

REAL-TIME DETECTION OF SUSPICIOUS ROADSIDE OBJECTS IN SURVEILLANCE IMAGERY

Submitted by

KRITHIKA M

22UEO028

NIKITHAA S

22UEO034

RAKSHA DEVI R J

22UEO043

RAKSHNA M

22UEO044

Under the guidance of

Mrs.Divya R

Assistant Professor

In partial fulfillment for the award of degree

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

**AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER
EDUCATION FOR WOMEN
COIMBATORE – 641108
NOVEMBER 2025**

ABSTRACT

Real-time detection of suspicious roadside objects is an important task for improving public safety and preventing potential threats in urban environments. This system focuses on automatically identifying unusual or potentially dangerous objects such as unattended bags, boxes, or debris from surveillance camera footage. By using advanced computer vision and deep learning techniques, the model analyzes live video streams and detects objects placed near roadsides in real time. The detected objects are classified and monitored to determine whether they pose a potential risk. When a suspicious object is identified, the system can generate alerts to assist security personnel and authorities in taking timely action. This approach helps enhance surveillance efficiency, reduces manual monitoring efforts, and supports faster response to possible security threats.

The rapid growth of urban environments and transportation networks has increased the need for intelligent surveillance systems to ensure public safety. This project presents a system for real-time detection of suspicious roadside objects in surveillance imagery using advanced computer vision and deep learning techniques. The proposed system analyzes live video streams from roadside surveillance cameras to automatically identify unusual or potentially dangerous objects such as abandoned bags, unattended packages, or hazardous materials. A deep learning-based object detection model is employed to accurately detect and classify objects in real time, while additional contextual analysis helps determine whether the object is suspicious based on its location, duration of presence, and surrounding activity. The system continuously monitors the scene and generates alerts to authorities when a suspicious object is detected, enabling faster response and preventive action. By integrating automated detection with surveillance infrastructure, the proposed approach enhances situational awareness, reduces the need for constant human monitoring, and improves the overall efficiency of public safety systems.

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Surveillance systems play an important role in maintaining public safety and security in places such as roads, highways, airports, railway stations, and public areas. With the increasing number of vehicles and people, it has become difficult for humans to manually monitor all surveillance cameras continuously.

Suspicious roadside object detection is a technology that uses computer vision and artificial intelligence (AI) to automatically detect unusual or abandoned objects on roadsides. These objects may include unattended bags, suspicious packages, or unknown items that could potentially cause danger or accidents.

The system analyzes video footage captured from surveillance cameras and identifies objects that remain in the same place for an unusual amount of time or appear in restricted areas. Once a suspicious object is detected, the system can send alerts to security personnel or authorities for immediate action.

This technology helps in preventing security threats, improving traffic safety, and assisting law enforcement agencies. By automating the monitoring process, it reduces human effort and increases the efficiency of surveillance systems.

1.1 NECESSITY

The necessity of Artificial Intelligence (AI) in detecting suspicious roadside objects has grown significantly due to increasing security concerns, rising traffic density, and the limitations of human surveillance. Roads, highways, and public spaces often experience situations where unattended bags, fallen cargo, hazardous materials, or deliberately placed harmful objects pose risks to commuters and infrastructure. Traditional monitoring methods rely heavily on manual inspection, which can be slow, inconsistent, and prone to error, especially in large or crowded areas. AI-powered systems, however, can continuously analyze video footage

from CCTV cameras, recognize unusual patterns, detect anomalies in object behavior or placement, and immediately alert authorities.

1.2 CURRENT SITUATION

The current situation of AI in detecting suspicious roadside objects shows rapid progress but also several practical limitations. Today, many smart cities and transportation departments have started integrating AI-powered video analytics into CCTV networks, traffic cameras, and roadside surveillance systems. These AI systems are capable of continuously scanning roads for anomalies such as unattended bags, dropped cargo, potholes, debris, or any object that appears unusual or potentially dangerous. In several cities across the world—and increasingly in India—AI is already being used to identify hazards, alert authorities, and reduce manual monitoring. Research in 2025 has focused heavily on improving accuracy through deep learning and edge computing, allowing real-time alerts even in low-light or crowded environments. However, despite technological advancement, widespread implementation is still limited because of the need for high-quality camera infrastructure, strong network connectivity, and consistent power supply. AI models also struggle with unpredictable environments, such as cluttered roadside areas, weather changes, and moving objects. Privacy concerns and regulatory restrictions further slow adoption in public spaces. Overall, while AI-based roadside anomaly and suspicious object detection is becoming more reliable and actively tested, it is still transitioning from experimental deployments to large-scale, everyday use. As infrastructure improves and models become more robust, the technology is expected to become a standard part of urban safety and intelligent transportation systems.

1.3 RELATED WORK

Several related works focus on detecting abandoned or suspicious objects in public surveillance environments, which is closely connected to roadside safety. Researchers have developed systems using advanced deep-learning models such

as YOLOv8, YOLOv11, and RetinaNet to identify objects like bags, luggage, or packages that remain unattended for a certain period. These methods usually combine object detection with temporal analysis to track whether the object moves or if the person associated with it leaves the scene. Studies published in recent years demonstrate that such systems can achieve high accuracy, often exceeding 85–95% for medium-sized objects in real-time CCTV footage. Although originally intended for airports, bus stands, and public transit areas, these methods are now being adapted for roadside surveillance, where detecting unattended or suspicious objects can help prevent accidents or security threats.

1.4 PROBLEM STATEMENT

The increasing number of security threats, accidents, and road hazards has highlighted the urgent need for a reliable system to detect suspicious or foreign objects along roadsides. Traditional surveillance methods depend heavily on manual monitoring, which is time-consuming, inconsistent, and prone to human error—especially in busy highways, urban streets, and low-visibility environments. Many roadside objects such as unattended bags, fallen cargo, debris, or unusual packages often go unnoticed until they cause traffic obstruction, accidents, or potential security risks. Existing monitoring systems lack the intelligence to automatically identify these anomalies in real time. Therefore, there is a critical need for an automated, AI-powered detection system capable of continuously analyzing video feeds, recognizing abnormal or suspicious objects, and sending instant alerts to authorities. Such a solution must be robust enough to work under varying weather, lighting, and traffic conditions while maintaining high accuracy and low false alarms. Addressing this problem will significantly improve public safety, reduce accidents, and enhance the efficiency of road surveillance systems.

1.5 OBJECTIVES OF THE WORK

- To develop an AI-powered system capable of detecting suspicious or foreign objects along roadsides in real time.
- To analyze and process continuous video feeds from CCTV or traffic cameras using advanced computer vision techniques.
- To accurately identify unattended items, debris, fallen objects, or any unusual roadside anomalies.
- To reduce reliance on manual surveillance and minimize human error in monitoring large road networks.
- To generate instant alerts for authorities when a suspicious object is detected.
- To improve detection accuracy under varying lighting, weather, and traffic conditions.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

The rapid growth of urbanization and transportation networks has made roadside safety a critical concern, prompting researchers to explore advanced technologies for intelligent monitoring. Traditional surveillance methods, which depend heavily on human operators, are often inefficient in detecting hazardous or suspicious objects due to fatigue, limited attention, and large monitoring areas. Recent advancements in Artificial Intelligence, particularly in computer vision and deep learning, have enabled automated systems to analyze video streams and identify unusual objects or activities with greater accuracy and speed.

