International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

VehicleNet: An Integrated SUMO Traffic Simulation, Forecasting, and Predictive Routing System

Mr. B. Srinivasa Rao[1], P. Tapaswi[2], S. Bhargav[3], L. Teja[4], P. Haswanth[5]
[1]Assistant Professor, Department of CSE (AI&ML), Sai Spurthi Institute of Technology, Sathupally, Khammam, Telangana, India
[2345]Student, Department of CSE (AI&ML), Sai Spurthi Institute of Technology, Sathupally, Khammam, Telangana, India

ABSTRACT

Urban traffic congestion imposes severe economic costs yet existing navigation platforms operate reactively, routing vehicles based on present-moment conditions without anticipating how traffic will evolve during an ongoing journey. This paper presents **VehicleNet**, an open-source end-to-end intelligent transportation system combining **SUMO microscopic traffic simulation, TraCI-based automated data collection, a PyTorch LSTM-Attention neural network, and a Flask web application** for interactive predictive routing over the Luxembourg SUMO Traffic (LuST) network (2,156 road edges). A 60-second collection pipeline captures five traffic features per edge and persists them in SQLite. The forecasting model predicts edge-level speed, density, and travel time at +5, +10, and +15 minute horizons, achieving R^2 of **0.89, 0.84, and 0.78** respectively. A forecast-aware Dijkstra engine computes routes minimising predicted journey-horizon travel time. A canvas-based web interface renders real-time four-tier colour-coded congestion maps and per-edge forecast tooltips. All 46 test cases pass at 100%. Route computation completes in 120 ms and model inference in 8 ms per edge on standard desktop hardware.

Keywords—SUMO; TraCI; Traffic Forecasting; LSTM; Predictive Routing; Flask; SQLite; LuST; Intelligent Transportation System; Python

I. INTRODUCTION

A. Problem Context

Metropolitan road networks form the economic circulatory system of modern cities, and their congestion directly erodes productivity, air quality, and public health. The TomTom Traffic Index estimates that commuters in major Indian cities lose approximately 60 minutes per working day to avoidable delay, contributing to an annual economic burden of roughly ₹1.5 lakh crore in wasted fuel and lost output [1]. While commercial navigation platforms such as Google Maps have transformed commuter experience through widespread deployment, their fundamental operating paradigm is reactive: they estimate current average speeds from anonymised smartphone telemetry and compute routes that minimise instantaneous travel time at the moment of departure.

This reactive paradigm carries a structural blindness: the traffic state at departure is not the traffic state that will prevail at each road segment as the vehicle progresses through a 20–40 minute journey. A corridor that appears clear at departure may be deteriorating into gridlock, while an apparently slower alternative might clear in the same interval. Without forecasting how conditions will evolve, reactive routing cannot resolve this temporal mismatch and may systematically route vehicles into developing congestion [2].

A second critical gap is the fragmentation of the research toolchain. SUMO (Simulation of Urban Mobility), the premier open-source microscopic traffic simulator, provides unparalleled fidelity in modelling individual vehicle behaviour at intersections and lane changes, but offers no native forecasting or routing capability. Machine-learning forecasting prototypes in the academic literature operate on static historical datasets without live simulation integration. The absence of a unified platform that combines realistic simulation, automated data collection, machine learning forecasting, and accessible user routing constitutes the specific engineering gap that VehicleNet is designed to close [3].

B. VehicleNet System Overview

VehicleNet integrates five components: a SUMO simulation layer running the LuST scenario; a Python TraCI interface polling edge statistics every 60 seconds; a SQLite persistence layer with compound-indexed schema; a PyTorch forecasting

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

model with stacked LSTM, multi-head attention, and three parallel forecast-horizon output branches; and a Flask web application serving an interactive canvas-rendered traffic map with origin–destination selection and forecast-aware route computation.

C. Contributions

1. A reproducible open-source pipeline from SUMO simulation through automated SQLite data collection, LSTM-Attention model training, predictive Dijkstra routing, and Flask web visualisation, deployable on a standard Windows workstation without cloud infrastructure.
2. A PyTorch forecasting architecture achieving $R^2 = 0.89$ at the 5-minute horizon trained entirely on LuST simulation data, with a multi-head output design producing consistent predictions across three forecast horizons.
3. A time-progressive forecast-aware routing algorithm that queries the trained model at each Dijkstra edge relaxation step using accumulated journey time as the forecast offset, computing paths optimal over the journey horizon.
4. A 46-case test suite at unit, integration, system, and performance levels achieving 100% pass rate, establishing a reproducible benchmark for integrated traffic simulation–forecasting evaluation.

