
AI ACCIDENT PREDICTION USING MACHINE LEARNING AND DEEP LEARNING

Dr. Abhishek K.*¹, Abhishek Kumar A.², Amogh M.², Manjunath G. S.²

¹Associate Professor, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

²Student, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

Article Received: 18 September 2026

Article Revised: 08 October 2026

Published on: 28 October 2026

*Corresponding Author: Dr. Abhishek K.

Associate Professor, Dept. of CSE, Jyothy Institute of Technology, Karnataka, India.

DOI: <https://doi-doi.org/101555/ijrpa.1137>

ABSTRACT

Traffic accidents are a major concern worldwide, causing about 1.19 million deaths every year along with substantial economic damage. With the growing availability of traffic and environmental data and increasing computing power, Machine Learning (ML) and Deep Learning (DL) are being widely used to predict traffic accidents. This paper reviews the different ways in which ML and DL are used for accident prediction. It covers model families such as CNN and RNN, the datasets used, the factors that influence accidents, and how data is used to improve predictions. The paper also presents the implementation of RouteGuard, a route-risk assessment system that uses a Random Forest model to estimate road-segment risk.

KEYWORDS: Traffic Accident Prediction, Machine Learning, Deep Learning, Road Safety Analysis, Artificial Intelligence, Accident Severity Prediction, Traffic Risk Assessment, Intelligent Transportation Systems, Predictive Modeling, Data Mining, Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, Big Data Analytics, Accident Hotspot Detection.

I. INTRODUCTION

Traffic accidents are a major problem worldwide. They cause many deaths and badly hurt the economy, people, and road infrastructure. As more people move to cities and the number of vehicles grows, roads are becoming more crowded and complex. Transportation safety reports identify road accidents as a leading cause of death, especially among working-age young adults. Accidents also impose heavy costs through hospital bills, property damage, traffic congestion, and lost productivity.

Traditional statistical methods for understanding traffic are no longer sufficient. Artificial Intelligence (AI) now helps predict when accidents are likely to occur by finding patterns in large amounts of data, which in turn makes transportation systems smarter. In recent years, ML and DL have changed the way intelligent transportation systems are built. ML models such as decision trees, random forests, support vector machines, and logistic regression are used to find accident-prone locations, estimate accident severity, and analyze the factors that cause accidents. These models are popular because they are easy to interpret and work well with data. However, they may not perform well on very large, unstructured data. DL models such as convolutional and recurrent neural networks, including long short-term memory (LSTM) networks, are very good at handling spatial and temporal data.

Despite this progress, important problems remain: a lack of data, limited model interpretability, and the need for real-time deployment. This paper reviews the research on using ML and DL to predict traffic accidents, covering the methods, the data used, the factors that affect accidents, the evaluation measures, and directions for future research. By examining what is known and identifying research gaps, this survey aims to guide progress in the field and help make transportation systems safer, smarter, and more efficient. It also presents the implementation of RouteGuard, a route-risk assessment application.

LITERATURE SURVEY

A. Deep Learning Advances in Vision-Based Accident Anticipation

This paper looks at how vision can be used to predict accidents with deep learning techniques [1]. It examines about 147 studies and groups them into feature-based methods, spatio-temporal methods, scene-understanding methods, and multimodal methods. The study shows that deep learning models using videos and images are very effective at predicting accidents because they can capture complicated traffic patterns. Problems remain, including limited data, poor generalization across situations, and long processing times in real-time use. Future work should explore transformer models and the combination of data types.

B. Crash Severity Prediction Using Hybrid CNN-RNN Framework

This study uses a deep learning model that combines a CNN and an RNN to predict the severity of crashes on highways [2]. The model captures both what is happening at a location and how conditions change over time, and it performs better than traditional machine learning and statistical approaches. The authors found that combining the two kinds of models is a good idea because it helps in understanding how the factors that cause accidents are connected. This makes the approach useful for intelligent transportation systems.

