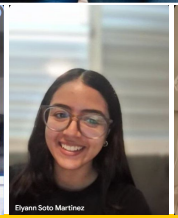
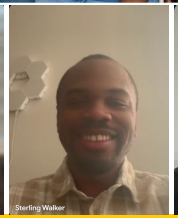
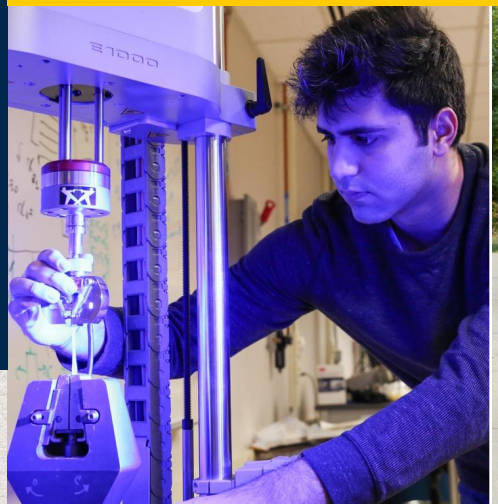
Spatio-Temporal Deep Learning for Traffic Forecasting

A Comparative Study of LSTM and DCRNN on PEMS-BAY

DEARBORN

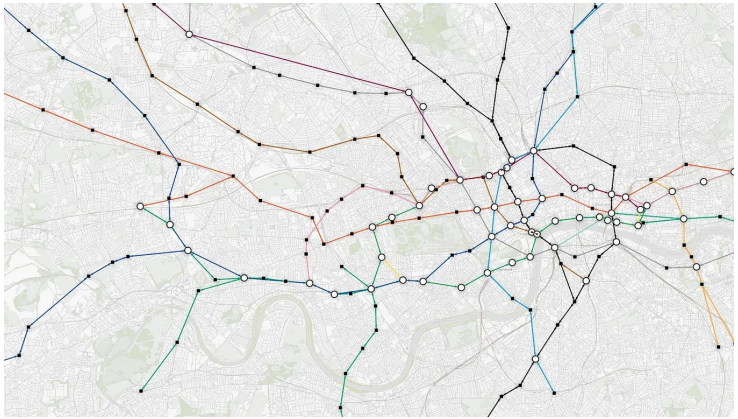






Motivation

- **Traffic congestion continues to grow worldwide**, leading to significant losses in time, fuel consumption, and economic productivity.
- **Real-time traffic prediction is critical** for enabling dynamic routing and navigation,adaptive traffic signal control, incident detection and emergency response, improved public transportation scheduling.
- Urban road networks exhibit strong **spatial and temporal** dependencies.



Objectives & Research Questions

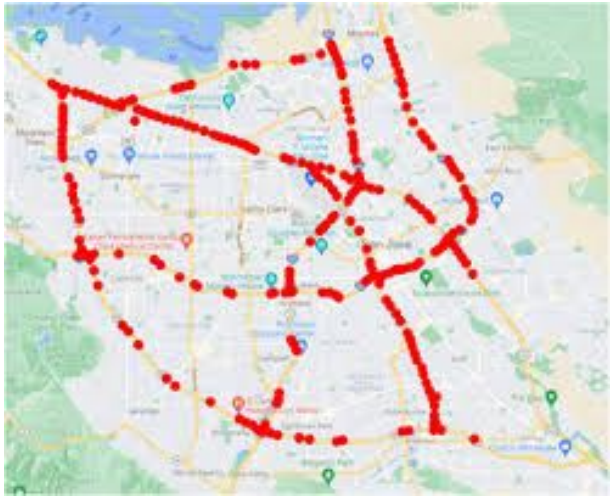
Main Goal: Evaluate how temporal-only (LSTM) vs. spatio-temporal (DCRNN) deep learning models perform on short-term traffic forecasting.

Questions:

1. Does modeling spatial graph connectivity significantly improve forecasting accuracy?
2. How does prediction accuracy change across different forecasting horizons (5-30 minutes)?
3. What advantages does diffusion-based graph modeling offer over sequence-only models?
4. How well do LSTM and DCRNN generalize to real-world traffic patterns in PEMS-BAY?

PEMS-BAY Overview

Dataset	PEMS-BAY
Sensors	325
Period	1 Jan - 30 Jun 2017
Region	San Francisco Bay Area, California
Sampling Rate	Every 5 minutes
Data Type	Traffic speed measurements (multivariate spatio-temporal series)



PEMS Bay Dataset: Every red dot here in the map depicts the sensor spatially located across the freeways.

