

AI-Based Smart Road Surveillance System for Accident and Road Anomaly Detection using Deep Learning

Alpit Pathak, Harshit Singh Solanki, Dr. Vishal Shrivastava, Dr. Akhil Pandey

Artificial intelligence & data science, Arya College of Engineering & I.T., Jaipur, India

Abstract - Road accidents and road-surface anomalies such as potholes, cracks, and debris create serious safety and maintenance challenges. This paper presents a software-only prototype concept for smart road surveillance using traffic video, designed to run on a standard laptop with 4 GB RAM and without additional hardware. The first implementation stage focuses on YOLO-based vehicle detection, multi-object tracking/counting, and a Streamlit dashboard. A lightweight pretrained YOLO model and reduced input resolution are proposed to keep resource use manageable. Accident classification and road-damage detection require suitable labelled data and separate evaluation; they are not claimed as completed capabilities unless implemented and tested. LSTM-based temporal modelling is therefore reserved for future work and will be included only after implementation and evaluation. The paper defines a practical prototype workflow and an evaluation plan; measured performance results must be added only after experiments have been run.

Keywords: Road Surveillance, Vehicle Detection, Object Tracking, Vehicle Counting, Yolo, Streamlit, Computer Vision, Traffic Monitoring.

1 INTRODUCTION

Road traffic accidents are a major public-health and economic burden across the world. According to the World Health Organization, approximately 1.19 million people die every year as a result of road traffic crashes, and tens of millions more suffer non-fatal injuries. A large proportion of these incidents occur on roads that are already fitted with CCTV or traffic-camera infrastructure, yet the footage is typically reviewed only after an incident is reported, rather than being used to detect the incident as it happens. In parallel, poorly maintained road surfaces — potholes, cracks, and debris — are a frequent contributing factor in both vehicle damage and secondary accidents, but municipal bodies still largely rely on manual road inspection to identify them.

Advances in deep learning based computer vision have made it possible to automatically analyse video streams for both events (accidents) and static defects (road anomalies) with high accuracy. Convolutional neural networks (CNNs) are effective at extracting spatial features from individual frames, while recurrent architectures such as LSTM and GRU networks capture how those features change over time — for example, the sudden change in vehicle trajectory and speed that characterises a collision. Object-detection frameworks such as the YOLO (You Only Look Once) family further allow such analysis to run in, or close to, real time, which is essential for a surveillance system that must alert responders quickly.

This study proposes a staged, software-only road-surveillance workflow. The initial prototype uses a pretrained YOLO detector to identify common vehicle classes in recorded traffic video, a tracking component to maintain IDs across frames, and a Streamlit dashboard to display annotated video and vehicle counts. The current scope does not assume that accidents or potholes can be reliably classified by a generic pretrained vehicle detector. Those capabilities require appropriate datasets, task-specific training, and independent evaluation. An LSTM component is outside the initial prototype and will be considered only if it is implemented and evaluated.

1.1 Convolutional Neural Networks (CNNs):

CNNs are the standard backbone for extracting spatial features — edges, textures, and shapes — from individual video frames. In road surveillance, CNNs are used to recognise vehicles, pedestrians, and road-surface defects within a single frame, before that information is passed to a detection or temporal-modelling stage.

1.2 YOLO-Based Object Detectors:

YOLO models treat detection as a single regression problem, predicting bounding boxes and class probabilities directly from full images in one pass. This makes YOLO variants (YOLOv3 through YOLOv8) well suited to real-time traffic surveillance, where both vehicles involved in a possible collision and road-surface anomalies must be localised quickly.

1.3 Recurrent Networks (RNN / LSTM / GRU):

Recurrent architectures model how a scene evolves across consecutive frames. In accident detection, an LSTM or GRU layer placed after the CNN/YOLO feature extractor can learn the characteristic sudden deceleration, trajectory change, or object overlap that precedes and accompanies a crash, which a single-frame model cannot capture on its own.

1.4 Autoencoders (AE) for Anomaly Detection:

Autoencoders learn a compressed representation of “normal” traffic scenes in an unsupervised manner; frames that reconstruct poorly are flagged as anomalies. This is useful for detecting accidents or road defects that are rare and therefore hard to collect enough labelled examples for.

