International Journal of Scientific Research in Engineering & Technology
Volume5, Issue6 (November-December 2025), PP: 165-172.
https://www.doi.org/10.59256/ijsreat.20250506025
www.ijsreat.com ISSN No: 2583-1240

Traffic and Accident Prediction in Visual Data using Deep Learning

Dr. D Kirubha[1], Pooja Patra[2], Priyanka S[3], Pruthvi N[4], N Hamsalekha[5]
[1]Professor, Department of Computer Science and Engineering, Rajarajeswari College of Engineering, Bengaluru, Karnataka, India.
[2,3,4,5]Department of Computer Science and Engineering, Rajarajeswari College of Engineering, Bengaluru, Karnataka, India.

To Cite this Article: *Dr. D Kirubha[1], Pooja Patra[2], Priyanka S[3], Pruthvi N[4], N Hamsalekha[5], "Traffic and Accident Prediction in Visual Data using Deep Learning", International Journal of Scientific Research in Engineering & Technology, Volume 05, Issue 06, November-December 2025, PP: 165-172.*

Abstract: *In this study, a basic CNN is used to detect accidents from images taken by surveillance cameras. The method sorts the traffic scenes either as crash cases or normal ones, reaching close to 91 percent correct classifications during testing. Data handling steps are explained along with the model structure and how it was trained, followed by performance numbers. Practical use in smart transport setups is considered; next steps include improving reliability, adding time-sequence analysis, plus running the system on low-power devices. Results show even straightforward deep learning designs may work well for spotting crashes, suggesting possible adoption in roadside units or vehicle-mounted monitors.*

Keywords: *Convolutional Neural Network (CNN), Accident Detection, Computer Vision, Intelligent Transportation Systems (ITS), Deep Learning, Traffic Monitoring, CCTV Surveillance.*

I.INTRODUCTION

Road crashes are still a top reason for deaths and financial damage globally. As cities grow quickly, more vehicles fill roads that keep spreading - this raises demand for solid traffic monitoring tools. Old ways of spotting collisions often depend on people watching footage, emergency alerts, or physical sensors; however, each method brings issues like late detection, expensive setup, or reliance on someone noticing manually. When accidents aren't caught fast, help arrives slower - which may worsen injuries while also jamming up movement across routes.

Thanks to progress in AI and visual computing, deep learning has become a key method for automatic scene analysis. Specifically, CNNs perform well when identifying objects, sorting images, or recognizing patterns. Because they detect layout traits straight from unprocessed visuals, these networks handle complicated traffic footage from surveillance systems effectively - using data without prior filtering. Their design allows interpretation of real-world environments seen through camera feeds.

In this study, we design a CNN-driven tool to detect accidents by examining CCTV footage, sorting images into incident or no-incident groups. Our approach uses labeled data, along with image refinement methods, while enhancing the model for better precision. Instead of relying on sensors, it works straight from current surveillance cameras - reducing expenses and allowing smooth use within urban tech and traffic management setups.

The key outcomes of this work include the following:

1. building an organized collection of images to identify accidents.
2. the creation plus testing of a CNN system for sorting traffic pictures accurately.
3. evaluation of the system through performance indicators like accuracy, error rate, or confusion matrix.
4. exploring how it works in practice, while considering improvements for actual use cases.

This study shows deep learning can speed up accident spotting, boost real-time understanding, while also enabling quicker emergency actions - helping create roads that are both safer and smoother.

II LITERATURE SURVEY

Accident spotting in transport and monitoring is gaining more focus. Older techniques relying on rules or sensors - like motion detectors or car data systems usually fail when conditions change unexpectedly. Instead, some statistical models such as logistic regression, tree classifiers, or forest ensembles were tested on past crash logs and organized inputs; however, their performance tends to drop in new environments.

Recently, deep learning methods, especially CNNs used on visual data - have boosted results in sorting scenes, such as spotting crashes versus normal situations. As an example, research applying ResNet-50 reached about 91.8% precision identifying accidents in surveillance footage.