C. Graph Neural Networks for Road Safety Modeling

This paper uses Graph Neural Networks (GNNs) to predict road accidents [3]. The road network is treated as a graph of connected nodes. The authors built a dataset of over 9 million accidents and used it to test models such as GraphSAGE, which predict accidents well. The study shows that road connectivity and traffic movement are important, and that modeling roads as a graph improves road safety analysis.

D. US Road Accident Prediction Using Machine Learning

This study uses machine learning models such as Random Forest to estimate how many accidents will occur [4]. It uses a multi-state dataset and considers environmental factors, driver behavior, and road characteristics. The main goal is to predict when accidents will occur rather than only how severe they are, so that authorities can take action before accidents happen and allocate resources wisely.

E. Recent Advances in Traffic Accident Analysis (ML Review)

This survey reviews over 190 research papers on traffic accident prediction with machine learning [5]. It groups the studies into four areas: accident risk, accident frequency, accident severity, and accident duration. The paper points out that ML and DL methods are very good at making predictions, especially when data from several sources is used. Challenges include a lack of accident data, the absence of standard datasets for comparing results, and the need for better models that handle data from different places and times.

F. Road Accident Prediction Using Machine Learning Algorithms

This study uses Random Forest, Decision Trees, and Support Vector Machines to predict when road accidents will happen [6]. The models use information such as traffic density, weather, road characteristics, and driver behavior. By learning from past events, the system can indicate whether accidents are likely in certain areas or at certain times. The goal is to predict occurrence rather than severity, which helps traffic authorities keep roads safe and use resources wisely. ML is better than statistical methods at predicting accidents because it can capture complicated relationships and handle many variables at the same time.

G. Road Traffic Accident Risk Prediction Based on Deep Learning

This paper presents a method for predicting accident risk, especially at intersections [7]. It compares approaches such as Naïve Bayes and Decision Trees, and reports that Deep Neural Networks are very good at finding patterns. The study discusses factors that affect accident risk, such as road width, speed limit, and infrastructure. The findings are based on real accident-risk and traffic-safety data and can help decision makers improve traffic safety.

H. Artificial Intelligence in Road Traffic Accident Prediction

This paper reviews many studies on how AI is used to predict accidents [8]. It finds that, among machine learning methods, Random Forest is especially popular for estimating accident severity and reducing the number of accidents. The authors state that AI makes transportation systems smarter by allowing problems to be predicted early, risks to be analyzed, and plans to prevent accidents to be developed, which makes roads safer for everyone.

II. SYSTEM ARCHITECTURE

The training data is synthetically labelled, while road geometry is obtained from OpenStreetMap (OSM). The hotspot and speed-limit files also contain sample data. The risk scores are therefore estimates and not validated real-world accident probabilities.

System Architecture

RouteGuard adopts a client–server architecture that integrates a web-based user interface, RESTful backend services, a machine learning-based route-risk assessment module, external mapping services, and a persistent database. The architecture supports route planning, road-segment risk estimation, navigation assistance, hazard reporting, and traffic-related complaint management.

The frontend is developed using React and JavaScript, with MapLibre GL providing interactive map visualization. The backend is implemented in Python Flask and exposes REST API endpoints for route scoring, user authentication, ride management, incident reporting, and pothole-related complaints. User authentication uses JSON Web Tokens (JWT), and SQLite stores user accounts, ride records, incident reports, and complaint information.

OpenStreetMap provides road-network information, while OSRM and Nominatim support route calculation and geographical search, respectively. The risk-assessment module divides candidate routes into smaller road segments and extracts relevant features. These features are supplied to a trained Random Forest model to estimate segment-level risk scores. The scores are aggregated into route-level risk to compare alternative routes.

The system also includes a frontend navigation interface that supports location tracking and hazard notifications, and an officer-facing dashboard that gives access to complaint and incident information. If the trained model is unavailable, a heuristic-based fallback may be used, with its results distinguished from model-based predictions.