Table 1: Summary of existing researches using CNN-based detection

Dataset Name	Architecture	Category	Strength	Limitations
KITTI	CNN	Vehicle and lane detection on urban roads	High detection accuracy in structured scenes	Requires heavy computation for real-time use
UA-DETRAC	Faster R-CNN	Traffic monitoring and vehicle counting	Robust to partial occlusion	Slow inference speed on edge devices
RDD2020	ResNet50	Road surface anomaly (pothole/crack) detection	Strong feature extraction from road texture	Sensitive to lighting and weather variation
BDD100K	CNN (VGG-based)	Road anomaly and obstacle detection	Large and diverse driving scenario dataset	High annotation cost for training
CADP	CNN	Accident scene classification	Works well on CCTV footage	Limited accuracy during night-time conditions

Table 2: Research based on YOLO detection techniques

Dataset Name	Architecture	Category	Strength	Limitations
COCO	YOLOv3	General object detection	Real-time detection speed	Lower accuracy on small/distant objects
CADP	YOLOv4	Road accident detection	High frames-per-second on video streams	False positives in dense traffic scenes
RDD2020	YOLOv5	Pothole and crack detection	Lightweight and deployable on edge devices	Misses hairline cracks and faint damages
UCF-Crime	YOLOv7	Anomalous event detection in surveillance	Good balance of speed and accuracy	Needs GPU acceleration for deployment
Custom CCTV dataset	YOLOv8	Accident and road anomaly detection	High accuracy with real-time inference	Requires large labeled traffic-camera dataset

Table 3: State-of-the-art studies based on RNN / LSTM

Dataset Name	Architecture	Category	Strength	Limitations
CADP	CNN + LSTM	Accident anticipation from video	Learns temporal motion patterns before impact	Computationally expensive to train
DAD (Dashcam Accident Dataset)	LSTM	Early accident prediction	Predicts accidents a few seconds in advance	Sensitive to camera shake and noise
UCF-Crime	ConvLSTM	Spatio-temporal anomaly detection	Captures both spatial and motion features	Long training time on video sequences
A3D	GRU	Accident detection from dashcam video	Efficient sequence modelling with fewer parameters	Limited generalization on small datasets
Custom traffic video	CNN + RNN	Road anomaly and congestion detection	Captures vehicle motion trajectories well	Prone to overfitting on limited footage

Table 4: Research based on Autoencoder anomaly-detection techniques

Dataset Name	Architecture	Category	Strength	Limitations
UCSD Pedestrian	Convolutional Autoencoder	Abnormal event detection in surveillance	Unsupervised learning, no labelled anomalies needed	Sensitive to camera angle and crowd density
Avenue Dataset	Convolutional AE	Abnormal traffic event detection	Learns a compact model of normal behaviour	High false-alarm rate in cluttered scenes

Dataset Name	Architecture	Category	Strength	Limitations
ShanghaiTech Campus	Variational Autoencoder	Anomaly detection in road scenes	Good generalization across scenes	Complex and slow to train
RDD2022	Autoencoder + CNN	Road surface damage detection	Detects previously unseen anomaly types	Requires a very clean training distribution
Custom CCTV footage	LSTM-Autoencoder	Traffic anomaly and accident detection	Effective for temporal anomaly scoring	Needs long, continuous video sequences

Table 5: Hybrid and transfer-learning based researches

Dataset Name	Architecture	Category	Strength	Limitations
BDD100K	ResNet50 + YOLO	Joint accident and anomaly detection	Combines classification with localization	Higher model complexity and memory use
RDD2020	EfficientDet	Pothole and road-damage detection	Efficient scaling of accuracy vs. compute	Needs careful fine-tuning per road type
CADP	Faster R-CNN + LSTM	Accident detection and short-term prediction	High detection accuracy	Slow for strict real-time deployment
Custom dataset	MobileNet + SSD	Lightweight road anomaly detection	Suitable for low-power edge cameras	Lower accuracy on small, distant objects
UA-DETRAC	Attention-based CNN	Traffic accident detection	Focuses computation on relevant regions	More complex architecture to train and tune

The practical study objective is to build and document a reproducible laptop-based prototype for vehicle detection, tracking/counting, and dashboard visualisation, then evaluate it using recorded video. Accident and road-surface anomaly detection are longer-term extensions requiring task-specific data and validation.

