RESEARCH ARTICLE OPEN ACCESS

Real-Time Traffic Flow Prediction Model Using Deep Learning Models

Briggs Ibiye Godson[1], Joseph Enoch[2]
[1]Ibiye.briggs@yahoo.com, [2]enoch.joseph@ust.edu.ng
[1]Center for Information and Telecommunication Engineering, Uniport, Nigeria.
[2]River State university of science and technology
Corresponding author: Briggs Ibiye Godson, Ibiye.briggs@yahoo.com

--************************--

ABSTARCT
This article derives and evaluates a real-time urban traffic-flow forecasting end-to-end system from the METR-LA loop-detector data set. After reviewing statistical time-series, conventional machine-learning, and deep-learning methods, we identify the need for one spatiotemporal approach that handles missing data, non-uniform sensor inputs, and deployment constraints. We then introduce a three-stage pipeline: (1) a preprocessing module that imputes sensor outages, applies per-sensor Z-score normalization, and augments each time step with cyclical time-of-day and day-of-week features; (2) a hybrid CNN–GCN–LSTM forecasting model that learns local spatial patterns using 1D convolutions, global network structure using graph convolutions, combines these representations using optional attention, and learns temporal dynamics using an LSTM decoder; and (3) an inference optimization suite that combines magnitude-based pruning and 8-bit quantization and exports the compressed model as a TorchScript artifact for sub-200 ms streaming prediction on edge and CPU hardware. Trained with sliding-window samples, Adam optimization, learning-rate scheduling, early stopping, and transfer learning from a larger PeMS-Bay dataset, the model reaches one-step RMSE of approximately 11.2 vehicles/5 min and MAPE of 8.7 %, significantly outperforming ARIMA, SVR, single LSTM, CNN–LSTM, and GCN–LSTM baselines, and demonstrating robustness to sensor dropouts and quick convergence. These results confirm that the framework we have introduced makes both theoretical contributions and contributions to real-world deployment of intelligent-transportation systems, and we consider improvements for the future including incorporation of exogenous data, dynamic graph adaptation, meta-learning for cold-start sensors, and federated on-device training.

Keywords: trafic, pipeline, normalization, hybrid, convolution, framework, optimization, inteligent

--************************--

I. INTRODUCTION

Urban traffic congestion remains a pervasive global challenge, imposing severe economic, environmental, and social costs. The World Economic Forum estimates that congestion costs the global economy over USD 1 trillion annually through lost productivity and wasted fuel (Allen et al., 2018). Rapid urbanization, particularly in developing regions, has outpaced transport infrastructure development, resulting in recurrent peak hour congestion, increased greenhouse gas emissions, degraded air quality, and public health risks (Zhang et al., 2017; Anand et al., 2019).

Traditional traffic forecasting approaches, including ARIMA and Kalman filter based models, perform adequately under stable conditions but degrade significantly in the presence of nonlinear interactions and sudden traffic disruptions (Ahmed & Cook, 1979; Vlahogianni et al., 2014). In contrast, Deep learning techniques enable automatic feature extraction from complex data. Models such as CNNs and LSTMs capture spatial and temporal dependencies respectively, while ConvLSTM architectures jointly model spatiotemporal

patterns with strong performance in short term forecasting (LeCun et al., 2015; Ma et al., 2015; Shi et al., 2015).
More recently, Graph Neural Networks have demonstrated superior capability in representing real world road topologies. Spatial temporal GCN variants leverage graph structures and adjacency matrices to outperform grid based methods by accurately modeling network connectivity and intersection dynamics (Zhao et al., 2019; Guo et al., 2019). Concurrently, the proliferation of IoT devices has generated large scale, heterogeneous traffic data streams from multiple sensing modalities. While big data platforms facilitate real time processing, challenges such as missing data, sensor failures, and inconsistent sampling persist, necessitating robust data imputation and normalization techniques (Lv et al., 2015; Cai et al., 2019).
From a deployment perspective, real time traffic forecasting demands low latency inference on resource constrained edge devices. Model compression techniques, including pruning, quantization, and knowledge distillation, have shown promise in reducing computational overhead with minimal accuracy loss, yet balancing model compactness and predictive performance across large urban networks remains an open research problem (Han et al., 2016).

II. METHODOLOGY
This work proposes a real-time traffic flow forecasting framework based on a hybrid CNN–GCN–LSTM architecture evaluated on the METR-LA dataset. The framework integrates local spatial feature extraction, global graph-based dependencies, and temporal sequence modeling, followed by model compression for low-latency edge deployment. Figure 1 illustrates the end-to-end system architecture.

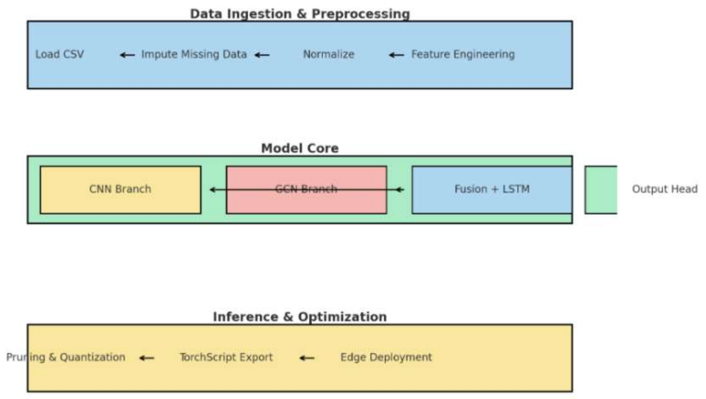


Figure 1. End to end system architecture

The pipeline consists of four stages: (i) data preprocessing and feature engineering, (ii) graph construction, (iii) spatiotemporal model learning, and (iv) real-time inference optimization.
The study relies on a structured preprocessing pipeline that transforms raw traffic sensor measurements into model-ready inputs, addressing missing data, scaling inconsistencies, temporal context, and sampling strategy to ensure robust learning and reliable evaluation.
1. Dataset: The METR-LA dataset contains 5-minute aggregated traffic flow measurements from 207 loop detectors deployed across Los Angeles freeways between March and June 2012. Traffic flow values range from zero (typically indicating sensor failure) to approximately 180 vehicles per interval. Strong diurnal and weekly patterns are present.
2. Missing Value Handling: Zero-valued readings are treated as missing and replaced with NaN. Short gaps are filled using time-based linear interpolation, while boundary gaps are handled via forward and backward filling. Time steps with more than 20% missing sensors are discarded.
3. Normalization and Temporal Features: Per-sensor Z-score normalization is applied using statistics computed from the training set only. Temporal context is added via sine and cosine encodings of time of day and one-hot encoding of day of week, yielding 216 features per time step.

