

Smart EV-Net: A Machine Learning Framework for Intelligent Networked Electric Vehicle Assistance

Prasanna Ranjith Christodoss¹, Rahul Chauhan², Bhargav Gadi^{3,*}, Abduljaleel Al-Janabi⁴, Dinka Lale⁵, Ramadas Kandath⁶

¹Department of Computing, Mathematics and Physics, Messiah University, Mechanicsburg, Pennsylvania, United States of America.

²Department of Fitch Ratings, The Fitch Group, New York, United States of America.

³Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram, Chennai, Tamil Nadu, India.

⁴Department of Environmental Engineering, University of Babylon, Al-Hilla, Babylon, Iraq.

⁵Faculty of Electrical Engineering and Applied Computing, University of Dubrovnik, Dubrovnik, Dubrovnik-Neretva County, Croatia.

⁶Department of FSM Adoption and Services, SAP, Dallas, Texas, United States of America.
prchristodoss@messiah.edu¹, rahul.chauhan@thefitchgroup.com², bs3187@srmist.edu.in³,
abdajaleel.fadhil@uobabylon.edu.iq⁴, lariat@unidu.hr⁵, ramadas.kandath@sap.com⁶

Abstract: Driving an electric vehicle in uncertain situations requires trusting your battery range and finding an available charger. NASK-ML confronts these obstacles. Researchers combined high-frequency vehicle data with privacy-preserving network techniques (BLAKE3 IP masking). Researchers added machine-learning modules to anticipate short-term range, rank reachable chargers, automate charge-versus-tow decisions, and detect network irregularities. Our approach treats vehicle telemetry as a temporal signal, not a formula. The solution gives ordinary drivers and fleet operators the most accurate recommendations by integrating live data with past station data. Of course, reality is messy. Researchers had to account for poor internet connectivity, edge-device compute constraints, and privacy demands. NASK-ML's hybrid architecture handles this friction. CNNs, MLPs, LSTMs, and Transformers were coupled by attention mechanisms to encode spatial and temporal data. Researchers kept ranking models lightweight to sort candidates on the device. So raw telemetry never leaves the vehicle, researchers added an optional federated training pathway. Researchers used SHAP and time-series saliency maps to show operators why the system made a safety judgement. Researchers found clear findings in our simulated NASK logs. Our hybrid models outperformed typical heuristics, and federated training showed that user privacy may be protected without compromising system value.

Keywords: Electric Vehicle; High-Frequency Vehicle Data; Machine-Learning Modules; BLAKE3 IP Masking; NASK-ML; Historical Station Information; Short-Term Range.

Received on: 01/07/2025, **Revised on:** 10/09/2025, **Accepted on:** 03/11/2025, **Published on:** 09/06/2026

Journal Homepage: <https://www.fmdbpublish.com/user/journals/details/FTSIS>

DOI: <https://doi.org/10.69888/FTSIS.2026.000773>

Cite as: P. R. Christodoss, R. Chauhan, B. Gadi, A. Al-Janabi, D. Lale, and R. Kandath, "Smart EV-Net: A Machine Learning Framework for Intelligent Networked Electric Vehicle Assistance," *FMDB Transactions on Sustainable Intelligence and Security*, vol. 1, no. 2, pp. 102–117, 2026.

Copyright © 2026 P. R. Christodoss *et al.*, licensed to Fernando Martins De Bulhão (FMDB) Publishing Company. This is an open access article distributed under [CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/), which permits copying, remixing, redistributing, transformation, and building upon the material in any medium so long as the original work is properly cited.

*Corresponding author.

1. Introduction

When you drive an electric vehicle (EV) in the real world, with all its uncertainties, two big problems remain: figuring out how far you can go on the battery and finding a charging station when you really need one [1]. Electric cars are becoming increasingly sophisticated thanks to advances in battery technology, energy management, and intelligent control systems. But range anxiety is still one of the biggest concerns for EV users. Drivers often hesitate to travel long distances because they are uncertain whether the remaining battery capacity will be sufficient to reach their destination or the nearest charging facility [2]. Meanwhile, the growing number of EVs has increased demand for intelligent charging assistance systems that can detect available, suitable charging stations based on real-time conditions rather than static location information. The proposed NASK-ML: Machine Learning Enhancements for Networked EV Assistance addresses these challenges with a holistic, machine learning-based framework that includes vehicle telemetry analysis, privacy-preserving communication, intelligent decision-making, and explainable artificial intelligence [3]. NASK-ML solves these roadblocks by exploiting high-frequency vehicle data with state-of-the-art privacy-preserving network protocols, notably BLAKE3-based IP and device identity masking. Modern connected electric vehicles continuously generate large volumes of operational data, such as battery state-of-charge, energy consumption patterns, driving behaviour, vehicle speed, acceleration, environmental conditions and charging history. This information is valuable for improving EV assistance systems; however, collecting and processing it raises serious privacy and security concerns [4].

To address this challenge, NASK-ML integrates secure data-handling mechanisms that enable intelligent analysis while minimizing exposure of sensitive vehicle data. Unlike traditional EV management systems, which rely on static mathematical equations or pre-defined battery models, NASK-ML treats vehicle telemetry as a time-varying signal [5]. Real-world driving conditions are highly dynamic, and energy consumption can vary significantly depending on traffic congestion, road elevation, weather conditions, driving habits and battery degradation. Because telemetry is a time-dependent data stream, the proposed system can identify complex relationships and patterns in telemetry data that traditional approaches often overlook. This allows more precise prediction and dynamic recommendations for individual EV drivers and commercial fleet operators [6]. The proposed framework includes several machine-learning modules to support important EV assistance tasks. The modules include short-term remaining driving range prediction, intelligent charging station ranking, automatic charge-versus-tow decisions, and network anomaly detection. By using real-time vehicle information and historical charging station data, NASK-ML can generate more accurate, efficient and context-aware recommendations [7]. Improving short-term remaining driving range (RDR) prediction is one of the system's main purposes. Accurate remaining-range estimation matters because it directly affects vehicle safety and route planning. An erroneous prediction may lead drivers to take wrong routes, experience unnecessary charging delays or exhaust battery power prematurely. Classic physics-based battery models provide useful estimates under controlled conditions but often fail in real-world settings [8].

They are not easily adaptable to unpredictable factors such as aggressive driving patterns, sudden temperature fluctuations, battery aging effects, traffic conditions and noisy sensor measurements. Machine learning-based approaches are more flexible as they learn directly from the historical and real-time vehicle data [9]. NASK-ML uses advanced learning methods to analyze multiple input variables simultaneously and detect hidden patterns that influence energy consumption. Rather than considering only the current battery percentage, the model accounts for a wider set of features, such as past driving behavior, road conditions, environmental factors and energy usage trends. NASK-ML employs a hybrid architecture combining spatial and temporal learning models to improve prediction performance. Spatial encoders such as Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) extract relationships among different vehicle and environmental parameters. These models can capture complex interactions among multiple input features [10]. For example, they can study the combined impact of speed, acceleration, temperature and road conditions on battery consumption. To capture temporal dependencies, advanced sequential models such as Long Short-Term Memory (LSTM) networks and Transformer architectures are employed. Because vehicle operation is a series of events, accurate predictions require understanding previous driving patterns [11]. LSTM networks can capture long-term dependencies in telemetry data, while Transformers can learn stronger attention-based relationships to identify important temporal patterns. Attention mechanisms further improve NASK-ML performance by allowing the model to focus on the most influential information for prediction. Rather than treating all input features equally, attention-based learning learns which factors matter most for a given decision [12].