A. Technology Stack

The technologies used in RouteGuard support interactive visualization, backend processing, geographical data integration, machine learning inference, and data persistence (Table I).

TABLE I. TECHNOLOGY STACK OF ROUTEGUARD.

Component	Technology	Purpose
Frontend	React, JavaScript, Vite	User interface and client-side processing
UI styling	Tailwind CSS, CSS	Responsive interface design
Map visualization	MapLibre GL, React Map GL	Displaying routes, road segments, and geographical information
Backend	Python, Flask	REST API and application logic
Authentication	PyJWT	Token-based authentication
Machine learning	Scikit-learn, Random Forest	Road-segment risk estimation
Numerical processing	NumPy	Feature processing and risk aggregation
Model serialization	Joblib	Saving and loading the trained model
Database	SQLite	Storing users, rides, incidents, and complaints
Routing service	OSRM	Generating routes and obtaining distance and duration
Geocoding	Nominatim	Place search and geographic coordinates

2) Level 1 — Internal data flow.

Component	Technology	Purpose
Road network	OpenStreetMap / Overpass API	Road geometry used in training-data generation
Browser-side vision	TensorFlow.js	Experimental camera-based pothole detection

B. Data Flow Model

The system has two principal data flows: an offline model-training pipeline and an online route-risk prediction pipeline.

1) Model-training data flow

Road geometry is retrieved from OpenStreetMap through the Overpass API. Training examples and labels are generated synthetically from road geometry and predefined risk rules, together with the hotspot and speed-limit data. Features are extracted from each road segment and used to train the Random Forest classifier, and the trained model and its metadata are saved as a Joblib bundle (model.pkl).

2) Online prediction data flow

The frontend sends route-scoring requests to the Flask endpoint POST /api/score-routes. The backend evaluates the road segments and returns route-level scores, risk levels, segment information, and the factors contributing to the estimated risk. The interface uses these outputs to compare alternative routes.

3) Route-risk calculation

For each road segment, the Random Forest produces a risk score p_i . The route score is calculated from the length-weighted mean segment risk and the 90th percentile of segment risks:

$$R = 100 (0.55 \bar{p}_w + 0.45 P_{90}) \quad (1)$$

Where $\bar{p}_w$ is the length-weighted mean segment probability, P_{90} is the 90th percentile of segment probabilities, and R is the route-level risk score on a 0–100 scale.

The implementation classifies scores below 30 as low risk, scores from 30 up to (but not including) 55 as moderate risk, and scores of 55 or above as high risk. These thresholds and probabilities are relative risk estimates of the current implementation and not calibrated probabilities of real-world crashes.

C. Data Flow Diagrams (DFD)

1) Level 0 — Context diagram

The entire RouteGuard application is represented as one system interacting with external entities (Fig. 1).

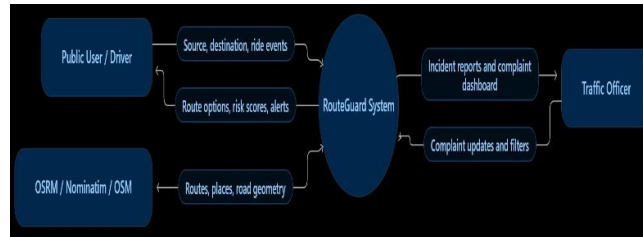


Fig. 1. Level 0 DFD: context diagram of RouteGuard.