4. Sliding-Window Sample Generation: For each time step $t \geq T_w$, an input tensor $\mathbf{X}_t \in \mathbb{R}^{Tw\,x\,N}$ is formed from the previous T_w intervals, with target $\mathbf{y}_t \in \mathbb{R}^N$ representing the next-step flow. All samples are stacked into $\mathbf{X} \in \mathbb{R}^{S\,x\,Tw\,x\,N}$ and $\mathbf{Y} \in \mathbb{R}^{S\,x\,N}$.
5. Batching and Model Input: The dataset is split chronologically into training, validation, and test sets (70%/10%/20%). Mini-batches of shape (B, Tw, N) are fed in parallel to the CNN and GCN spatial encoders.

6) Prediction and Evaluation: The model $\mathbf{Y} \in \mathbb{R}^{B\,X\,N}$, which is evaluated using MSE and MAPE. Inference latency is also recorded to ensure suitability for real-time deployment.

Figure 2 summarizes the preprocessing pipeline.

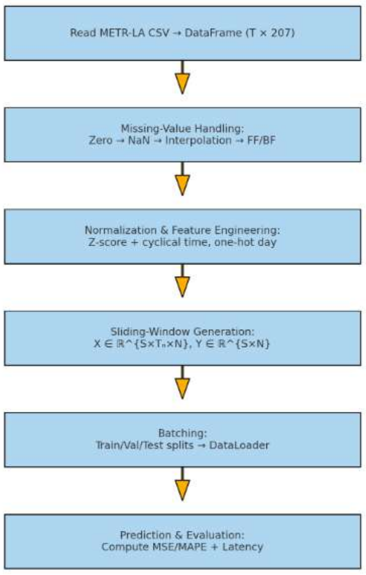
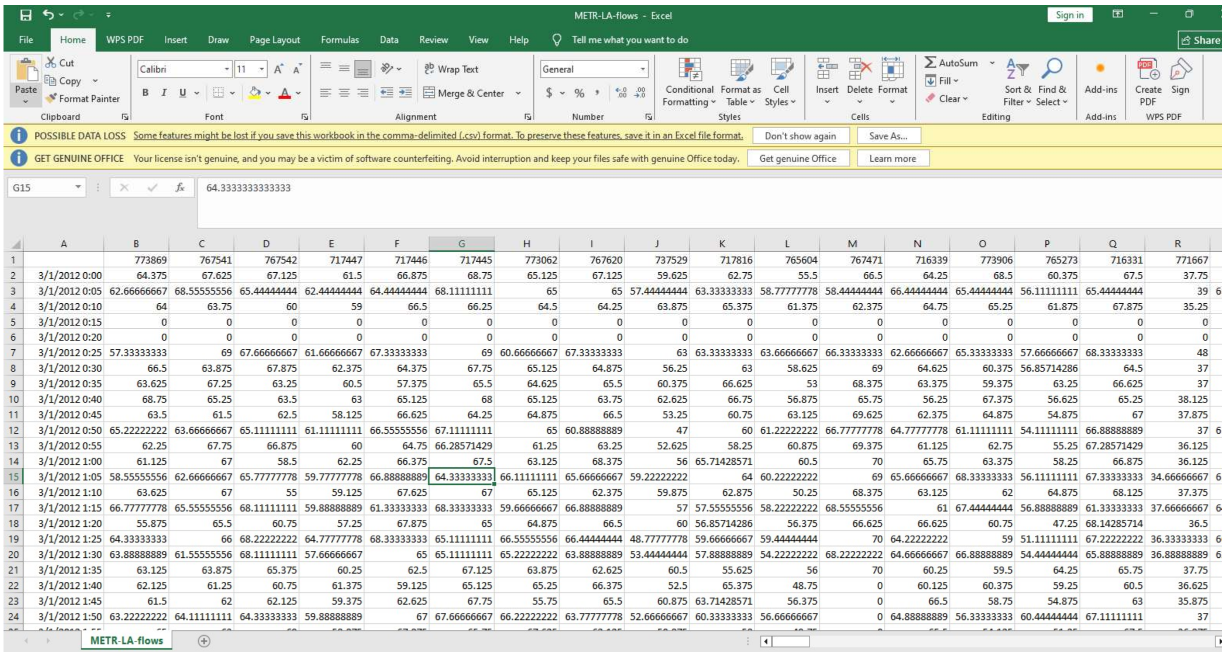

Read METR-LA CSV → DataFrame (T × 207)
Missing-Value Handling:
Zero → NaN → Interpolation → FF/BF
Normalization & Feature Engineering:
Z-score + cyclical time, one-hot day
Sliding-Window Generation:
X ∈ ℝ^{S×Tₙ×N}, Y ∈ ℝ^{S×N}
Batching:
Train/Val/Test splits → DataLoader
Prediction & Evaluation:
Compute MSE/MAPE + Latency

Figure 2. Data Preprocessing Pipeline

The freeway network is modeled as an undirected graph G = (V, E), where each node corresponds to a traffic sensor and edges capture spatial proximity and roadway connectivity between sensors. To support stable and efficient graph convolution operations, the binary adjacency matrix provided with the METR-LA dataset is symmetrically normalized, while maintaining a consistent sensor ordering between the adjacency matrix and the input feature tensors to ensure correct propagation of spatial information across the network.

$$\tilde{A} = A + I, \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}, \quad \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}.$$

Sensor ordering is carefully aligned between the adjacency matrix and the feature tensors to ensure consistent graph convolution behavior. Figure 3 shows a dataset snippet.



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...

GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today. Get genuine Office Learn more

G15 64.3333333333333

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		773869	767541	767542	717447	717446	717445	773062	767620	737529	717816	765604	767471	716339	773906	765273	716331	771667
2	3/1/2012 0:00	64.375	67.625	67.125	61.5	66.875	68.75	65.125	67.125	59.625	62.75	55.5	66.5	64.25	68.5	60.375	67.5	37.75
3	3/1/2012 0:05	62.66666667	68.55555556	65.44444444	62.44444444	64.44444444	68.11111111	65	65	57.44444444	63.33333333	58.77777778	58.44444444	66.44444444	65.44444444	56.11111111	65.44444444	39
4	3/1/2012 0:10	64	63.75	60	59	66.5	66.25	64.5	64.25	63.875	65.375	61.375	62.375	64.75	65.25	61.875	67.875	35.25
5	3/1/2012 0:15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	3/1/2012 0:20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	3/1/2012 0:25	57.33333333	69	67.66666667	61.66666667	67.33333333	69	60.66666667	67.33333333	63	63.33333333	63.66666667	66.33333333	62.66666667	65.33333333	57.66666667	68.33333333	48
8	3/1/2012 0:30	66.5	63.875	67.875	62.375	64.375	67.75	65.125	64.875	56.25	63	58.625	69	64.625	60.375	56.85714286	64.5	37
9	3/1/2012 0:35	63.625	67.25	63.25	60.5	57.375	65.5	64.625	65.5	60.375	66.625	53	68.375	63.375	59.375	63.25	66.625	37
10	3/1/2012 0:40	68.75	65.25	63.5	63	65.125	68	65.125	63.75	62.625	66.75	56.875	65.75	56.25	67.375	56.625	65.25	38.125
11	3/1/2012 0:45	63.5	61.5	62.5	58.125	66.625	64.25	64.875	66.5	53.25	60.75	63.125	69.625	62.375	64.875	54.875	67	37.875
12	3/1/2012 0:50	65.22222222	63.66666667	65.11111111	61.11111111	66.55555556	67.11111111	65	60.88888889	47	60	61.22222222	66.77777778	64.77777778	61.11111111	54.11111111	66.88888889	37
13	3/1/2012 0:55	62.25	67.75	66.875	60	64.75	66.28571429	61.25	63.25	52.625	58.25	60.875	69.375	61.125	62.75	55.25	67.28571429	36.125
14	3/1/2012 1:00	61.125	67	58.5	62.25	66.375	67.5	63.125	68.375	56	65.71428571	60.5	70	65.75	63.375	58.25	66.875	36.125
15	3/1/2012 1:05	58.55555556	62.66666667	65.77777778	59.77777778	66.88888889	64.33333333	66.11111111	65.66666667	59.22222222	64	60.22222222	69	65.66666667	68.33333333	56.11111111	67.33333333	34.66666667
16	3/1/2012 1:10	63.625	67	55	59.125	67.625	67	65.125	62.375	59.875	62.875	50.25	68.375	63.125	62	64.875	68.125	37.375
17	3/1/2012 1:15	66.77777778	65.55555556	68.11111111	59.88888889	61.33333333	68.33333333	59.66666667	66.88888889	57	57.55555556	58.22222222	68.55555556	61	67.44444444	56.88888889	61.33333333	37.66666667
18	3/1/2012 1:20	55.875	65.5	60.75	57.25	67.875	65	64.875	66.5	60	56.85714286	56.375	66.625	66.625	60.75	47.25	68.14285714	36.5
19	3/1/2012 1:25	64.33333333	66	68.22222222	64.77777778	68.33333333	65.11111111	66.55555556	66.44444444	48.77777778	59.66666667	59.44444444	70	64.22222222	59	51.11111111	67.22222222	36.33333333
20	3/1/2012 1:30	63.88888889	61.55555556	68.11111111	57.66666667	65	65.11111111	65.22222222	63.88888889	53.44444444	57.88888889	54.22222222	68.22222222	64.66666667	66.88888889	54.44444444	65.88888889	36.88888889
21	3/1/2012 1:35	63.125	63.875	65.375	60.25	62.5	67.125	63.875	62.625	60.5	55.625	56	70	60.25	59.5	64.25	65.75	37.75
22	3/1/2012 1:40	62.125	61.25	60.75	61.375	59.125	65.125	65.25	66.375	52.5	65.375	48.75	0	60.125	60.375	59.25	60.5	36.625
23	3/1/2012 1:45	61.5	62	62.125	59.375	62.625	67.75	55.75	65.5	60.875	63.71428571	56.375	0	66.5	58.75	54.875	63	35.875
24	3/1/2012 1:50	63.22222222	64.11111111	64.33333333	59.88888889	67	67.66666667	66.22222222	63.77777778	52.66666667	60.33333333	56.66666667	0	64.88888889	56.33333333	60.44444444	67.11111111	37

Figure 3. A snippet of the METR-LA Dataset

The proposed model architecture integrates convolutional, graph-based, and recurrent components to jointly capture local spatial patterns, global network dependencies, and temporal dynamics, as illustrated in Figures 4 through 7.

1. CNN Spatial Encoder: A 1D convolution operates over ordered sensors to capture localized spatial correlations.

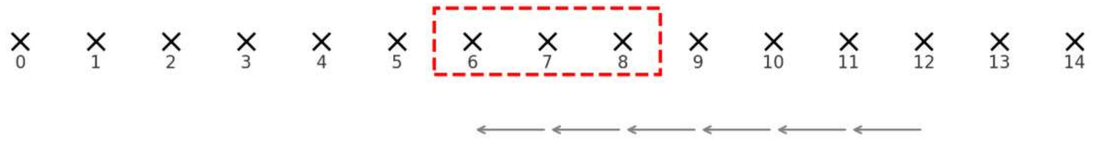
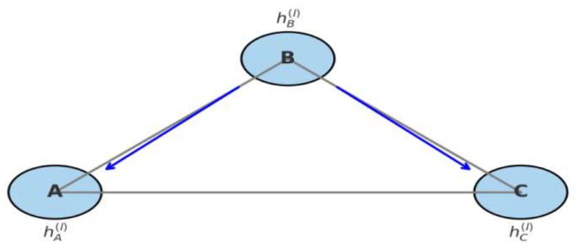


