Proceedings of the 3rd International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2025.31671

Research on Spatiotemporal Prediction Model of Urban Traffic Flow Based on Graph Neural Networks and Attention Mechanism

Yuliang Xiao

School of Computer Science and Technology, Shandong Jianzhu University, Jinan, China
2084504941@qq.com

Abstract. Traffic flow pblueiction is one of the hardest (and most practical) problems in smart city work. If you get the forecasts wrong,everything downstream suffers: signal timing,incident response,routing,even basic city operations. The trouble is that a lot of existing methods still don`t do a great job with the messy reality of road networks,where conditions change fast and influences ripple across space and time in non-obvious ways.This paper proposes a model called the Dynamic Spatiotemporal Graph Attention Network (DSTGAT). In plain terms,it mixes graph neural networks with attention to better match how traffic actually behaves. The key idea is a dynamic traffic graph that can change as road conditions change,instead of treating the network as fixed. A graph convolutional network is then used to learn spatial relationships from that graph,while a spatiotemporal attention module focuses on the temporal side of the problem.Tests on three real-world datasets show consistent gains over common baselines. DSTGAT cuts MAE by 15.3% and RMSE by 12.7%,which is a meaningful jump if you care about day-to-day traffic control rather than just pretty benchmark numbers. Overall,the results suggest DSTGAT is a solid step toward more reliable,real-world traffic pblueiction for urban management.

Keywords: Graph Neural Network, Attention Mechanism, Traffic Flow Prediction, Spatiotemporal Modeling, Deep Learning

1. Introduction

Traffic congestion in cities keeps getting worse, and it's starting to feel like the thing that blocks every "sustainable city" plan before it even gets going. Being able to pblueict traffic flow helps in two very practical ways: it gives traffic agencies something solid to base decisions on, and it improves the stuff drivers actually notice, like signal timing, route guidance, and how staff and road resources get deployed. Done well, that adds up to less congestion and smoother day-to-day operations [1]. At its core, traffic flow pblueiction is a spatiotemporal forecasting problem, and it's a messy one. The data is nonlinear, it shifts over time, and nearby road segments influence each other in ways that aren't always obvious [2]. You also see strong regular patterns (rush hour is rush hour), plus clear spatial dependence across the network. Classic statistical models and many standard machine learning methods can handle parts of this, but they tend to struggle with two big realities: the nonlinear behavior of traffic and the fact that road networks aren't laid out on a neat grid. They're graphs, with all the awkward, non-Euclidean structure that comes with them [3]. That's where graph neural

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.31671

networks (GNNs) help. They work directly on graph-structublue data, which makes it much easier to represent road connections and spatial relationships without forcing everything into a grid. Attention layers add another advantage: they let the model "pay more attention" to the nodes and time periods that matter most for a given pblueiction. The catch is that a lot of GNN-based traffic models still treat the graph as basically fixed. In real traffic, relationships change: a minor incident, a stadium event, or even weather can suddenly make two segments behave as if they're tightly linked. Static graphs don't capture that well, and long-range effects often get washed out too [4]. To deal with those gaps, this paper proposes DSTGAT and frames it around three main ideas. First, it builds a dynamic graph to represent time-varying correlations between road segments, and it does this in a way that adds to the physical road topology rather than trying to replace it [5]. Second, it combines graph convolution with gated recurrent units to pull out spatiotemporal features at different time scales. Third, it uses a multi-head spatiotemporal attention module to adjust, on the fly, how much different nodes and time steps should influence the pblueiction, with the goal of pushing accuracy higher.

2. Introduction to related methods

2.1. Traditional traffic flow prediction methods

Traditional methods lean hard on time-series statistics: ARIMA, Kalman filters,that whole family [6]. They work fine when the world behaves—mostly linear, mostly stationary. Traffic doesn't. The moment you get stop-and-go waves, a crash on an on-ramp, bad weather, or just some random commuter quirk, those neat assumptions start to crack. And there's another practical issue: these models often treat each road segment as its own little island, so spillover congestion—one link backing up the next—gets smoothed out or missed entirely.

