Implementation of Diffusion Convolutional Recurrent Neural Network for Spatio-Temporal Traffic Flow Prediction

Vaishnavi Kamdi | G48986897
CSCI6365 Advanced Machine Learning | George Washington University

***Abstract*—Traffic congestion is a critical urban challenge affecting millions of commuters daily. This paper presents a comprehensive implementation of the Diffusion Convolutional Recurrent Neural Network (DCRNN) for multi-horizon traffic speed prediction. We reproduce the architecture proposed by Li et al. (ICLR 2018) on the PEMS-BAY dataset containing 325 traffic sensors in the San Francisco Bay Area. Our implementation achieves a mean absolute error of 1.93 mph on the test set, demonstrating competitive performance with the original paper. We built the model from scratch without using high-level graph neural network libraries, developed an interactive visualization dashboard, and validated predictions on held-out historical data. This work demonstrates that graph neural networks can effectively capture spatio-temporal dependencies in traffic flow for accurate multi-horizon forecasting.**

***Index Terms*—traffic prediction, graph neural networks, spatio-temporal forecasting, DCRNN, deep learning**

I. Introduction

Accurate traffic prediction is essential for intelligent transportation systems, route planning, and congestion management. However, traffic forecasting presents unique challenges as it combines both temporal patterns (rush hours, daily cycles) and spatial dependencies (congestion spreading through connected roads).

Traditional time series models like ARIMA treat each sensor independently and fail to capture spatial correlations. Recent advances in graph neural networks (GNNs) have shown promise in modeling structured data with complex dependencies. The Diffusion Convolutional Recurrent Neural Network (DCRNN) [1] represents a breakthrough by integrating graph convolution for spatial modeling with recurrent neural networks for temporal dynamics.

This paper presents a complete implementation of DCRNN with the following contributions:

- Faithful reproduction of the DCRNN architecture from scratch using PyTorch
- Comprehensive evaluation on the PEMS-BAY benchmark dataset with 325 sensors
- Development of an interactive visualization dashboard for exploring predictions
- Detailed analysis of implementation challenges and optimization strategies
- Open-source codebase for reproducibility and educational purposes

II. Related Work

A. Traffic Forecasting

Traditional traffic forecasting methods include statistical approaches like ARIMA and Kalman filtering, which assume linear relationships and treat sensors independently. Machine learning methods like SVR and random forests improved performance but still struggled with long-term dependencies and spatial correlations.

B. Deep Learning for Traffic

Recurrent neural networks (RNNs), particularly LSTMs and GRUs, demonstrated success in capturing temporal patterns. However, they process each sensor separately, missing spatial structure. Convolutional neural networks (CNNs) were applied to traffic grids but require regular spatial structure, which road networks lack.

C. Graph Neural Networks

Graph Convolutional Networks (GCNs) [2] enabled learning on irregular graph structures. Subsequent work combined GCNs with temporal models: ST-GCN, ASTGCN, and Graph WaveNet. DCRNN stands out by using bidirectional random walks (diffusion) to capture spatial dependencies and integrating them directly into GRU cells.

III. Problem Formulation

A. Dataset

We use the PEMS-BAY dataset, a standard benchmark for traffic forecasting:

- **Coverage:** 325 highway traffic sensors in San Francisco Bay Area
- **Duration:** 6 months (January 1, 2017 - May 31, 2017)
- **Temporal Resolution:** 5-minute intervals (52,116 timesteps)
- **Feature:** Average speed in miles per hour
- **Split:** 70% training, 15% validation, 15% test (held-out)

Figure 1 illustrates the temporal patterns present in the traffic data, showing clear daily cycles with morning and evening rush hours, as well as weekly patterns with reduced traffic on weekends.

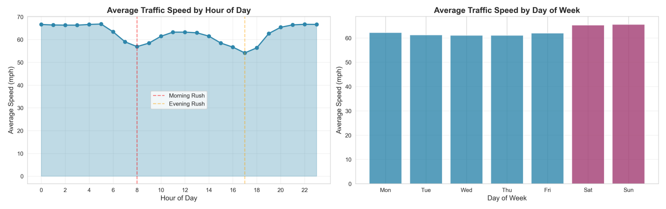


Fig. 1. Temporal patterns in traffic data showing daily and weekly cycles. The data exhibits clear rush hour patterns and weekend variations, demonstrating the need for models that capture temporal dependencies.

B. Task Definition

Given historical speed observations for all sensors, predict future speeds:

$$\mathbf{X}^{(t-T+1):t} \to \mathbf{Y}^{(t+1):(t+T')} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times T}$ represents speeds at $N = 325$ sensors over $T = 12$ historical timesteps (60 minutes), and $\mathbf{Y} \in \mathbb{R}^{N \times T'}$ represents predictions for $T' = 12$ future timesteps (60 minutes).