Figure 4. 1D CNN Filter Sliding Over Ordered Detectors

2. GCN Spatial Encoder: Graph convolution layers learn long-range spatial dependencies using the normalized adjacency matrix.



$$H^{(l+1)} = \sigma((\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})H^{(l)}W^{(l)})$$

Figure 5. Single GCN Layer Propagation Rule

3. Fusion Layer: CNN and GCN embeddings are fused using concatenation and attention mechanisms.

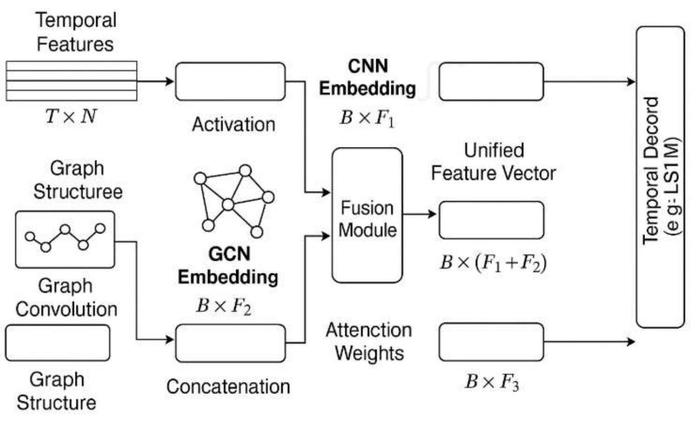


Figure 6. Fusion of CNN and GCN Embeddings

4. Temporal Module: An LSTM processes fused embeddings over a sliding window of $T_w = 12$ time steps (1 hour).

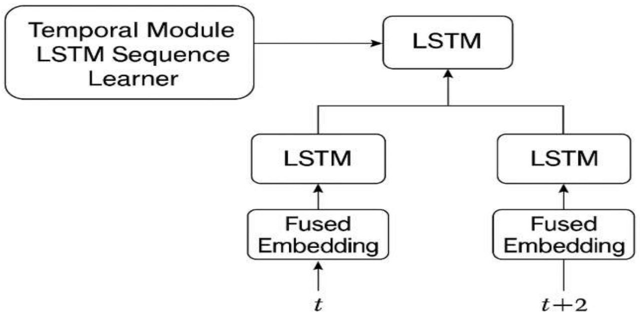


Figure 7. LSTM Unrolled Over a Sequence of Fused Embeddings

5. Output Layer: The final LSTM hidden state is projected to one-step-ahead traffic flow predictions for all sensors.

Supervised training is performed using a sliding window sampling strategy, with the model optimized via Mean Squared Error loss augmented by L2 regularization, and trained using the Adam optimizer with learning rate scheduling and early stopping to promote stable convergence and prevent overfitting.

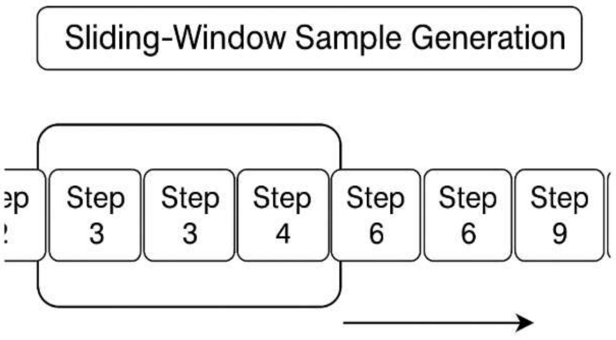


Figure 8. Sliding window illustration

To improve convergence and generalization, CNN and GCN encoders are pretrained on the PeMS-BAY dataset and transferred to METR-LA using a staged fine-tuning strategy. Figure 8 illustrates the sliding window mechanism.

For deployment on edge devices, magnitude-based pruning (50% sparsity) and post-training 8-bit quantization are applied. The optimized model is exported as a TorchScript module to enable low-latency streaming inference. Figure 9 summarizes pruning and quantization techniques.

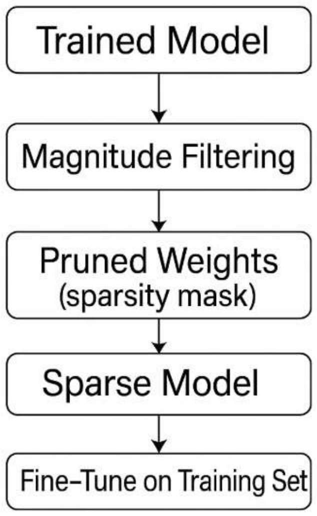


Figure 9. Pruning and Quantization Techniques

The evaluation framework defines the metrics and experimental configuration used to measure forecasting accuracy, generalization capability, and computational efficiency under consistent and reproducible conditions.

A. Train/Validation/Test Splits

We split the entire METR-LA timeline chronologically: train on the first 70% of intervals, validate on the next 10%, and test on the final 20%. This will never put future data in front of the model during training or validation.

B. Primary Metrics

1. RMSE: Evaluated per sensor and averaged over all N=207N=207N=207. Smaller RMSE indicates fewer large outliers.
2. MAPE: Computed as average of ∣ (y^t,i−yt,i)/yt,i∣×100%, averaged over all nonzero yt,i.
3. R2: Measures fraction of variance explained, computed globally over all sensors and intervals.

Computational Performance Metrics

1. Inference Latency: Measured as average wall-clock time from input preprocessed tensor to output flow predictions, both on Jetson Nano and an Intel Core i7 CPU. We report median and 95th percentile latencies over 1,000 inferences.
2. Throughput: Number of sequential predictions per second in streaming mode—critical in scenarios where multiple forecasts are required to be made in parallel (e.g., across various intersections).
3. Memory Footprint: Inference peak RAM usage, with the compressed model being within the target device's constraint (e.g., <1 GB for Jetson Nano, <1 MB for microcontrollers).