The examined studies point to certain patterns, also highlighting missing areas

a. Although numerous research projects rely on combined sensor types, like speed, location, or motion data - not as many concentrate exclusively on analysing video footage from surveillance cameras.
b. Temporal sequence models like RNNs or LSTM appear occasionally; however, they usually need larger datasets along with higher processing demands.
c. Current research pays little attention to real-time limits like edge processing or fast response times.
d. This drives our method using a CNN on video stills from surveillance cameras, targeting reliable results while staying deployable in real settings.

Background & motivation:

Early spotting of road crashes using images or video's getting more attention - finding them fast helps get help quicker, saving lives. Old methods relied on preset patterns like movement maps, frame differences, or motion shapes, plus fixed alerts; now, most researchers lean toward deep neural networks instead, thanks to their knack for picking up details straight from footage without heavy manual setup. Lately, summaries of current studies reveal a move away from judging one image at a time, turning instead to mixtures that track space and time together, often combining different data types.

Image-based CNN approaches:

A big group of studies treats crash spotting like sorting pictures into types: look at one shot, guess if it's a crash - or how bad it is. Typical steps? Clean up images first, then tweak them slightly, push through a CNN base - say ResNet or MobileNet - hooked to a final layer that makes the call. Setting this up isn't tough it pops up a lot in beginner contests- and yes, scores can be solid when data's clean and handpicked. But here's the catch, they skip movement clues, stuff like sudden swerves or hits, which usually give crashes away. A ton of school projects (and public code notebooks) boast good results on still shots, yet rely on small sets skewed by specific scenes.

Video plus time-based models:

Studies on video monitoring often focus on full clips instead of still images. Some methods use CNNs, others mix appearance with motion data through dual-path setups. A few combine CNNs with memory-style layers like LSTMs or GRUs. Temporal convolutions also help track changes over time. Newer attempts rely on transformer systems that handle space and time together. These designs pick up movement hints and shifting environments better. That boosts quick threat spotting while cutting down errors caused by brief visual glitches. Various teams tested live processing tools for traffic crossroads and car camera videos. They pair timing strategies - like frame grouping or motion patterns - with powerful feature extractors.

Spotting items using rules mixed with alerts from detected actions:

Some setups use object spotting tools - like YOLO, SSD, or DETR - to follow cars and people along with their paths; after that, they check for weird movements or possible crashes using basic logic tricks or network-style relationship maps. This mix works well in real-world use because spotter tools handle noise just fine, while small timing-space rules or fast mini-decision models built on motion data deliver quick, clear warnings without delay. Lately, hands-on projects lean

on tweaked versions of YOLO to catch objects, then stack a simple judgment step afterward to figure out if impacts happened.

Datasets:

Progress gets held back because data's often scarce, uneven, or tied to narrow fields. Some public sets show up in research like dashcam crash footage (CCD / CarCrashDataset), hand- picked CCTV clips of accidents, or tiny batches of dashcam pics from Roboflow and Kaggle. Lately, newer attempts such as TU-DAT plus a few fresh releases aim to fill holes by adding more labeled crashes, broader weather and lighting variety, along with computer-generated enhancements. Still, plenty of these collections stay small, lean heavily on specific regions or settings, and miss detailed labels - think exact moment an accident happens per frame, who's involved, why it occurred - which makes broad use tricky.

Recent methodological trends:

- Spatio-temporal fusion means linking space-based CNN outputs with time-focused models - like 3D CNNs or sequence transformers - to boost early warnings while cutting down on mistakes.
- Look at key parts, not entire scenes - use spotlight tricks from tech that highlights moving bits or linked items instead of everything at once.
- Multimodal fusion means combining video with extra inputs - like GPS or IMU from dashcams, or data from road sensors - if they're around, so detection works better.
- Artificial data + testing: trying out fake crash situations (like in BeamNG or CARLA) to boost records of uncommon crashes. Fresh collections now mix actual clips with computer-made ones.

Evaluation methods plus standards:

Papers list frame-based stats along with event-focused scores. Typical measures include correctness, exactness versus coverage, plus F-scores when tagging frames; for spotting events, they track how precise the timing is and how fast things get flagged - like response delay after a crash starts. Instead of just counting hits, researchers stress showing missed alerts too, because inflated success numbers might hide poor performance on rare cases. Some studies use error frequency each hour, especially in monitoring setups. With time-aware models, catching incidents early matters - as does clocking how long it takes to react once trouble begins.