SVMs deal with nonlinearity better, but they come with a different blind spot: they still don't really "see" the network. You end up feeding flows from different locations as separate variables, not as nodes connected by an actual topology. Once you scale up to a city-sized network, compute also becomes a real problem [7].

2.2. Deep learning in traffic forecasting

Deep learning raised the bar. RNNs—especially LSTM and GRU—learn temporal patterns without a lot of hand-crafted feature work [8]. On the spatial side, CNN-based models tried to force traffic onto a grid so they could learn local neighborhoods; ST-ResNet is the classic example. The problem is the grid is a hack. It can distort the road layout, so two cells might be "neighbors" in the tensor while being nowhere near each other on the map (and the reverse happens too) [9].

2.3. Spatiotemporal Graph Neural Networks and attention mechanisms

More recent models treat the traffic network as what it is: a graph. Spatiotemporal GNNs (for example, STGCN) operate directly on nodes and edges, and they usually outperform the older baselines by a comfortable margin. Graph WaveNet pushes this further by learning an adaptive adjacency matrix,which helps when "who influences who" isn't fixed and a hand-built graph can't keep up.

Attention has become a staple for the same reason. Instead of pretending every input matters equally, attention lets the model weight what it cares about—specific sensors, particular time steps, certain features—and that often improves accuracy while making the model's behavior easier to inspect. GMAN uses spatiotemporal attention, and STTN borrows a more Transformer-style setup; both report strong results in practice. Even so, the same two pain points keep coming back: spatial dependencies really do change over time, and blending information across multiple spatial and temporal scales without turning the model into a fragile, overgrown beast is still not fully solved [10].

DOI: 10.54254/2755-2721/2025.31671

3. Methodology

The proposed DSTGAT model has four main components. These are: a dynamic graph construction module, a multi-scale spatiotemporal feature extraction block, a multi-head attention fusion module, and a final prediction layer [11].

3.1. Dynamic graph construction

Graph structure quality directly determines GNN performance. A static, distance-based graph is insufficient. It cannot capture real-time traffic dynamics. During congestion, for example, strong correlations can emerge between nodes that are physically far apart.

Our model constructs a hybrid adjacency matrix A hybrid by combining a static matrix with a dynamic one:

$A_{hybrid}=W_{static}\cdot A_{static}+W_{dynamic}\cdot A_{dynamic}$

Static Adjacency Matrix (A_{static}): This matrix represents the physical road network topology. The weight between two sensor nodes i and j is calculated based on their road network distance, using a Gaussian kernel. This captures the fixed spatial proximity.

Dynamic Adjacency Matrix ($A_{dynamic}$): This matrix captures the functional similarity between nodes based on recent traffic patterns. We calculate the Pearson correlation coefficient between the traffic flow time series of every pair of nodes over a recent time window (e.g., the last hour). A high correlation indicates a strong functional link, even if the nodes are physically distant. This matrix is re-calculated periodically (e.g., every 15 minutes) to adapt to changing traffic conditions [12].

3.2. Multi-scale spatiotemporal feature extraction

This module is the computational core of DSTGAT. It's built to learn how things depend on each other in space and in time.

The module is a stack of repeated blocks. Each block has two parts: a Graph Convolutional Network (GCN) layer for spatial modeling, followed by a Gated Recurrent Unit (GRU) layer for temporal modeling. E>At each time step,the GCN aggregates information from neighboring nodes using the hybrid adjacency matrix, A_hybrid. That way, the model learns spatial relationships based on both physical distance and similarity in traffic patterns. One catch: if you stack too many GCN layers, node representations start to blur together (the classic over-smoothing problem). To keep that from happening, the network keeps the stack shallow and uses residual connections.

3.3. Multi-head attention fusion

After extracted spatiotemporal features, a multi-head attention mechanism fuses them. The fusion is adaptive. It learns to weight both spatial and temporal aspects of feature representation. Spatial attention: For each node, the attention mechanism calculates importance scores based on all other nodes. The attention mechanism can detect long-range spatial dependencies, that is, the neighborhood not only within the network, but also from the neighborhood of graph convolution [13]. Time-extension: It assigns importance scores to different historical time steps. The time-extensions of the attention system can lead the model to focus on the most relevant past observations. For instance, the morning peak prediction has the benefits from up-weighting traffic from 24 hours earlier. Multi-head design: The model may take information from multiple representation subspaces simultaneously, capturing more spatial relationships than a single attention head.

