Spatio-Temporal Deep Learning for Traffic Forecasting: Comparative Study of LSTM & DCRNN

Elyann Soto[#1], Martin Gasevski[#2], Sterling Walker[#3], Krishna Sharma[#4]

[#]*Computer and Information Science, University of Michigan-Dearborn*
105 Computer and Information Science Building, 4901 Evergreen Road, Dearborn, MI 48128, USA
[1]elyannsm@umich.edu,[2]voidtest@umich.edu, [3]walkersw@umich.edu, [4]krishsh@umich.edu

***Abstract*— Short-term traffic forecasting is essential for routing, congestion mitigation, and intelligent transportation management across large-scale urban road networks. While temporal deep learning models such as Long Short-Term Memory networks capture sequential patterns effectively, they do not explicitly model how congestion propagates through a spatially connected sensor graph. Graph-based spatio-temporal models, such as the Diffusion Convolutional Recurrent Neural Network, are designed to leverage road-network structure to improve long-horizon predictions. In this study, we conduct a controlled comparison between a Sequence-to-Sequence LSTM baseline and two DCRNN variants on the PEMS-BAY dataset, using identical preprocessing, forecasting horizons, and training configurations. All models receive 12 historical time steps as input and predict the next 12 steps, evaluated at 5-30-minute horizons using MAE, RMSE, and MSE. Contrary to expectations from prior work, our results show that a well-trained LSTM consistently outperforms both a speed-only DCRNN and a graph-based DCRNN with adjacency information. These findings highlight the strength of simple temporal baselines and reveal the sensitivity of spatial models to adjacency construction, normalization, and training procedures, underscoring the importance of careful graph design when applying DCRNN-like architectures to real-world traffic forecasting tasks.**

I. INTRODUCTION

Accurate short-term traffic forecasting plays a central role in modern intelligent transportation systems. Reliable predictions of future traffic speed enable dynamic route guidance, congestion mitigation, adaptive traffic signal control, and safety-critical decision making. As cities continue to experience growth in travel demand, forecasting models must scale to dense sensor networks and provide robust multi-step predictions.

Classical time-series models, including ARIMA, VAR, and SVR, have historically been applied to traffic forecasting, but they assume linearity, treat each sensor independently, and fail to capture the nonlinear spatial interactions present in real highway networks. Deep learning approaches have addressed many of these limitations. Recurrent neural networks and Long Short-Term Memory networks can learn complex temporal dependencies over long sequences, making them natural candidates for sequential traffic modeling. However, purely temporal models lack explicit mechanisms to capture spatial propagation, such as how downstream congestion affects neighboring segments.

Graph-based neural networks bridge this gap by modeling traffic as a spatio-temporal process defined over a road network graph. Among these, the Diffusion Convolutional Recurrent Neural Network integrates directed graph diffusion with recurrent modeling, enabling it to learn how traffic disturbances flow across connected sensors. These spatial-temporal architectures have shown strong performance when the underlying road graph is well constructed and when training procedures are tuned appropriately.

In this study, we directly compare a Sequence-to-Sequence LSTM baseline with the DCRNN architecture, training both models exclusively on the PEMS-BAY dataset, a large-scale and spatially complex freeway sensor network. Using a single dataset ensures that both architectures operate under the same data distribution, sensor density, and temporal patterns, allowing a clean evaluation of how temporal-only and spatial-temporal models differ in their ability to capture traffic dynamics.

Our work contributes in five key ways. First, we implement a Seq2Seq LSTM model on PEMS-BAY to establish a strong temporal baseline for multi-step forecasting. Second, we implement a DCRNN under matched preprocessing steps, sequence lengths, and training settings, enabling a controlled architectural comparison. Third, by evaluating all models under

identical experimental conditions and on a single dataset, we provide a fair and transparent benchmark. Fourth, our empirical findings show that a properly tuned LSTM can outperform both DCRNN variants, including the graph-based version, which contrasts with prior literature and highlights the sensitivity of spatial models to adjacency construction and training details. Finally, we identify several potential factors contributing to the weak performance of the adjacency-based DCRNN and outline opportunities for improving graph-based modeling in future work.

This introduction establishes the motivation, context, and purpose of the study and aligns with the methodology and empirical findings presented in the remainder of the paper.

II. RELATED WORK

This section reviews prior work on temporal, graph-based, and hybrid deep learning models for traffic forecasting.