Data Preprocessing

Normalization

- Applied **z-score normalization** to speed values of all sensors to scale them.
- Loaded and used the **precomputed 325×325 adjacency matrix** (distance-based graph) to encode spatial relationships between sensors.

Sequence Construction

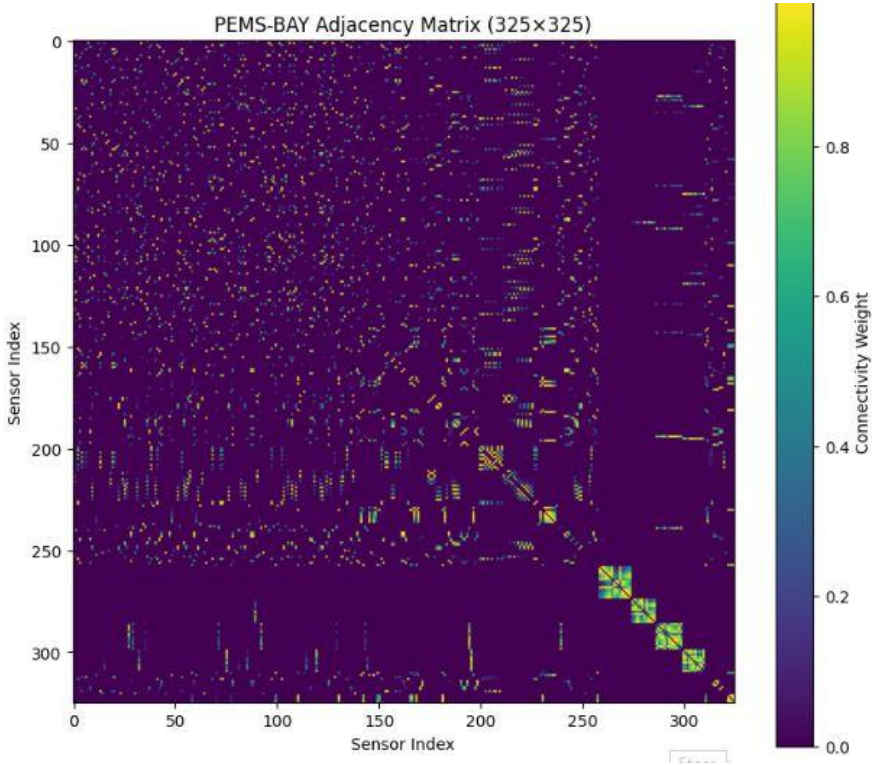
- Converted continuous data into **sliding windows of 12 past time steps : 1st, 2nd and 3rd future predictions**

Dataset Split

- 70% training, 10% validation, 20% testing

Graph Structure Visualization

- Adjacency matrix visualized using a heatmap to show connectivity strength between sensors and the spatial structure learned by DCRNN.

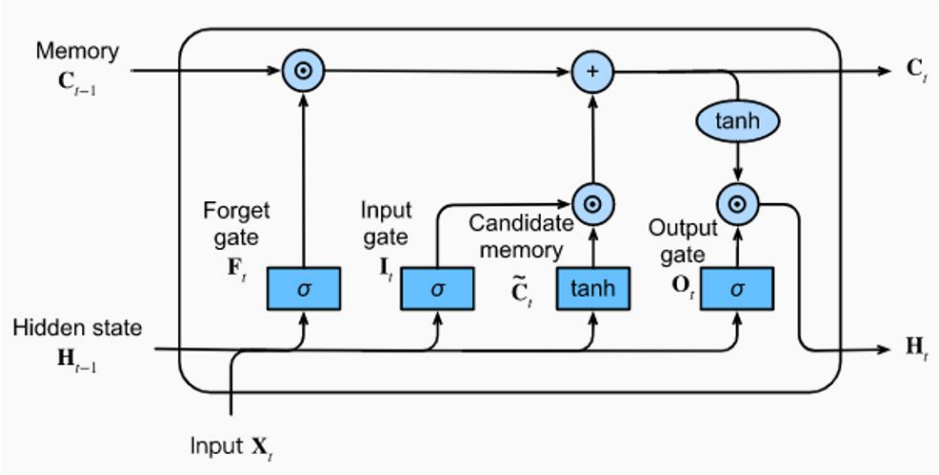


Baseline Model: LSTM

LSTM learns temporal patterns but not spatial structure.