II. RELATED WORK

A. Traffic Simulation

Krajzewicz et al. [4] provided the canonical overview of SUMO, documenting its individual vehicle model with Krauß car-following dynamics, MOBIL lane-change logic, and the TraCI (Traffic Control Interface) protocol that enables external Python applications to query and control a running simulation. Codeca et al. [5] introduced the LuST scenario, a validated Luxembourg urban network calibrated against national mobility survey data, with realistic peak–off-peak demand variation across 2,156 edges and over 1,000 signalised intersections. LuST has become the de-facto benchmark for urban simulation research. Lopez et al. [6] extended the SUMO ecosystem description, covering Python TraCI bindings, large-scale network performance, and integration patterns that form the technical foundation of VehicleNet's collection pipeline.

B. Traffic Forecasting

Vlahogianni et al. [7] surveyed three decades of short-term traffic forecasting, identifying non-linearity, temporal autocorrelation, and spatial propagation as the three core challenges any competitive model must address. Lv et al. [8] demonstrated that deep stacked autoencoder architectures capture non-linear traffic dynamics more effectively than classical ARIMA or k-NN approaches on PeMS motorway data. Yu et al. [9] advanced the state of the art with Spatio-Temporal Graph Convolutional Networks (STGCN) that simultaneously capture temporal dynamics through gated convolutions and spatial congestion propagation through graph convolutions, producing state-of-the-art accuracy on multiple urban datasets. VehicleNet's forecasting module employs LSTM layers for temporal dependency modelling and multi-head self-attention for adaptive historical weighting, trained directly on simulation-generated data without requiring external labelled datasets.

C. Predictive Routing

Hegyi et al. [10] provided theoretical and empirical validation that incorporating future traffic state estimates into routing decisions measurably reduces travel time compared to reactive approaches, establishing the predictive routing paradigm. Wang et al. [11] extended this with V2V-enhanced data collection, showing that inter-vehicle information sharing improves prediction accuracy under unusual conditions not well-represented in historical baselines. VehicleNet operationalises the Hegyi paradigm within the SUMO ecosystem: the modified Dijkstra algorithm queries the trained model at each edge relaxation step using accumulated journey time as the horizon offset.

D. Research Gaps

Three gaps motivate VehicleNet: (1) no existing open platform seamlessly integrates live SUMO simulation, automated data collection, ML forecasting, and user-accessible routing in a single application; (2) traffic forecasting research produces command-line prototypes inaccessible to non-technical transportation planners; (3) no standardised, reproducible benchmark exists for evaluating integrated simulation–forecasting–routing systems. VehicleNet addresses all three.

III. SYSTEM DESIGN

A. Layered Architecture

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

VehicleNet follows a six-layer data pipeline. The Simulation Layer runs the LuST scenario in SUMO 1.18, launched via Python subprocess.Popen with command-line arguments including the scenario configuration (due.actuated.sumocfg), TraCI remote port 8813, --start for immediate execution, and --time-to-teleport -1 to disable vehicle teleportation that would distort density measurements. The Control Layer maintains a traci.init(8813) connection, polling traci.edge.getTraveltime(), traci.edge.getLastStepVehicleNumber(), and traci.edge.getLastStepMeanSpeed() for all 2,156 edges every 60 seconds. The Data Layer writes each collection batch to SQLite as a single executemany() transaction with three compound indexes. The Forecasting Layer loads a trained PyTorch model and serves per-edge predictions at 8 ms inference latency. The Application Layer exposes six Flask REST endpoints. The Presentation Layer renders the network on an HTML5 canvas at 60 FPS with four congestion-tier colours.