A. *Time-series approaches*

LSTM-based Sequence-to-Sequence models [1] introduced the idea of mapping past sequences to future sequences using an encoder–decoder design. The encoder compresses historical traffic readings into a context vector, while the decoder generates future predictions step by step. LSTMs effectively handle vanishing gradients and can learn medium- and long-range temporal patterns such as daily congestion cycles. However, because the entire input sequence is reduced to a single vector, important temporal information may be lost for long horizons. These models also process data sequentially, which slows training, and they treat each sensor independently, ignoring spatial relationships within a traffic network.

Transformer-based architectures [11] eliminate recurrence entirely and instead use self-attention to capture dependencies between all timesteps in parallel. This allows the model to learn long-range temporal relationships more efficiently than RNNs. Transformers scale well and can model complex interactions across multivariate traffic sequences. Their limitations include quadratic computational cost with respect to sequence length and the absence of built-in spatial structure, requiring additional mechanisms to model road-network topology. Despite these challenges, attention-based models significantly advanced time-series forecasting through their ability to model global temporal dependencies.

CNN-based “Traffic as Images” approaches [10] convert time–space traffic data into two-dimensional matrices and apply stacked convolution layers to jointly learn local and global spatiotemporal features. This method is computationally efficient and generalizes well across large road networks. However, transforming irregular sensor layouts into image grids distorts true road topology, and CNNs lack memory mechanisms for long-term temporal dependencies. These limitations inspired later graph-based models, which preserved genuine road-network structure rather than imposing a grid.

B. *Graph-based approaches*

DCRNN (Diffusion Convolutional Recurrent Neural Network) [3] models traffic flow through a directional diffusion process on the road graph. The diffusion convolution captures upstream and downstream propagation of congestion, while a recurrent GRU-like structure models temporal dependencies. The model demonstrated strong long-horizon performance on METR-LA and PEMS-BAY. Its main limitations include the need for a predefined adjacency matrix and higher computational cost due to graph operations and recurrent processing.

STGCN (Spatio-Temporal Graph Convolutional Network) [4] combines one-dimensional temporal convolutions with graph convolutions arranged in a block structure. By removing recurrence, STGCN enables faster and fully parallel training, while still leveraging graph structure for spatial modeling. However, its temporal receptive field is limited by the convolution window size, and its fixed undirected adjacency matrix prevents modeling directional traffic flow.

T-GCN (Temporal Graph Convolutional Network) [5] integrates a basic graph convolution with a GRU, providing a simple mechanism for learning spatial and temporal features jointly. Although lightweight and fast to train, it does not incorporate direction and struggles with complex networks or long-horizon forecasting tasks. As a result, it typically underperforms DCRNN and STGCN.

C. *Attention/Transformers:*

AASTGCN (Attention-Based STGCN) [6] enhances STGCN by introducing spatial and temporal attention mechanisms, enabling the model to focus on the most relevant sensors and time steps. It also models recent, daily, and weekly patterns separately, capturing periodic behaviors. While effective, ASTGCN has higher complexity, increased training time, and a greater risk of overfitting.

AGCRN (Adaptive Graph Convolutional Recurrent Network) [7] learns both node embeddings and adjacency matrices dynamically, allowing it to discover hidden spatial

relationships beyond simple distance-based connections. This flexibility often improves forecasting accuracy. However, the learned graph can be difficult to interpret, and training becomes more challenging because the model must learn spatial and temporal patterns simultaneously.

Hybrid STGCN variants [8], [9] extend the core STGCN architecture by incorporating multiple adjacency matrices or additional data sources, such as navigation queries or vehicle volumes. These models capture richer spatial dependencies that cannot be explained by sensor speeds alone. However, they depend on external datasets that may be unavailable in practice and increase the complexity of training and tuning.

More advanced hybrids combine distance-, correlation-, and travel-time-based adjacency definitions using learned weights or Transformer-style components. These architectures achieve strong performance but are computationally expensive and less interpretable, making them less suitable for smaller deployments or use cases requiring transparency.

D. Strengths / Limitations

Traffic forecasting research can be organized into three major families of models.

The first family consists of temporal-only models, including CNN-based "Traffic as Images" approaches [10], Sequence-to-Sequence LSTMs [1], and Transformer models [11]. These models are computationally efficient and easy to train, but they do not incorporate real road-network structure. As a result, they struggle to model interactions between sensors and show reduced accuracy for long-term predictions.

The second family comprises graph-based models such as DCRNN [3], STGCN [4], and T-GCN [5]. By explicitly modeling spatial dependencies, these methods achieve substantially better multi-step forecasting performance on standard benchmarks like METR-LA and PEMS-BAY. Their main drawbacks are reliance on a fixed adjacency matrix and increased computational requirements.