2.2 LITERATURE SURVEY

The study contributes significantly to the field of AI-based vision systems operating in low-visibility environments. The authors focus on enhancing object detection accuracy under foggy conditions by jointly integrating image restoration and domain adaptation techniques, allowing detectors trained on clear-weather datasets to perform reliably in adverse weather scenarios. Their approach improves visual clarity, reduces domain shift, and enhances feature extraction without requiring extensive retraining. While the work advances object detection performance in challenging environments, it does not address higher-level functionalities such as automated threat analysis or real-time response mechanisms. Nevertheless, it provides valuable insights into improving detection robustness under degraded visibility, which can serve as a foundational component for systems dealing with roadside hazards or suspicious objects. (1)

The work presents a comprehensive survey on video anomaly detection using deep learning techniques, highlighting the growing necessity of automated systems capable of recognizing unusual events in real time. Their survey examines various deep learning models—such as CNNs, RNNs, autoencoders, and GAN-based frameworks—used for detecting irregular patterns across surveillance footage. The authors emphasize the importance of automatic anomaly detection for enhancing public safety, especially in scenarios where manual monitoring is inefficient or impractical. They also discuss the challenges posed by complex environments, varying motion patterns, and limited annotated datasets, while underscoring the need for robust models that can generalize across diverse real-world conditions. Although the survey provides valuable insights into real-time anomaly detection, it mainly focuses on behavioral anomalies rather than domain-specific threats, but its findings are highly relevant for systems requiring rapid identification of suspicious or hazardous road side events. (2)

The study on Integrating AI with Public Safety Portals explores how artificial intelligence can be embedded within authority dashboards to enhance situational awareness and streamline emergency responses. The work primarily focuses on system-level integration, where AI algorithms assist in aggregating data, identifying emerging issues, and enabling faster decision-making through centralized public safety platforms. By improving data flow between field sensors, citizen reports, and administrative portals, the system helps authorities detect problems earlier and respond more efficiently. However, the study does not delve deeply into advanced computer vision or real-time object detection techniques; instead, it emphasizes the operational benefits of AI-enabled dashboards for monitoring incidents, coordinating responders, and improving overall public safety management. Despite its limitations regarding visual detection capabilities, the work provides valuable insights into how AI-driven interfaces can strengthen emergency response frameworks and support rapid identification of suspicious or hazardous situations when integrated with real-time surveillance systems. (3)

The study introduces an attribute-driven modulation framework designed to enhance object detection accuracy by incorporating example-specific attributes into the prediction process. Their method improves feature representation and enables detectors to better adapt to variations across object classes, visual contexts, and scene complexities, ultimately increasing detection robustness. The research demonstrates significant performance gains over conventional deep learning detectors, particularly in challenging scenarios with diverse object attributes. However, while the approach strengthens detection precision, it does not address real-time integrations with large language models (LLMs) or automated recommendation systems, nor does it focus on autonomous threat interpretation or emergency response. Still, the paper contributes valuable advancements in adaptive detection mechanisms, offering insights that can support more intelligent

and context-aware systems for tasks such as identifying suspicious roadside objects. (4)

It provides a comprehensive examination of 3D object detection and tracking methods using deep learning for various computer-vision applications, highlighting how modern neural architectures significantly improve the accuracy and robustness of 3D perception tasks. Their study compares different deep learning models and detection pipelines, emphasizing advancements that enhance spatial understanding and object localization in complex scenes. However, despite offering valuable insights into algorithmic performance, the work notes a gap in the integration of these 3D tracking methods with real-time roadside surveillance systems, where constraints such as processing speed, environmental variability, and deployment scalability remain underexplored. Overall, the literature reinforces that deep learning has substantially advanced 3D object detection and tracking, yet practical, real-time implementations in surveillance contexts require further research. (5)

The study investigates the use of Faster R-CNN for roadside surveillance, demonstrating that the model delivers strong detection accuracy and reliable vehicle tracking performance under varied traffic conditions. The work highlights Faster R-CNN's effectiveness in identifying vehicles with high precision, reinforcing its suitability for intelligent transportation and monitoring systems. However, the study also acknowledges a key limitation: slower inference times, which hinder its deployment in real-time roadside surveillance scenarios that demand rapid processing and immediate response. Overall, the literature suggests that while Faster R-CNN offers robust accuracy for vehicle detection and tracking, its computational latency presents challenges for real-time operational use. (6) It proposes an edge detection-based bounding box construction algorithm designed

to enhance the precision of object detection in YOLOv3, demonstrating that integrating edge information can significantly refine object boundaries and reduce localization errors. Their work shows that incorporating edge cues leads to more accurate and stable detection outputs, particularly in scenes with complex object contours or low-contrast conditions. While the study effectively improves YOLOv3's detection precision, it does not extend its contributions toward higher-level functionalities such as automated decision-making or recommendation systems, focusing instead solely on architectural improvements to the detection pipeline. Overall, the literature indicates that edge-based enhancements can meaningfully boost YOLOv3 performance but leave opportunities for integrating advanced intelligence or automation features. (7)

It presents an embedded real-time object detection system capable of identifying objects and accurately measuring their size on resource-constrained devices, demonstrating the feasibility of performing efficient detection and measurement directly on embedded hardware. Their approach emphasizes low-latency processing and practical implementation, making it suitable for applications where portability and on-device computation are essential. However, while the system performs well in detecting objects and determining their dimensions, it does not extend its functionality to higher-level analytical tasks such as suspicious object classification or contextual decision-making. Overall, the literature shows that real-time embedded detection and measurement systems can achieve reliable performance, though their scope often remains limited to basic detection rather than advanced interpretive capabilities. (8)

CHAPTER 3

METHODOLOGY

3.1 METHODOLOGY

Module 1 - Data Collection and Surveillance Input Module

Module 2 - Object Detection, Training and validation

Module 3 - Performance evaluation

Module 4 - LLM-Based Recommendation Module

Module 5 - Alert & Communication Module

Module 6 - Data Storage & Analytics Module

3.2 EXISTING SYSTEM:

In the existing surveillance systems used for monitoring roads and public areas, the detection of suspicious roadside objects is largely dependent on **manual observation** and **traditional CCTV monitoring**. Security personnel or traffic operators must continuously watch live video feeds to identify unusual items such as unattended bags, debris, fallen objects, or any foreign materials on the roadside. This manual process is highly inefficient because human operators can easily miss critical events due to fatigue, distraction, or the overwhelming number of cameras installed in modern smart cities. Most existing systems lack intelligent automation and operate only as recording tools, not analysis tools.

Additionally, traditional object detection systems—if present—use older techniques such as motion detection, background subtraction, or simple threshold-based algorithms. These methods work poorly in real-world environments because changing weather, shadows, moving vehicles, night-time lighting, and crowded backgrounds cause false alarms. They cannot distinguish between normal roadside objects and suspicious ones, nor can they provide real-time alerts with high accuracy.

The absence of **AI-based detection models** in these existing systems means they cannot recognize specific object types, track how long an object has been stationary, or analyze whether it poses a risk. As a result, timely detection of hazardous or suspicious roadside items becomes slow and unreliable. This creates safety risks for travelers and delays emergency responses. Therefore, the limitations of the existing system highlight the need for an intelligent, automated, real-time AI solution.

3.3 PROPOSED SYSTEM:

The proposed system aims to improve roadside surveillance by integrating advanced computer vision and AI-based decision-making. It uses a hybrid **CNN–YOLO model**, where CNN helps extract important visual features and YOLO performs fast, real-time detection of objects. This combination ensures that suspicious or abnormal objects on the roadside are identified with high accuracy and minimal delay. The system continuously monitors the environment and instantly flags any potentially dangerous object. After detection, the information is passed to a **Large Language Model (LLM)**, which interprets the scenario in a more human-like manner. The LLM analyzes the context, understands the risk level, and generates suitable recommendations or actions. These may include alerting authorities, sending warnings, or suggesting preventive steps. By merging vision-based detection with intelligent reasoning, the system offers a more reliable, proactive, and automated approach to roadside safety and security.