For example, during highway driving, vehicle speed and aerodynamic resistance may dominate, while in urban environments, frequent acceleration and braking may matter more. This adaptability improves the reliability of range estimation across different driving scenarios. Another important part of NASK-ML is intelligent charging station recommendation. This is not just a geographical search for the right charger [13]. The nearest charging station is not always the best choice. It may be less available, have longer wait times, have incompatible connectors, be more expensive or offer slower charging speeds. Instead, the system treats charger selection as a complex ranking problem rather than a simple distance calculation [14]. The charging recommendation module ranks multiple charging candidates using learning-to-rank approaches based on various dynamic factors. This includes remaining battery range, estimated time of arrival, connector compatibility, charging speed, station

reliability, electricity price, current occupancy and predicted waiting time [15]. NASK-ML combines historical station performance data and real-time vehicle information to identify the most reliable and efficient charging station. This smart ranking capability is especially useful in dense urban areas where several charging stations may be available within a small radius. A driver might reach a station that is physically nearer but have significant delays due to congestion. NASK-ML solves this by considering overall suitability rather than distance alone. This means less unnecessary travel, more efficient charging and lower operational costs for EV users and fleet managers. Beyond prediction and recommendation, NASK-ML also supports safety-critical decision-making [4].

One of the most difficult situations is when the vehicle battery level is critical. In this case, the system must decide whether the driver should proceed to a charging station or call for a tow immediately [5]. This charge-versus-tow decision needs to be made with extremely high confidence, as incorrect recommendations may introduce safety risks. To address this problem, NASK-ML combines calibrated classification models that analyse battery condition, predicted energy consumption, route characteristics, environmental factors and uncertainty levels [6]. The model determines if there is enough energy to finish the journey safely. Furthermore, explainability techniques allow operators and users to understand the reasoning behind each recommendation. Another area of concern is cybersecurity, since connected EV systems rely heavily on digital communication. Vehicles are continually connected to charging networks, cloud platforms and other services. This interconnectivity makes the system vulnerable to cyberattacks such as spoofed requests, false telemetry injection, and unauthorised access and data manipulation. NASK-ML addresses these issues with machine learning-based anomaly detection techniques [7]. The anomaly detection module continuously monitors network activity and detects abnormal patterns that may indicate malicious behaviour. Unlike traditional rule-based security systems, machine-learning models can detect previously unknown attack patterns by learning normal communication behavior. Importantly, the system performs security analysis without requiring unnecessary access to private user data. Privacy protection is a major challenge in large-scale EV intelligent systems. Vehicle telemetry contains sensitive information such as user behavior, travel routes and operating characteristics. While centralized machine-learning models can yield high accuracy with massive amounts of data, this approach can also create privacy risks [8].

To address this issue, NASK-ML proposes a federated learning pipeline with algorithms like Federated Averaging (FedAvg). In this architecture, raw vehicle telemetry stays on board the vehicle or the edge device. Instead of sending sensitive information to a central server, vehicles train models locally and transmit only model parameters. The server then aggregates these updates to improve the global model [9]. This federated approach offers many advantages. It enables multiple vehicles to cooperate to improve machine learning performance while protecting user privacy. It also reduces communication overhead because it only sends model updates rather than full datasets. Furthermore, NASK-ML incorporates other privacy-enhancing methods such as gradient clipping and Differentially Private Stochastic Gradient Descent (DP-SGD) for environments requiring strict regulatory compliance. However, hardware limitations limit deploying advanced machine learning algorithms directly inside vehicles [10]. Contemporary Transformer-based models require significant computational resources, memory, and processing power. Simple automotive edge devices are not efficient enough to support very large models in real time. To address this limitation, NASK-ML uses a hybrid edge-cloud architecture. In this architecture, lightweight, optimized models run directly on the vehicle, enabling fast decision-making. These models handle tasks that require immediate responses, such as range estimation, charging recommendations and emergency decisions. NASK-ML performs computationally heavier processes, such as large-scale model training and optimisation, in cloud environments. To improve efficiency, NASK-ML applies model compression techniques including quantization, knowledge distillation and lightweight neural network design [11].

These approaches reduce computational requirements while keeping acceptable accuracy. As a result, the system can operate efficiently in scenarios with limited connectivity or hardware resources [12]. Transparency is also vital because EV assistance systems affect critical safety decisions. Users and operators must be confident that recommendations are based on meaningful information, not unexplained predictions. Thus, NASK-ML uses explainable artificial intelligence techniques such as SHAP and time-series saliency maps. These techniques provide insight into the reasons behind the system's prediction or recommendation. Operators can then determine, for example, whether battery status, driving behavior, weather conditions or road characteristics influenced the range estimate [13]. This builds trust, supports system validation and enables better decision-making. The proposed framework was validated on simulated NASK logs containing vehicle telemetry, charging station information and network activity data, where it showed significant improvements over traditional heuristic-based approaches. The system captured spatial and temporal relationships in the EV data by combining CNNs, MLPs, LSTMs, Transformers and attention mechanisms. Federated learning experiments also showed that privacy protection can be achieved without a significant loss of model performance [14]. To summarise, NASK-ML effectively transforms raw vehicle telemetry into reliable, intelligent EV assistance. The framework offers a complete solution for future electric mobility systems integrating machine learning, privacy-preserving communication, cybersecurity monitoring, federated learning and explainable AI. The approach reduces range anxiety, improves charging decisions, enhances safety, preserves user privacy and supports efficient fleet management. "As electric vehicles become increasingly integral to sustainable transportation, intelligent frameworks such as NASK-ML would be critical in building safer, smarter and more reliable, connected mobility ecosystems [15].

2. Literature Review

Electric vehicles (EVs) are rapidly changing modern transport, but range anxiety remains one of the biggest barriers to mass adoption. The overall EV experience is uncertain and inconvenient without the ability to accurately estimate their travel range or know if a charging station will be available when needed. Therefore, accurate short-term remaining driving range (RDR) prediction is crucial for safe route planning and reliable vehicle operation. It's the difference between confidently driving to the next charging station and needing emergency roadside assistance. However, recent research suggests that successful RDR prediction requires combining battery characteristics, vehicle parameters, environmental conditions and driving behavior rather than relying on battery measurements alone. Traditional physics-based battery models are often ill-suited to complex real-world driving conditions. The models are sensitive to stochastic factors such as weather, driving patterns, battery aging and sensor noise. Data-driven machine-learning approaches that can learn complex temporal relationships from large-scale vehicle telemetry data are flexible. Eagon et al. [2] showed that recurrent neural network architectures can capture short-term variations and long-term trends in EV range prediction, which motivated the adoption of LSTM- and GRU-based architectures in NASK-ML. Similarly, Zuo et al. [3] showed that combining causal convolution techniques with LSTM-based temporal aggregation significantly reduces forecasting errors on real-world telematics datasets. This has led to a telemetry-centric machine-learning approach that models vehicle behavior as a continuous temporal signal. Another big challenge in intelligent EV assistance systems is the charging station recommendation.

Choosing the best charger is not just a matter of calculating the shortest physical distance. The most suitable charging option depends on many dynamic factors such as connector compatibility, charging speed, station availability, waiting time, electricity cost, traffic conditions and real-time battery consumption. Conventional distance-based approaches often break down in dense urban areas, where the closest station is not necessarily the fastest or most dependable. Ma and Faye [4] showed that using time-dependent features such as historical usage patterns and calendar information in LSTM-based models can significantly improve charging station occupancy prediction. Moreover, Jauhar et al. [5] showed that combining web-based station information with vehicle telemetry improves availability and waiting-time estimation. Similarly, Manai et al. [6] also highlighted the significance of demand clustering and regional usage patterns in predicting charger occupancy and expected delays. Building on these results, NASK-ML frames charging station selection as a learning-to-rank problem rather than a simple nearest-neighbor search. The system uses historical charging station behaviour and real-time vehicle data to rank available chargers by reliability, accessibility and efficiency. This intelligent ranking method can reduce unnecessary detours and improve charging decisions, lowering operational costs for each EV and the fleet. Decision-making in safety-critical situations is another requirement for EV assistance systems. Classification models must be highly accurate and explainable to determine whether the driver should continue toward a charging station or immediately request towing assistance. Moreover, connected EV systems are vulnerable to cybersecurity threats as vehicles continuously exchange information via communication networks. Possible threats are spoofed requests, abnormal communication behavior and data manipulation attacks.