2 RELATED WORKS

2.1 Accident Detection from Surveillance Video

One line of work uses CNN and LSTM combinations to learn the spatio-temporal signature of a collision directly from dashcam or CCTV footage [2]. Other researchers [3] use frame-differencing together with a lightweight CNN classifier to flag sudden, large changes in the scene as candidate accidents, prioritising speed over precision. Unsupervised approaches [4] instead learn what “normal” traffic looks like and flag frames that deviate strongly from that model, which avoids the need for large amounts of labelled accident footage but tends to produce more false positives.

2.2 Road Anomaly and Pothole Detection

A second line of work focuses on static road-surface defects. Benchmark datasets such as RDD2020 [6] have enabled CNN- and YOLO-based models to localise potholes, cracks, and other damage directly from road images. These models are typically evaluated using mean average precision (mAP) rather than the accuracy/recall metrics used for accident detection, since the task is closer to object detection than binary classification.

2.3 Real-Time Detection with YOLO

Successive YOLO variants [7–9] have progressively improved the trade-off between detection accuracy and inference speed, making them attractive for surveillance systems that must process many camera feeds simultaneously. Later versions add anchor-free detection heads and improved backbones that further reduce false positives on small or partially occluded objects, which is common in traffic scenes.

This paper therefore adopts a staged approach: a lightweight YOLO-based detector with tracking/counting and a Streamlit interface forms the initial prototype. An LSTM temporal layer is not part of the initial implementation and will be added only if it can be implemented and evaluated reliably.

3 PROPOSED METHODOLOGY

The proposed software-only methodology is designed for a laptop with 4 GB RAM and does not require dedicated external hardware. The first prototype focuses on vehicle detection, tracking/counting, and dashboard visualisation. The following subsections describe the data and implementation plan; final settings should be documented from the actual run.

3.1 Dataset

For the initial demonstration, use a short, legally usable recorded traffic video or a small public sample. The prototype can detect the vehicle classes supported by the pretrained YOLO model. Accident clips and road-damage datasets may

be considered in later stages, but they must not be described as a combined 48,000-frame dataset unless the files are actually collected, labelled, and checked.

3.2 Pre-Processing

Video frames are read with OpenCV and resized to a lower inference resolution to reduce memory and processing load. The detector processes frames sequentially. No 16-frame LSTM sequences are used in the initial prototype. Record the selected resolution, video frame rate, confidence threshold, and any skipped-frame setting in the experiment log.

3.3 Data Splitting

For a prototype demonstration, use separate short video clips for development and testing where possible. Avoid splitting adjacent frames from the same clip across training and test sets if a custom model is trained, because this can inflate evaluation results. The pretrained detector demo itself should be reported as an inference demonstration, not as a newly trained model.

3.4 Proposed Model

The initial prototype consists of (1) OpenCV video input, (2) a lightweight pretrained YOLO model for vehicle detection, (3) a tracking component that assigns persistent IDs, (4) line-crossing or track-based counting, and (5) a Streamlit dashboard for video display and summary counts. This is a practical software pipeline rather than a newly trained CNN-YOLOv8-LSTM architecture.

- i. Input Layer: Holds the frame's attributes such as height, width, and colour depth.
- ii. Convolution Layer: Extracts spatial features by applying learned filters to each frame.
- iii. Pooling Layer: Reduces feature-map size to speed up computation and reduce overfitting.
- iv. Detection Head (YOLOv8): Predicts bounding boxes and class scores for vehicles and road anomalies.

LSTM temporal modelling: not included in the initial prototype. It may be explored as future work after collecting labelled sequences and defining an evaluation protocol.

Use the pretrained detector for inference initially. Training settings, optimiser, loss, epochs, and model-specific metrics should be reported only if a custom training experiment is actually performed.

3.5 Proposed Algorithm

Input: recorded traffic video file

Output: annotated video frames, vehicle track IDs, counts, and dashboard summaries

Process:

1. Create a Python virtual environment and install compatible versions of OpenCV, Ultralytics, Streamlit, and required dependencies.
2. Load a small pretrained YOLO model suitable for CPU inference.
3. Read the selected video with OpenCV and resize frames to a laptop-friendly resolution.
4. Run YOLO detection and retain vehicle classes of interest.
5. Apply the supported tracking method to associate detections across frames.
6. Update per-class and line-crossing counts from tracked vehicle IDs.
 - Use CPU inference and a small model.
 - Reduce frame resolution or process fewer frames if memory/latency is high.
 - Log the model name, settings, and environment versions.
 - Do not claim model training unless training was performed.
7. Display the annotated output and counts in a Streamlit dashboard.
8. Test on short clips and record processing speed, missed detections, duplicate counts, and tracking failures.
9. Report limitations; do not label a vehicle detection as an accident without a validated accident-detection method.
10. Add accident/anomaly classification or LSTM only after implementation and evaluation.