Each sensor is modeled independently: past speeds > future speeds.

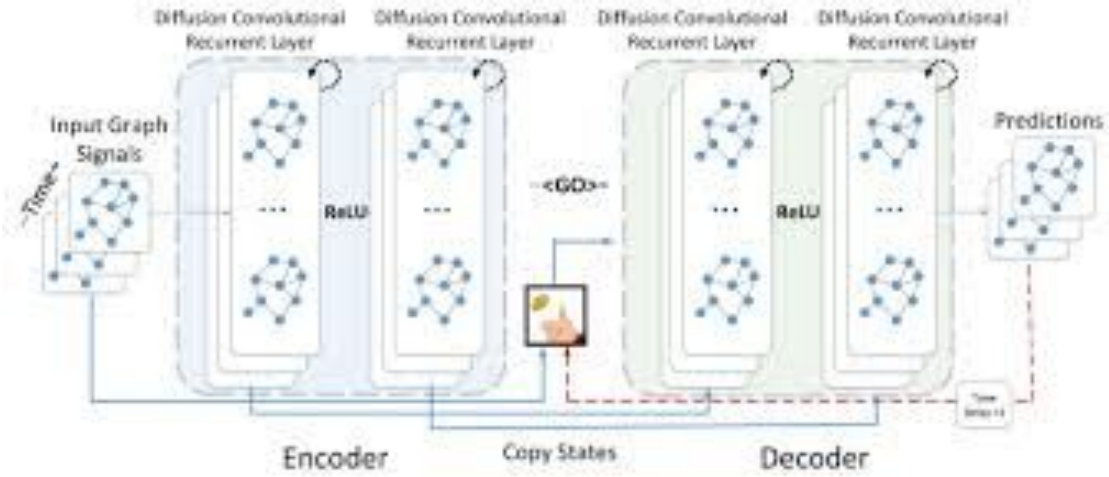
Flow: Input sequence > LSTM Layers > Fully Connected Layer > Predicted Future Speeds



Advanced Model: DCRNN

DCRNN learns both spatial (graph connectivity) and temporal patterns.
Uses diffusion graph convolutions + recurrent modeling to capture how traffic moves through the network.

Flow: Graph > DiffusionConv > Recurrent Layers > Output.



Training & Evaluation Setup

Training Configuration

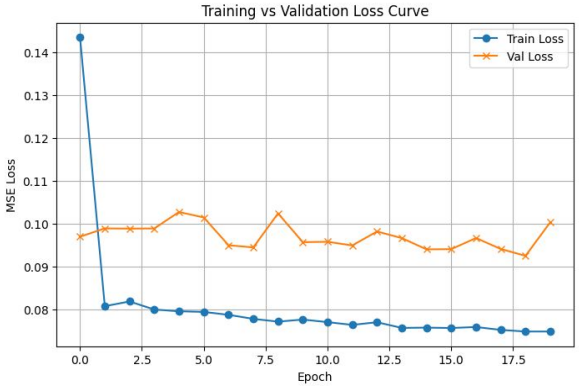
- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam (learning rate = 0.001)
- **Batch Size:** 64
- **Epochs:** 20
- **Hardware:** Google Colab GPU environment (Tesla T4)

Evaluation Metrics

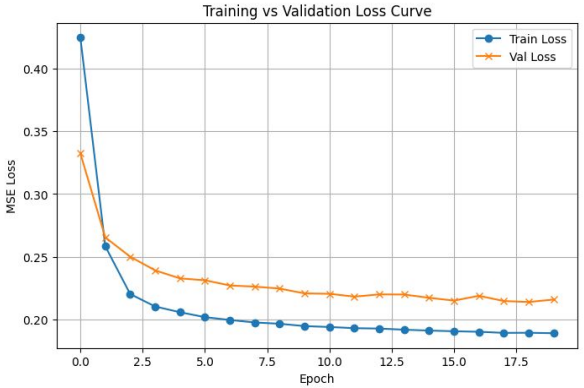
- **MAE** - Mean Absolute Error
- **RMSE** - Root Mean Squared Error

Model Validation

- Training and validation losses monitored to track convergence and detect overfitting.
- LSTM and DCRNN were evaluated under identical preprocessing, sequence lengths, and dataset splits for fair comparison.