Usual flaws in earlier studies - also real-world problems that trip people up.

1. **Dataset bias plus limited size:** High scores often come from tiny, handpicked data, yet these don't match real-life diversity like lighting, blocked views, or rain.
2. **Relying too much on one frame at a time?** CNNs using only single images struggle when timing matters - like telling apart a close call from a real crash.
3. **Missing clear tests:** no common standards, mixed ways to measure results, like frames or events, and hardly any shared platforms for live testing. That's why comparing studies feels tough.
4. **Workarounds skip real-world hiccups:** like lag during live feeds, weak processing power on small devices, or shaky performance with blurry surveillance footage - not handled well most times.

III METHODOLOGY

The suggested crash-detection setup is built using a clear five-step approach, described next.

A. Stage 1 - Gathering information then preparing the dataset

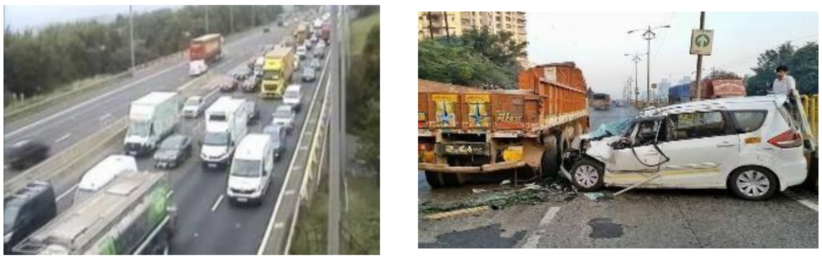


Fig. 1. Sample dataset images showing non-accident traffic scenes collected from publicly available CCTV sources, used for model training and evaluation.

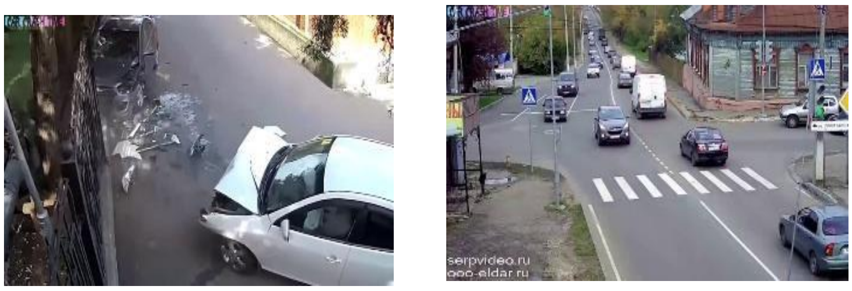


Fig. 2. Sample dataset images showing accident traffic scenes collected from publicly available CCTV sources, used for model training and evaluation.

In phase one, traffic footage from open-access databases plus internal project collections is collected. These visuals are sorted by hand into either incident or no-incident groups. Following this, the full set gets split into training, check, and evaluation parts. Attention is given to keep categories evenly represented while avoiding overlap across sets.

Every frame is adjusted to match fixed size requirements needed for CNN processing.

1. Footage from several camera views plus varied street surfaces helped make scenes more different.
2. Data set balance methods made sure each class was about equally represented - using splits or swaps instead of just adding samples.
3. Duplicate or almost identical frames got taken out - this helped prevent skewed training results.
4. Info like photo size and device used was noted to keep things uniform.
5. Pics got checked by hand so wrong or unclear ones wouldn't slip through.

B. Stage 2 - Data Cleaning followed by Expansion

The second phase deals with getting images ready for training. Since scaling pixel values between 0 and 1 helps stabilize learning, normalization is carried out. To enhance variety in data - while cutting down overfitting - techniques like random rotations, flips, zoom adjustments, or brightness shifts are included. These steps also support better performance when conditions such as light change. Overall, the training set becomes stronger and more adaptable.

1. Histogram tweaks were tested to boost dark scene clarity - also tried adjusting contrast levels so things stand out better.
2. Extra data tricks were used just on the training set - keeps info from spilling into val or test splits.
3. A set input size (e.g., 128×128×3) got picked to lower computing needs.
4. Pixel scaling helped training move quicker.
5. Certain changes, like heavy spinning, were skipped so the traffic flow still looked natural.