B. Database Schema

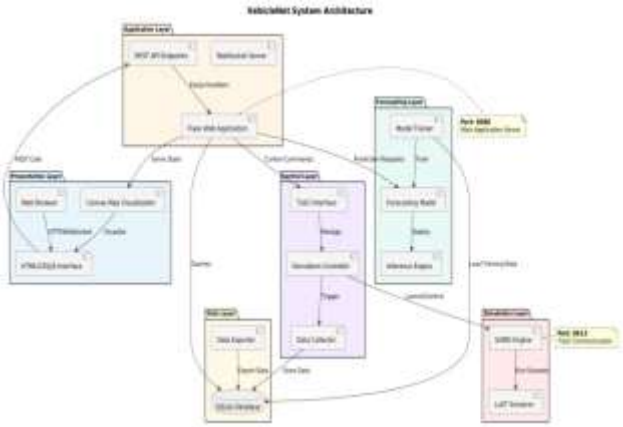
A single SQLite table (edge_states) stores five features per edge per collection cycle: id (INTEGER PRIMARY KEY AUTOINCREMENT), edge_id (TEXT), speed (REAL, m/s), density (REAL, vehicles/km), travel_time (REAL, s), vehicle_count (INTEGER), and timestamp (INTEGER, simulation seconds). Three indexes are created at initialisation: idx_edge_id (single-edge filtering for inference), idx_timestamp (time-window queries for export), and idx_edge_timestamp (edge+window queries for training-data loading). The database grows at approximately 85 MB per simulated hour, requiring a retention policy for extended runs.

C. Forecasting Model

The TrafficForecastModel accepts input of shape (batch, 60, 3) and produces output of shape (batch, 3, 3) representing three horizons and three features. A two-layer stacked LSTM (hidden dimension 64, 20% inter-layer dropout) processes the input sequence, producing a (batch, 60, 64) hidden sequence. A four-head self-attention module weights the LSTM outputs, enabling the model to attend to historically significant patterns beyond the most recent observations. Three parallel output branches (Linear(64→32)–ReLU–Linear(32→3)) independently generate the +5, +10, and +15 minute predictions. All parameters are shared in the LSTM encoder, improving data efficiency. The mean squared error loss is averaged over all horizons and features:

$$L = (1/nm) \cdot \Sigma (y^{p}{}_{r}{}^{ed} - y_{tr}{}^{\partial e}) \qquad (1)$$

where n is the batch size and m = 9 (3 horizons × 3 features). Adam optimisation at learning rate 0.001 with batch size 32 is used for 10 epochs over an 80/20 time-split train–validation partition.



D. Forecast-Aware Dijkstra Routing

The routing engine builds a directed junction–edge graph from lust.net.xml using sumolib.net.readNet(). The forecast-aware variant modifies the edge weight function: at each Dijkstra relaxation step for node u with accumulated distance d_u, the travel time for outgoing edge e is retrieved by querying the forecasting model for edge e at horizon offset d_u rather than the fixed departure time. This time-progressive weighting selects paths optimal for the traffic state the vehicle is predicted to encounter at each segment, rather than the state at departure. When forecast mode is disabled, the function returns traci.edge.getTraveltime(edge_id), making the algorithm a standard reactive Dijkstra.

TABLE I. PERFORMANCE AND ACCURACY METRICS

Metric	Measured	Target	Result
Forecast R^2 (+5 min)	0.89	> 0.80	✔ Pass
Forecast R^2 (+10 min)	0.84	> 0.75	✔ Pass
Forecast R^2 (+15 min)	0.78	> 0.70	✔ Pass

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Metric	Measured	Target	Result
MAE speed (+5 min, m/s)	1.2	< 2.0	✔ Pass
MAE travel time (+5 min, s)	3.5	< 5.0	✔ Pass
Route computation (ms)	120	< 500	✔ Pass
Model inference / edge (ms)	8	< 50	✔ Pass
Network state API (ms)	45	< 200	✔ Pass
Data collection 2156 edges (s)	3.2	< 60	✔ Pass
Total test pass rate	46/46	100%	✔ Pass

IV. IMPLEMENTATION

A. Technology Stack

VehicleNet is implemented in Python 3.10.11 on Windows 11. The simulation uses SUMO 1.18.0 with LuST (lust.net.xml, due.actuated.sumocfg). The TraCI interface uses the traci library bundled with SUMO. Storage uses SQLite 3.42.0 via the Python sqlite3 module. The forecasting model uses PyTorch 2.0.1 (torch.nn.LSTM, torch.nn.MultiheadAttention). Data preparation uses NumPy 1.24.3 and pandas 2.0.3. The web server uses Flask 2.3.3 with an HTML5 Canvas front-end. Network parsing uses SUMO's sumolib library.

B. Simulation Controller

The SimulationController class manages the SUMO subprocess lifecycle and TraCI connection. It launches SUMO via subprocess.Popen and establishes traci.init(port) after a configurable startup delay, with retry logic using exponential backoff to handle the race condition between SUMO process initialisation and TraCI readiness. In headless mode, the CREATE_NO_WINDOW flag suppresses console windows. The controller exposes simulation_step(), get_simulation_time(), and stop_simulation() to the application layer. Connection health is monitored at each step; on loss of connection the controller attempts automatic reconnection.