3.3 LIMITATIONS OF EXISTING METHOD:

- Heavily dependent on manual monitoring, which leads to human fatigue, slow response time, and a high chance of missing suspicious objects.
- Traditional CCTV systems lack intelligence and only record video without analyzing or detecting anomalies automatically.

- Older computer vision techniques like motion detection and background subtraction give poor accuracy in real-world roadside environments.
- High number of false alarms due to shadows, lighting changes, weather conditions, and moving vehicles interfering with detection.

3.4 ADVANTAGES:

- The combination of CNN feature extraction and YOLO's detection capability improves the precision of identifying suspicious objects.
- YOLO enables fast processing, making the system suitable for continuous roadside monitoring.
- The LLM provides intelligent reasoning and understands the scenario beyond just detection.
- Contextual analysis by the LLM helps eliminate unnecessary alerts by evaluating the situation more accurately.
- Minimizes human involvement, reducing manual monitoring effort and chances of human error.

3.4 BLOCK DIAGRAM:

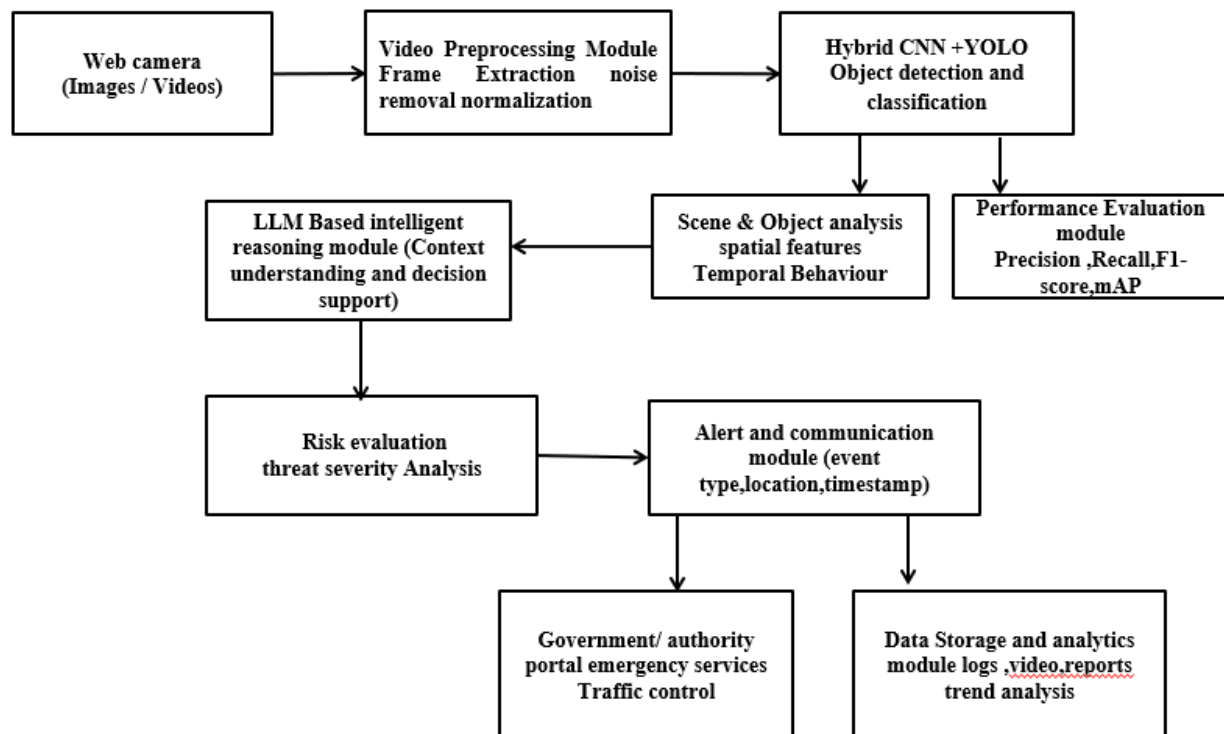


Fig 3.1 Block Diagram – Real-Time Detection of Suspicious Roadside Objects In Surveillance Imagery

CHAPTER 4

4.1 SYSTEM REQUIREMENTS

Programming Language : Python

Dataset : Anomaly Dataset

Frameworks & Libraries : TensorFlow, PyTorch, OpenCV, YOLOv8, Transformers (Hugging Face)

Hardware : High-resolution surveillance cameras, GPU-enabled server (NVIDIA)

Software Tools : Jupyter Notebook, Flask, Docker, Git, Government API integration tools, Processor, RAM, Hard Disk, Operating System, Display

4.2 Programming Language: Python

Python serves as the primary programming language for developing the **Surveillance Suspicious Roadside Object Detection System**. It is widely used in artificial intelligence and machine learning applications because of its simple syntax, readability, and extensive library support. Python enables developers to efficiently process images, train deep learning models, and build backend systems. In this project, Python supports modules for video processing, machine learning model integration, and alert generation. Its large ecosystem allows seamless integration with libraries such as TensorFlow, PyTorch, and OpenCV. Python also enables quick experimentation and model optimization, making it highly suitable for building intelligent surveillance systems.

4.3 Dataset: Anomaly Dataset

The system uses an **Anomaly Dataset** containing images and video samples of both normal roadside scenes and unusual or suspicious situations. These datasets

include examples of abandoned bags, unknown packages, or objects placed in unusual locations near roads or public areas.

The dataset is used to train and test the machine learning models so that the system can learn to distinguish between normal objects and suspicious activities. By analyzing patterns in the dataset, the model improves its detection accuracy and becomes capable of identifying potential threats in real-time surveillance footage.

4.4 Frameworks and Libraries

The system relies on several powerful frameworks and libraries to perform deep learning and computer vision tasks. Libraries such as TensorFlow and PyTorch are used to develop and train neural network models capable of recognizing objects in video frames.

OpenCV is used for image and video processing operations such as frame extraction, noise reduction, and object tracking. The object detection algorithm used in this project is YOLOv8, which enables fast and accurate real-time detection of objects in surveillance footage. Additionally, Hugging Face Transformers are used to enhance intelligent reasoning and contextual understanding within the system.

4.5 Hardware: Surveillance Cameras and GPU Server

High-resolution surveillance cameras are used to continuously monitor roadside environments and capture real-time video data. These cameras act as the primary input source for the system, providing visual information that can be analyzed by the AI model.

A GPU-enabled server powered by NVIDIA graphics processors is used to handle the computational workload of deep learning models. GPUs significantly accelerate the processing of large volumes of image data and improve the speed of object detection algorithms, making real-time monitoring possible.

4.6 Software Tools

Several software tools are used to develop, test, and deploy the system. Jupyter Notebook is used for building and experimenting with machine learning models and analyzing datasets.

The web framework Flask is used to create APIs and backend services that manage alerts and communication with authorities. Docker is used for containerizing the application to ensure consistent deployment across different environments. Version control is managed using Git, which allows developers to track changes and collaborate effectively.

4.7 Processor: Intel Core i3 or Higher

A processor such as Intel Core i3 or any equivalent multi-core CPU is sufficient for running the surveillance detection system. The processor executes Python scripts, processes video frames, and performs object detection tasks.

Higher-performance processors such as Core i5 or Core i7 can improve system efficiency, especially when handling multiple camera feeds or processing large video datasets.

4.8 RAM: Minimum 8 GB

A minimum of 8 GB RAM is recommended for running deep learning models and video processing applications efficiently. The system requires memory to load datasets, run object detection models, and process video frames simultaneously.

Higher RAM capacity allows faster processing and better multitasking performance when working with high-resolution images and large volumes of surveillance data.