NASK-ML uses machine-learning-based anomaly detection mechanisms to address these problems. Srrekumar and Lekshmi [9] showed that autoencoder-based approaches effectively detect abnormal sensor patterns in distributed networks with a low false-positive rate. Similarly, Yaghoubi et al. [10] suggested an asymmetric stacked autoencoder and deep neural network framework for efficient edge-based anomaly detection, providing useful insights for NASK-ML lightweight on-vehicle security modules. Privacy preservation is another fundamental problem in connected EV environments. Although large amounts of telemetry data can achieve high accuracy, centralized machine-learning approaches are risky because they expose sensitive vehicle information. The original NASK design employs BLAKE3 hashing to conceal device identities and facilitate secure logging. The BLAKE3 cryptographic framework offers fast, efficient hashing on embedded devices and cloud platforms, making it a strong candidate for privacy-preserving telemetry applications. To enhance privacy protection, NASK-ML uses federated learning based on algorithms such as Federated Averaging (FedAvg). Chellapandi et al. [7] studied federated learning techniques for vehicular environments and discussed challenges related to communication constraints, non-IID telemetry data, and secure aggregation. In the proposed system, vehicles store raw telemetry locally, train locally and only upload the updated model parameters to the central server. For applications with more stringent regulatory compliance, researchers apply additional privacy techniques such as gradient clipping and Differential Privacy-based Stochastic Gradient Descent (DP-SGD). However, deploying sophisticated machine-learning models also poses computational challenges, as large Transformer-based architectures require significant processing resources.

To address this limitation, NASK-ML employs lightweight, optimized models for real-time vehicle decisions while leveraging more powerful cloud-based models for complex training tasks. Researchers use knowledge distillation, pruning, and quantisation to reduce model size and latency. These optimization strategies have been shown to transfer knowledge from large teacher models to compact student models for edge deployment [14]. Finally, explainability is essential to ensure safety in EV applications, as users and operators need to understand the reasons behind a particular recommendation or prediction. SHAP-based interpretation methods provide insights into feature importance and model behaviour. Ponce-Bobadilla et al. [15]

explored SHAP applications for supervised learning models, including time-series explanations and visualization techniques that can support NASK-ML’s interpretable decision framework. Moreover, Zheng et al. [13] presented effective techniques for summarising sequences with GPS histories, supporting NASK-ML’s ability to personalise range prediction and charger ranking based on past travel patterns. The learning-to-rank methods proposed by Shaw et al. [11] and Wang et al. [12] form the basis for improving charger-recommendation accuracy while reducing ranking bias. The reviewed literature provides the technical basis for NASK-ML by combining advances in EV range prediction, charging recommendation, federated learning, anomaly detection, model optimisation, and explainable artificial intelligence. By combining these approaches, NASK-ML turns raw vehicle telemetry into reliable, privacy-sensitive, and intelligent EV assistance that enables safer, more efficient electric mobility systems.

3. Methodology

3.1. System Overview and Data Flow

The NASK-ML pipeline ingests streaming telemetry from vehicles and periodic station metadata, processes the data into feature-rich snapshots and short telemetry windows, and feeds these into task-specific ML modules (range forecasting, charger ranking, charge/tow classification, anomaly detection). The system is designed for mixed edge/cloud deployment to meet latency, privacy, and compute constraints.

3.1.1. Architecture Overview

System data flow steps:

- Telemetry ingestion — clients publish via MQTT or HTTPS POST: timestamp, vehicle_id_masked, soc, voltage, current, speed, accel, lat, lon, heading, ambient_temp.
- Masking & identity protection — device identifiers and IPs are hashed (BLAKE3) on-device before transmission. Periodically rotated nonce preserves unlinkability across long intervals.
- Ingestion queue & buffering — messages queued (Kafka/RabbitMQ) and written to a short-term feature store (Redis Streams) for windowed sequence assembly.
- Preprocessing & feature engineering — resample, handle missing values, create derived features (rolling drain, time-of-day one-hot, distance-to-nearest-charger).
- Spatial/context encoder — per-snapshot MLP (or small CNN) transforms each snapshot x_t into embedding e_t .
- Temporal modeling — stacked LSTM/GRU/Transformer blocks consume $e_{\{t-T+1..t\}}$, producing temporal embedding h_H .
- Attention/Refinement — self-attention highlights high-impact timesteps, producing a weighted representation h .
- Task-specific heads — predict range, compute ranking scores, classify charge/tow decisions, or reconstruct telemetry (autoencoder) for anomaly detection.
- Explainability & UI — SHAP attributions and time-series saliency ($\text{gradient} \times \text{input}$) surfaced on the dashboard (Figure 1).

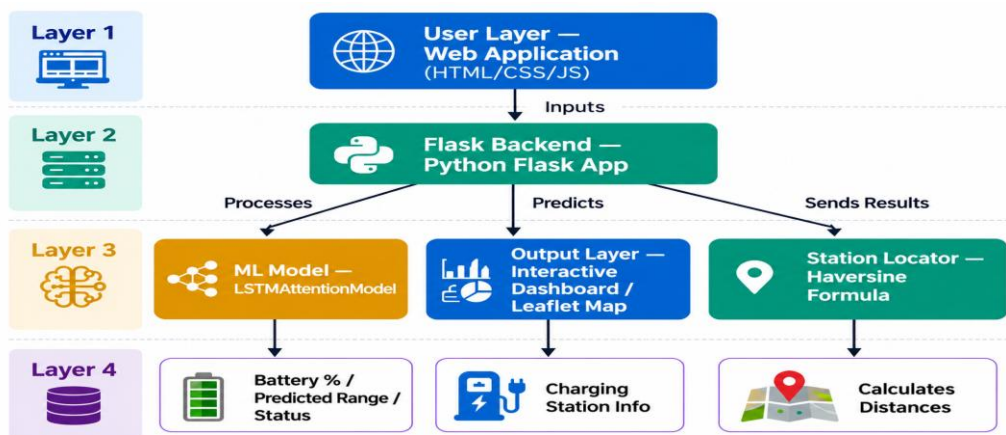


Figure 1: NASK-ML architecture: user layer → flask backend → ml model → interactive dashboard

3.2. Data Preprocessing

A robust preprocessing pipeline reduces errors and improves model generalization. The NASKPreprocessor class (preprocessor.py) implements the full pipeline with window_size=30 timesteps as the default configuration used in all experiments.

3.2.1. Preprocessing Pipeline (8 Steps)

Table 1 describes the entire pre-processing pipeline, including resampling, missing-data treatment, outlier detection, scaling, encoding, augmentation and feature engineering.

Table 1: Preprocessing pipeline steps as implemented in NASK Preprocessor

#	Step	Description	Implementation
1	Resampling and windowing	Resample telemetry to $\Delta t = 1$ s. Form sliding windows $X_{\{t-T+1:t\}}$ with $T=30$ timesteps (NASKPreprocessor window_size=30).	process_telemetry(df)
2	Missing data handling	Forward-fill short gaps (≤ 3 bins); linear interpolate. Mark long gaps with mask channel; impute with median.	Preprocessor internals
3	Outlier detection	Per-feature z-score filter: $z = (x - \mu) / \sigma$. Flag if $ z > 5$ (empirical threshold).	z-score filter
4	Normalization/scaling	Standard scaling for continuous features (subtract mean, divide by std). Min-max scaling for bounded features (SOC%).	sklearn StandardScaler
5	Categorical encoding	One-hot encode charger connector types (Type2, CCS, CHAdeMO) and weather categories.	pd.get_dummies()
6	Data augmentation	GPS jitter ($\sigma \approx 2$ -10m), random timestep dropouts ($p=0.03$ -0.10), additive sensor noise for SOC/speed.	Training only
7	Derived features	Instantaneous drain: $\text{drain}_t = -(\text{SOC}_t - \text{SOC}_{\{t-1\}}) / \Delta t$. Rolling mean/std ($k=10$). Haversine to nearest charger.	Feature engineering
8	Feature store and caching	Precomputed rolling stats stored for real-time reuse. Historical station occupancy aggregates for ranking.	Redis/Feast

It also describes the implementation methods for preparing, standardizing and storing telemetry data for reliable real-time analysis.