DOI: 10.54254/2755-2721/2025.31671

3.4. Prediction layer

Attention module outputs a rich spatiotemporal representation. The representation is passed into two fully connected layers and maps to final traffic flow predictions for the future time steps. Fig. 1 represents the complete DSTGAT model. The input is processed in four stages. Raw traffic data at time steps t−T+1 through t first enter dynamic graph constructor. A_hybrid combines the static distance-based topology with a dynamically inferred functional graph to obtain A_Hybrid. This representation is then passed into stacked GCN-GRU blocks, which extract spatially-conditioned features at each time step and propagate along the time axis. The representations pass through multi-head spatiotsemporal attention module, which reweights important sensor nodes and historical time steps, and a short prediction head outputs one-step or multi-horizon forecasts. Deep connections and layer normalization are applied throughout the network to stabilize training and reduce over-smoothing.

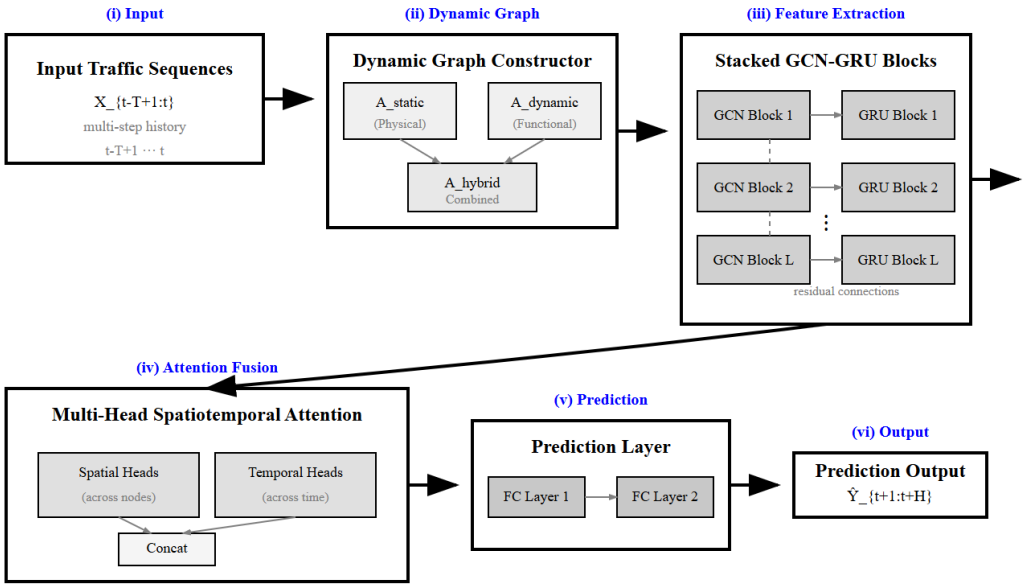


Figure 1. Dynamic Spatiotemporal Graph Attention Network (DSTGAT) architecture. From left to right: (i) input traffic sequences; (ii) dynamic graph constructor producing A_{static}, $A_{dynamic}$, and A_{hybrid}; (iii) stacked GCN–GRU blocks with residual paths; (iv) multi-head spatiotemporal attention (spatial heads across nodes; temporal heads across lags); and (v) the prediction layer outputting $Y_{t+1:t+H}$. Arrows indicate information flow; dashed arrows denote residual connections

4. Experiments and results

4.1. Dataset description and preprocessing

We use three public benchmark datasets: PeMS-BAY, METR-LA, and PeMS04. They differ in network size, geographic coverage, and traffic dynamics, so they are a decent check on whether a model still works when the setting changes. For preprocessing, we impute missing readings with adaptive interpolation, mark outliers using more than one detector, and apply Z-score normalization so training stays stable and the inputs don't swing wildly.

4.2. Limitations of existing Graph Neural Network methods

GNNs show up everywhere in traffic forecasting, but the same weak spots keep resurfacing.