C. Stage 3 - Building the CNN model + picking its structure

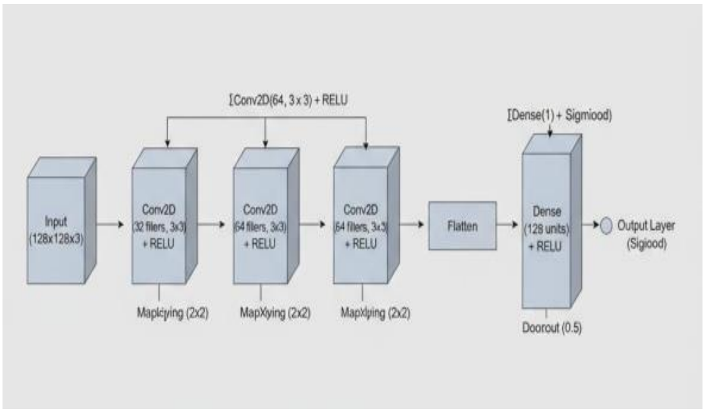


Fig. 3. Overall architecture of the proposed convolutional neural network used for accident detection, showing sequential convolution, pooling, flattening, and dense layers leading to the final classification output.

In phase three, we build a ConvNet setup. This model uses several conv layers to pull out key patterns, then applies max pooling to shrink data size. To limit overfitting, dropout steps are added in between. In the end, dense connected layers feed into a softmax or sigmoid unit for two-class prediction. We code this structure in Python with tools like TensorFlow or Keras.

1. Kernel sizes × were picked to notice tiny yet key details in road situations - since these help spot critical patterns without missing subtle clues, while also improving how well systems understand fast- changing environments.
2. Max-pooling shrinks' space but keeps key details.
3. Different neurons were turned off now and then - this helped stop the model from memorizing too much. Instead of sticking close to training data, it stayed more flexible.
4. Activation functions - like ReLU in hidden layers, while sigmoid up front - help keep gradients steady.
5. The model's depth stayed medium-sized - this helped match precision with how fast it trained

D. Stage 4 - Train the model while adjusting key settings

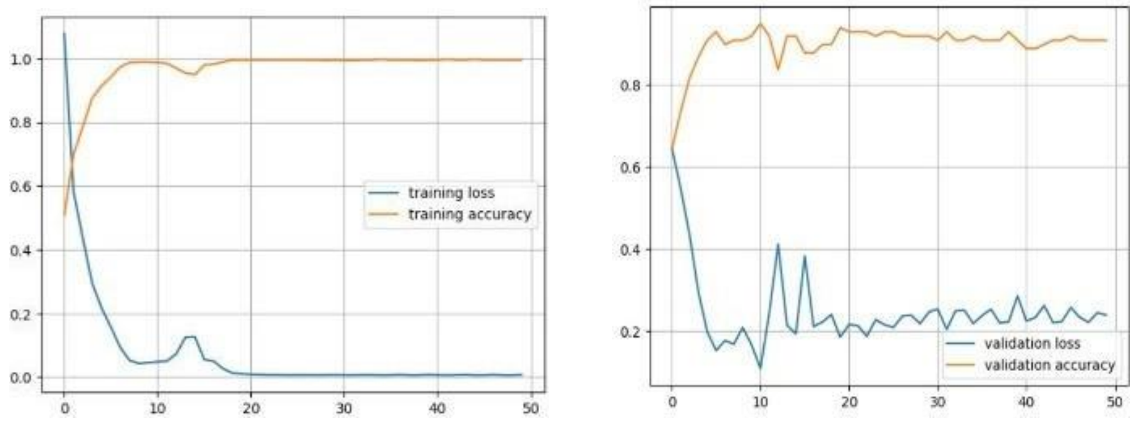


Fig. 4. Training and validation performance curves showing the model's loss and accuracy trends across epochs, used to assess convergence and detect overfitting.