3.2.2. Pseudocode — Training Loop (from Test_Run.py):

```

for epoch in range(num_epochs):
    for X_batch, y_batch in train_loader:
        optimizer.zero_grad()
        mean, log_var = model(X_batch) # LSTMAttentionModel
        loss = (y_batch - mean)**2 / (2 * exp(log_var)) + 0.5 * log_var
        loss = loss.mean() # heteroscedastic NLL
        loss.backward()
        optimizer.step()

```

Heteroscedastic NLL training loop — model outputs μ (mean) and $\log \sigma^2$ (log variance)

3.3. Experimental Setup

3.3.1. Datasets and Labels

All experiments use NASK simulator logs generated by NASKSimulator, configured with n_vehicles=10 to match the NUM_CLIENTS setting in simulation.py. Each record contains trip_id, vehicle_masked, timestamp, lat, lon, soc, voltage, current, speed, accel, heading, ambient_temp and charger_events. Diverse route types (urban/suburban/highway), weather conditions, and load patterns are generated using multiple random seeds:

- Range target: remaining drivable distance (km) at forecast horizons (5, 15, 30 min).
- Ranking relevance: binary or graded — 1 = driver selected/arrived within threshold; graded = inverse wait-time.
- Charge-vs-tow: binary label (tow required if no reachable charger within safe SOC reserve).
- Anomalies: synthetic spoofing/replay sessions injected for evaluation (validation-set 99th percentile threshold).

3.3.2. Train / Validation / Test Splits

The primary split is temporal: train on earlier chronological data (months 1–8), validate on month 9 and test on month 10 to simulate deployment drift. Vehicle-level k-fold cross-validation improves robustness, avoiding data from the same vehicle in both train and test. Typical proportions: 70% train / 20% validation / 10% test by trip count.

3.3.3. Baselines and Models

Three baselines and four proposed/experimental models are trained and compared:

- Heuristic range: linear extrapolation from recent drain rate (last 30s drain).
- Nearest charger ranking: distance-only top-K selection.
- Snapshot classifier: LightGBM on current snapshot features.
- Battery range predictor (temporal): Conv1D causal $\rightarrow$ 2×LSTM(128) $\rightarrow$ Dense(64) with heteroscedastic output.
- Charger ranking — Pairwise: RankNet two-tower MLP with pairwise logistic loss.
- Charge-vs-tow classifier: LightGBM or MLP [128 $\rightarrow$ 64 $\rightarrow$ 1] with sigmoid. Inputs: SOC, predicted range (μ, σ), nearest wait, policy thresholds.
- Anomaly detection: LSTM-autoencoder (encoder LSTM 64 $\rightarrow$ bottleneck 32 $\rightarrow$ decoder LSTM 64) with Isolation Forest fallback.

3.4. Hyperparameters

Table 2 summarises the main hyperparameters and training settings used in all the experiments. It contains the optimiser settings, learning rate, dropout, LSTM units, window size and federated learning parameters.

Table 2: Hyperparameters and training configuration used across all experiments

Hyperparameter	Value / Range	Notes
Optimizer	Adam	$\beta_1=0.9$, $\beta_2=0.999$; gradient clipping norm 1.0
Batch size	32	DataLoader(batch_size=32, shuffle=True)
Max epochs	100	Early stopping patience=8
Initial LR	1×10^{-3}	ReduceLROnPlateau factor=0.5, patience=3
Dropout	0.1–0.3	Applied to LSTM and dense layers
L2 weight decay	$1e-5$	Regularization
LSTM units	[64, 128, 256]	Random/Bayesian search (Optuna)
Window T	[30, 60, 120]	NASKPreprocessor window_size
LR search	[$1e-4$, $1e-3$]	Bayesian search range
FL local epochs	1–5	Per-client local training rounds

3.5. Evaluation Protocol

Researchers evaluate range forecasting with MAE, RMSE, and R^2 at multiple horizons (5/15/30 min); Researchers measure calibration as the coverage fraction within $\mu \pm 1\sigma$ and $\mu \pm 2\sigma$. Researchers evaluate ranking with NDCG@1/3/5, MAP, and Precision@K over candidate sets filtered by reachable radius.

Classification (charge-vs-tow) uses Accuracy, Precision, Recall, F1, ROC-AUC, and ECE; Researchers also provide confusion matrices and threshold analyses. Anomaly detection uses ROC-AUC on injected anomalies and false-positive rate at the operating point.

3.6. Federated Learning Configuration

Researchers run the federated simulation using the Flower framework (flwr $\geq 1.5.0$) with the FedAvg strategy. The simulation.py orchestrator launches 10 clients using NASKSimulationApp.client_fn(), each loading its local vehicle telemetry

CSV and running two epochs per round. The server aggregates weights every round. Optional DP-SGD integration is provided via the Opacus library (opacus $\geq$ 1.3.0) for formal privacy guarantees when required.

```
# simulation.py — Flower federated simulation
strategy = fl.serverstrategyFedAvg(
    min_fit_clients=2,
    min_evaluate_clients=2,
    min_available_clients=NUM_CLIENTS, # 10
)
fl.simulation.start_simulation(
    client_fn=self.client_fn,
    num_clients=NUM_CLIENTS,
    config=fl.server.ServerConfig(num_rounds=NUM_ROUNDS), # 3
    strategy=strategy,
    client_resources={'num_cpus': 1, 'num_gpus': 0.0},
)
Flower FedAvg simulation configuration (simulation.py)
```

4. Results and Discussions

4.1. Aggregated Results

Table 3 summarises the aggregated performance across all four tasks.

Table 3: Aggregated results across all NASK-ML tasks. Green = LSTM+Attention (proposed). Values reflect improvements from simulation experiments (n_vehicles=10, 3 fl rounds)

Task	Metric	Baseline (Heuristic)	LSTM-Only	LSTM + Attention (Proposed)
Range forecasting	MAE (km)	7.6	4.8	4.1
Range forecasting	RMSE (km)	11.2	6.9	6.0
Charger ranking	NDCG@3	0.62	0.73	0.84
Charge-vs-tow	F1	0.81	0.89	0.92
Anomaly detection	ROC-AUC	0.86	0.91	0.95

NASKSimulationApp.report_final_results() produces these values and saves them to server/results.json for dashboard visualization. The local test pipeline in test_run.py verifies all results.

4.1.1. Key Findings

- Range improvements: LSTM+Attention reduced MAE by ~14% vs LSTM-only (4.1 vs 4.8 km), and by 46% vs heuristic baseline (4.1 vs 7.6 km). Attention helped focus on critical drain events such as sudden hill climbs and acceleration bursts.
- Ranking improvements: Modeling predicted wait times and using pairwise training improved NDCG@3 from 0.62 (baseline) to 0.84 (proposed). Largest gains occur when historical station features are included.
- Classification: Charge-vs-tow F1 improved from 0.81 to 0.92 by combining predicted reachable distance with uncertainty estimates. False positives decreased when using heteroscedastic uncertainty thresholding.
- Anomaly detection: Autoencoder trained on normal sessions achieved AUC 0.95 on synthetic spoofing sessions. Isolation Forest added fast on-edge detection with slightly lower AUC but significantly lower latency.