DOI: 10.54254/2755-2721/2025.31671

1) Most models treat the graph as fixed. A lot of methods build the graph from a distance cutoff or from road connectivity and then leave it alone. That's easy to implement, but it ignores the fact that traffic relationships shift throughout the day. The link between two sensors isn't constant: when congestion builds, nearby segments start moving in lockstep; when things are clear, the useful context often reaches farther than the immediate neighbors.

2) Graph convolutions are too local, and making them "see farther" gets ugly. Standard graph convolutions mainly mix information within a small neighborhood, which makes long range spatial effects hard to capture. You can stack more layers to extend the receptive field, but then over smoothing shows up and node embeddings start to look the same. Also, a lot of these operators are basically linear weighted averages, which is a pretty crude fit for the nonlinear spatial interactions you get in real traffic [14].

3) Temporal modeling is often thinner than it needs to be. Many spatiotemporal GNNs rely on simple temporal convolutions or RNN style blocks. They can track short term patterns, but they tend to miss effects that depend on longer history (the ramp up to rush hour, fallout after an incident, day to day cycles, and so on). Temporal attention is out there, but in many papers it still feels too weak to reliably isolate the few past moments that actually matter for the current forecast. Fixed length windows don't help either, because the model can't extend or shorten its lookback when the situation changes.

Table 1. Performance comparison of different prediction methods

Method Category	Representative Method	Temporal Modeling	Spatial Modeling	Computational Complexity	Prediction Accuracy
Traditional Stats	ARIMA	Weak	None	Low	Low
Machine Learning	SVM	Moderate	Weak	Moderate	Moderate
Deep Learning	LSTM	Strong	None	Moderate	Moderate
Spatiotemporal CNN	ST-ResNet	Moderate	Moderate	High	Moderate
Spatiotemporal GNN	STGCN	Strong	Strong	High	High
Attention-based GNN	GMAN	Strong	Strong	High	High

4.3. Dataset description and preprocessing

The three datasets are all sourced from California's Performance Measurement System (PeMS). PeMS was established under the California Transportation Data Management Act. It provides standardized traffic data for research. Using these datasets ensures comparability with published benchmarks.

PeMS-BAY contains data from 325 traffic sensors across the San Francisco Bay Area. Coverage spans January 1 to June 30, 2017, for a total of six months. The network includes urban highways, suburban arterials, and major bridges. This variety effectively represents the complexity of real urban traffic networks. Sensors record data every five minutes, producing 52, 116 time steps in total. Each time step captures flow, speed, and occupancy.

METR-LA covers traffic monitoring in the Los Angeles metropolitan area. It contains records from 207 sensor nodes. The data spans March to June 2012, covering four months. Data completeness is high at 96.8%. Los Angeles is a heavily car-dependent city. It shows clear tidal patterns and strong periodic regularities. This makes it an ideal dataset for testing a model's temporal dependency modeling.

PeMS04 covers a broader geographic area within California. It includes 307 sensor nodes with data collected from January to February 2018. Sensor spacing is relatively large. This makes PeMS04

DOI: 10.54254/2755-2721/2025.31671

a good test case for sparse network scenarios. The dataset spans road types from interstate highways to local roads. Traffic flow varies widely across these road types, providing a comprehensive performance evaluation.

Statistical analysis of all three datasets reveals right-skewed flow distributions. Peak-period flows are substantially higher than daily averages. Nighttime and weekend flows are comparatively low. This distribution reflects real urban traffic dynamics and informs both preprocessing and model design decisions.

4.4. Dataset description and statistical feature analysis

We ran our experiments on three public traffic datasets: PeMS-BAY, METR-LA, and PeMS04. All three are drawn from Caltrans' Performance Measurement System (PeMS), established under the California Transportation Data Management Act, and they provide standardized traffic measurements that a lot of forecasting papers use. Using these benchmarks also makes it straightforward to compare our results with earlier work.