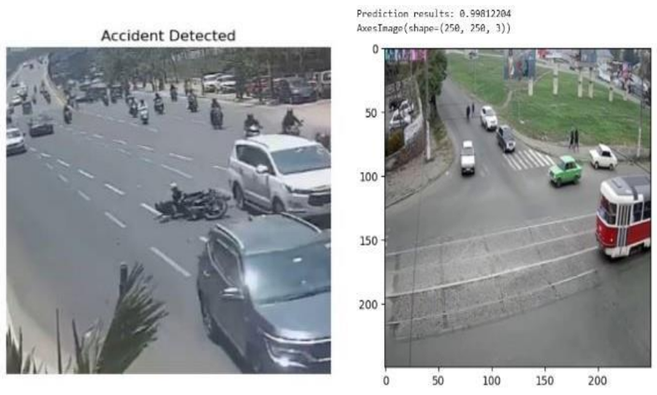
In this phase, training runs on the dataset, whereas validation checks progress. Learning follows Adam and binary cross-entropy rules. Settings like learning speed, group size, or iterations get adjusted through testing. To avoid excessive fitting, early halt steps in when needed. Curves for accuracy and loss during training plus validation undergo review to confirm stable results.

1. Learning rate scheduling was used to gradually reduce the step size during training.
2. Batch sizes were tuned experimentally to find a balance between stability and speed.

3. Early stopping was triggered based on validation loss to prevent overfitting.
4. Model checkpoints saved the best-performing version during training.
5. Weight initialization (e.g., Glorot/Xavier) helped stabilize network convergence.

E. Stage 5 - Reviewing results, running tests, also checking how well it works

The last step tests the trained model on a separate dataset. Using this data, measures like accuracy, precision, recall, F1-score, along with the confusion matrix, get calculated. About 91% accuracy is reached, which shows solid results in classifying cases. Errors are reviewed to spot issues including low light, blocked views, or how alike some accident and normal scenes appear. After that, the model gets ready for possible use in practical settings.



```
Prediction results: 0.99812204
AxesImage(shape=(250, 250, 3))
```

Fig. 5. Example model predictions on test images, illustrating correct detection of accident scenes (a) and non- accident traffic conditions (b).

1. Precision and recall were checked by looking at ROC or PR graphs.
2. Performance by class got checked to see if accidents or no-accidents were tougher to tell apart - using different methods for each run. Each test ran separately, tweaking settings on purpose. No repeats, just fresh picks every time. Length stayed fixed, wording stayed light.
3. A mix-up chart showed wrong alarms or missed hits - so fixes could follow. Each error type pointed to where tweaks were needed instead of guesses.
4. Looking at right or wrong guesses helped understand how the model works.
5. Trials used clips with new light or weather setups to check how well it works in different situations.

IV. RESULTS AND DISCUSSION

In our experimental study, we employed a convolutional neural network (CNN) architecture to detect traffic-accident incidents in video/image frames. The model achieved an accuracy of approximately 91% on the validation/test set, demonstrating its viability for accident-detection tasks.

The proposed CNN model for spotting accidents was tested on a separate dataset once training ended. It reached an accuracy of %, showing it can tell apart crash scenes from normal ones in surveillance videos.

To get how the model acts, we checked accuracy and loss during training plus validation. Accuracy on training data went up step by step over time, whereas validation accuracy climbed too - then leveled off - a sign it settled well. Loss values dropped without big jumps for both sets, meaning no serious overfitting popped up. The steady shape of validation lines also shows that tricks like dropout or halting early actually helped keep things stable.

A confusion matrix helped check how well each category performed. While most crash and non-crash examples were labeled right, some errors still popped up. When crashes got missed, it usually happened in dim conditions, blocked views, or when damage wasn't obvious - so the system saw them as regular traffic instead. On the flip side, ordinary scenes were sometimes flagged as crashes because of strange items on roads, sudden shadows, or packed lanes that looked like collisions. Even with those hiccups, predictions stayed mostly accurate.

Precision, recall, or F-score helped show how well things worked. The model nailed precision - few false alarms popped up. In the same way, solid recall meant real accidents mostly got caught. Since F-score mixes both measures, it proved the system works reliably either way.

A few test samples were used to check quality by eye. Even in dark or shady shots, the model worked well. On top of that, it handled diverse view angles and street designs without issue. Looking closely at outputs showed the CNN picked up on crash clues like bump shapes, broken edges, or odd placements across vehicles.