4.9 Hard Disk: 500 GB or Above

The system requires sufficient storage space to store surveillance videos, training datasets, and generated reports. A hard disk of 500 GB or more is recommended to maintain logs, model files, and detection records.

Using a Solid State Drive (SSD) can significantly improve system performance by providing faster read and write speeds when handling large video files.

4.10 Operating System: Windows 10/11 or Ubuntu 20.04+

The system can run on both Windows and Linux operating systems. Windows 10 or Windows 11 provides a user-friendly environment for development and testing. Linux distributions such as Ubuntu 20.04 or later provide excellent performance for AI-based applications and are commonly used for deploying machine learning systems on servers.

4.11 Display: 15.6-inch Monitor with 1366×768 Resolution or Higher

A display with at least 1366×768 resolution is required to clearly view the system interface and surveillance outputs. Higher resolution displays allow better visualization of detected objects, alert notifications, and monitoring dashboards.

This ensures that security personnel or authorities can easily monitor detected suspicious activities and respond quickly to potential threats.

CHAPTER 5

MODULE DESCRIPTION

5.1 DATASET DESCRIPTION

- An anomaly dataset is a collection of data used in machine learning to identify rare items, events, or observations that deviate significantly from the norm or expected behavior.

- <https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset>

Field	Details
Dataset Name	UCF Crime Dataset
Source	Kaggle
Original Provider	University of Central Florida (UCF)
Kaggle Uploaded By	odins0n
Dataset Type	Video Dataset
Domain	Computer Vision / Video Analysis
Purpose	Crime & Anomaly Detection
Total Videos	1900+
Normal Videos	~950
Crime Videos	~950
Number of Classes	13 Crime Categories
Crime Categories	Abuse, Arrest, Arson, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Vandalism, RoadAccidents, Shoplifting, Assault
Video Format	MP4
Annotation Type	Video-level labels
Typical Usage	Deep Learning, CNN, LSTM, 3D CNN, Anomaly Detection
Difficulty Level	Intermediate to Advanced
Kaggle Link	https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset

Anomaly-Detection-Dataset-UCF

Data Card Code (7) Discussion (1) Suggestions (0)

***** A few notes on using the dataset *****

Chen Chen, UNC-Charlotte
<https://webpages.uncc.edu/echen62/>
chen.chen@uncc.edu

Cite our paper:
Waqas Sultan, Chen Chen, Mubarak Shah, "Real-world Anomaly Detection in Surveillance Videos"
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018

1. Anomaly Detection Experiment

- A) Videos:
Anomaly-Videos-Part-1 -- Part-4 (4 zip files)
Training-Normal-Videos-Part-1.zip and Training-Normal-Videos-Part-2.zip (normal videos for training)
Testing-Normal-Videos.zip (normal videos for testing)

UCF_Crimes-Train-Test-Split.zip contains the training and testing split in our experiments
(folder: Anomaly_Detection_splits)

- B) Temporal Annotations for Testing Videos (anomaly videos)

Temporal_Anomaly_Annotation_for_Testing_Videos.txt

Each row of 'Temporal_Anomaly_Annotation.txt' is the annotation for a video, for example:
Abuse028_x264.mp4 Abuse 165 240 -1 -1

- The first column is the name of the video
- The second column is the name of the anomalous event
- The third column is the starting frame of the event (you will have to convert each video to image frames first)
- The fourth column is the ending frame of the event.
- For videos in which second instance of event occurs, fifth and sixth contains starting and ending frames of

Negative number means no anomalous event instance. In this example, abuse (instance) only occurs once.

Note: Our videos have 30 frames per second.

2. Anomaly Event Recognition Experiment

Classify 13 anomaly events and normal event

Normal_Videos_for_Event_Recognition.zip contains the normal videos we used for event recognition experiment
Note: rename the unzipped folder to 'Normal_Videos_event'

UCF_Crimes-Train-Test-Split.zip also contains the training and testing split for event recognition experiment
(folder: Action_Recognition_splits)

- Augmented_2D_Features
- Augmented_Anomaly_Vid
- Burglary
- Explosion
- Extracted_C3D_Features
- Extracted_Features-Incep
- FightingA_Part1
- FightingA_Part11
- FightingA_Part2
- FightingA_Part3
- Model_Trained
- Normal_Videos_for_Event
- Per_Frame_Features2D-In
- Shooting
- Testing-Normal-Videos_A
- Training-Normal-Videos-I
- Training-Normal-Videos-I
- UCF_Crimes-Train-Test-5
- Anomaly_Test.txt
- Anomaly_Test_Inception.t
- Anomaly_Train.txt
- Anomaly_Train_Inception.I
- Anomaly_Train_New.txt
- Augmented_Anomaly_Tra
- Normal_Train.txt
- Normal_Train_Inception.ts
- ReadMe_Anomaly_Detect
- Temporal_Anomaly_Annot
- UCF_Crimes_instruction.d
- model.json
- weights_L1L2.mat

Summary

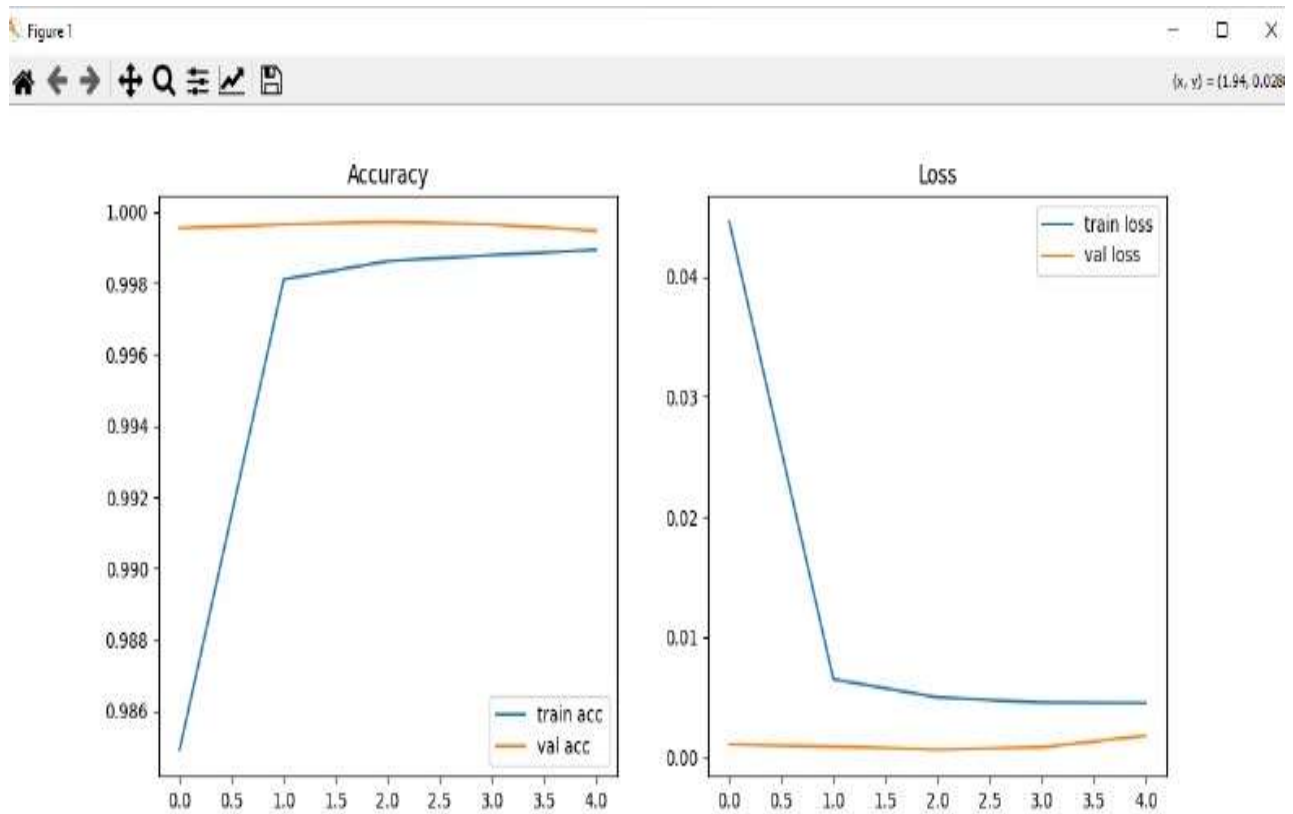
8369 files

5.2 DATA COLLECTION

- Gather and preprocess relevant data for training the detection model.
- Datasets are sourced from platforms like Kaggle, containing images of roadside scenarios, suspicious objects, and normal roadside environments.
- Data preprocessing includes resizing, normalization, augmentation (rotations, flips, brightness adjustments), and labeling objects for supervised learning.