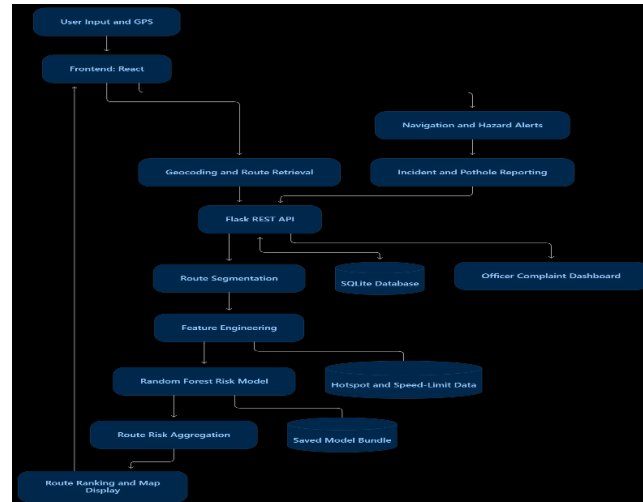


Fig. 2. Level 1 DFD: internal data flow of RouteGuard.

III. IMPLEMENTATION

A. Development Tools

- **Visual Studio Code:** Source-code development and project management
- **Node.js and npm:** Frontend dependency management and execution of development scripts
- **Python:** Backend development, data processing, and machine learning
- **Git and GitHub:** Version control and source-code management
- **Render:** Deployment configuration for hosting the Flask backend
- **Web Browser:** Testing the user interface, maps, and navigation features

B. Frontend Libraries

- **React and React DOM:** Component-based user interface development
- **Vite:** Frontend development server and production build tool
- **Tailwind CSS:** Responsive styling and layout design
- **MapLibre GL and React Map GL:** Interactive map rendering and route visualization
- **React Router DOM:** Client-side page navigation
- **Three.js, React Three Fiber, and Drei:** Three-dimensional visual effects
- **Framer Motion:** Interface animations and transitions
- **TensorFlow.js:** Experimental browser-based pothole detection

- **Lucide React:** User-interface icons

C. Backend Libraries

- **Flask:** REST API development and request handling
- **Flask-CORS:** Cross-origin request configuration
- **PyJWT:** JSON Web Token generation and verification
- **python-dotenv:** Loading environment variables
- **Requests:** HTTP communication with external services
- **NumPy:** Numerical operations and feature processing
- **Scikit-learn:** Random Forest model training and inference
- **Joblib:** Serialization and loading of the trained model
- **Gunicorn:** Production WSGI server on supported deployment platforms

D. Machine Learning and Data Processing

- **Random Forest Classifier:** Main model for estimating road-segment risk
- **Feature Engineering:** Extracts road class, curvature, hotspot proximity and severity, time-related indicators, and speed-limit features
- **Synthetic Data Generation:** Creates training examples and labels from road geometry and predefined risk rules
- **Model Evaluation:** Training code records model evaluation metrics; final reported performance should be measured on appropriately held-out data

E. Mapping and Geographical Services

- **OpenStreetMap:** Road-network and geographical information
- **Overpass API:** Retrieval of OpenStreetMap road data
- **OSRM:** Route generation, distance, and estimated travel duration
- **Nominatim:** Place search and geocoding
- **Browser Geolocation API:** Obtaining the user's location, subject to permission

F. Database and Storage

- **SQLite:** Persistent storage for application records
- **CSV files:** Hotspot information, speed-limit information, and synthetic training data
- **GeoJSON:** Geographical zone boundaries
- **Joblib model bundle (model.pkl):** Stores the trained model and associated metadata

G. Testing and Deployment

- **Vite build tools:** Building and previewing the frontend
- **Flask development server:** Local backend testing
- **Gunicorn:** Production backend serving

- **Render configuration:** Deployment setup for the backend.

IV. RESULTS AND DISCUSSION

This section describes the functional performance of the RouteGuard implementation, its route-risk estimation, visualization, and limitations. Since the current implementation uses synthetically generated risk labels, the results distinguish between demonstrated system functionality and validated real-world accident prediction performance.

A. System Implementation Results

RouteGuard integrates a React-based frontend with a Flask backend to provide risk-aware route planning and road-safety support. The application combines route generation, geographical data processing, road-segment risk estimation, and interactive visualization within a single system.