The third family includes attention-based, adaptive-graph, and hybrid models [6]–[9], which dynamically learn spatial importance or integrate multiple adjacency types. While these models often outperform earlier graph techniques, they introduce higher model complexity, training cost, and reduced interpretability.

Compared with these families, our project focuses on a clear and meaningful contrast between a simple LSTM baseline [1] and the graph-based DCRNN [3]. The LSTM represents the strengths and limitations of temporal-only modeling: it is easy to train and handles short-term patterns well but lacks spatial awareness. In contrast, DCRNN incorporates directed spatial modeling, diffusion dynamics, and improved long-term forecasting, exemplifying the advantages of spatio-temporal graph learning. This comparison mirrors the broader shift in the literature from purely temporal models to graph-based approaches. Although attention and adaptive-graph models may provide further improvements, they introduce significantly more complexity and computational cost. Thus, our study focuses on the essential question: How much improvement comes from adding graph structure to a basic temporal model? Prior work strongly suggests that graph-based models outperform LSTM-style baselines, particularly for multi-step forecasting and larger sensor networks.

Building on this foundation from prior work, the next section introduces the PEMS-BAY dataset used in our comparative evaluation.

III. DATASET DESCRIPTION

The PEMS-BAY traffic dataset is a publicly available real-world freeway speed dataset collected by the Caltrans Performance Measurement System (PeMS) across the Bay Area, California. It contains measurements from 325 sensors distributed throughout the regional highway network, covering six months of recorded continuous speed data from 2017, sampled at 5-minute intervals starting from January 1 to June 30 . Each sensor records average traffic speed, resulting in a multivariate spatiotemporal time series data widely used for evaluating traffic forecasting models such as GNNs, RNNs and diffusion models. The dataset also provides the corresponding road network topology through an adjacency graph derived from real roadway distances, enabling research on graph-structured traffic prediction under realistic conditions. Prior to use, the data were cleaned to remove empty or erroneous readings, and all sensor measurements were normalized using z-score normalization,

$$z = \frac{Xi - X\mu}{\sigma} \quad (1)$$

to ensure a consistent scale across sensors. Following preprocessing, the dataset was partitioned into 70% training, 10% validation, and 20% testing, consistent with standard practices in spatiotemporal forecasting.

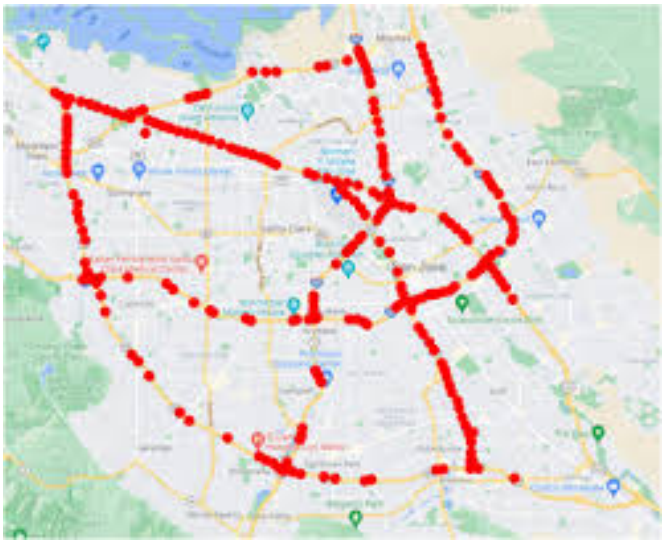

Fig. 3.1. Spatial distribution of loop-detector sensors in PEMS-BAY dataset.

The map in Fig. 3.1 illustrates the spatial distribution of traffic sensors deployed across the San Francisco Bay Area in the PEMS-BAY dataset. Each red marker represents an individual loop-detector sensor positioned along major freeways and urban corridors. The dense and widespread placement of these sensors captures regional traffic dynamics with high spatial granularity, enabling accurate modeling of both local and network-level traffic patterns.

IV. OUR MODEL ARCHITECTURE

With the dataset defined, we now describe the model architectures and training methodology used to evaluate temporal-only and spatial–temporal forecasting approaches.

A. *Baseline Model: LSTM*

The baseline model adopts a sequence-to-sequence Long Short-Term Memory network trained on the PEMS-BAY dataset. The LSTM receives 12 historical time steps (60 minutes) as input and predicts the next 12 time steps (60 minutes).

The model architecture comprises two stacked LSTM layers with 64 hidden units each, followed by a fully connected (dense) output layer. A dropout layer with rate = 0.2 is applied between recurrent layers to reduce overfitting. The input data are normalized using z-score normalization across sensors to stabilize training.