- Page 21

- The model is trained iteratively with techniques like transfer learning and hyperparameter tuning.
- Validation ensures that the model correctly detects suspicious objects without overfitting.



5.4 PERFORMANCE EVALUATION

- Assess the accuracy and reliability of the trained model.
- Metrics such as Precision, Recall, F1-score, and mAP (mean Average Precision) are used to evaluate detection accuracy.
- The system is tested on unseen images to check generalization and robustness in real-world scenarios.

- Evaluating performance ensures the model is ready for integration with the next stage—LLM-based recommendation.

```

17 output_placeholder = st.empty()
18 cwi_result_placeholder = st.empty()
19 yolo_result_placeholder = st.empty()
20 illegal_result_placeholder = st.empty()
21
22 # ===== UPLOADED VIDEO =====
23 if option == "Upload Video/Image":
24     uploaded_file = st.file_uploader("Choose a file", type=["jpg", "jpeg", "png", "avi"])
25     if uploaded_file:
26         stfile = tempfile.NamedTemporaryFile(delete=False)
27         stfile.write(uploaded_file.read())
28         file_path = stfile.name
29
30         if uploaded_file.type.startswith("image"):
31             frame = cv2.imread(file_path)
32             cwi_result, cwi_conf = detect_cwi(file_path)
33             results = yolo_model(frame)
34             annotated_frame = results[0].plot().copy()
35
36             abnormal_detected = False
37             illegal_detected = False
38
39             # Check YOLO detection
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```

1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

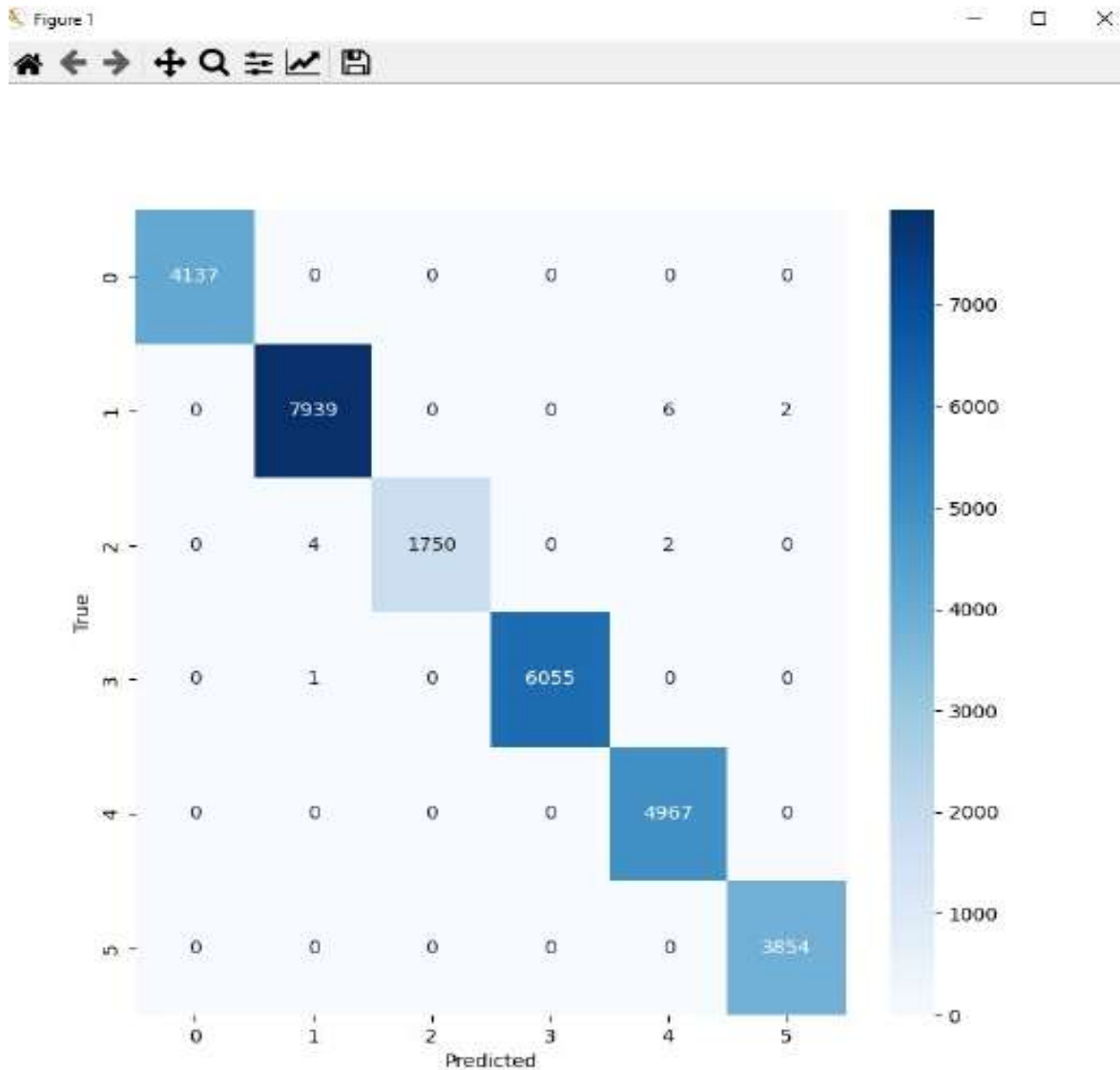
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500