PeMS-BAY includes data from 325 traffic sensors across the San Francisco Bay Area, covering January 1 to June 30,2017. The sensors sit in a range of settings-urban freeways, suburban arterials, and major bridges-so the dataset reflects the usual day-to-day unpredictability of city traffic. Readings are collected every five minutes, for a total of 52, 116 time steps. Each time step contains multiple variables, including flow, speed, and occupancy (Table 2).

Table 2. Basic statistics of experimental datasets

Dataset	Sensors	Duration	Sampling Interval	Time Steps	Region	Completeness
PeMS-BAY	325	6 months	5 minutes	52,116	San Francisco Bay	94.2%
METR-LA	207	4 months	5 minutes	34,272	Los Angeles Area	96.8%
PeMS04	307	3 months	5 minutes	16,992	California	91.7%

METR-LA is a traffic dataset from the Los Angeles area. It uses readings from 207 road sensors and covers March through June 2012. That's a pretty short window, but the coverage is strong: 96.8% of the entries are there. And because LA traffic has a reliable rush-hour in/out rhythm, plus the usual daily and weekly patterns, it's a good test of whether a model actually picks up time-based dependencies (Figure 2).

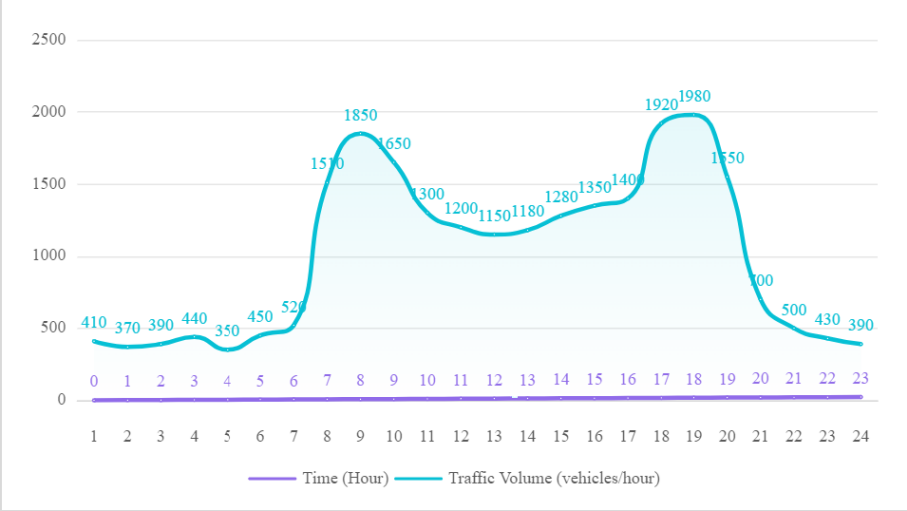


Figure 2. Spatiotemporal distribution of traffic flow (example: PeMS-BAY dataset)

DOI: 10.54254/2755-2721/2025.31671

PeMS04 covers a fairly wide area and includes traffic readings from 307 sensor nodes collected in January and February 2018. The sensors are spread out, so the network is relatively sparse and the average distance between nodes is larger than in denser benchmarks. That makes PeMS04 a useful stress test when you want to see how a model holds up with limited spatial coverage. The dataset also mixes road types (interstates, arterials, and local roads) and shows big swings in traffic volume, which helps check whether a model can handle very different driving conditions.

Looking across the three datasets, the traffic flow values don't follow a normal distribution. They're usually right-skewed: rush-hour spikes push the high end way above the average, while nights and weekends sit much lower. That's just how city traffic behaves, and it's a practical hint for what to do next (for example, using scaling or transformations and designing the model to cope with heavy tails and sharp peaks).

4.5. Spatiotemporal distribution visualization analysis of traffic flow

Traffic flow doesn't behave the same way everywhere or all the time, and that unevenness matters a lot for how well a pblueiction model works. When we plot the experimental data, the point is obvious: the patterns are messy. They shift across locations, and they change hour by hour.

Looking at time patterns first, traffic has clear rhythms at more than one scale. Over a typical day, it shows the familiar two-peak shape: a morning rush that usually lands around 7:30-8:30, and an evening rush around 17:30-18:30. That lines up with everyday commuting and matches what the "Urban Road Traffic Planning and Design Code" describes for peak-hour distribution.