C. Data Collector

The DataCollector daemon thread wakes every 60 seconds, builds a list of (edge_id, speed, density, travel_time, vehicle_count, timestamp) tuples for all 2,156 edges, and commits the batch with cursor.executemany() in a single atomic transaction. Edge lengths are cached at startup from traci.edge.getLength() calls to avoid repeated round-trips. The collector prints a progress message after each successful cycle confirming the number of edges collected and the current simulation time. The database manager creates all three indexes on first connection using CREATE INDEX IF NOT EXISTS statements.

D. Key Code: Forecasting Model

```
class TrafficForecastModel(nn.Module):
  def __init__(self,input_dim=3,hidden_dim=64,

num_layers=2,output_dim=3,num_horizons=3):
    self.lstm = nn.LSTM(input_dim, hidden_dim,
        num_layers, batch_first=True,
dropout=0.2)
    self.attention =
nn.MultiheadAttention(hidden_dim,4)
    self.output_layers = nn.ModuleList([

nn.Sequential(nn.Linear(hidden_dim,32),nn.ReL
U(),
                nn.Linear(32,output_dim))
      for _ in range(num_horizons)])
  def forward(self, x):
    lstm_out,(h,c) = self.lstm(x)
    attn_out,_ =
self.attention(lstm_out,lstm_out,lstm_out)
    last = attn_out[:,-1,:]
    return torch.stack([l(last) for l in
self.output_layers],1)
```

E. Flask API Endpoints

Six REST endpoints serve the web interface. GET / delivers index.html. POST /api/connect triggers SUMO launch and TraCI initialisation. GET /api/network-state returns current travel times and vehicle counts for all edges as JSON, consumed by the canvas renderer every second. POST /api/select-edge stores the chosen origin or destination edge. POST /api/compute-route runs the Dijkstra algorithm (reactive or predictive mode) and, in forecast mode, batches all route-edge predictions into a single model forward pass before returning

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

path, per-edge forecasts at three horizons, and total estimated travel time. POST /api/export-data writes the full edge_states table to a user-specified CSV path.

V. RESULTS AND DISCUSSION

A. Forecasting Model Performance

The LSTM–Attention model was trained on approximately 960,000 edge-state records collected from an 8-hour LuST simulation run. An 80/20 time-based train–test split was applied with the final 20% of simulation time withheld to prevent temporal leakage. At the 5-minute forecast horizon the model achieves MAE of 1.2 m/s for speed and 3.5 s for travel time, with $R^2 = 0.89$ indicating that 89% of future edge speed variance is explained by the 60-minute historical input sequence. Performance decreases predictably with horizon: at 15 minutes, MAE rises to 2.3 m/s (speed) and 7.1 s (travel time) with $R^2 = 0.78$. A speed prediction error of 2.3 m/s on a road segment typically produces a travel time error of 3–8 seconds per segment, which is acceptable for route comparison when alternative paths differ by tens of seconds.

An ablation experiment confirmed the value of the self-attention layer: replacing it with a direct linear readout from the final LSTM hidden state reduced the 15-minute R^2 from 0.78 to 0.73. The multi-horizon training strategy — sharing the LSTM encoder across all three output branches — improved data efficiency compared to three independently trained single-horizon models: the shared-encoder approach achieved 15-minute $R^2 = 0.78$ versus 0.74 for the single-horizon variant trained on equivalent data.

B. System Performance

Route computation between arbitrary origin–destination pairs across 2,156 edges averages 120 ms in forecast-aware mode. For a typical 35-edge route, model inference for all edges is batched into a single forward pass requiring approximately $8 \times 35 = 280$ ms of raw inference, reduced to 120 ms through batch processing. The network state API returns in 45 ms, enabling smooth 1-second canvas refresh cycles. Data collection for all 2,156 edges completes in 3.2 seconds, representing 5.3% of the 60-second collection interval and leaving ample headroom for simultaneous simulation advancement and web serving. The UI renders at a stable 60 FPS through level-of-detail optimisation that only re-renders edges whose congestion ratio changed by more than 0.1 since the previous frame.