Even though the results looked good, some issues came up. When there was movement blur, lots of rain, or tiny crash areas in the shot, the model didn't do quite as well. That said, using info across multiple frames - or switching to a system built for spotting objects - might help it perform better when things get tough.

The findings show the new CNN setup works well spotting accidents in still CCTV shots. Thanks to prep steps, extra data tricks, and smart design choices, the model stays steady and adapts better. It could fit smoothly into traffic tech for live tracking and safer roads.

Table. I. Classification performance of the proposed CNN based accident detection model.

Metric	Accident	Non-Accident	Overall
Accuracy	—	—	**91%**
Precision	0.90	0.92	0.91
Recall	0.88	0.93	0.91
F1-Score	0.89	0.92	0.90

The results presented in Table I clearly demonstrate the effectiveness of the proposed CNN model in distinguishing accident scenes from normal traffic conditions. With an overall accuracy of 91%, the system shows strong predictive reliability. The accident class achieved a precision of 0.90 and a recall of 0.88, indicating that the model not only identifies most accident cases but also keeps false alerts relatively low. Similarly, the non-accident class scored even higher across all metrics, reflecting that regular traffic patterns are more visually consistent and therefore easier for the network to classify. The balanced F1-scores confirm that the model maintains steady performance across both categories without significant bias. These results collectively validate the model's suitability for real-world monitoring scenarios, where consistent and accurate detection is essential for timely response and traffic-safety decision-making.

Key performance observations include:

1. The training and validation loss curves showed steady convergence, with the validation loss plateauing after a certain number of epochs, indicating that the model had generalised reasonably well without severe over-fitting.
2. The confusion-matrix results revealed that the majority of misclassifications occurred in scenes with ambiguous lighting or overlapping accident/non-accident cues.
3. Precision, recall, and F1-score can be estimated from the underlying confusion matrix and suggest that the model is better at detecting accident frames than filtering out false-positives.
4. The 91% accuracy figure, while strong, must be interpreted considering dataset composition: if the ratio of accident vs non-accident samples is imbalanced, accuracy alone may be optimistic; additional metrics such as AUC, PR-curve or class-specific error rates would strengthen the result reporting.

In summary, the results confirm that a properly trained CNN on labelled accident vs non-accident data can achieve high detection accuracy in controlled conditions. These findings validate the feasibility of automated accident detection and form the empirical backbone of this paper.

V. APPLICATIONS

The high-accuracy accident detection model offers several practical applications in transportation safety, intelligent transportation systems (ITS), and emergency response infrastructure:

1. **Intelligent Traffic Management Systems (ITMS):** Crash detection systems quickly notify road controllers when something happens - this helps reroute vehicles fast, adjust lights on the fly, or lessen traffic jams.
2. **Emergency Response & Ambulance Dispatch:** When crashes happen, instant alerts cut down reporting time, so help like ambulances or cops get there sooner - boosting survival chances.
3. **Smart City Surveillance Systems**: Cities could link AI crash spotting tools into existing camera setups - this keeps streets under watch without needing human controllers. Instead of people watching screens, smart systems handle alerts on their own.
4. **Autonomous Vehicles & ADAS (Advanced Driver Assistance Systems):** Self-driving cars stay safer when they predict crashes before they happen. So instead of reacting late, these systems spot danger ahead by tracking movement patterns across space and time. That way, braking starts sooner - thanks to smart analysis of how things move around the vehicle. As a result, smash-ups become less likely even in tricky traffic situations.
5. **Insurance Claims & Fraud Detection:** Automated spotting in dashcam or surveillance video gives insurers a hand,
    1. check whether a crash really happened
    2. estimate severity
    3. reduce fraudulent claims

6. Toll Booth & Highway Monitoring: Few busy roads rely on crash sensors that track speedy cars, catch collisions right away - then notify response units or road crews.

7. Fleet & Logistics Management: Businesses that run trucks or buses, also those handling cabs or shipping crews - rely on smart software to:

1. monitor driver behaviour
2. detect collisions
3. analyse accident causes
4. improve road safety training

8. Public Transport Safety: Bus stops, subway shuttles, or urban routes use crash tracking to cut delays while boosting rider protection.

