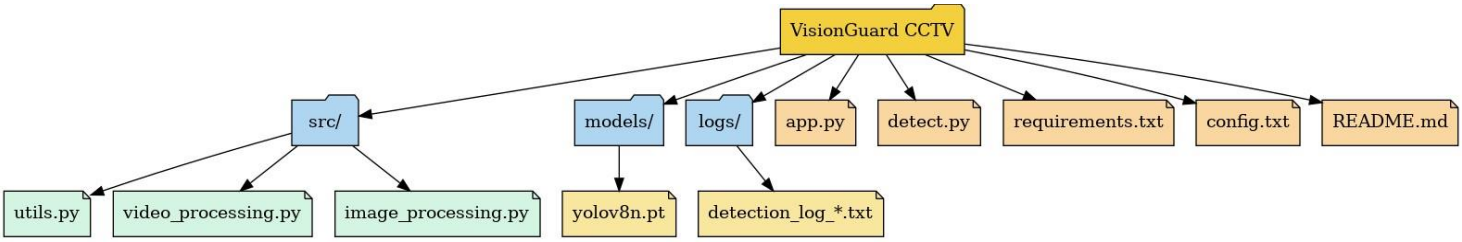


Figure 1: Project Structure

Required Modules:

Module	Purpose
opencv-python	Access and process webcam/video frames
ultralytics	YOLOv8 object detection model
pyttsx3	Text-to-speech announcements for detections
streamlit	Web dashboard for live monitoring & file uploads
Pillow	Image handling in Streamlit
numpy	Array & image data processing
datetime	Timestamp logs & filenames
tempfile	Handling temporary files in dashboard
os	File operations
subprocess	Run detection script from dashboard

3.2 Implementation

The VisionGuard CCTV project is implemented in **three major modules**:

1. Model Training:

- **Dataset:** CIFAR-10 (10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck).
- **Preprocessing:** Normalization of pixel values (0–255 to 0–1), one-hot encoding for labels.
- CNN Architecture:
    - Multiple convolutional layers for feature extraction.
    - MaxPooling layers for down-sampling.
    - Dense layers for classification.
    - Softmax activation for multi-class output

- Model evaluation on test dataset**.**
- Saving the trained model (.h5 format) for later use.

2. Real-Time Monitoring

- OpenCV captures frames from webcam.
- Model predicts class for each frame.
- If class matches target (e.g., car, person, animal), system triggers recording and logs event.

3. Image Upload Mode

- User uploads image via Streamlit UI.
- Model predicts object category.
- Result displayed with confidence percentage.
- Option to save the classified image in logs.

4. Event Logging & Visualization

- All detections recorded with timestamp and confidence.
- Detection history viewable from dashboard

3.3 Input-Output

User Inputs:

- **Training Phase:** Dataset input (CIFAR-10 images).
- **Live Mode:** Webcam video feed.
- **Upload Mode:** Single image file (JPG, PNG).
- **Settings:** Confidence threshold selection.

System Outputs:

- **Predicted Class:** Name of detected object (e.g., “Car”).
- **Confidence Score:** Probability percentage of prediction.
- **Detection Log:** Timestamp, object name, and image snapshot
- **Recorded Video Clips:** Saved only when target object detected.
- **Visual Feedback:** Bounding boxes and labels displayed on live feed.

Figure 1: Image Upload Interface

Figure 2: Image uploaded

Figure 3: Predicted Name

Prediction: frog (99.98%)

Class Probabilities:

- airplane: 0.00%
- automobile: 0.00%
- bird: 0.00%
- cat: 0.00%
- deer: 0.00%
- dog: 0.00%
- frog: 99.98%
- horse: 0.00%
- ship: 0.00%
- truck: 0.00%