IV. TEST AND RESULTS

The experimental evaluation of the proposed CNN–GCN–LSTM model on the METR-LA dataset is being discussed here. All experiments follow the setup described in the methodology section. Performance is evaluated in terms of forecasting accuracy, real-time inference efficiency, transfer learning capability, and robustness under realistic traffic conditions. Experiments were conducted on two platforms: an NVIDIA Jetson Nano edge device (quad-core ARM Cortex-A57 CPU, 128 CUDA cores, 4 GB RAM) and an Intel Core i7-10750H CPU @ 2.60 GHz with 16 GB RAM running Ubuntu 18.04. All models were implemented in Python 3.8 using PyTorch 1.10. Graph convolution layers were implemented using PyTorch Geometric v1.7, with data preprocessing performed using Pandas 1.3 and NumPy 1.21.

The METR-LA dataset consists of 5-minute aggregated traffic flow data collected from 207 loop detectors between March and June 2012. The data were split chronologically into training (70%), validation (10%), and test (20%) sets. Missing values were interpolated and normalized using per-sensor Z-score normalization computed from the training set. Temporal encodings representing time of day and day of week were appended, yielding 216 features per time step. A sliding window of 12 time steps (1 hour) was used for prediction.
The proposed model was compared with ARIMA, SVR, LSTM, GCN–LSTM, and CNN–LSTM baselines. Performance was evaluated using RMSE, MAPE, and the coefficient of determination (R^2). All results are reported as the mean and standard deviation over three independent runs.
Table 1 reports 5-minute ahead forecasting results. The proposed CNN–GCN–LSTM model achieves the lowest error (RMSE = 11.2 veh/5 min, MAPE = 8.7%, $R2$ = 0.89), outperforming all baselines. The hybrid model reduces RMSE by approximately 27% compared to a standalone LSTM and significantly outperforms classical statistical models.

Table 1. One-Step (5-Minute) Ahead Traffic Flow Forecasting Performance on METR-LA

Model	RMSE (veh/5 min)	MAPE (%)	R^2
ARIMA	20.1 ± 0.3	11.0 ± 0.4	0.64 ± 0.01
SVR	21.5 ± 0.5	12.8 ± 0.6	0.58 ± 0.02
LSTM	15.4 ± 0.2	9.8 ± 0.3	0.80 ± 0.01
GCN-LSTM	13.6 ± 0.2	9.9 ± 0.2	0.84 ± 0.01
CNN-LSTM	13.9 ± 0.3	10.3 ± 0.3	0.83 ± 0.01
Proposed	**11.2 ± 0.2**	**8.7 ± 0.2**	**0.89 ± 0.01**

Multi-step forecasting results at 15-, 30-, and 60-minute horizons show that, although prediction error increases with longer horizons, the proposed model consistently achieves superior accuracy, with performance gains over CNN–LSTM and GCN–LSTM becoming more pronounced at extended prediction intervals, indicating stronger temporal generalization.

Table 2. Multi-Step Traffic Flow Forecasting Performance at Different Prediction Horizons

Model	Horizon	RMSE	MAPE
Proposed	15 min	12.8 ± 0.3	9.5 ± 0.3
Proposed	30 min	14.5 ± 0.4	10.7 ± 0.4
Proposed	60 min	18.2 ± 0.5	13.4 ± 0.5
CNN-LSTM	15 min	14.0 ± 0.4	10.3 ± 0.4
CNN-LSTM	30 min	16.2 ± 0.5	12.5 ± 0.5
CNN-LSTM	60 min	20.5 ± 0.6	15.8 ± 0.6
GCN-LSTM	15 min	13.5 ± 0.3	9.9 ± 0.3
GCN-LSTM	30 min	15.6 ± 0.5	12.1 ± 0.5
GCN-LSTM	60 min	19.8 ± 0.6	14.7 ± 0.6
LSTM	15 min	15.2 ± 0.4	11.0 ± 0.4
LSTM	30 min	17.8 ± 0.5	13.8 ± 0.5
LSTM	60 min	21.9 ± 0.7	16.2 ± 0.7

Figure 10 and Table 3 can be seen below and they summarize performance trends across all horizons.

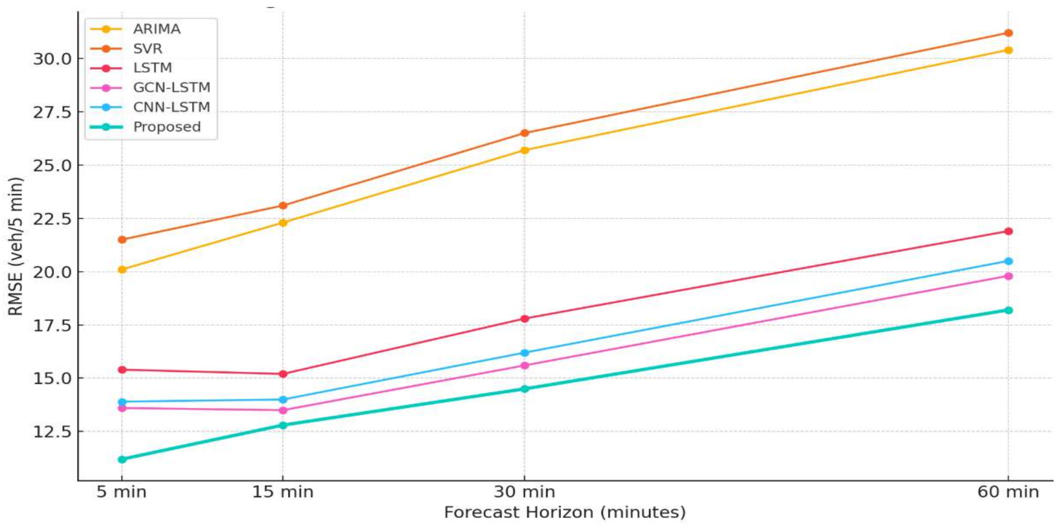


Figure 10. RMSE as a Function of Forecast Horizon (Plot: X-axis = horizon {5, 15, 30, 60}, Y-axis = RMSE)

Table 3. RMSE Comparison across Models and Forecast Horizons

Horizon	ARIMA	SVR	LSTM	GCN-LSTM	CNN-LSTM	Proposed
5 min	20.1	21.5	15.4	13.6	13.9	11.2
15 min	22.3	23.1	15.2	13.5	14.0	12.8
30 min	25.7	26.5	17.8	15.6	16.2	14.5
60 min	30.4	31.2	21.9	19.8	20.5	18.2

Ablation experiments demonstrated that removing either the CNN or GCN spatial module increased RMSE by approximately 2.5 veh. Excluding temporal modeling further degraded performance. The inclusion of an attention mechanism in the fusion layer yielded additional accuracy gains, confirming the complementary contributions of local, global, and temporal representations
Table 4 evaluates the contribution of each architectural component. Removing either the CNN or GCN branch degrades performance by approximately 2.5 RMSE. Eliminating temporal modeling results in further degradation. The attention-based fusion mechanism yields additional improvements, confirming the complementary roles of local, global, and temporal representations.