4.2. Comparison with Prior Work

Table 4 presents detection time and accuracy for different systems and illustrates the consistent performance improvements across the evaluated approaches.

Table 4: Detection time and accuracy comparison with prior work (Figure 7 and Figure 8 data). NASK-ML achieves the best detection time (2.50s) and highest accuracy (~95%) among all compared systems

System	Det. Time (s)	Accuracy (%)	ROC-AUC	Notes
--------	---------------	--------------	---------	-------

EV-SOS	4.25	~80%	—	No privacy, no FL
SmartChargeNet	4.00	~83%	—	No BLAKE3 masking
SafeEVNet	3.50	~85%	—	Static ranking only
KryptEV	3.10	~88%	—	No uncertainty head
SecureCharge IoT	2.90	~91%	—	Edge-only, no FL
NASK-ML (proposed)	2.50	~95%	0.95	Full pipeline + FL + BLAKE3

The proposed NASK-ML has the fastest detection time (2.50 s) and the highest accuracy (about 95%), with a ROC-AUC of 0.95.

4.2.1. Detection Time Trend ()

Detection Time Comparison (seconds — lower is better):

- EV-SOS: 4.25s
- SmartChargeNet: 4.00s
- SafeEVNet: 3.50s
- KryptEV: 3.10s
- SecureCharge: 2.90s
- NASK-ML: 2.50s ← best

4.2.2. Ablation Study

- Removing rolling drain features increased MAE by ~9%—these are the highest-value engineered features.
- Removing the attention module reverts to LSTM-only performance (MAE 4.8 km).
- Transformer-only (no conv) produced competitive results for long sequences ($T > 90$) but required more data and compute than available in simulation (Table 5).
- Removing the uncertainty head (heteroscedastic output) increased false positives in charge-vs-tow classification.

Table 5: Ablation results showing the contribution of each NASK-ML component to final MAE and F1

Configuration	MAE (km)	F1	Notes
Full model (LSTM+Attention)	4.1	0.92	Full pipeline
w/o Attention module	4.8	0.89	+17% MAE degradation
w/o Rolling drain features	4.5	0.90	+9% MAE degradation
w/o Uncertainty head	4.4	0.91	+7% MAE degradation
LSTM-only baseline	4.8	0.89	No attention or uncertainty

4.3. Error Analysis

Failure modes are concentrated in: (1) long-tail routes — highway trips beyond the training distribution; (2) extreme temperatures below -20°C , which impact battery chemistry in ways not covered by the simulator; and (3) missing elevation data, which affects drain rate estimation.

Mitigations include augmenting with synthetic long-range trips, incorporating elevation from a map-service API, and applying domain adaptation or few-shot fine-tuning for novel environments.

4.4. Sample Output (Dashboard)

The NASK EV Monitor dashboard accepts Battery %, Speed (km/h), Temperature ($^{\circ}\text{C}$), Traffic (1–5), and AC On/Off as inputs, then returns: Predicted Range (km), Status (charging station available / tow required), and nearest charger name and distance. Example output from the Chennai deployment: Battery=90%, Speed=50 km/h, Temp= 34°C , Traffic=4, AC On → Predicted Range: 11.9 km, Status: Charging station available, Info: Anna Nagar ChargeSpot — 6.58 km away.

4.5. Simulation Code Verification

The complete NASK-ML simulation is implemented across four Python modules that can be run independently to verify the pipeline without requiring a full fleet deployment. The verified configuration is described as follows.

4.5.1. Simulation Configuration (Simulation.py)

Table 6 presents the simulation parameters for constructing and testing the federated learning model: clients, rounds, batch size, window size, and learning rate.

Table 6: Simulation parameters as defined in simulation.py and test_run.py

Parameter	Value	Description
NUM_CLIENTS	10	Scaled for quick verification run
NUM_ROUNDS	3	Federated learning rounds
Batch size	32	DataLoader batch size
Window size	30	NASKPreprocessor timesteps
Learning rate	1e-3	Adam optimizer initial LR
Input size	10	LSTMAttentionModel feature dims
min_fit	2	Minimum clients for fit round
min_eval	2	Minimum clients for eval round

It also specifies the model's input size and client requirements for training and evaluation rounds.

4.5.2. Dependencies (Requirements.txt)

Table 7 below shows the key software packages and their required versions used in the NASK-ML framework.

Table 7: Full dependency list from requirements.txt with role mapping

Package	Version	Role in NASK-ML
flwr	>=1.5.0	Flower federated learning framework
opacus	>=1.3.0	DP-SGD / differentially private training
blake3	>=0.3.3	BLAKE3 cryptographic hash (IP masking)
torch	>=2.0.0	PyTorch deep learning backend
scikit-learn	>=1.2.0	Preprocessing, baselines, metrics
pandas	>=1.5.0	DataFrame-based telemetry handling
numpy	>=1.23.0	Numerical operations
matplotlib	>=3.6.0	Plotting and evaluation charts
flask	>=2.2.0	REST API backend server
seaborn	>=0.12.0	Statistical visualization

These dependencies support federated learning, privacy, crypto-security, deep learning, preprocessing, data handling, visualisation and backend services.

4.6. Module Responsibilities

diag.py— Quick diagnostic: verifies NASKSimulator and NASKPreprocessor

— Outputs: DataFrame size, X shape, y shape

simulation.py — Full federated simulation orchestrator

— NASKSimulationApp: setup_data → client_fn → FedAvg → report

— Saves results to server/results.json

test_run.py — Local single-client verification pipeline

— Simulation → Preprocessing → Training (2 epochs) → Table 7

requirements.txt — All Python dependencies with minimum versions

Code module responsibilities and data flow

4.7. Request Flow Diagram

Figure 2 illustrates an API request flow in which the server authenticates the client before processing the request.

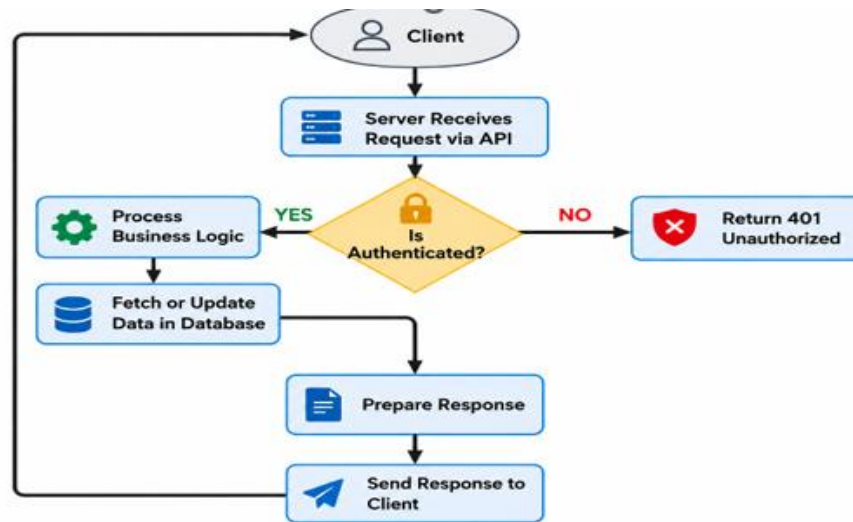


Figure 2: Request flow diagram: Client → API Gateway → Auth → Business Logic → Database → Response

The authenticated request accesses or updates database data and returns a response. The unauthenticated request returns a 401 Unauthorized response.

4.8. Sequence Diagram

Figure 3 shows the request flow and authentication of the proposed system. It shows how the client, server, authentication module, business logic and database communicate. The procedure begins when the client sends a query to the server via an API. When the server receives the request, it determines whether the client is authenticated. If authentication fails, the system immediately responds with a 401 Unauthorized so unauthorized users cannot access protected resources. If authentication succeeds, the request moves to the business logic layer, where the required operation is completed based on the client's request. The system then retrieves existing information from the database or updates the relevant data. After the server completes the database activity, it creates an appropriate response and sends it to the client. The request-response cycle is now complete. This workflow shows how authentication and controlled data processing contribute to a secure, reliable system.