Figure 4: Prediction classes and percentage

4. VISUALISATION

4.1 Flowchart

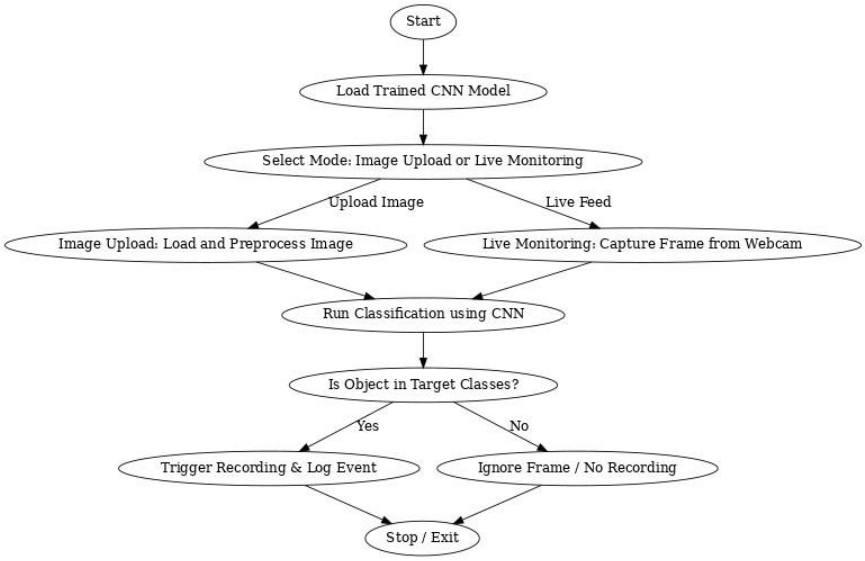


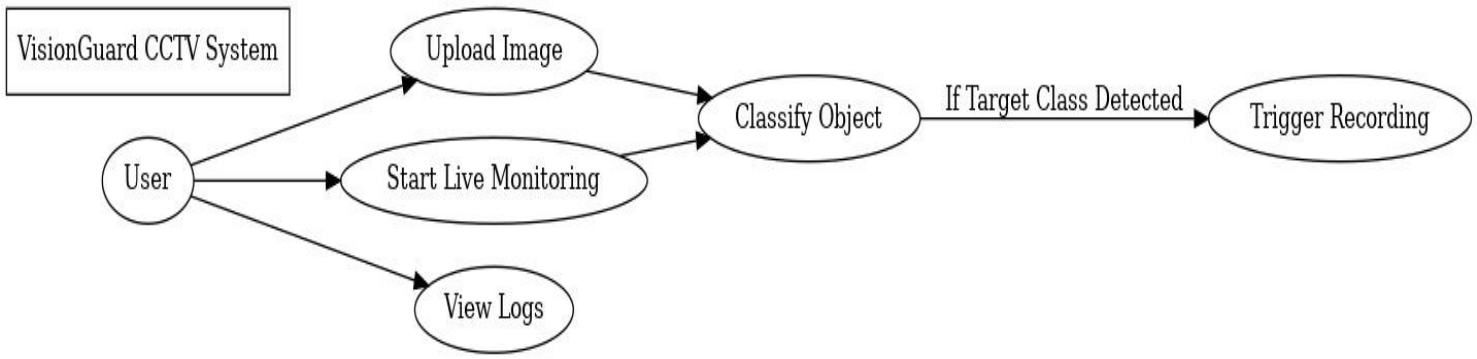
Figure 1: Complete Project Flowchart of Object Detection

Description:

1. Start application.
2. User selects mode: Live Monitoring / Image Upload.
3. If Live Monitoring:
    - Capture frame from webcam.
    - Classify object.
    - If target object → start recording & log event.
    - Else → continue monitoring.
4. If Image Upload:
    - Classify uploaded image.
    - Display predicted class & confidence.
5. End.

4.2 Use Case Diagram:

Figure 2: Use Case Diagram for Object Detection



Actors:

- **User: Selects mode, uploads image, reviews logs.**
- **System: Processes input, classifies object, logs results.**

Use Cases:

- **Live Monitoring**.
- **Image Classification**
- **Event Logging**
- **Video Recording**

4.3 Class Diagram

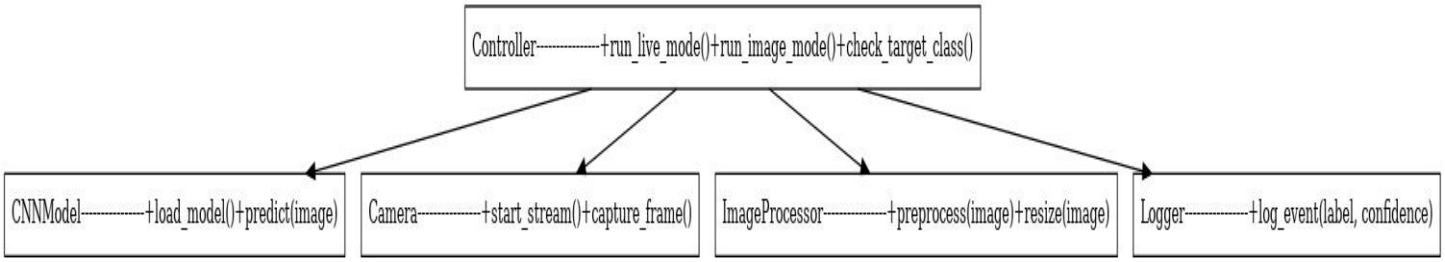


Figure 3: Class Diagram of the Object detection

1. **VisionGuardApp**

- Attributes: mode, threshold
- Methods: start_live_monitor(), upload_image(), save_log()

2. **CNNModel**

- Attributes: architecture, weights
- Methods: train(), predict(), load_model()

3. **VideoRecorder**

- Attributes: output_path, is_recording
- Methods: start(), stop(), save()

4. **Logger**

- Attributes: log_file
- Methods: write_log(), read_logs()

5.1 Future Scope:

1. Integrate person detection with facial recognition for security.
2. Add **custom dataset support** for training with domain-specific objects.
3. Implement **cloud storage** for storing recordings.
4. Add **SMS/Email alert system** for critical detections.
5. Deploy on **Raspberry Pi** for portable CCTV monitoring.

5.2 Conclusion:

The VisionGuard CCTV system successfully demonstrates how AI-based object classification can make surveillance smarter, storage-efficient, and more actionable. By combining deep learning with real-time monitoring, it reduces unnecessary recordings while ensuring important events are captured. The system's modular design allows easy adaptation to different environments and datasets, making it a flexible solution for modern surveillance needs.

5.3 References

1. TensorFlow Documentation – https://www.tensorflow.org/
2. Keras API Reference – https://keras.io/
3. OpenCV Documentation – https://docs.opencv.org/
4. CIFAR-10 Dataset – https://www.cs.toronto.edu/~kriz/cifar.html