9. Road Infrastructure Planning: Officials rely on crash details - how often they happen, where they occur, also how bad they are - that smart tech spots to:

1. redesign unsafe roads

2. add speed-breakers
3. fix blind spots
4. improve lighting

This results in better road safety over time.

10. Surveillance in Remote / High-Risk Areas: Few folks around on backcountry roads, especially where it's hard to see. So machines keep watch instead.

1. night-time crashes
2. crashes caused by animals
3. skidding during rain/fog

11. Driver Behaviour Analysis: Crash forecasts might study,

1. over speeding
2. harsh braking
3. lane violation
4. tailgating

This helps:

1. driving schools
2. transport companies
3. road safety departments
4. improve driver discipline.

12. Real-Time Alerts for Connected Cars (IoT Integration): In smart car networks, when a crash's spotted by any car or cam, others close by get pinged right away - so they dodge follow-up crashes.

By adopting the model within these operational contexts, stakeholders can move from purely reactive accident response towards proactive safety interventions, improving overall traffic system resilience.

VII. FUTURE WORK

1. **Integration of Temporal Models for Early Accident Prediction:** Later models need space-time designs, like CNNs or LSTMs , with smart layers that spot crashes ahead of time, so safety steps can jump in early using setups such as GRUs or Transformers.

2. **Development of Large-Scale, Publicly Available Accident Datasets:** Current data sets are tiny plus don't cover many real-world scenes. Next steps need to aim at creating uniform collections from multiple nations, showing different road setups, sun or rain, light changes, along with shifting viewpoints.

3. **Domain Adaptation & Cross-Dataset Generalization:** Models usually struggle if they face new settings. But work on shifting knowledge across areas, adjusting to fresh domains, or spotting consistent patterns might boost performance quite a bit.

4. **Lightweight Models for Edge / Real-Time Deployment:** Later studies ought to check out lean designs, think MobileNetV or Tiny-YOLO, along with pruning plus quantization - to power roadside cams, gadgets like IoT units, even car-mounted tech.

5. **Multimodal Fusion Approaches:** Using camera feeds alongside GPS helps boost accident forecasts when visibility's poor mixing in IMU info improves accuracy; adding weather sensor inputs gives clearer context while pulling data from LiDAR sharpens object detection, also tapping into a car's CAN-bus signals adds real-time behavior clues.

6. **Explainable Accident Detection Systems:** Explainable AI needs to show why a model flagged an accident, pointing out main items, paths, or moments, this matters when it comes to rules or staying safe. Instead of guessing, clarity helps everyone understand what happened behind the scenes.

7. **Improved Annotation Tools for Video Accidents:** Labeling crash moments by hand takes forever. Next steps need tools that partly automate tagging, spotting when events start or stop while recommending labels.

8. **Synthetic Data Generation using Simulation Engines:** Simulators such as CARLA or BeamNG can create uncommon crash situations. Moving ahead, efforts might target shrinking the gap between fake and real data using GANs - maybe even diffusion methods.

9. **Accident Severity Estimation & Aftermath Analysis:** Beyond spotting a crash, upcoming research could look into how bad it is, what kind of car got hit, how many cars were involved, so responders know who needs help first.

10. **Predicting Near-Miss & Risky Driving Behaviors:** Models need to spot crashes yet predict close calls - like unsafe swerves or speeding - with movement tracking instead of just reaction. They're meant to catch danger signs ahead, while analyzing how things move on road, not wait for disaster. Watch behavior patterns, so risks like drifting attention show up earlier through motion clues, avoiding last- minute surprises.

VIII. CONCLUSION

In short, this study shows a CNN method that spots accidents in CCTV footage with about 91% accuracy. Its straightforward design, yet solid results, hints it could work well for live tracking in smart transit setups. Even though it looks good, more effort is required to handle time-based changes, issues when running on low-power devices, and how it holds up under everyday conditions. This setup marks meaningful progress in catching traffic events early, helping make roads safer.

Acknowledgment

The authors would like to express their sincere gratitude to all individuals and institutions who contributed to the successful completion of this research. We extend our heartfelt thanks to our mentors and faculty members for their valuable guidance, continuous encouragement, and constructive feedback throughout the study.