```

```

2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598

```



5.5 LLM-BASED RECOMMENDATION

- Receives object detection results and time-based behavior data.
- Understands scene context and identifies suspicious or abnormal activities.
- Converts detection outputs into high-level insights.
- Generates alerts and action recommendations for security personnel.
- Acts as an intelligent reasoning layer to support faster decision-making.

```
Thony - D:\Charlie\2025\2026\AI-ROAD SIDE\main.py @ 7:25
File Edit View Run Tools Help

main.py
120
121 print(f"\nCategory: {category}")
122 print(f" Sensitivity (Recall): {sensitivity:.2f}")
123 print(f" Specificity: {specificity:.2f}")
124
125 # Classification report
126 print("\nClassification Report:\n")
127 print(classification_report(y_test, y_pred_classes, target_names=CATEGORIES))
128
129 # Confusion matrix plot
130 plt.figure(figsize=(8,8))
131 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
132 plt.xlabel("Predicted")
133 plt.ylabel("True")
134 plt.show()
135

Shell
Processed video: Accreat015_x264.mp4
Processed video: Accreat029_x264.mp4
Processed video: Accreat030_x264.mp4
Processed video: Accreat033_x264.mp4
Processed video: Accreat042_x264.mp4
Processed video: Accreat043_x264.mp4
Processed video: Assault000_x264.mp4
Processed video: Assault011_x264.mp4
Processed video: Assault018_x264.mp4
Processed video: Assault029_x264.mp4
Processed video: Assault033_x264.mp4
Processed video: Assault045_x264.mp4
Processed video: Assault051_x264.mp4
Processed video: Burglary009_x264.mp4

Local Python 3 - C:\Users\ARJ\AppData\Local\Programs\Python\Python310\python.exe
```

```
Thony - D:\Charlie\2025\2026\AI-ROAD SIDE\main.py @ 10:11
File Edit View Run Tools Help

main.py
100 AT OUTLINE: VIDEO: VIDEO ID: 100
101 print(f"Sending GPS (Latitude: {lat}, Longitude: {lon})")
102 url = f"https://api.thingspeak.com/update?api_key=3CGD9P3UKUFI6H8&field2=ANOMALY&field3={lat}&field4={lon}"
103 try:
104     urllib.request.urlopen(url)
105 except:
106     print("Could not send data to Thingspeak.")
107
108 # Show annotated frame
109 cv2.imshow("Detection", annotated_frame)
110 if cv2.waitKey(1) & 0xFF == ord('q'):
111     break
112
113 cap.release()
114 cv2.destroyAllWindows()
115

Shell
Processed video: Fighting013_x264.mp4
Processed video: Fighting022_x264.mp4
Processed video: Fighting030_x264.mp4
Processed video: Fighting036_x264.mp4
Processed video: Fighting037_x264.mp4
Processed video: Fighting048_x264.mp4
Processed video: Normal_Videos_010_x264.mp4
Processed video: Normal_Videos_185_x264.mp4
Processed video: Normal_Videos_365_x264.mp4
Processed video: Normal_Videos_606_x264.mp4
Processed video: Normal_Videos_696_x264.mp4
Processed video: Normal_Videos_781_x264.mp4
Processed video: Normal_Videos_941_x264.mp4
Training samples: 114866, Testing samples: 20717

Local Python 3 - C:\Users\ARJ\AppData\Local\Programs\Python\Python310\python.exe
```





5.6 ALERT AND COMMUNICATION

- Triggers alerts based on suspicious or abnormal activities detected by the system.
- Communicates alerts to security personnel through notifications, messages, or alarms.
- Provides essential details such as event type, camera location, and time.
- Ensures quick response and timely intervention.

Enhances situational awareness and overall surveillance effectiveness

CNN Prediction: Fighting (100.0%)

YOLO Detections: Fighting, person, person, person

⚠ Detected: Fighting, person, person, person. Recommendation: Stay calm and move to a safe location. Alert nearby people and authorities. Stay calm and move to a safe location. Alert nearby people and authorities. Stay calm and move to a safe location. Alert nearby people and authorities. Stay calm and move to a safe location. Alert nearby people and authorities.

✓ Run Camera

5.7 DATA STORAGE AND ANALYSIS

- Stores detection results, images, and system recommendations for auditing and training purposes.
 - Maintains historical records to support compliance, review, and traceability.
 - Enables performance analysis by tracking accuracy, false positives, and false negatives.
 - Supports continuous improvement through data-driven model retraining
- Ensures data integrity and accessibility for long-term analysis



tion Records

Timestamp	Status	Latitude	Longitude
13:51:21	assault	8.7139	77.7567
13:51:37	assault	8.7139	77.7567
13:51:54	assault	8.7139	77.7567
06:45:19	fighting	11.0055	76.9661
06:45:42	fighting	11.0055	76.9661
08:33:53	fighting	11.0055	76.9661
08:36:25	fighting	11.0055	76.9661
08:36:49	fighting	11.0055	76.9661
08:36:05	burglary	11.0055	76.9661
08:40:45	burglary	11.0055	76.9661

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

- Detects suspicious activities in real time and sends instant alerts with key details, enabling quick response and effective intervention.
- Securely stores surveillance data and supports analysis, pattern recognition, and historical review.
- Enhances situational awareness and aids informed decision-making for continuous system improvement.
- It combines precise, real-time object detection with intelligent, automated recommendations, ensuring that authorities can respond swiftly to threats.

APPENDIX A

```
import streamlit as st
import cv2
import numpy as np
from PIL import Image
import tempfile
import time
from ultralytics import YOLO
import av
from streamlit_webrtc import webrtc_streamer, VideoTransformerBase,
WebRtcMode
import pandas as pd
from datetime import datetime
import plotly.graph_objects as go
import threading
import queue
# —— Module-level queue (safe to access from any thread, no
session_state) ——
_ALERT_QUEUE: queue.Queue = queue.Queue(maxsize=50)

# Page configuration
st.set_page_config(
    page_title="RoadGuard - AI Roadside Object Detection",
    page_icon="🚧",
    layout="wide",
    initial_sidebar_state="expanded"
)

st.markdown("""
```

<style>

@import

url('https://fonts.googleapis.com/css2?family=Rajdhani:wght@400;600;700&display=swap');

html, body, [class*="css"] { font-family: 'Rajdhani', sans-serif; }

.title-container {

background: linear-gradient(135deg, #0f2027 0%, #203a43 50%, #2c5364 100%);

padding: 2rem; border-radius: 12px; text-align: center;

margin-bottom: 1.5rem; border: 1px solid #00e5ff33;

box-shadow: 0 0 40px rgba(0,229,255,0.08);

}

.main-title {

color: #00e5ff; font-size: 2.8rem; font-weight: 700;

letter-spacing: 4px; text-transform: uppercase; margin: 0;

text-shadow: 0 0 20px rgba(0,229,255,0.5);

}

.subtitle {

color: #80cbc4; font-size: 1rem; letter-spacing: 2px;

margin-top: 0.4rem; text-transform: uppercase;

}

.alert-container { padding: 1.2rem 1.5rem; border-radius: 10px; margin: 0.8rem 0; }

.alert-danger {

background: linear-gradient(135deg, #1a0000 0%, #3d0000 100%);

border: 2px solid #ff1744; box-shadow: 0 0 20px rgba(255,23,68,0.3);

animation: danger-pulse 1.2s infinite;

}

.alert-safe {

```
background: linear-gradient(135deg, #001a0f 0%, #00251a 100%);
border: 2px solid #00e676; box-shadow: 0 0 15px rgba(0,230,118,0.15);
}

@keyframes danger-pulse {
0% { box-shadow: 0 0 20px rgba(255,23,68,0.3); }
50% { box-shadow: 0 0 40px rgba(255,23,68,0.7); }
100% { box-shadow: 0 0 20px rgba(255,23,68,0.3); }
}

.stButton > button {
background: linear-gradient(135deg, #006064 0%, #00838f 100%);
color: #e0f7fa; border: 1px solid #00e5ff44; padding: 0.6rem 1.5rem;
border-radius: 6px; font-family: 'Rajdhani', sans-serif; font-weight: 600;
font-size: 0.95rem; letter-spacing: 1.5px; text-transform: uppercase;
transition: all 0.25s ease;
}

.stButton > button:hover {
transform: translateY(-2px); box-shadow: 0 6px 20px rgba(0,229,255,0.25);
border-color: #00e5ff99;
}

.section-header {
color: #00e5ff; font-size: 1rem; font-weight: 700; letter-spacing: 3px;
text-transform: uppercase; border-bottom: 1px solid #00e5ff33;
padding-bottom: 0.4rem; margin-bottom: 1rem;
}

</style>

""", unsafe_allow_html=True)
```