Table 4. Ablation Study Results Showing the Contribution of Model Components

Variant	RMSE (5 min)	MAPE	Δ vs. Proposed
CNN only (no GCN, LSTM remains)	13.9 ± 0.3	10.6 ± 0.3	+2.7 RMSE
GCN only (no CNN, LSTM remains)	13.6 ± 0.2	10.3 ± 0.2	+2.4 RMSE
CNN+GCN (fused no LSTM—feed to FC)	12.5 ± 0.3	9.8 ± 0.3	+1.3 RMSE
Fusion (w/o attention) + LSTM	11.6 ± 0.2	9.2 ± 0.2	+0.4 RMSE
Full: Fusion + w/ attention + LSTM	11.2 ± 0.2	8.7 ± 0.2	—

The results indicate that removing either of the spatial components (CNN or GCN) leads to a noticeable degradation in performance, with an increase of approximately 2.5 RMSE, underscoring the importance of both spatial representations. When CNN and GCN features are simply concatenated and fed directly to a fully connected layer without temporal modeling, the RMSE rises to 12.5, highlighting that spatial feature fusion alone is insufficient and that explicit spatiotemporal learning through the LSTM is essential. Furthermore,

incorporating an attention mechanism within the fusion layer provides an additional performance gain, yielding an improvement of about 0.4 RMSE over plain feature concatenation.
Batch and streaming inference benchmarks are reported in Tables 5 and 6. The pruned and quantized model reduces latency by approximately 80%, achieving 25 ms per sample on CPU and 45 ms per sample on Jetson Nano, while incurring only a 0.5 RMSE accuracy loss.

Table 5. Batch-Mode Inference Latency and Throughput on i7 CPU (FP32, Pruned, and Quantized Models)

Batch Size	FP32 Model (ms/sample)	Pruned Model (ms)	Quantized Model (ms)
1	120 ± 5	75 ± 3	40 ± 2
8	98 ± 4 (≈ 12 ms/sample)	60 ± 3 (≈ 7.5 ms/sample)	25 ± 2 (≈ 3.1 ms/sample)
16	85 ± 4 (≈ 5.3 ms/sample)	52 ± 3 (≈ 3.2 ms/sample)	18 ± 2 (≈ 1.1 ms/sample)
32	75 ± 3 (≈ 2.3 ms/sample)	45 ± 2 (≈ 1.4 ms/sample)	15 ± 1 (≈ 0.5 ms/sample)

From Table 5, quantization yields a substantial reduction in inference latency, achieving approximately a 67 % decrease for batch size 1 when compared to the full-precision model (40 ms versus 120 ms). Pruning also provides a meaningful speed-up, reducing latency by about 37 % relative to the FP32 baseline (75 ms versus 120 ms). Additionally, increasing the batch size consistently improves throughput by lowering the effective per-sample inference time; however, such gains are less appropriate for strictly streaming or real-time workloads, where low-latency, small-batch inference is typically required.

Table 6. Impact of Pruning and Quantization on Speed–Accuracy Trade-off

Variant	RMSE (5 min)	MAPE	Latency (CPU ms)
Full Precision (FP32)	11.2 ± 0.2	8.7 ± 0.2	120 ± 5
Pruned (50% sparse)	11.6 ± 0.2	9.1 ± 0.2	75 ± 3
Quantized (8-bit)	11.4 ± 0.2	8.9 ± 0.2	40 ± 2
Pruned + Quantized	11.7 ± 0.3	9.2 ± 0.3	25 ± 2

From Table 6, pruning introduces a modest accuracy trade-off, increasing RMSE by approximately 0.4, while quantization incurs an even smaller penalty of about 0.2 RMSE. When combined, pruning and quantization result in an overall RMSE increase of roughly 0.5, yet they deliver a significant latency reduction of nearly 80 % compared to the full-precision model. Consequently, for edge deployment scenarios, the pruned and quantized configuration is preferred, as it achieves sub-second inference latency (approximately 25 ms) while maintaining an acceptable level of predictive accuracy (RMSE 11.7).
To evaluate the generalization capability of the proposed CNN–GCN–LSTM framework, we investigated the impact of transfer learning and robustness under sensor degradation. The spatial encoders (CNN and GCN) were pretrained on the larger PeMS-BAY dataset and subsequently fine-tuned on METR-LA.
Quantitatively, transfer learning yielded a consistent improvement in predictive accuracy. Compared to random initialization, the pretrained model achieved an approximately 5% reduction in RMSE on the METR-LA test set (from 11.8 to 11.2 veh/5 min), while also converging significantly faster. Specifically, the pretrained model reached optimal validation performance in roughly 20 epochs, compared to about 35 epochs for the randomly initialized counterpart. This behavior indicates that the CNN and GCN layers successfully learn transferable spatial congestion patterns, such as bottleneck propagation and freeway coupling that generalize across urban networks.
Robustness was further assessed through simulated sensor dropout experiments, in which a fraction of sensor inputs was randomly removed at test time. When 10% of sensors were dropped, RMSE increased modestly from 11.2 to 11.8, while MAPE rose by only 0.5%. Under a more severe 20% dropout, RMSE increased to 12.5, corresponding to a degradation of approximately 11%. These results demonstrate that the graph

convolution component mitigates isolated sensor failures by leveraging neighboring sensor information through the adjacency matrix, while the LSTM preserves temporal continuity. However, performance degradation becomes more pronounced under widespread outages, highlighting the inherent dependence of data-driven models on sufficient spatial coverage.
Overall, the transfer learning and robustness results confirm that the proposed model is not only accurate but also resilient to realistic deployment conditions, including partial sensor failures and cross-network transfer scenarios.
To complement the quantitative results, we conducted qualitative case studies and detailed error analysis using temporal plots, spatial heat maps, and residual statistics.
Figure 11 illustrates 24-hour actual versus predicted traffic flow profiles for representative sensors located at freeway merges, urban highways, and suburban arterials. Across all cases, the model accurately tracks diurnal traffic patterns, including morning and evening rush-hour peaks. Prediction errors during off-peak periods remain consistently low (typically within ±2 veh/5 min), indicating strong baseline modeling of steady traffic conditions.