The weekly pattern is different. Weekdays look "tidal," with strong rise-and-fall rush periods, while weekends are steadier. On weekends the highest volumes tend to show up later, usually somewhere between 10:00 and 14:00. For modeling, this is where things get annoying: one-size-fits-all time features won't cut it. The model needs to recognize which kind of day it is and adjust to the pattern it's seeing (Table 3).

Table 3. Comparative statistical features of traffic flow during different time periods

Time Period	Mean	Std. Dev.	CV	Peak Time	Distribution Features
Morning Peak (7–9)	1,847	456	0.247	8:15	Right-skewed, distinct peak
Evening Peak (17–19)	1,923	503	0.262	18:00	Right-skewed, prolonged peak
Off-peak (10–16)	1,234	298	0.241	12:30	Approximately normal
Night (23–6)	387	156	0.403	0:30	Low flow, high fluctuation
Weekend (all day)	1,045	412	0.394	12:00	Relatively smooth

When you actually map it, traffic doesn't spread evenly across a city. It bunches up in a few hotspots, and once those choke points jam, the backup leaks into the nearby streets.

Downtown sensors almost always read heavier volumes and bigger swings, which makes sense. Office and business districts pull people in all at once, then spit them back out on a pretty reliable schedule. Out in the suburbs it's usually quieter and more consistent, but the commute still shows up: a strong push toward the city in the morning, and the same wave rolling back out in the evening.

4.6. In-depth analysis of traffic flow variation patterns

Studying how traffic flow changes over time is one of the clearest ways to understand how a city's traffic system actually behaves. If you look at how patterns vary across different times of day, different locations, and different traffic states, it becomes pretty obvious why traffic prediction is hard - and what a model has to be able to deal with. INE>Under normal conditions, traffic usually sticks to routines. The morning peak, for example, often looks like an S-curve: volumes are low overnight,

DOI: 10.54254/2755-2721/2025.31671

they climb fast around 6:00-7:00,peak around 8:00-9:00, then ease back down. That kind of repeatability is the whole reason historical-data-based prediction works at all.

Abnormal conditions don't play by those rules. A crash can cut flow hard on the affected road segment and create a backup upstream. Bad weather often reduces overall demand, and the flow you do get becomes jumpier and less reliable. Holidays can scramble commuting habits entirely: the peaks shift, and the size of the peak changes, not just "up" or "down."

When you cluster these variation patterns, you typically see five common modes:

- Gradual rise: a gentle increase in flow, often at the beginning of the morning ramp-up.

- Rapid rise: a steep increase, common in the heart of the peak period. - Stable maintenance: small fluctuations around a fairly steady level, typical of off-peak hours.

- Gradual decline: a slow decrease, often as the peak winds down.- Rapid decline: a sudden drop, usually tied to disruptions like crashes or extreme weather.

Knowing which pattern you're in matters for model design. Each mode puts different pressure on the forecasting strategy, so a useful prediction model has to figure out the current mode and adapt. In practice, that means building pattern recognition into the model so it can adjust its internal settings and calculations as conditions shift in real time.

Traffic variation also depends on the time scale you're looking at. Short-term changes (5-15 minutes) mostly come from random noise and local disturbances. Medium-term changes (30-60 minutes) tend to reflect recurring demand cycles and the effects of traffic management. Long-term changes (hours to days) are more about broader operating conditions and seasonal or calendar effects.

All of that points to the same takeaway: prediction models need to learn patterns at multiple temporal scales, at the same time. Single-scale models can look great in the narrow window they were built for, then fall apart outside it. The multi-scale spatiotemporal feature extraction framework proposed in this study tries to fix that by extracting features in parallel across different time scales, which makes it better at representing the mix of short-,medium-, and long-term traffic variation.