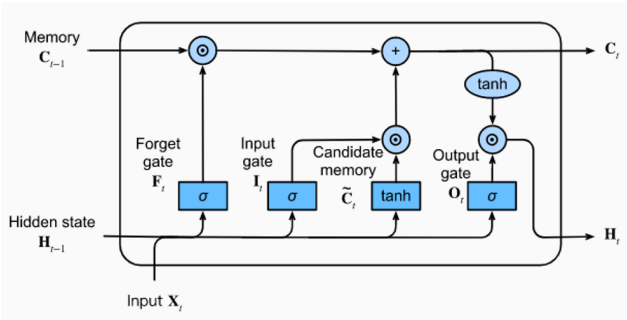

Fig. 4.1. Baseline Sequence-to-Sequence LSTM architecture for traffic speed forecasting.

Fig. 4.1 presents the baseline architecture of the LSTM model. In this structure, H_{t-1} represents the hidden state from the previous time step, which is propagated into the LSTM cell, while C_{t-1} denotes the corresponding memory (cell state) carried forward through the recurrent sequence. The input X_t corresponds to the sensor speed value at time step t, serving as the current observation provided to the LSTM cell. Together, these components enable the LSTM to capture temporal dependencies and learn long-term patterns within the traffic speed time series.

The LSTM was selected as a baseline due to its strength in modelling temporal dependencies in sequential data. Through gating mechanisms, it effectively mitigates the vanishing-gradient problem and learns long-range correlations in time-series sequences, all are properties that make it suitable for traffic flow forecasting where temporal dynamics dominate.

The network was implemented in PyTorch and trained with the Adam optimizer using a learning rate of 0.001, batch size of 64, and Mean Squared Error (MSE) as the loss function. Training proceeded for 20 epochs.

B. *Advanced Model: DCRNN*

To model both temporal and spatial dependencies, we used a Diffusion Convolutional Recurrent Neural Network (DCRNN) introduced by Li et al. [3]. Unlike the LSTM, which treats each sensor as independent, DCRNN represents sensors as nodes in a directed graph whose weighted edges encode spatial proximity based on a Gaussian distance kernel.

The DCRNN uses the same 12-step input and 12-step forecasting horizon for comparability with the LSTM baseline. The bidirectional diffusion process allows information to flow between upstream and downstream nodes, capturing how congestion propagates along road segments.

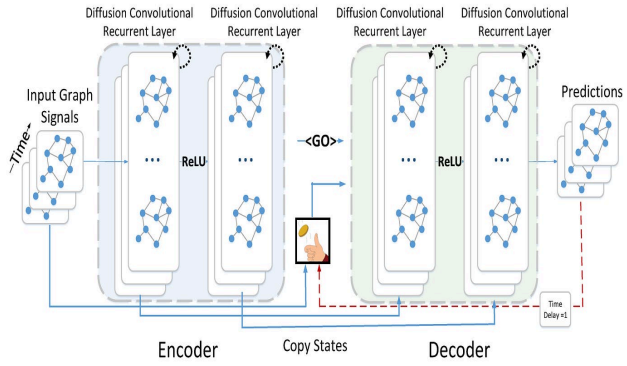

Fig. 4.2. Encoder–decoder structure of the DCRNN architecture with diffusion convolution and scheduled sampling.

Fig. 4.2 illustrates the overall architecture of the Diffusion Convolutional Recurrent Neural Network (DCRNN). The model follows an encoder-decoder framework designed to capture both the temporal dependencies and the spatial diffusion dynamics present in traffic sensor networks. In the encoder, a sequence of historical graph signals is passed through stacked Diffusion Convolutional Recurrent (DCR) layers, where each layer integrates recurrent processing with bidirectional graph diffusion to model spatial correlations across the road network. The encoder progressively updates its hidden states using the learned diffusion process, effectively capturing how traffic conditions propagate through the connected sensor graph.

At the start of decoding, a special <GO> token initializes the decoder, which mirrors the encoder structure with its own stack of DCR layers. The decoder receives the hidden states copied from the encoder and generates future graph signals autoregressively, one time step at a time. During this process, the decoder uses scheduled sampling (represented by the coin-flip symbol), gradually transitioning from ground-truth inputs to its own predicted outputs to improve long-term forecasting stability. The final predictions represent the future traffic states across all sensors in the network.

The DCRNN was trained using the PEMS-BAY dataset, which contains 325 sensors and a dense spatial topology. The model leverages both the adjacency matrix and temporal recurrence to jointly learn spatial-temporal correlations. To ensure comparability with the LSTM baseline, all training procedures, data loading, batching, and evaluation steps were implemented in PyTorch. The implementation was adapted from publicly available DCRNN reference code and restructured to operate within a unified PyTorch workflow.