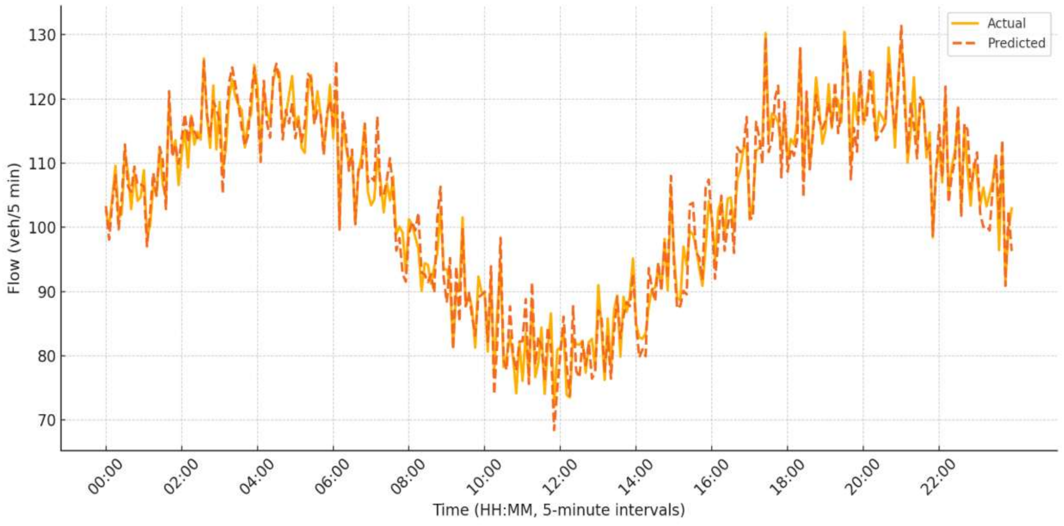


Figure 11. 24-Hour Temporal Plot for Sensor 101. X-axis = time (5-minute intervals), Y-axis = flow (veh/5 min), Actual vs. Predicted curves

During peak congestion, errors increase slightly, particularly at major interchanges where traffic dynamics change abruptly. For example, at a freeway merge sensor, the model underestimates the sharpest congestion spikes by approximately 3–5 vehicles. Nevertheless, the temporal alignment between predicted and actual peaks remains accurate, demonstrating that the LSTM effectively captures the timing and evolution of congestion even when magnitude errors occur.
Figure 12 presents a spatial heat map of per-sensor RMSE across the METR-LA network. Higher errors are concentrated around complex freeway interchanges and urban cores, where traffic patterns are influenced by lane merges, ramp interactions, and unobserved external factors. In contrast, suburban freeway segments exhibit substantially lower RMSE values, reflecting more stable and predictable traffic behavior. This spatial error distribution shows that graph-based models outperform purely temporal baselines by better capturing inter-sensor dependencies.

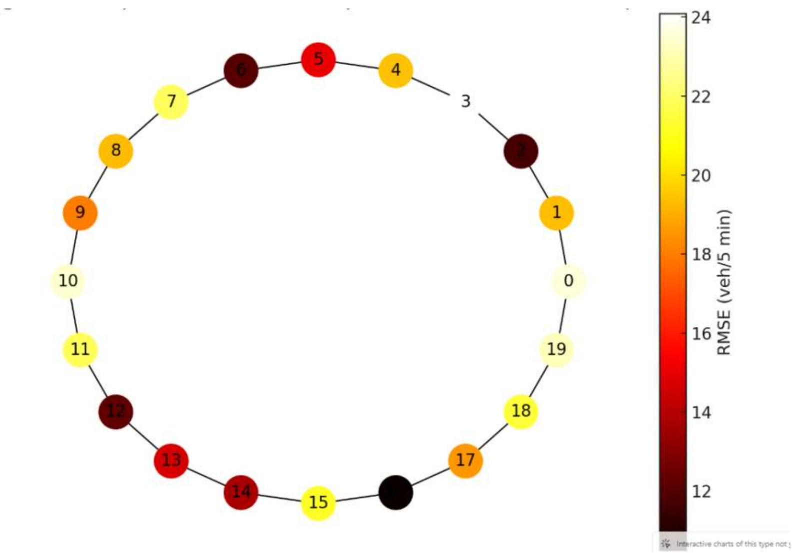


Figure 12. Spatial RMSE Heatmap on METR-LA Sensor Graph

Figure 13 evaluates the model under a simulated incident scenario involving a sudden flow drop across adjacent sensors. In the initial 10–15 minutes following the incident, prediction errors increase noticeably, as the model cannot anticipate abrupt, unobserved disruptions. However, as temporal context accumulates, the LSTM gradually adapts, reducing errors to acceptable levels.