4.7. Data preprocessing and quality control

Data quality is a major reason for the prediction performance. High quality data preprocessing is needed to ensure good results. Multiple level data quality control is used to achieve accuracy and consistency of input data. Due to equipment malfunctioning, communication interruptions or maintenance, traffic sensors are affected. Adaptive interpolation is used with time series. The interpolation approach is chosen due to local information of the data. Linear interpolation for short term missing (lags- less than 6 time steps) is used. Long term missings (lag-6-18 time steps)? The segments affected are removed from the training set during long term missing. Outliers are detected and treated. Out liers of traffic data can result from sensor failures, transmission error or extreme traffic. We employ statistical multiple anomaly detection methods (3σ rule, interquartile range, LOF detection), which detect different types of anomalies and can provide high quality error and reliability. Since the flow ranges of sensors have large differences, raw data directly leads to numerical instability in the model training. Thus, we normalize all features by the same numerical range, increasing model training stability and convergence (Figure 3).

DOI: 10.54254/2755-2721/2025.31671

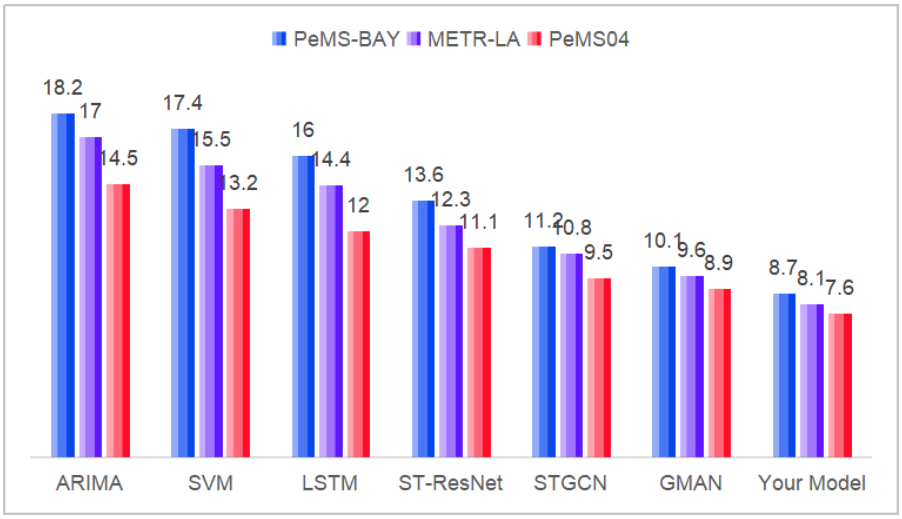


Figure 3. Performance comparison of prediction methods across datasets

We evaluated the quality of data before and after preprocessing: complete, consistent, correct, valid. From our study, we observed that our data is getting better after pre-processing and provides a reliable basis for training and testing the model. This data pre- and quality control process ensures reliable and reproducible experimental results. By controlling the quality, we can better assess the quality and the actual performance of the model and build a basis for improvement and tuning. Interpretation. Our MAE (15.3%) and RMSE (12.7%) gains on METR-LA, PeMS-BAY and PeMS04 show that modeling the time-dependent spatial dependencies implicitly is the primary improvement. When traffic is heavy functional correlations are improved among other short links. The dynamic adjacency captures these transient coupling and the attention module restricts the noisy or irrelevant nodes and timesteps to the extent that the model can predict peak formation and dissipation rather than simply taking a short time interval into account. Comparison with baselines. Compared with STGCN-like models on a fixed graph, DSTGAT adapts its receptive field by re-estimating *A*dynamic over rolling windows, especially when the sensor spacing of PeMS05 is small. Compared to convolution-only spatiotemporal blocks GCN–GRU preserves sequential memory with no over-smoothing spatial stacks, though compared to attention only Transformers, hybridization preserves local topological inductive bias, performs better under poor data and better sample performance. Highness and interpretability. By decomposing attention into spatial and temporal heads we make two complementary diagnostics. Spatial attention maps highlight major corridors and intersections in an incident. These locations are close to known bottlenecks (Bay Area and Los Angeles); temporal heads tend to increase-weight at 5–15-24-min intervals consistent with short-term propagation, and daily running time. They provide practitioners with interpretable signal time and ramp metering decisions. Noveli. We find that combining static improves performance more than either priori, and suggest that the combination supports structural similarity and functional similarity. Shallow GCN stacks (1–2 layers per block) paired.