The training configuration matched the baseline setup and used the Adam optimizer with a learning rate of 0.001, a batch size of 64, and Mean Squared Error (MSE) as the loss function. Early stopping based on validation loss was used to prevent overfitting. Model performance was evaluated using MAE and RMSE across forecasting horizons of 5, 10, 15, and 30 minutes.

C. *Training Setup*

Both the LSTM and DCRNN models were trained under controlled and comparable conditions to ensure a fair evaluation. All experiments were conducted on GPU-accelerated environments, and the full training pipeline, data loading, and evaluation procedures were implemented in PyTorch. The DCRNN implementation was adapted from publicly available reference code to ensure consistency while allowing integration into the PyTorch-based workflow.

The Adam optimizer was used for both models with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1e\text{-}8$. Training was performed using a batch size of 64, with models trained for up to 50 epochs and early stopping triggered based on the lowest validation loss.

All input features were normalized and partitioned according to the preprocessing described in Section III. Both models used an input sequence length of 12 steps (representing 60 minutes of past data) to forecast the next 12 steps. During training, data was shuffled within each batch to promote generalization while preserving the temporal order within each sequence.

Training was performed on NVIDIA Tesla T4 GPUs (16 GB VRAM). On this hardware, the LSTM model required approximately 15 minutes to train, while the graph-based DCRNN model required roughly 45 minutes due to the additional computational cost of diffusion graph operations.

D. *Loss & Metrics*

The Mean Squared Error (MSE) was adopted as the primary loss function for both models during training, emphasizing the minimization of large deviations between predicted and observed traffic speeds:

$$'MSE = \frac{1}{n}\sum i = 1^{n}(y_i - \hat{y}_i)^2, \tag{2}$$

where y_i and $\hat{y}_i$ denote the actual and predicted values, respectively. In evaluation, the models' performances were compared using two complementary metrics that provide a holistic view of predictive quality:

$$\text{MAE} = \frac{1}{n}\sum_i |y_i - \hat{y}_i|, \quad \text{RMSE} = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2}, \tag{3}$$

Mean Absolute Error (MAE): This metric quantifies average absolute deviation and is robust to outliers.

Root Mean Squared Error (RMSE): This metric penalizes larger errors more heavily, making it sensitive to sharp traffic anomalies.

V. OUR EXPERIMENTS

A. *Environment and Setup*

All experiments were conducted using Google Colab with GPU acceleration, leveraging the PyTorch deep learning framework for model development and training. Both the LSTM and DCRNN models were implemented by adapting components from the official DCRNN repository to ensure consistency with established benchmark practices. The training pipeline, data preprocessing procedures, and evaluation routines were fully implemented in PyTorch to enable flexible experimentation and efficient execution in a cloud-based environment. This setup provides a reproducible and scalable platform for training spatiotemporal forecasting models on the PEMS-BAY dataset.

B. *Training Dynamics*

All metrics were computed at multiple forecasting horizons (5, 10, 15, and 30 minutes) to evaluate short-term versus long-term temporal generalization. To visualize convergence, the training and validation losses were recorded per epoch, demonstrating that both models stabilized without overfitting.

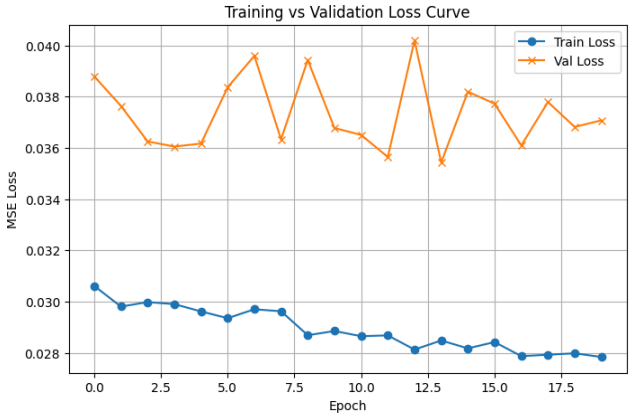

Fig. 5.1 . LSTM training/validation loss (5 min)