3.6 Practical Prototype Implementation

The prototype will use Python, OpenCV, Ultralytics YOLO, a tracking interface supported by the installed version, and Streamlit. To suit 4 GB RAM, start with a small pretrained model, CPU inference, short clips, and reduced frame

resolution. The dashboard should show the uploaded video, annotated detections, track IDs, per-class counts, and basic run information. The exact package versions and settings must be recorded after the environment is tested.

3.7 Evaluation and Reporting

Run the same test clips more than once and record whether detections and counts are stable. Where labelled ground truth is available, calculate precision, recall, and mAP for detection; compare vehicle counts against manual counts and report absolute counting error. Measure approximate processing speed on the target laptop. Include screenshots and logs as evidence. Do not report accuracy values unless calculated from actual tests.

3.8 Prototype Flow

Start → Select recorded video → Read and resize frames → YOLO vehicle detection → Track IDs → Update vehicle counts → Display annotated frames and summary metrics in Streamlit → Save experiment notes → End.

This workflow demonstrates the first software-only stage on an ordinary laptop. It does not require extra cameras or sensors. Live CCTV integration, reliable accident detection, and pothole/crack detection remain extensions that require suitable input sources, data, and evaluation.

4 RESULTS AND DISCUSSION

The prototype is intended to be evaluated through repeatable tests on recorded video. No experimental accuracy, F1-score, or training/validation result is asserted in this revised draft because such figures must come from an actual run of the implementation.

4.1 Dataset Description

Record the source and licence of each video, clip duration, resolution, and conditions (day/night, traffic density, camera angle). For a small demonstration, describe the actual clips used; do not claim a 48,000-frame labelled dataset unless it has been assembled and verified.

4.2 Performance Metrics

Accuracy measures the proportion of correct predictions: $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$.

Precision is the ratio of true positives to all predicted positives: $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$.

Recall is the ratio of true positives to all actual positives: $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$.

F1-score combines precision and recall into a single metric: $\text{F1} = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$.

No measured training or validation accuracy is reported at this stage. After implementation, record the exact test clips and report only measured values. For vehicle detection, suitable measures may include precision, recall, mAP (if ground-truth labels are available), and inference speed. For tracking/counting, report ID switches or tracking quality where ground truth exists, plus manual count error. A pretrained demo without labelled ground truth should be described as a functional demonstration, not a quantitative accuracy evaluation.

The held-out test-set metrics are pending implementation and evaluation. Do not insert illustrative or estimated percentages as results. Evaluate the system on separate clips and report limitations such as occlusion, small/distant vehicles, camera motion, lighting, and CPU processing speed.

The expected contribution is a reproducible low-cost prototype workflow. Whether it improves accuracy or counting reliability must be established experimentally; no comparative improvement is claimed before those tests are completed.

5 CONCLUSION AND FUTURE WORK

This paper outlines a software-only smart road-surveillance prototype for a laptop with 4 GB RAM. The initial scope is YOLO-based vehicle detection, tracking/counting, and a Streamlit dashboard, without additional hardware. Reliable accident detection and road-surface anomaly detection are not claimed as completed features unless implemented and tested with appropriate data. Likewise, LSTM temporal modelling is reserved for future work until it has been implemented and evaluated.

Future work includes testing on diverse traffic clips, evaluating vehicle-counting error and inference speed, training a task-specific road-damage detector if labelled data are available, investigating validated accident-detection methods,

and considering LSTM temporal modelling only after a reproducible experiment. The final paper should include screenshots, environment details, dataset/source information, and actual measured results from the running prototype.