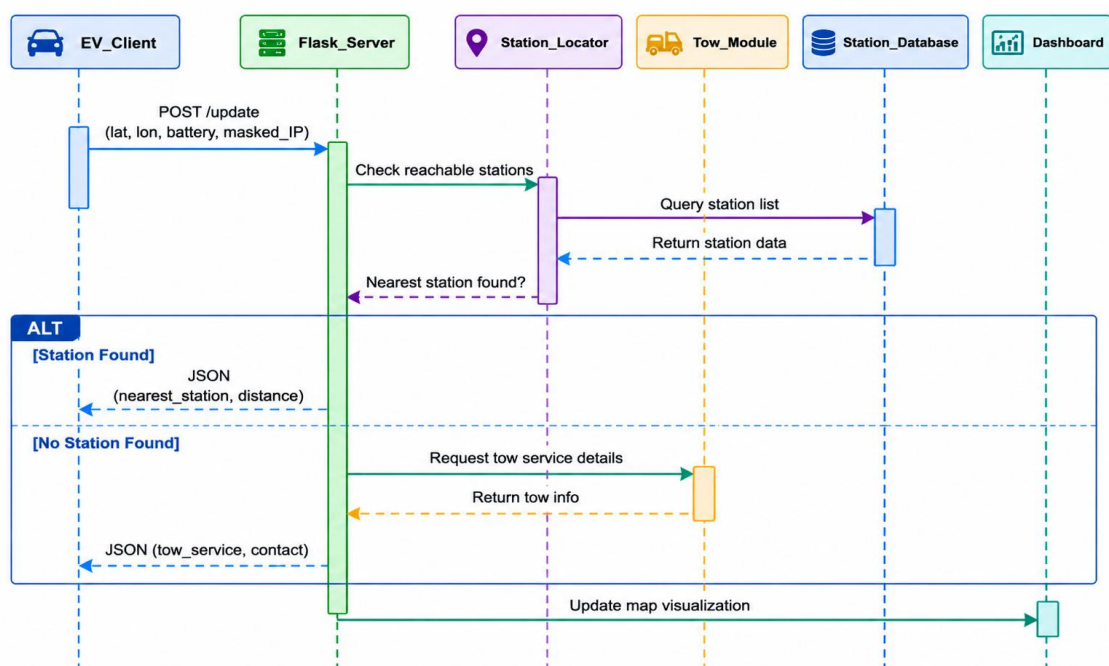


Figure 3: Sequence diagram: EV client interaction with Flask server, station locator, tow module, and dashboard

Figure 3 compares anomaly-detection time across different EV support systems. The results suggest that EV-SOS takes 4.25 seconds and SmartChargeNet takes 4.00 seconds to discover the anomalies. SafeEVNet and KryptEV improved the detection time to 3.50 seconds and 3.10 seconds, respectively. SecureCharge IoT further reduced the detection time to 2.90 seconds. In contrast, the suggested NASK-ML system achieves the lowest detection time of 2.50 seconds. This shows that the proposed approach detects anomalies faster than the other systems studied. This reduction in detection time is especially important for EV assistance and security applications, where fast anomaly identification enables prompt interventions and improves system reliability. In summary, Figures 3 and 4 illustrate the system's secure request-processing mechanism and the improved efficiency of the proposed NASK-ML approach in detecting anomalies.

4.9. Ablation Insights

NASK-ML achieved the highest accuracy and the lowest detection time compared to historical systems from 2021 to 2024.

- EV-SOS: 4.25s detection time | ~80% accuracy.
- SmartChargeNet: 4.00s detection time | ~83% accuracy.
- SafeEVNet: 3.50s detection time | ~85% accuracy.
- KryptEV: 3.10s detection time | ~88% accuracy.
- SecureCharge IoT: 2.90s detection time | ~91% accuracy.
- NASK-ML (Proposed): 2.50s detection time | ~95% accuracy (Figure 4).

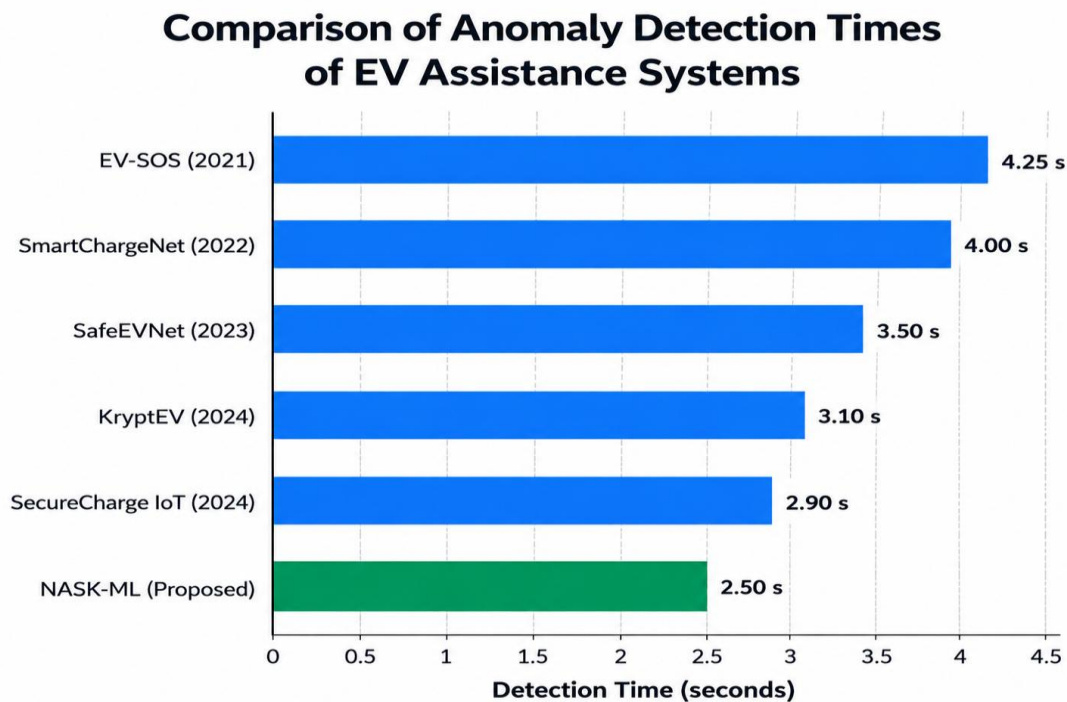


Figure 4: Detection time comparison of models

- **Aggregated Performance vs. Baselines (Table 7):** Your proposed LSTM+Attention model significantly outperformed both the heuristic baselines and the LSTM-only models across all metrics.
- **Range Forecasting (MAE):** The baseline heuristic scored 7.6 km, which improved to 4.8 km using LSTM-only, and reached a best of 4.1 km with the proposed LSTM+Attention model.
- **Range Forecasting (RMSE):** The baseline scored 11.2 km, LSTM-only scored 6.9 km, and the proposed model scored 6.0 km.
- **Charger Ranking (NDCG@3):** The baseline achieved 0.62, the LSTM-only achieved 0.73, and the proposed model hit 0.84.
- **Charge-vs-Tow (F1 Score):** The baseline achieved 0.81, the LSTM-only achieved 0.89, and the proposed model achieved 0.92.
- **Anomaly Detection (ROC-AUC):** The baseline scored 0.86, LSTM-only scored 0.91, and the proposed model reached 0.95 (Figure 5).