The frontend enables users to enter source and destination locations, view candidate routes, and compare their estimated risk levels. The backend processes route geometry, extracts road-segment features, and uses a Random Forest model to estimate risk scores. The results are returned to the frontend for presentation through map-based visualizations and route comparisons.

The system also provides ride management, incident reporting, pothole-related functionality, and an officer-facing complaint dashboard. External mapping services provide routing and geocoding, while SQLite stores application records.

B. Machine Learning Model Performance

The project documentation reports a ROC-AUC of approximately 0.66 for the Random Forest model trained on synthetically labelled data, and an approximate upper limit of 0.71 for reproducing the noisy synthetic labelling rule.

ROC-AUC measures the model's ability to distinguish between the two synthetic classes across classification thresholds. A value of 0.66 indicates limited-to-moderate discrimination in this experimental setting; it does not demonstrate equivalent performance on actual road accidents.

C. Route-Risk Assessment and Visualization

RouteGuard estimates risk at the road-segment level and aggregates these estimates to compare alternative routes. The route-ranking mechanism supports three categories: fastest, safest, and riskiest. The fastest route is selected on estimated travel duration, whereas the safest and riskiest routes are determined from the calculated risk scores.

Using a weighted mean together with the 90th percentile gives greater consideration to routes containing high-risk segments. This is useful for route comparison because a route with a low average score may still contain an isolated segment with an elevated estimated risk.

D. Functional Testing

The implementation supports the following functional workflows. These describe implemented capabilities rather than independently measured performance results.

- User authentication using JSON Web Tokens
- Route scoring and segment-level risk estimation through POST /api/score-routes
- Comparison of fastest, safest, and riskiest routes on an interactive map
- Navigation with location tracking and hazard notifications
- Ride management and incident reporting
- Pothole-related complaints and an officer-facing complaint dashboard.

E. DISCUSSION

The implementation demonstrates the feasibility of integrating machine learning-based risk estimation with geographical routing and a web-based navigation interface. Its principal contribution is the integration of segment-level risk scoring, alternative-route comparison, hazard notifications, and incident reporting in one application.

However, the quality of the risk estimates depends on the underlying data. The current training dataset uses synthetic labels generated from predefined rules, while the hotspot and speed-limit information includes sample values. Consequently, the reported ROC-AUC measures agreement with a synthetic target rather than the model's ability to predict observed accidents.

F. Summary of Findings

The current implementation establishes a working framework for combining geographical route planning, Random Forest-based risk estimation, navigation assistance, and road-hazard reporting. The documented ROC-AUC of approximately 0.66 provides an initial evaluation of the synthetic-data experiment, but it is not sufficient to establish real-world accident prediction effectiveness.

Future improvements should focus on validated accident datasets, more reliable geographical information, independent model evaluation, and field testing. These steps are essential before RouteGuard can be considered a dependable decision-support system for real-world road safety.

G. Future Directions

Emerging deep learning architectures are expected to enhance prediction capabilities and can better capture complex relationships in traffic systems than traditional ML models:

- **Transformer-based models:** capture long-range temporal dependencies
- **Graph Neural Networks (GNNs):** model road-network structure
- **Hybrid models (CNN + LSTM + GNN):** support spatial-temporal learning
- **Attention mechanisms:** focus on critical features

V. CONCLUSION

Predicting traffic accidents has become very important for making roads safer and for reducing deaths, injuries, and economic losses. This survey traces the move from traditional statistics to ML and DL. Traditional statistics are good for understanding relationships but are not suited to large, complicated data. Machine learning predicts better because it finds patterns that are hard to see, but these models depend on manually chosen features and struggle with spatial and temporal dependencies. Deep learning methods such as CNN, RNN, LSTM, and Graph Neural Networks can learn features by themselves and capture complicated relationships across time and place.