Train vs Epoch curve: LSTM

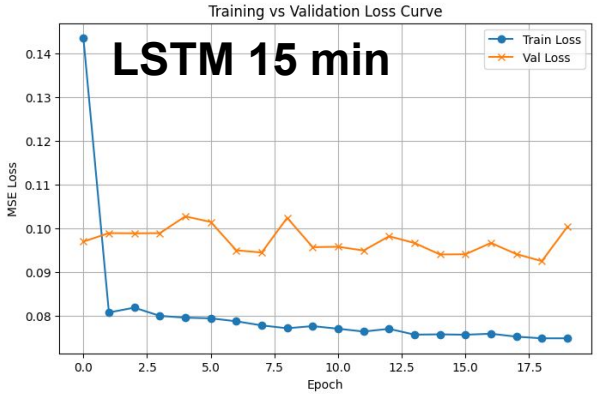


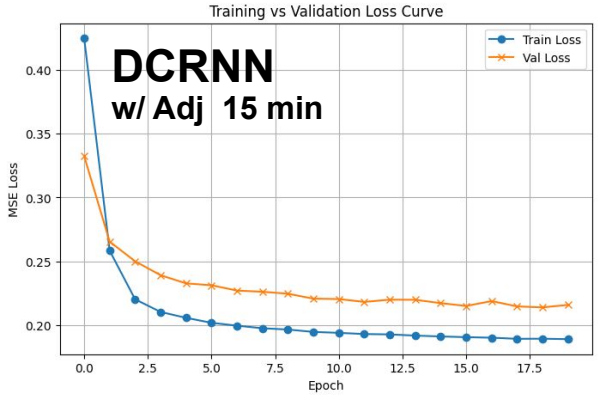
Train vs Epoch curve: DCRNN

Experimental Results: PEMS-BAY

Training and validation curves converge for all models:

- LSTM achieves the lowest plateaued loss at each horizon
- DCRNN without adjacency converges to moderately higher loss
- Graph-based DCRNN with adjacency has the highest loss in our configuration





Performance Metrics: PEMS-BAY

Metrics computed on held-out test set MSE, MAE, RMSE:

- Across 5-30 minute horizons, LSTM consistently performs best
- DCRNN without adjacency is slightly worse but still competitive
- DCRNN with adjacency has substantially larger errors at every horizon

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

Contradictory findings

Expectation: Research suggests Spatio-Temporal Graph models like DCRNN consistently outperform temporal-only baselines like LSTM.

Our Empirical Reality:

- **LSTM Superiority:** A well-tuned LSTM consistently achieved lower error rates across 5, 10, 15, and 30-minute horizons.
- **The "Complexity Tax":** The full DCRNN model using sensor adjacency performed significantly worse than the LSTM, with Validation Loss nearly **2x-3x higher** in some trials.

Unexpected Result: The "Speed-only" DCRNN (no graph adjacency) outperformed the full Graph DCRNN, suggesting the spatial data was hindering, not helping, our specific implementation.

Discussion

Adjacency Matrix Limitations: The pre-defined sensor graph may not accurately reflect true traffic dynamics (e.g., functional connectivity vs. physical distance).

Hyperparameter Sensitivity:

- Graph operations increase model complexity, making convergence much harder.
- Parameters tuned for sequence models (LSTM) do not transfer directly to diffusion convolutional architectures.

Architecture Isolation:

- Comparing **LSTM vs. DCRNN (No-Adj)** isolates the architecture.
- Since DCRNN (No-Adj) was competitive but still slightly worse, the deficit lies in the *graph construction*, not necessarily the diffusion mechanism.

Conclusion

Simplicity remains competitive: For short-term forecasting (5-15 mins), a strong temporal baseline (LSTM) is efficient and robust.

Complexity is not a guarantee: Simply switching to a graph-based architecture does not guarantee performance gains without rigorous tuning and graph definition.

- Our DCRNN with adjacency did not reach performance reported in prior work
- Results underscore the importance of careful graph design and replication of reference setups

Future Work:

- **Dynamic Graph Learning:** Instead of a static distance matrix, let the model learn the graph structure (e.g., Graph WaveNet).
- **Hybrid Tuning:** Investigate separate hyperparameter optimization strategies specifically for the diffusion layers.

Acknowledgments

Special Thanks to:

Our Team:

- Elyann Soto
- Krishna Sharma
- Martin Gasevski
- Sterling Walker

Our Data:

- California Department of Transportation - Performance Measurement System (PeMS)

DCRNN Reference Implementation:

- https://github.com/liyaguang/DCRNN