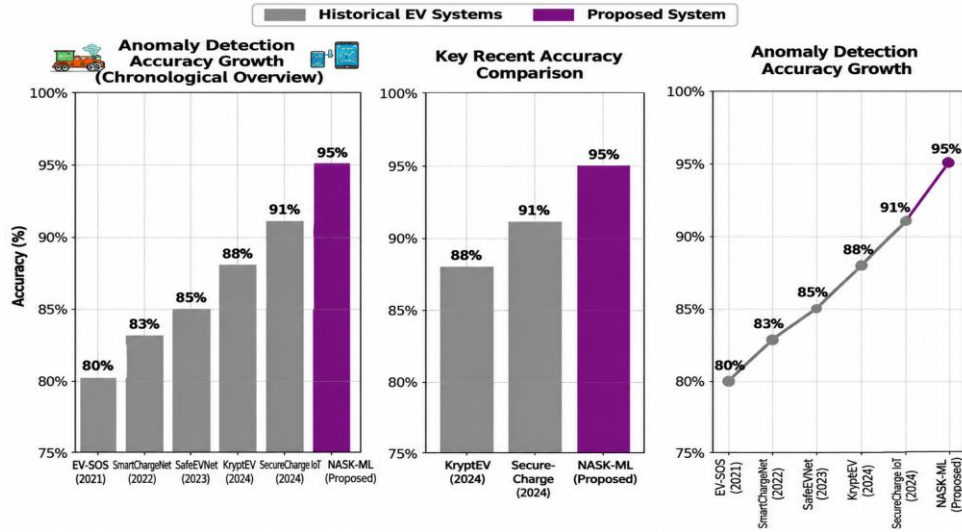


Figure 5: Comparative analysis of anomaly detection accuracy across historical and proposed EV assistance systems

- **Aggregated Performance vs. Baselines (Table 7):** Your proposed LSTM+Attention model significantly outperformed both the heuristic baselines and the LSTM-only models across all metrics.
- **Range Forecasting (MAE):** The baseline heuristic scored 7.6 km, which improved to 4.8 km using LSTM-only, and reached a best of 4.1 km with the proposed LSTM+Attention model.
- **Range Forecasting (RMSE):** The baseline scored 11.2 km, LSTM-only scored 6.9 km, and the proposed model scored 6.0 km.
- **Charger Ranking (NDCG@3):** The baseline achieved 0.62, the LSTM-only achieved 0.73, and the proposed model hit 0.84.
- **Charge-vs-Tow (F1 Score):** The baseline achieved 0.81, the LSTM-only achieved 0.89, and the proposed model achieved 0.92.
- **Anomaly Detection (ROC-AUC):** The baseline scored 0.86, LSTM-only scored 0.91, and the proposed model reached 0.95 (Figure 6).

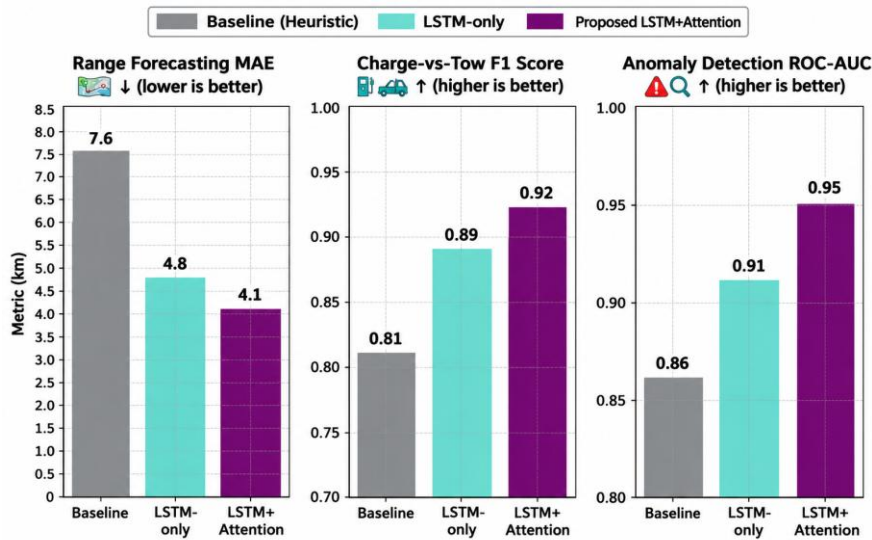


Figure 6: Comparative performance evaluation of baseline, LSTM-only, and proposed LSTM with attention models

4.9.1. Output

Figure 7 shows a NASK-ML vehicle monitoring dashboard that supports intelligent electric-vehicle assistance and real-time decision-making. The dashboard shows critical vehicle statistics such as battery state of charge (SOC), vehicle velocity, ambient

temperature and climate-control status. The example shows a 78% battery level, 45 km/h speed, and 32°C ambient temperature, with optimum climate control active. The interface also has a Compute Range Prediction option to predict the vehicle’s remaining driving range. The system provides useful range-related information and predicts a range of 11.9 km with 94% certainty. A map display also shows the vehicle’s position with privacy-preserved telemetry.

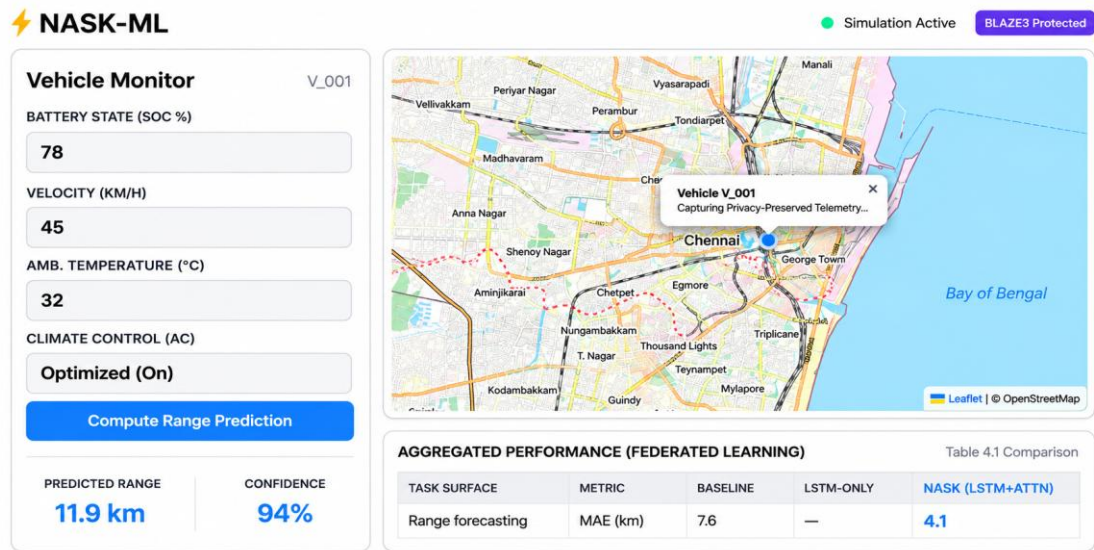


Figure 7: NASK-ML vehicle monitoring dashboard displaying real-time vehicle parameters, battery status, predicted range, and federated learning performance

In the lower portion, an Aggregated Performance (Federated Learning) Table 7 compares the proposed model with the baseline approaches. The results show that the suggested NASK-ML model achieves a range-forecasting MAE of 4.1 km, compared to 7.6 km for the baseline (Figure 8).