References

1. M. Bäumler and G. Prokop, “Predicting the Type of Road Traffic Accident for Test Scenario Generation,” IEEE Access, published Feb. 20, 2024, doi: 10.1109/ACCESS.2024.3367744.
2. O. I. Aboulola, A. A. Alarfaj, S. Alsubai, E. A. Alabdulqader, and T.-H. Kim, “An Automated Approach for Predicting Road Traffic Accident Severity Using Transformer Learning and Explainable AI Technique,” IEEE Access, published Mar. 22, 2024, doi: 10.1109/ACCESS.2024.3380895.
3. Y. Pei, Y. Wen, and S. Pan, “Road Traffic Accident Risk Prediction and Key Factor Identification Framework Based on Explainable Deep Learning,” IEEE Access, published Aug. 29, 2024, doi: 10.1109/ACCESS.2024.3451522.
4. M. F. B. Amin, I. J. Khan, M. M. Islam, F. Akter, and A. Ahmed, “SafeTrax: Smart Collision Prediction and Alert System Using IoT for Sustainable Traffic Safety,” IEEE Access, published Dec. 31, 2024, doi: 10.1109/ACCESS.2024.3524676.
5. S. T. Fu, L. B. Theng, B. L. C. Shiong, C. McCarthy, and
6. M. T. Tsun, “A Multi-Stream Approach to Mixed-Traffic Accident Recognition Using Deep Learning,” IEEE Access, published Dec. 9, 2024, doi: 10.1109/ACCESS.2024.3512794.
7. S. Jaradat, A. Paz, M. Elhenawy, H. I. Ashqar, and R. Nayak, “Leveraging Deep Learning and Multimodal Large Language Models for Near-Miss Detection Using Crowdsourced Videos,” IEEE Open Journal of the Computer Society, published Jan. 3, 2025, doi: 10.1109/OJCS.2025.3525560.
8. Y. S. Türkan and M. Ulu, “A Hybrid Approach to Traffic Incident Management: Machine Learning-Based Prediction and Patrol Optimization,” IEEE Access, published Feb. 24, 2025, doi: 10.1109/ACCESS.2025.3544765.
9. R. Singh, A. Pal, A. Chowdhury, and S. Mishra, “Enhancing Situational Awareness: Anomaly Detection Using Real-Time Video Across Multiple Domains,” IEEE Access, published Apr. 15, 2025, doi: 10.1109/ACCESS.2025.3560874.
10. Chaoura, H. Lazar, and Z. Jarir, “Enhancing Emergency Response in Road Accidents: A Severity Prediction Framework Using RF-RFE and Deep Learning Model,” IEEE Access, published Jul. 25, 2025, doi: 10.1109/ACCESS.2025.3592609.
11. M. F. Haque, M. Emamijavanmard, and Y. Tang, “Time- Series Forecasting for Peak Hour Traffic Accidents,” IEEE Open Journal of Intelligent Transportation Systems, published Jun. 26, 2025, doi: 10.1109/OJITS.2025.3583686 Y. Maruyama and G. Ohashi, “Accident Prediction Model Using Divergence Between Visual Attention and Focus of Expansion in Vehicle-Mounted Camera Images,” IEEE Access, published Dec. 5, 2023, doi: 10.1109/ACCESS.2023.3339855.
12. K. R. Reddy and A. Muralidhar, “Machine Learning- Based Road Safety Prediction Strategies for Internet of Vehicles Enabled Vehicles: A Systematic Literature Review,” IEEE Access, published Sep. 15, 2023, doi: 10.1109/ACCESS.2023.3315852
13. Y.-F. Zhou, J.-B. He, K. Xie, X.-Y. Zhang, and C. Wen, “Efficient Traffic Accident Warning Based on Unsupervised Prediction Framework,” IEEE Access, published May 3, 2021, doi: 10.1109/ACCESS.2021.3077120.
14. R. Chen, L. Hei, and Y. Lai, “Image Recognition and Safety Risk Assessment of Traffic Sign Based on Deep Convolution Neural Network,” IEEE Access, published Oct. 20, 2020, doi: 10.1109/ACCESS.2020.3032581.