| # | Session | State | Init |
|---|---------|-------|------|
|---|---------|-------|------|

```
if 'detection_stats' not in st.session_state:
st.session_state.detection_stats = {
'elephant': 0, 'tiger': 0, 'bear': 0, 'lion': 0,
'person': 0, 'car': 0, 'truck': 0, 'dog': 0
}
if 'stop_live_feed' not in st.session_state:
st.session_state.stop_live_feed = False
if 'live_mode_active' not in st.session_state:
st.session_state.live_mode_active = False
DANGER_ANIMALS = ['elephant', 'bear', 'tiger', 'lion']
```

| # | Detector |
|---|----------|
|---|----------|

```
class RoadsideDetector:
def __init__(self, model_path='yolov8n.pt', conf=0.25):
self.model = YOLO(model_path)
self.class_names = self.model.names
self.conf = conf

def detect(self, image):
return self.model(image, conf=self.conf, iou=0.45, verbose=False)
def draw_boxes(self, image, results):
"""
Annotates image. Returns (img, danger_bool, danger_list, counts_dict).
Never touches st.session_state — safe from any thread.
"""
```

```
danger_detected = False
detected_dangers = []
local_counts = {}
for result in results:
    boxes = result.boxes
    if boxes is None:
        continue
    for box in boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        conf_val = float(box.conf[0])
        cls_id = int(box.cls[0])
        class_name = self.class_names[cls_id]
        is_danger = class_name in DANGER_ANIMALS
        if is_danger:
            danger_detected = True
            detected_dangers.append(class_name)
            color, thickness = (0, 0, 255), 3
        else:
            color, thickness = (0, 230, 118), 2
        cv2.rectangle(image, (x1, y1), (x2, y2), color, thickness)
        label = f"{class_name} {conf_val:.2f}"
        (lw, lh), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.65, 2)
        cv2.rectangle(image, (x1, y1 - lh - 8), (x1 + lw + 4, y1), color, -1)
        cv2.putText(image, label, (x1 + 2, y1 - 4),
            cv2.FONT_HERSHEY_SIMPLEX, 0.65, (255, 255, 255), 2)
        local_counts[class_name] = local_counts.get(class_name, 0) + 1
    return image, danger_detected, detected_dangers, local_counts
```

| # | WebRTC | Transformer |
|---|--------|-------------|
|---|--------|-------------|

```
class VideoTransformer(VideoTransformerBase):
```

```
    """
```

```
    Background thread — uses module-level _ALERT_QUEUE only.
```

```
    Never reads or writes st.session_state.
```

```
    """
```

```
    def __init__(self):
```

```
        self.detector = RoadsideDetector() # loads YOLO once per connection
```

```
    def transform(self, frame):
```

```
        img = frame.to_ndarray(format="bgr24")
```

```
        results = self.detector.detect(img)
```

```
        img, danger, dangers, counts = self.detector.draw_boxes(img, results)
```

```
        try:
```

```
            _ALERT_QUEUE.put_nowait({'danger': danger, 'animals': dangers,
                                     'counts': counts})
```

```
        except queue.Full:
```

```
            pass # drop silently — never block the video thread
```

```
        return img
```

| # | Helpers |
|---|---------|
|---|---------|

```
def alert_html(animal_name: str) -> str:
```

```
    return f"""
```

```
<div class="alert-container alert-danger">
```

```
<h2 style="color:#ff1744;margin:0;font-family:'Rajdhani',sans-serif;letter-
spacing:3px;">
```

```
⚠ DANGER ALERT</h2>
```

```
<h3 style="color:#ff8a80;margin:8px 0 4px;font-family:'Rajdhani',sans-serif;">
```

```
{animal_name.upper()} DETECTED ON ROADSIDE</h3>
```

```
<p style="color:#ffcdd2;margin:0;font-size:0.95rem;">
```

```
Slow down · Keep windows closed · Contact forest dept.</p>
```

```
</div>"""
```

```
def safe_html() -> str:
```

```
return ""
```

```
<div class="alert-container alert-safe">
```

```
<h2 style="color:#00e676;margin:0;font-family:'Rajdhani',sans-serif;letter-spacing:3px;">
```

```
✓ ROADSIDE CLEAR</h2>
```

```
<p style="color:#b9f6ca;margin:4px 0 0;font-size:0.95rem;">
```

```
No dangerous wildlife detected. Drive safely.</p>
```

```
</div>"""
```

```
def play_alert_sound():
```

```
try:
```

```
import winsound
```

```
winsound.Beep(1000, 300); time.sleep(0.05); winsound.Beep(1500, 300)
```

```
except Exception:
```

```
pass
```

```
def refresh_stats(placeholder):
```

```
s = st.session_state.detection_stats
```

```
placeholder.markdown(
```

```
f"🐘 Elephant: **{s['elephant']}** \n 🐅 Tiger: **{s['tiger']}** \n"
```

```
f"🐻 Bear: **{s['bear']}** \n 🦁 Lion: **{s['lion']}** \n"
```

```
f"👤 Person: **{s['person']}** \n 🚗 Car: **{s['car']}** \n"
```

```
f"🚚 Truck: **{s['truck']}** \n 🐕 Dog: **{s['dog']}**"
```

```
def drain_queue():
    """Drain _ALERT_QUEUE. Call ONLY from main thread."""
    latest_danger, latest_animals, merged_counts = False, [], {}
    while True:
        try:
            msg = _ALERT_QUEUE.get_nowait()
            if msg['danger']:
                latest_danger = True
                latest_animals = msg['animals']
                for k, v in msg.get('counts', {}).items():
                    merged_counts[k] = merged_counts.get(k, 0) + v
        except queue.Empty:
            break
    return latest_danger, latest_animals, merged_counts

# _____ Main
```

```
def main():
    st.markdown("""
<div class="title-container">
<h1 class="main-title"> 🚧 RoadGuard AI</h1>
<p class="subtitle">Roadside Object & Wildlife Detection System ·
Powered by YOLOv8</p>
</div>""", unsafe_allow_html=True)
    with st.sidebar:
        st.markdown('<div class="section-header">Control Panel</div>',
            unsafe_allow_html=True)
```

```
confidence_threshold = st.slider("Confidence Threshold", 0.10, 1.0, 0.25,
0.05)
```

```
# Stop live feed when switching modes
```

```
detection_mode = st.radio(
```

```
"Input Source",
```

```
["  Live Camera", "  Upload Image", "  Upload Video"],
```

```
on_change=lambda: st.session_state.update(stop_live_feed=True)
```

```
)
```

```
st.markdown('<div          class="section-header"          style="margin-
top:1rem;">Alerts</div>',
```

```
unsafe_allow_html=True)
```

```
enable_sound = st.checkbox("Sound Alert (Windows only)", value=False)
```

```
enable_visual = st.checkbox("Visual Alert Banner", value=True)
```

```
st.markdown('<div          class="section-header"          style="margin-
top:1rem;">Stats</div>',
```

```
unsafe_allow_html=True)
```

```
stats_ph = st.empty()
```

```
refresh_stats(stats_ph)
```

```
col_feed, col_right = st.columns([3, 1])
```

```
with col_feed:
```

```
st.markdown('<div          class="section-header">Detection          Feed</div>',
unsafe_allow_html=True)
```

```
status_ph = st.empty()
```

```
#          —          LIVE          CAMERA
```

```
if detection_mode == "  Live Camera":
```

```
st.session_state.stop_live_feed = False
```

```
st.session_state.live_mode_active = True

webrtc_streamer(
    key="roadguard-detection",
    mode=WebRtcMode.SENDRECV,
    video_transformer_factory=VideoTransformer,
    async_processing=True,
    media_stream_constraints={"video": True, "audio": False},
)

# Create a container for live updates
live_status = st.empty()

# Main thread: continuously drain queue and update UI
try:
    while not st.session_state.stop_live_feed:
        danger, animals, counts = drain_queue()
        # Update session state with new counts if any detections
        if counts:
            for k, v in counts.items():
                if k in st.session_state.detection_stats:
                    st.session_state.detection_stats[k] += v
            # Update UI based on danger status
            if enable_visual:
                if danger:
                    animal = animals[0] if animals else "Wild Animal"
                    status_ph.markdown(alert_html(animal), unsafe_allow_html=True)
            if enable_sound:
                play_alert_sound()
            else:
                # Don't overwrite danger alert immediately
                current_status = status_ph.empty()
```