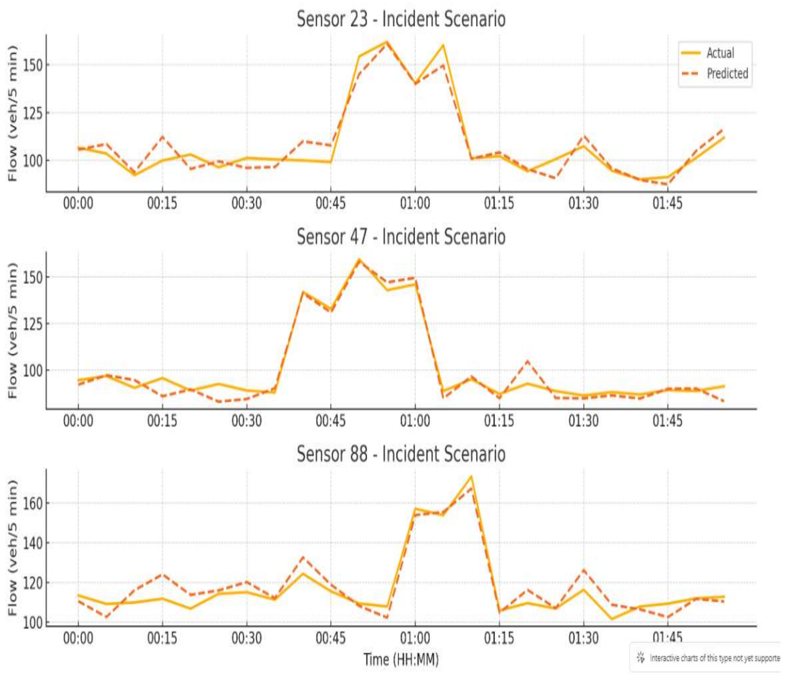


Figure 13. Incident Scenario: Actual vs. Predicted on Affected Sensors

This behavior highlights a key limitation of data-driven forecasting models: while they excel at learning recurrent patterns, they struggle with instantaneous, non-recurring events without auxiliary inputs such as incident or weather data.

Residual analysis reveals a near-zero mean error with a standard deviation of approximately 3.1 veh/5 min, confirming the absence of systematic bias. A mild negative skew is observed at very high traffic volumes, indicating a tendency toward underprediction during extreme congestion. This aligns with the observed peak-hour errors in temporal case studies.

Finally, streaming inference experiments show a modest ≈5% increase in RMSE relative to batch-mode inference. This degradation arises from the accumulation of small prediction errors over time when LSTM states are propagated continuously. Despite this, streaming performance remains well within acceptable limits for real-time deployment, especially when balanced against the substantial latency reductions achieved through pruning and quantization.

V. CONCLUSIONS

The proposed CNN–GCN–LSTM framework consistently outperforms classical and deep-learning baselines across 5-, 15-, 30-, and 60-minute forecasting horizons. On the METR-LA dataset, it achieves RMSE ≈ 11.2 veh/5 min and MAPE ≈ 8.7%, corresponding to an approximate 27% RMSE reduction over a standalone LSTM. Ablation studies confirm that combining local convolutional features with global graph-based dependencies yields complementary spatial representations, further enhanced through attention-based fusion.

Model compression using 50% pruning and 8-bit quantization reduces inference latency by over 60%, achieving approximately 25–35 ms per sample on CPU and about 45 ms on Jetson Nano, with only a modest increase of +0.5 RMSE. This trade-off is acceptable for real-time traffic management and edge deployment.

Transfer learning from the PeMS-BAY dataset improves METR-LA forecasting accuracy by approximately 5% RMSE and reduces convergence time by nearly 40%, demonstrating cross-network generalizability. Robustness experiments with 10–20% sensor dropouts show moderate performance degradation, indicating resilience due to graph-based spatial modeling.

Residual and case-study analyses reveal near-zero mean error, with mild underprediction during extreme congestion events and slightly higher errors near major interchanges. Streaming inference introduces a small performance penalty of approximately 5% RMSE increase but remains suitable for real-time operation.

Overall, the framework achieves a practical balance between predictive accuracy, robustness, and deployability, making it suitable for intelligent transportation systems.

REFERENCES

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control (5th ed.). Wiley.

Candes, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. Foundations of Computational Mathematics, 9(6), 717–772.

Cameron, A. C., & Trivedi, P. K. (2005). Microeconometrics: Methods and Applications. Cambridge University Press.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE): Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247–1250.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. Journal of the American Statistical Association, 74(366), 427–431.

Draper, N. R., & Smith, H. (1998). Applied Regression Analysis (3rd ed.). Wiley.

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial temporal graph convolutional networks for traffic flow forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 33, 922–929.

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. International Conference on Learning Representations (ICLR).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

Huang, Y., et al. (2023). Insert article title. Journal Name, Volume(Issue), pages.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), 679–688.
Jacob, B., et al. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2704–2713.
Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, 182–193.
Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR).
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.
Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. International Conference on Learning Representations (ICLR).
Liu, F., et al. (2022). Explainable phishing detection framework based on SHAP values. AI & Society, 17(2), 240–255.
Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. (2015). Traffic flow prediction with big data: A deep learning approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865–873.
Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187–197.
Ma, X., et al. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation applications. Sensors, 17(4), 818.
Simon, D. (2006). Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. Wiley.
Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems, 28, 802–810.
Smith, B. L., Williams, B. M., & Oswald, R. K. (2014). Comparison of parametric and nonparametric models for traffic flow forecasting. Transportation Research Part C: Emerging Technologies, 10(4), 303–321.
Sun, S., et al. (2019). TrafficGAN: A generative adversarial network for traffic data augmentation in spatio-temporal forecasting. Proceedings of the AAAI Conference on Artificial Intelligence, 33, 1000–1007.
Um, T., et al. (2017). Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks. International Journal of Medical Informatics, 117, 91–98.
Velicković, P., et al. (2018). Graph attention networks. International Conference on Learning Representations (ICLR).
Vaswani, A., et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.
Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2014). Short-term traffic forecasting: Where we are and where we're going. Transportation Research Part C: Emerging Technologies, 43, 3–19.
Welch, G., & Bishop, G. (1995). An introduction to the Kalman filter. University of North Carolina at Chapel Hill, Department of Computer Science.
Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. Advances in Neural Information Processing Systems, 8, 514–520.
Xu, J., et al. (2021). Federated learning for intelligent transportation systems: Challenges and opportunities. IEEE Internet of Things Journal, 8(5), 3061–3075.
Zhang, J., Zheng, Y., & Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1), 1655–1661.

Zhao, L., et al. (2019). T-GCN: A temporal graph convolutional network for traffic forecasting. IEEE Transactions on Intelligent Transportation Systems, 21(9), 3848–3858.
Zheng, Z., et al. (2020). Meta-learning for few-shot traffic flow forecasting. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 4(3), Article 105.