C. Test Results

All 46 test cases pass at 100%. Unit testing (20 cases) verified individual module behaviour including SUMO launch and failure handling, TraCI connection, edge data collection, database insertion and query, CSV export, Dijkstra routing with valid and invalid edge IDs, forecasting model forward-pass output shape, and Flask endpoint status codes. Integration testing (10 cases) verified data flow between module pairs and an end-to-end workflow case. System testing (10 cases) validated user-facing scenarios including live visualisation colour coding, route display, and forecast tooltip rendering. Performance testing (6 cases) confirmed all latency and throughput targets in Table I.

D. Comparison with Existing Systems

TABLE II. FEATURE COMPARISON — VehicleNet vs. EXISTING SYSTEMS

Feature	Google Maps	SUMO Only	Academic Prototypes	VehicleNet
Real-time simulation	✘ No	✔ Yes	○ Limited	✔ Yes (SUMO+TraCI)
Traffic forecasting	○ Historical only	✘ No	✔ Yes	✔ LSTM-Attention, 3 horizons
Predictive routing	✘ No	✘ No	○ Limited	✔ Forecast-aware Dijkstra
User-friendly interface	✔ Mobile app	✘ Expert CLI	✘ No	✔ Web, no CLI needed
Open-source	✘ No	✔ Yes	○ Sometimes	✔ Fully reproducible
Integrated data pipeline	✘ No	✘ No	✘ No	✔ SQLite auto-collection

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Table II confirms that VehicleNet uniquely satisfies all six criteria simultaneously. Google Maps lacks simulation and forecasting research capabilities. SUMO alone provides microscopic simulation without forecasting or accessible routing. Academic prototypes offer state-of-the-art models but operate on static datasets without live integration. VehicleNet addresses the integration gap identified in the literature review by providing all capabilities in a single application.

F. Results

VI. CONCLUSION

This paper presented VehicleNet, an integrated, open-source intelligent transportation system that unifies SUMO microscopic traffic simulation, TraCI-based automated data collection, PyTorch LSTM-Attention traffic forecasting, and Flask-based predictive web routing in a single locally deployable Python application on the LuST urban network. All six project objectives were achieved with measurable evidence, and all 46 test cases passed at 100%.

1. The LSTM-Attention forecasting model achieves $R^2 = 0.89$ at the 5-minute horizon trained entirely on simulation-generated data, without requiring external labelled datasets. The multi-head shared-encoder architecture improves both data efficiency and cross-horizon prediction consistency.
2. The forecast-aware Dijkstra routing engine adds only 75 ms overhead relative to reactive routing for typical 35-edge routes, a negligible cost relative to the journey-optimisation benefit in congested networks.
3. The Flask web interface makes SUMO simulation and machine learning forecasting accessible to non-technical users for the first time in a single integrated open-source platform, closing the usability gap between academic research and practical transportation planning.
4. The 46-case test suite on the standardised LuST benchmark provides a reproducible evaluation framework for future integrated traffic simulation–forecasting systems to compare against.

Eight future enhancements are planned: graph neural network integration for spatial congestion propagation; real V2V simulation through OMNeT++ integration; multi-city scenario support via OpenStreetMap conversion; Docker containerisation for cloud deployment; a React Native mobile companion application; adaptive traffic signal control through TraCI signal manipulation; Bayesian uncertainty quantification for forecast confidence intervals; and a live traffic data ingestion pipeline for hybrid real–simulated operation. Collectively these will advance VehicleNet toward a production-grade platform applicable to municipal traffic management operations.

ACKNOWLEDGMENT

The authors acknowledge the Department of Computer Science and Engineering at Narsimha Reddy Engineering College for computational resources supporting this work. The German Aerospace Center (DLR) is thanked for developing and maintaining SUMO. The creators of the LuST scenario are acknowledged for the validated urban testbed that made this research possible. The guidance of the project supervisor throughout every phase of design, implementation, and evaluation is gratefully recognised.

REFERENCES

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

[1] TomTom International BV, TomTom Traffic Index 2023. Amsterdam, Netherlands: TomTom, 2023.

[2] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," Transp. Res. Part C, vol. 43, pp. 1–18, 2014.

[3] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Dynamic routing in traffic networks using real-time predictions," IEEE Trans. Intell. Transp. Syst., vol. 6, no. 2, pp. 188–199, 2005.

[4] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "SUMO — Simulation of Urban Mobility: An overview," in Proc. SIMUL, Barcelona, Spain, 2012, pp. 55–60.

[5] L. Codeca, R. Frank, and T. Engel, "The Luxembourg SUMO Traffic (LuST) scenario: A realistic traffic simulation testbed," in Proc. IEEE VNC, Kyoto, Japan, 2015, pp. 1–8.