Only show safe if we haven't had a danger in the last few seconds

if not danger:

status_ph.markdown(safe_html(), unsafe_allow_html=True)

Refresh stats display

refresh_stats(stats_ph)

Small delay to prevent overwhelming the UI

time.sleep(0.3)

except Exception as e:

st.error(f'Live feed error: {e}')

finally:

st.session_state.live_mode_active = False

————— IMAGE ————— UPLOAD

elif detection_mode == "📁 Upload Image":

st.session_state.stop_live_feed = True

st.session_state.live_mode_active = False

uploaded = st.file_uploader("Choose image", type=['jpg', 'jpeg', 'png'])

if uploaded:

image = Image.open(uploaded).convert("RGB")

image_array = np.array(image)

detector = RoadsideDetector(conf=confidence_threshold)

results = detector.detect(image_array)

proc, danger, dangers, counts = detector.draw_boxes(image_array.copy(), results)

for k, v in counts.items():

if k in st.session_state.detection_stats:

```

st.session_state.detection_stats[k] += v
c1, c2 = st.columns(2)
with c1:
st.markdown("**Original**"); st.image(image, use_container_width=True)
with c2:
st.markdown("**Detected**"); st.image(proc, use_container_width=True)
if enable_visual:
if danger:
animal = dangers[0] if dangers else "Wild Animal"
status_ph.markdown(alert_html(animal), unsafe_allow_html=True)
if enable_sound: play_alert_sound()
else:
status_ph.markdown(safe_html(), unsafe_allow_html=True)
refresh_stats(stats_ph)
#          VIDEO          UPLOAD

```

```

elif detection_mode == "🎥 Upload Video":
st.session_state.stop_live_feed = True
st.session_state.live_mode_active = False
uploaded_video = st.file_uploader("Choose video", type=['mp4', 'avi',
'mov'])
if uploaded_video:
tfile = tempfile.NamedTemporaryFile(delete=False, suffix=".mp4")
tfile.write(uploaded_video.read()); tfile.flush()
if st.button("🚀 Start Detection"):
cap = cv2.VideoCapture(tfile.name)
stframe = st.empty()

```

```
progress = st.progress(0)
frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) or 1
detector = RoadsideDetector(conf=confidence_threshold)
frame_idx = 0

while True:
    ret, frame = cap.read()
    if not ret: break
    if frame_idx % 2 == 0: # process every other frame
        results = detector.detect(frame)
        proc, danger, dangers, counts = detector.draw_boxes(
            frame.copy(), results)
        for k, v in counts.items():
            if k in st.session_state.detection_stats:
                st.session_state.detection_stats[k] += v
            rgb = cv2.cvtColor(proc, cv2.COLOR_BGR2RGB)
            stframe.image(rgb, use_container_width=True)
            if enable_visual:
                if danger:
                    animal = dangers[0] if dangers else "Wild Animal"
                    status_ph.markdown(alert_html(animal), unsafe_allow_html=True)
                if enable_sound: play_alert_sound()
            else:
                status_ph.markdown(safe_html(), unsafe_allow_html=True)
            progress.progress(min((frame_idx + 1) / frame_count, 1.0))
            frame_idx += 1
        cap.release()
        refresh_stats(stats_ph)
    st.success("✅ Video processing complete!")
```

with col_right:

```
st.markdown('<divclass="section-header">Analytics</div>',
unsafe_allow_html=True)

stats = st.session_state.detection_stats

if sum(stats.values()) > 0:

    labels = list(stats.keys())

    values = list(stats.values())

    colors = ['#ff1744','#ff6d00','#8d4513','#ffd600',
'#00e5ff','#69f0ae','#40c4ff','#ea80fc']

    fig = go.Figure(go.Bar(x=labels, y=values,
marker_color=colors[:len(labels)],
text=values, textposition='auto'))

    fig.update_layout(template="plotly_dark", height=260,
margin=dict(l=0,r=0,t=20,b=0),
paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='rgba(0,0,0,0)',
showlegend=False,
xaxis=dict(tickfont=dict(size=9)))

    st.plotly_chart(fig, use_container_width=True)

else:

    st.info("No detections yet.")

    st.markdown('<div          class="section-header"          style="margin-
top:1rem;">Quick Actions</div>',
unsafe_allow_html=True)

    if st.button("🚒 Emergency Alert", use_container_width=True):

        st.error("⚠️ Emergency protocol activated!"); play_alert_sound()

    if st.button("📞 Forest Dept Helpline", use_container_width=True):

        st.info("📞 Toll-free: 1800-123-4567")
```

```
if st.button(" 📍 Share Location", use_container_width=True):  
    st.info(" 📍 Location shared with authorities.")  
    st.markdown('<divclass="section-header"style="margin  
top:1rem;">Safety Tips</div>',unsafe_allow_html=True)  
    st.info(" 💎 Slow down near forests\n 💎 Keep windows closed\n"  
    " 💎 Don't honk at animals\n 💎 Never feed wildlife\n"  
    " 💎 Report sightings immediately")  
    st.markdown("---")  
    st.markdown(  
    "<div    style='text-align:center;color:#546e7a;font-family:Rajdhani,sans-  
serif;"  
    "letter-spacing:2px;font-size:0.85rem;">"  
    " 🚧 ROADGUARD AI · ROADSIDE OBJECT & WILDLIFE  
DETECTION · POWERED BY YOLOv8</div>",  
    unsafe_allow_html=True)  
    if __name__ == "__main__":  
        main()
```

BIBLIOGRAPHY

- [1] Zhihao Wu et al., "*Deep Object Detection with Example Attribute Based Prediction Modulation*", ICASSP 2022.
- [2] Jing Ma et al., "*Joint Image Restoration For Domain Adaptive Object Detection In Foggy Weather Condition*", ICIP 2024.
- [3] E. Shreyas et al., "*3D Object Detection and Tracking Methods using Deep Learning*", RTEICT 2021.
- [4] Shaji T. Blue et al., "*Edge detection based boundary box construction algorithm for YOLOv3*", ICCCNT 2019.
- [5] N.A. Othman et al., "*An Embedded Real-Time Object Detection and Measurement of its Size*", IDAP 2018.
- [6] M.H. Wani and A.R. Faridi, "Deep Learning-Based Video Surveillance System for Suspicious Activity Detection," 2024.
- [7] Z. Wu et al., "Real-Time Deep Learning Approach for Pedestrian Detection and Suspicious Activity Recognition," 2023.
- [8] Wang et al., "SED-NET: Real-Time Suspicious Event Detection via Deep Learning-Based Neural Network," 2025.
- [9] W.J. Jung and W.H. Choi, "Real-Time Pedestrian Surveillance System Using YOLO-Based Detection," 2024.
- [10] H. Ghahremannezhad et al., "Real-Time Accident Detection in Traffic Surveillance Using Deep Learning," 2022.
- [11] K. Oguine et al., "YOLOv3: Real-Time Object Detection Model for Smart Surveillance Systems," 2022.
- [12] Kumar et al., "Deep Learning-Based Real-World Object and Anomaly Detection for Surveillance Videos," 2023.