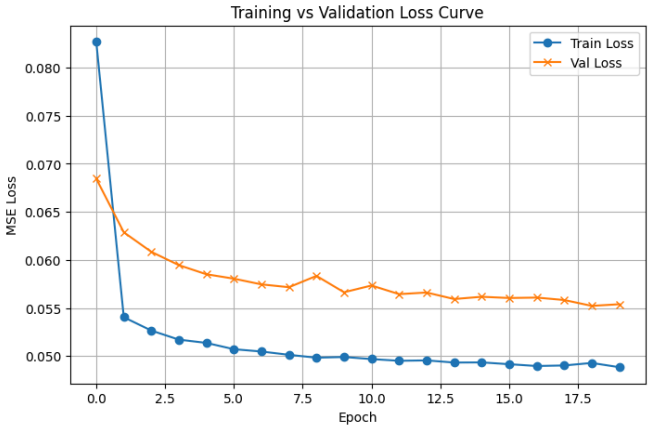

Fig. 5.2. DCRNN no-adj training/validation loss (5 min)

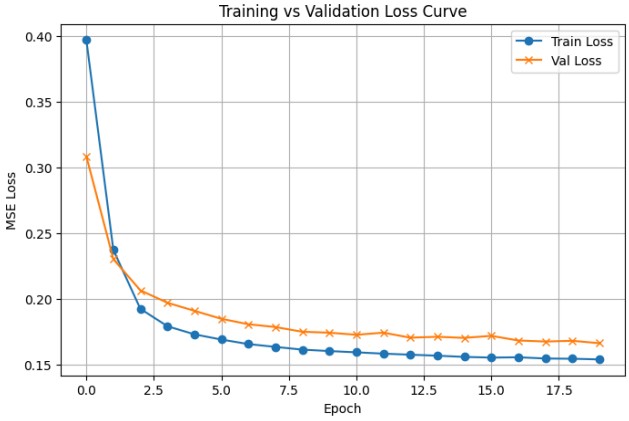

Fig. 5.3 . DCRNN with-adj training/validation loss (5 min)

Figures 5.1–5.3 compare training and validation loss curves for LSTM and both DCRNN variants on the 5-minute horizon. In all three cases, the validation loss closely follows the training loss and flattens after a small number of epochs, which indicates stable training without severe overfitting. However, the absolute loss levels differ: the LSTM curve settles at the lowest plateau, DCRNN without adjacency converges to a higher plateau, and DCRNN with adjacency converges to the highest plateau.

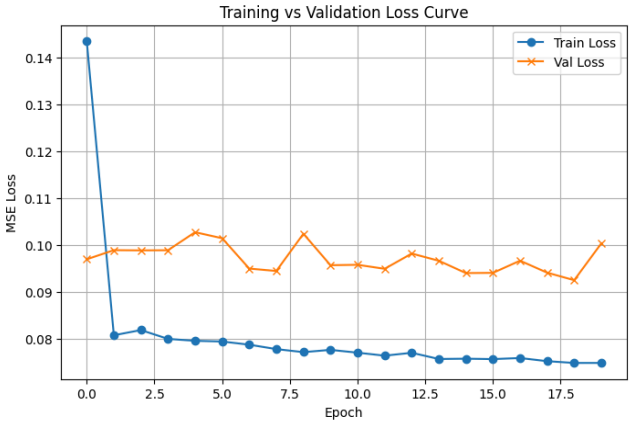

Fig. 5.4. LSTM (15 min)

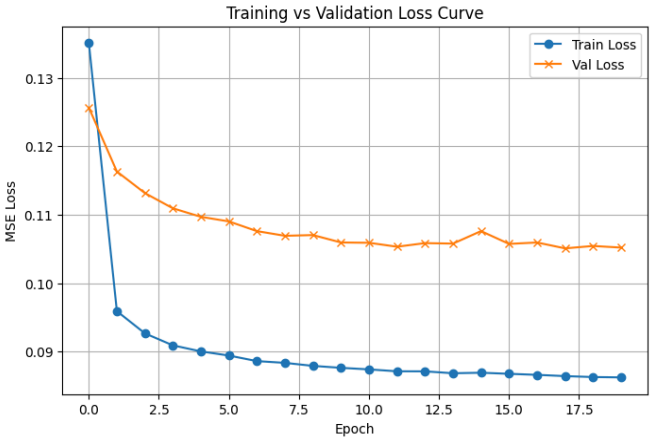

Fig. 5.5. DCRNN no-adj (15 min)