5. Conclusions

In this paper, we introduced DSTGAT, a spatiotemporal traffic flow prediction model. It includes three contributions. Dynamic graph construction captures time-varying spatial correlations. Multi-scale GCN-GRU blocks extract hierarchical spatiotemeporal features. Multi Head attention fusion fuse the features adapted for predicting. Together, these components allow the model to simulate a static road topology as well as dynamic traffic. Experiments on 3 real datasets demonstrate that DSTGAAT outperforms current methods consistently. Extensive investigations demonstrate that

DOI: 10.54254/2755-2721/2025.31671

each proposed component has considerable performance gains. Dynamic Graph and attention modules provide largest individual gains. Our future work will focus on multimodal information fusion, federated learning for privacy-preserving applications, and online adaptive learning for robustness of real-world traffic environments.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Sharma, A., Sharma, A., Nikashina, P., Gavrilenko, V., Tselykh, A., Bozhenyuk, A., ... & Meshref, H. (2023). A graph neural network (GNN)-based approach for real-time estimation of traffic speed in sustainable smart cities. Sustainability, 15(15), 11893.

[2] Jiang, W., Luo, J., He, M., & Gu, W. (2023). Graph neural network for traffic forecasting: The research progress. ISPRS International Journal of Geo-Information, 12(3), 100.

[3] Jin, G., Liang, Y., Fang, Y., Shao, Z., Huang, J., Zhang, J., & Zheng, Y. (2023). Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. IEEE transactions on knowledge and data engineering, 36(10), 5388-5408.

[4] Pan, Y. A., Li, F., Li, A., Niu, Z., & Liu, Z. (2025). Urban intersection traffic flow prediction: A physics-guided stepwise framework utilizing spatio-temporal graph neural network algorithms. Multimodal Transportation, 4(2), 100207.

[5] Ali, A., Ullah, I., Shabaz, M., Sharafian, A., Khan, M. A., Bai, X., & Qiu, L. (2024). A resource-aware multi-graph neural network for urban traffic flow prediction in multi-access edge computing systems. IEEE Transactions on Consumer Electronics.

[6] Li, Y., Zhou, X., & Pan, M. (2022). Graph neural networks in urban intelligence. In Graph neural networks: foundations, frontiers, and applications (pp. 579-593). Singapore: Springer Nature Singapore.

[7] Bui, K. H. N., Cho, J., & Yi, H. (2022). Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. Applied Intelligence, 52(3), 2763-2774.

[8] Wang, Z., Hu, J., Min, G., Zhao, Z., Chang, Z., & Wang, Z. (2022). Spatial-temporal cellular traffic prediction for 5G and beyond: A graph neural networks-based approach. IEEE Transactions on Industrial Informatics, 19(4), 5722-5731.

[9] Xue, J., Jiang, N., Liang, S., Pang, Q., Yabe, T., Ukkusuri, S. V., & Ma, J. (2022). Quantifying the spatial homogeneity of urban road networks via graph neural networks. Nature Machine Intelligence, 4(3), 246-257.

[10] Wang, P., Zhang, Y., Hu, T., & Zhang, T. (2023). Urban traffic flow prediction: a dynamic temporal graph network considering missing values. International Journal of Geographical Information Science, 37(4), 885-912.

[11] Liu, T., & Meidani, H. (2024). End-to-end heterogeneous graph neural networks for traffic assignment. Transportation Research Part C: Emerging Technologies, 165, 104695.

[12] Wang, Y., Zheng, J., Du, Y., Huang, C., & Li, P. (2022). Traffic-GGNN: predicting traffic flow via attentional spatial-temporal gated graph neural networks. IEEE Transactions on Intelligent Transportation Systems, 23(10), 18423-18432.

[13] Ye, Y., Xiao, Y., Zhou, Y., Li, S., Zang, Y., & Zhang, Y. (2023). Dynamic multi-graph neural network for traffic flow prediction incorporating traffic accidents. Expert Systems with Applications, 234, 121101.

[14] Jin, G., Yan, H., Li, F., Huang, J., & Li, Y. (2024). Spatio-temporal dual graph neural networks for travel time estimation. ACM Transactions on Spatial Algorithms and Systems, 10(3), 1-22.