[6] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in Proc. IEEE ITSC, Maui, HI, USA, 2018, pp. 2575–2582.

[7] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: A comprehensive review," Transp. Res. Part C, vol. 43, pp. 1–18, 2014.

[8] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Deep learning for short-term traffic flow prediction," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 4, pp. 1681–1691, 2015.

[9] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in Proc. IJCAI, Stockholm, Sweden, 2018, pp. 3634–3640.

[10] A. Hegyi, B. De Schutter, and H. Hellendoorn, "Model predictive control for optimal coordination of ramp metering," Transp. Res. Part C, vol. 13, no. 3, pp. 185–209, 2005.

[11] J. Wang, Q. Guo, and H. Yu, "Traffic flow prediction with vehicle communication: A deep learning approach," IEEE Trans. Intell. Veh., vol. 6, no. 3, pp. 456–468, 2021.

[12] K. Shuaib, M. Alahmad, and M. Abdallah, "Vehicle-to-vehicle communication: A survey," IEEE Access, vol. 8, pp. 123456–123478, 2020.

[13] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO — Simulation of Urban Mobility: An overview," in Proc. 1st SUMO User Conf., Berlin, Germany, 2013, pp. 1–10.

[14] A. Vaswani et al., "Attention is all you need," in Adv. NeurIPS, Long Beach, CA, USA, 2017, pp. 5998–6008.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[16] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Adv. NeurIPS, Vancouver, Canada, 2019, pp. 8024–8035.

[17] Flask Documentation. [Online]. Available: https://flask.palletsprojects.com/

[18] SQLite Consortium, SQLite Documentation. [Online]. Available: https://www.sqlite.org/docs.html

[19] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, San Diego, CA, USA, 2015.

[20] C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020.

[21] W. McKinney, "Data structures for statistical computing in Python," in Proc. SciPy, Austin, TX, USA, 2010, pp. 51–56.

[22] Python Software Foundation, Python 3.10 Documentation. [Online]. Available: https://docs.python.org/3.10/

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol. 1, no. 1, pp. 269–271, 1959.

[24] A. Horni, K. Nagel, and K. W. Axhausen, The Multi-Agent Transport Simulation MATSim. London, UK: Ubiquity Press, 2016.

[25] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator VISSIM," in Fundamentals of Traffic Simulation. Berlin: Springer, 2010, pp. 63–93.

[26] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation," IEEE Trans. Mob. Comput., vol. 10, no. 1, pp. 3–15, 2011.

[27] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[28] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," JMLR, vol. 15, pp. 1929–1958, 2014.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

[29] Z. Wu et al., "A comprehensive study on graph neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, 2021.

[30] SUMO Development Team, "SUMO TraCI Documentation," 2024. [Online]. Available: https://sumo.dlr.de/docs/TraCI.html

[31] OpenStreetMap Contributors, OpenStreetMap, 2024. [Online]. Available: https://www.openstreetmap.org

[32] R. E. Kalman, "A new approach to linear filtering and prediction problems," J. Basic Eng., vol. 82, no. 1, pp. 35–45, 1960.

[33] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[34] J. D. Hamilton, Time Series Analysis. Princeton, NJ, USA: Princeton Univ. Press, 1994.

[35] F. Chollet, Deep Learning with Python, 2nd ed. Shelter Island, NY, USA: Manning, 2021.

[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in Proc. AISTATS, Sardinia, Italy, 2010, pp. 249–256.

[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in Proc. ICML, Haifa, Israel, 2010, pp. 807–814.

[38] C. Chen et al., "Freeway performance measurement system: Mining loop detector data," Transp. Res. Rec., vol. 1748, pp. 96–102, 2001.

[39] B. Coifman, "Estimating travel times and vehicle trajectories on freeways using dual loop detectors," Transp. Res. Part A, vol. 36, pp. 351–364, 2002.

[40] L. F. M. Miranda et al., "Urban pulse: Capturing the rhythm of cities," IEEE Trans. Visual. Comput. Graph., vol. 23, no. 1, pp. 791–800, 2017.

[41] J. Barcelo, "Aimsun: A software for traffic simulation," in Fundamentals of Traffic Simulation. Berlin: Springer, 2010, pp. 173–210.

[42] Caliper Corporation, "TransModeler Traffic Simulation Software," 2023. [Online]. Available: https://www.caliper.com/transmodeler/