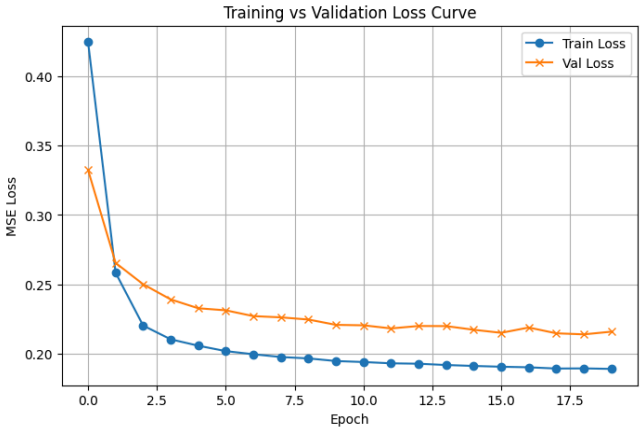

Fig. 5.6. DCRNN with-adj (15 min)

Similar behavior is seen for the 15 minutes window in Figures 5.4–5.6 above, which supports the quantitative results in Tables 1 and 2 below.

C. Per-sensor error analysis: Best vs. Worst sensors

We also examined how errors vary across individual sensors. For each DCRNN configuration and horizon, we computed per-sensor MAE on the test set and plotted the distribution ((Figures 5.7 and 5.8). A small set of sensors, such as #148, #117, and #177, consistently appear among the best (lowest MAE), while others like #134, #146, and #293, often rank among the worst.

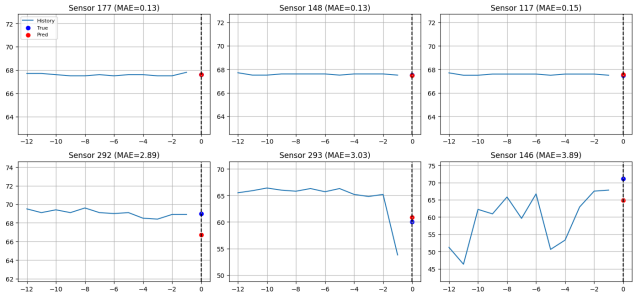

Fig. 5.7. Best and worst predicted sensors for DCRNN with adjacency at the 5-minute horizon.

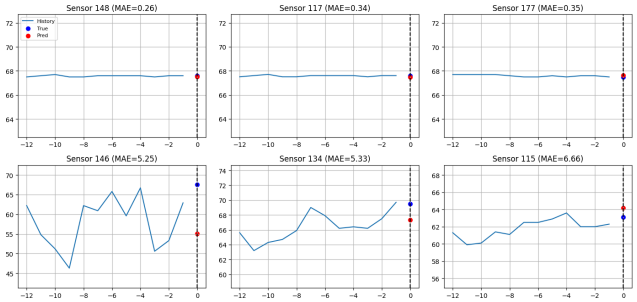

Fig. 5.8. Best and worst predicted sensors for DCRNN with adjacency at the 15-minute horizon.

This suggests that some locations are inherently easier to predict, perhaps because they show smoother, more regular traffic, while others exhibit more volatile behavior that all models struggle to capture. Importantly, the ranking of difficult sensors is similar across DCRNN variants, which indicates that the relatively poor performance of DCRNN with adjacency is not limited to a few outliers, but is spread across the network.

VI. OUR COMPARATIVE RESULTS

Table 1 reports the training and validation losses for all three models across the four prediction horizons (5, 10, 15, and 30 minutes) on the PEMS-BAY dataset. The three models are: (1) an LSTM baseline using only sensor speed, (2) a DCRNN variant without adjacency (“DCRNN no_adj”) that also uses only speed, and (3) a DCRNN variant with adjacency (“DCRNN with_adj”) that uses both sensor speed and graph structure.

Table I. Train/Validation loss by horizon and model (PEMS-BAY)

Time	LSTM (speed only)		DCRNN (speed-only)		DCRNN (speed & loc)	
Horizon	Train Loss	Val Loss	Train Loss	Val Loss	Train Loss	Val Loss
5 min	0.0285	0.0354	0.0493	0.0552	0.1538	0.1661
10 min	0.0481	0.0633	0.0682	0.0802	0.1714	0.1887
15 min	0.0748	0.0925	0.0864	0.1051	0.1893	0.2138
30 min	0.1546	0.1795	0.1373	0.1782	0.2375	0.2742

Across all horizons, the LSTM achieves the lowest training and validation losses. For example, at the 5-minute horizon, the LSTM converges to a training loss of 0.0285 and a validation loss of 0.0354, while DCRNN no_adj reaches 0.0493 and 0.0552, and DCRNN with_adj reaches 0.1538 and 0.1661. The same pattern holds at 10, 15, and 30 minutes: LSTM consistently has the smallest losses, DCRNN no_adj has slightly higher losses, and DCRNN with_adj has the highest.