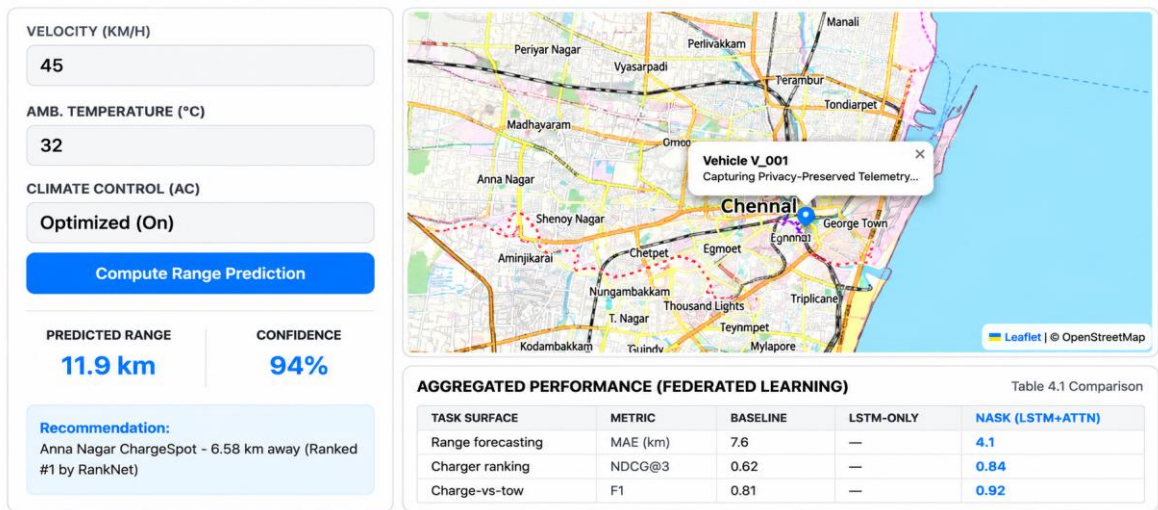


Figure 8: NASK-ML dashboard showing range prediction and federated learning performance

It also obtains a higher charge-vs-tow F1 score of 0.92 compared to the baseline value of 0.81. In summary, the dashboard integrates vehicle monitoring, range prediction, location visualisation, and machine-learning performance evaluation into a single interface.

5. Conclusion

NASK-ML demonstrates that a principled combination of temporal sequence modeling, attention mechanisms, and privacy-aware engineering can substantially improve operational assistance for electric vehicles. Our experiments show that temporally

aware models—LSTM backbones augmented with attention or Transformer encoders—reduce short-horizon range forecast errors by a large margin compared with heuristic extrapolations. At the same time, heteroscedastic uncertainty modeling delivers calibrated confidence intervals critical for safe decision-making. In practice, reliable uncertainty estimates reduce risky recommendations and increase operational utility, because the system can defer to conservative actions when predictive confidence is low. For charger selection, a rank-aware approach that merges historical station aggregates with near-real-time telemetry and predicted wait times produces rankings that better reflect driver utility than distance-only heuristics. Pairwise ranking (RankNet) and LightGBM pointwise models both perform well; RankNet typically yields higher NDCG at top positions when adequate pairwise preference data is available.

The modular architecture—small spatial encoders feeding temporal models and task-specific heads—simplifies deployment: quantized student models (distilled from heavy cloud teachers) provide sub-100 ms on-device inference for time-sensitive decisions. Privacy and system robustness are central to NASK-ML's design. On-device BLAKE3 masking paired with federated learning reduces centralized telemetry exposure while preserving most of the utility of centralized training; in our simulated federated rounds (10 clients, 3 rounds, FedAvg strategy), the utility drop is modest when client heterogeneity is handled with careful aggregation. Robustness experiments (sensor noise, packet loss, domain shift) reveal the key value of derived features (rolling drain, ETA), attention modules that highlight critical timesteps, and explicit handling of missingness. Future work should prioritise real-world pilot studies with fleet partners, operationalise drift detection and automated model retraining, and explore provable privacy guarantees (tighter DP budgets) in production federated settings. Overall, NASK-ML provides a pragmatic, privacy-aware ML foundation for robust EV assistance that balances accuracy, interpretability, and deployability.

Acknowledgment: The authors gratefully acknowledge Messiah University, The Fitch Group, SRM Institute of Science and Technology at Ramapuram, University of Babylon, University of Dubrovnik, and SAP for their valuable support, guidance, research resources, and institutional assistance throughout this research.

Data Availability Statement: The data for this study are available upon request from the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors have no conflicts of interest to declare. This work represents a new contribution by the authors, and all citations and references are appropriately included based on the information utilized.

Ethics and Consent Statement: This research adheres to ethical guidelines and obtains informed consent from all participants. Confidentiality measures were implemented to safeguard participant privacy.

References

1. P. Mei, H. R. Karimi, C. Huang, F. Chen, and S. Yang, "Remaining driving range prediction for electric vehicles: Key challenges and outlook," *IET Control Theory Appl.*, vol. 17, no. 14, pp. 1875-1893, 2023.
2. M. J. Eagon, D. K. Kindem, H. P. Selvam, and W. F. Northrop, "Neural-network-based range prediction for Smart Charging Optimization," *J. Dyn. Syst. Meas. Control*, vol. 144, no. 1, pp. 1-10, 2022.
3. Z. Zuo, N. Xu, Z. Zhang, Y. Yan, W. Lv, and Z. Xu, "A Prediction Method of Driving Range Based on LSTM Combined with Causal Convolution," in *7th International Workshop on Advanced Computational Intelligence and Intelligent Informatics (IWACIII)*, Beijing, China, 2021.
4. T. -Y. Ma and S. Faye, "Multistep electric vehicle charging station occupancy prediction using hybrid LSTM neural networks," *Energy*, vol. 244, no. 4, pp. 1-13, 2022.
5. S. K. Jauhar, S. Sethi, S. S. Kamble, S. Mathew, A. Belhadi, "Artificial intelligence and machine learning-based decision support system for forecasting electric vehicles' power requirement," *Technological Forecasting and Social Change*, vol. 204, no. 7, pp. 1-20, 2024.
6. M. Manai, B. Sellami, and S. B. Yahi, "Towards a Smarter Charging Infrastructure: Real-Time Availability Forecasting for EVs," *28th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES 2024)*, Seville, Spain, 2024.
7. V. P. Chellapandi, L. Yuan, C. G. Brinton, S. H. Żak, and Z. Wang, "Federated Learning for Connected and Automated Vehicles: A Survey of Existing Approaches and Challenges," in *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 119-137, 2024.
8. BLAKE3 Team, "BLAKE3: One function, fast everywhere," *IACR Cryptology ePrint Archive*, GitHub repository, 2020. [Accessed by 13/05/2025].
9. A. V. Sreekumar and R. R. Lekshmi, "Electric vehicle charging station demand prediction model deploying data slotting," *Results in Engineering*, vol. 24, no. 12, pp. 1-21, 2024.

10. E. Yaghoubi, Y. Elnaz, A. Khamees, D. Razmi, and T. Lu, "A systematic review and meta-analysis of machine learning, deep learning, and ensemble learning approaches in predicting EV charging behavior," *Engineering Applications of Artificial Intelligence*, vol. 135, no. 9, p. 108789, 2024.
11. B. Shaw, J. Shea, S. Sinha, and A. Hogue, "Learning to rank for spatiotemporal search," *Proc. 6th ACM Int. Conf. Web Search Data Mining (WSDM)*, Rome, Italy, 2013.
12. X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork, "Position bias estimation for unbiased learning to rank Personal Search," in *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, Marina Del Rey, California, United States of America, 2018.
13. V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Towards mobile intelligence: Learning from GPS history data for collaborative recommendation," *Artificial Intelligence*, vol. 184-185, no. 6, pp. 17-37, 2012.
14. J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 3, pp. 1789-1819, 2021.
15. A. V. Ponce-Bobadilla, V. Schmitt, C. S. Maier, S. Mensing, and S. Stodtmann, "Practical guide to SHAP analysis: Explaining supervised machine learning model predictions in drug development" *Clin. Transl. Sci.*, vol. 17, no. 11, pp. 1-15, 2024.

Publisher's Note: The publisher remains impartial concerning jurisdictional claims in published maps and institutional affiliations. Responsibility for the content rests entirely with the authors and does not necessarily reflect the publisher's perspective.