The survey also emphasizes the importance of diverse datasets, including traffic flow, weather, geographic data, and real-time sensor inputs. Integrating multi-source data has proven a key factor in improving prediction accuracy. Understanding the factors that influence accidents, such as human behavior, vehicle condition, road infrastructure, and environmental conditions, is also vital for reliable prediction systems.

Much remains to be done. Data unavailability, non-uniform datasets, and difficulty in interpreting models are unsolved challenges, and timely real-time deployment is hard to ensure. Existing models are narrow in scope and often fail in other settings. Future work should build stronger models that generalize across regions, favor interpretable AI and architectures, and combine multiple data types. Integrating these models with traffic systems, self-driving cars, and smart-city platforms will make them more useful in practice. With continued research and better data and models, ML and DL can support more intelligent and efficient transport systems and improve road safety.

VI. REFERENCES

1. X. Lin, Y. Zhang, and L. Chen, "Deep Learning Advances in Vision-Based Traffic Accident Anticipation," *IEEE Access*, vol. 12, pp. 34567–34589, 2024.

2. S. Koohfar and S. Kumar, "Crash Severity Prediction Using CNN-RNN Hybrid Model," *Journal of Intelligent Transportation Systems*, vol. 27, no. 4, pp. 512–525, 2023.
3. R. Nippani, A. Sharma, and P. Gupta, "Graph Neural Networks for Road Safety Modeling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 2, pp. 1450–1462, 2025.
4. S. Sivakoti, "Predictive Crash Analytics Using Deep Learning," *Transportation Research Part C: Emerging Technologies*, vol. 152, pp. 104123, 2023.
5. M. Behboudi, A. Hosseini, and H. Zarei, "Recent Advances in Traffic Accident Analysis Using Artificial Intelligence," *IEEE Access*, vol. 12, pp. 56789–56812, 2024.
6. K. Yamarthi, R. Singh, and P. Reddy, "Road Accident Prediction Using Machine Learning Algorithms," *International Journal of Transportation Science and Technology*, vol. 14, no. 1, pp. 98–110, 2025.
7. P. Pawar, S. Deshmukh, and A. Patil, "Deep Learning-Based Accident Risk Prediction for Smart Transportation Systems," *IEEE Access*, vol. 13, pp. 11234–11250, 2025.
8. H. Siswanto, B. Santoso, and D. Prabowo, "Artificial Intelligence in Road Traffic Accident Prediction," *Journal of Advanced Transportation*, vol. 2023, pp. 1–12, 2023.
9. A. Budzyński, M. Federowicz, A. Jabłoński, W. Hasan, J. Gorszánów, and T. Nikitishyn, "A Machine Learning Approach for Predicting Road Accidents," *Safety & Defense Journal*, vol. 15, no. 2, pp. 45–58, 2024.
10. J. Kim and S. Park, "Temporal Accident Risk Prediction Using Deep Learning," *Transportation Research Part C: Emerging Technologies*, vol. 148, pp. 103987, 2023.
11. L. Zhao and Y. Wang, "Graph Convolutional Networks for Accident Prediction in Road Networks," *Information Sciences*, vol. 650, pp. 120–135, 2024.
12. D. Patel, A. Chawla, and R. Gupta, "Multimodal Deep Learning for Traffic Accident Risk Prediction," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 6789–6801, 2024.
13. S. Dubey, A. Sharma, and R. Mishra, "AI-Based Road Accident Detection and Prediction Using Hybrid Models," *Educational Administration: Theory and Practice*, vol. 30, no. 1, pp. 210–220, 2024.
14. Y. Berhanu, "Machine Learning for Road Traffic Accident Prediction Using Spatial Data," *Transportation Research Procedia*, vol. 74, pp. 256–263, 2024.
15. H. Li, X. Zhang, and Y. Chen, "Traffic Accident Risk Prediction Using CNN-LSTM and Graph Neural Networks," *IEEE Access*, vol. 13, pp. 22345–22360, 2025.