Table 2 summarizes the corresponding test results in terms of MSE, MAE, and RMSE. The LSTM baseline again provides the best performance at every horizon. At 5 minutes, the LSTM achieves the best test MAE and RMSE, the DCRNN no_adj model is worse, while DCRNN with_adj is significantly worse. As the forecasting horizon increases, all three models degrade, but their relative ranking remains unchanged. At 30 minutes, the LSTM still performs best, the DCRNN no_adj trails, and DCRNN with_adj has the largest errors.

Table II. Test by horizon and model (PEMS-BAY)

Horizon	Model	Test MSE	Test MAE	Test RMSE
5 min	LSTM	0.0426	0.8685	1.6603
	DCRNN no-adj	0.0737	0.9841	1.8501
	DCRNN with-adj	0.2209	1.7321	3.2884
10 min	LSTM	0.0668	1.0357	2.0803
	DCRNN no-adj	0.1031	1.1601	2.2525
	DCRNN with-adj	0.2446	1.8163	3.4893
15 min	LSTM	0.0925	1.204	2.4471
	DCRNN no-adj	0.1322	1.332	2.5764
	DCRNN with-adj	0.2713	1.9151	3.6935
30 min	LSTM	0.1628	1.5399	3.2467
	DCRNN no-adj	0.2217	1.744	3.3933
	DCRNN with-adj	0.3367	2.1368	4.1964

In short, based on these metrics, we observe a consistent ordering of performance across all horizons: LSTM is best, followed by DCRNN no_adj, with DCRNN with_adj performing worst on this dataset in our current configuration.

Because the LSTM and DCRNN no_adj both use only speed information, they provide a fair head-to-head comparison of temporal modeling ability. The LSTM has lower train loss, lower validation loss, and lower test errors at every horizon. The differences are modest but systematic, typically on the order of 10-15% higher MAE and RMSE for DCRNN no_adj. This shows that in our setup, simply replacing the LSTM architecture with a DCRNN architecture, without giving it a graph to use, does not improve performance. The DCRNN with adjacency, which should in principle benefit from additional spatial information, instead shows substantially higher errors, indicating that the way we constructed and used the adjacency matrix did not produce the expected gains.

VII. OUR DISCUSSION AND CONCLUSION

This work presented a controlled comparison between a Sequence-to-Sequence LSTM baseline, a DCRNN variant without adjacency information, and a DCRNN model incorporating a road-network adjacency matrix. Although prior studies on spatial–temporal graph models consistently report strong improvements over temporal baselines on benchmark datasets such as PEMS-BAY, our results did not reproduce this pattern. Across all forecasting horizons, the LSTM achieved the lowest training loss, validation loss, and test errors. The speed-only DCRNN produced slightly higher errors, while the adjacency-based DCRNN showed the weakest performance overall.

These outcomes suggest that while LSTMs can serve as strong temporal baselines, the performance of graph-based architectures such as DCRNN is highly sensitive to implementation choices. Factors such as adjacency-matrix construction, normalization strategy, learning-rate scheduling, and diffusion-step configuration can substantially influence model behavior. Our findings do not contradict the value of spatial modeling; instead, they highlight that the benefits reported in prior work may not emerge unless the graph structure and training pipeline are carefully optimized.

This study underscores the importance of rigorous preprocessing and graph design when applying spatial–temporal deep learning models to real-world traffic forecasting. As future work, we plan to refine the adjacency matrix using alternative distance metrics, experiment with adaptive or learned graph structures, and more closely follow the reference DCRNN implementation to identify potential sources of degradation. We also aim to evaluate additional spatial–temporal architectures such as STGCN, ASTGCN, and AGCRN under the same unified framework, enabling a clearer assessment of how spatial information, when properly modeled, can enhance long-horizon traffic forecasting beyond temporal baselines.

VIII. REFERENCES

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[2] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," *International Conference on Learning Representations (ICLR)*, 2018.

[4] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3634–3640, 2018.

[5] L. Zhao et al., "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *arXiv preprint arXiv:1811.05320*, 2018.

[6] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting," *AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 922–929, 2019.

[7] Z. Wu, P. Pan, G. Long, C. Jiang, and S. Zhang, "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[8] Y. Xu, H. Li, J. Li, and Y. Yang, "Hybrid Spatio-Temporal Graph Convolutional Network for Travel Time Forecasting," *arXiv preprint arXiv:2007.15543*, 2020.

[9] W. Chen et al., "Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting," *arXiv preprint arXiv:1907.11611*, 2019.

[10] X. Ma et al., "Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction," *Sensors*, vol. 17, no. 4, 2017.

[11] Vaswani *et al.*, "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.