6 REFERENCES

- [1]. Zhou, Y., et al. "A Survey on Deep Learning for Intelligent Traffic Surveillance and Accident Detection." IEEE Transactions on Intelligent Transportation Systems, 2021. doi: 10.1109/TITS.2021.3061234.
- [2]. Singh, D., & Mohan, C. K. "Deep Spatio-Temporal Representation for Detection of Road Accidents Using Stacked Autoencoder." IEEE Transactions on Intelligent Transportation Systems, 2019. doi: 10.1109/TITS.2018.2835308.
- [3]. Arceda, V. M., & Riveros, E. L. "Fast Live Accident Detection in Traffic Video Using Deep Learning." 2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), 2018. doi: 10.1109/M2VIP.2018.8600844.
- [4]. Yao, Y., Xu, M., Wang, Y., Crandall, D. J., & Atkins, E. M. "Unsupervised Traffic Accident Detection in First-Person Videos." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019. doi: 10.1109/IROS40897.2019.8967556.
- [5]. Ki, Y. K., & Baik, D. K. "Model for Accurate Speed Measurement Using Double-Loop Detectors." IEEE Transactions on Vehicular Technology, 2006. doi: 10.1109/TVT.2006.877716.
- [6]. Arya, D., et al. "Global Road Damage Detection: State-of-the-Art Solutions." IEEE International Conference on Big Data (Big Data), 2020. doi: 10.1109/BigData50022.2020.9377790.
- [7]. Redmon, J., & Farhadi, A. "YOLOv3: An Incremental Improvement." arXiv preprint, 2018. arXiv:1804.02767.
- [8]. Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv preprint, 2020. arXiv:2004.10934.
- [9]. Jocher, G., et al. "Ultralytics YOLOv8." GitHub Repository, 2023. Available online: <https://github.com/ultralytics/ultralytics>.
- [10]. Sultani, W., Chen, C., & Shah, M. "Real-World Anomaly Detection in Surveillance Videos." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. doi: 10.1109/CVPR.2018.00678.
- [11]. Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., & Davis, L. S. "Learning Temporal Regularity in Video Sequences." IEEE CVPR, 2016. doi: 10.1109/CVPR.2016.86.
- [12]. Chan, F.-H., Chen, Y.-T., Xiang, Y., & Sun, M. "Anticipating Accidents in Dashcam Videos." Asian Conference on Computer Vision (ACCV), 2016. doi: 10.1007/978-3-319-54190-7_9.
- [13]. Huang, X., et al. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning." IEEE CVPR, 2020. doi: 10.1109/CVPR42600.2020.00271.
- [14]. Wen, L. J., et al. "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking." Computer Vision and Image Understanding, 2020. doi: 10.1016/j.cviu.2020.102907.
- [15]. Lin, T.-Y., et al. "Microsoft COCO: Common Objects in Context." European Conference on Computer Vision (ECCV), 2014. doi: 10.1007/978-3-319-10602-1_48.
- [16]. Girshick, R. "Fast R-CNN." IEEE International Conference on Computer Vision (ICCV), 2015. doi: 10.1109/ICCV.2015.169.
- [17]. Ren, S., He, K., Girshick, R., & Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." IEEE TPAMI, 2017. doi: 10.1109/TPAMI.2016.2577031.
- [18]. He, K., Zhang, X., Ren, S., & Sun, J. "Deep Residual Learning for Image Recognition." IEEE CVPR, 2016. doi: 10.1109/CVPR.2016.90.
- [19]. Tan, M., Pang, R., & Le, Q. V. "EfficientDet: Scalable and Efficient Object Detection." IEEE CVPR, 2020. doi: 10.1109/CVPR42600.2020.01079.
- [20]. Howard, A. G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv preprint, 2017. arXiv:1704.04861.
- [21]. World Health Organization. "Global Status Report on Road Safety 2023." Geneva: WHO Press, 2023.
- [22]. Ministry of Road Transport and Highways, Government of India. "Road Accidents in India 2022." Annual Report, Transport Research Wing, 2023.
- [23]. Hochreiter, S., & Schmidhuber, J. "Long Short-Term Memory." Neural Computation, 1997. doi: 10.1162/neco.1997.9.8.1735.
- [24]. Cho, K., et al. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation." Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. doi: 10.3115/v1/D14-1179.
- [25]. Simonyan, K., & Zisserman, A. "Very Deep Convolutional Networks for Large-Scale Image Recognition." International Conference on Learning Representations (ICLR), 2015. arXiv:1409.1556.