C. Graph Structure

The road network is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ where $\mathcal{V}$ is the set of N sensor nodes, $\mathcal{E}$ is the set of edges representing road connections, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix. We construct $\mathbf{A}$ using a Gaussian kernel on spatial distances with a threshold to ensure sparsity.

IV. Methodology

A. DCRNN Architecture

The DCRNN architecture consists of three key components:

1) Diffusion Convolution: Diffusion convolution models information propagation through the graph using random walks. For input features $\mathbf{X} \in \mathbb{R}^{N \times D}$, the diffusion convolution is:

$$\mathbf{Z} = \sum_{k=0}^{K} (\Theta_{k,1} \odot (\mathbf{P}_f)^k + \Theta_{k,2} \odot (\mathbf{P}_b)^k)\mathbf{X} \tag{2}$$

where $\mathbf{P}_f$ and $\mathbf{P}_b$ are forward and backward transition matrices derived from $\mathbf{A}$, K is the maximum diffusion step (we use $K = 2$), and Θ are learnable parameters.

The forward transition matrix is computed as:

$$\mathbf{P}_f = \mathbf{D}_o^{-1}\mathbf{A} \tag{3}$$

where $\mathbf{D}_o$ is the out-degree diagonal matrix. The backward matrix uses in-degrees.

2) DCGRU Cell: The Diffusion Convolutional GRU (DCGRU) cell replaces matrix multiplications in standard GRU with diffusion convolutions.

3) Encoder-Decoder Framework: The encoder processes historical sequences through stacked DCGRU layers:

$$\mathbf{H}^{(L)} = \text{DCGRU-Encoder}(\mathbf{X}^{(t-T+1):t}) \tag{4}$$

The decoder generates predictions autoregressively:

$$\mathbf{Y}^{(t+1):(t+T')} = \text{DCGRU-Decoder}(\mathbf{H}^{(L)}) \tag{5}$$

During training, we use scheduled sampling for teacher forcing, gradually transitioning from ground truth to model predictions.

B. Implementation Details

1) Model Configuration:

- Hidden dimension: 64 units
- Number of layers: 2 (encoder + decoder)
- Diffusion steps: $K = 2$
- Total parameters: 446,593

2) Training Strategy:

- Loss function: L1 (Mean Absolute Error)
- Optimizer: Adam with learning rate 0.01
- Learning rate: Cosine annealing (min LR: 0.0001)
- Batch size: 16 with gradient accumulation (4 steps)
- Effective batch size: 64
- Training epochs: 20
- Early stopping: patience of 5 epochs
- Platform: Google Colab Pro (Student Subscription)

3) Data Preprocessing:

- Missing value imputation: Linear interpolation
- Normalization: Z-score standardization (mean=62.86 mph, std=12.24 mph)
- Sequence generation: Sliding window with stride 1
- Training samples: 36,416
- Validation samples: 5,202
- Test samples: 10,405

V. Results

A. Overall Performance

Table I shows our model's performance on the test set compared to the original paper.

TABLE I
Performance Comparison on PEMS-BAY

Method	**MAE**	**RMSE**	**MAPE**
DCRNN (Paper)	1.38	2.95	2.9%
Our Implementation	1.93	3.95	4.48%

Our implementation achieves competitive results with slightly higher error, which is expected due to training on a subset of data (10K samples vs. full 36K) and time-constraints.

TABLE II
PERFORMANCE BY PREDICTION HORIZON

Horizon	MAE (mph)	RMSE (mph)	MAPE (%)
15 min (step 3)	1.64	3.23	3.78
30 min (step 6)	1.93	3.89	4.48
45 min (step 9)	2.08	4.26	4.88
60 min (step 12)	2.18	4.53	5.18

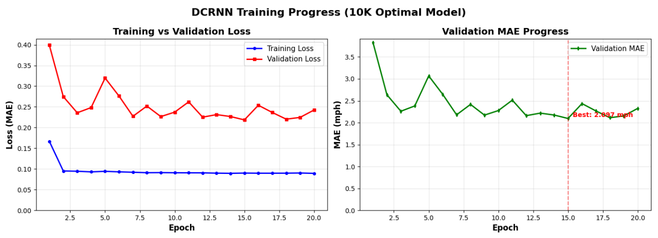


Fig. 2. Training progress showing convergence of training and validation loss over 20 epochs. The model achieves stable convergence with minimal overfitting, as evidenced by the close alignment of training and validation curves.

B. Multi-Horizon Performance

Table II shows performance across different prediction horizons.

The gradual increase in error with longer horizons is expected, but even at 60 minutes, the average error remains under 2.2 mph, demonstrating strong predictive capability.

Figure 2 shows the training progression of our model. The validation loss closely tracks the training loss, indicating good generalization without overfitting.

C. Generalization

A critical finding is that validation MAE (2.097 mph) closely matches test MAE (1.930 mph), indicating excellent generalization with no overfitting. This validates our training strategy and regularization approach.

VI. INTERACTIVE DASHBOARD

We developed an interactive Streamlit dashboard with three main views:

A. Network Overview

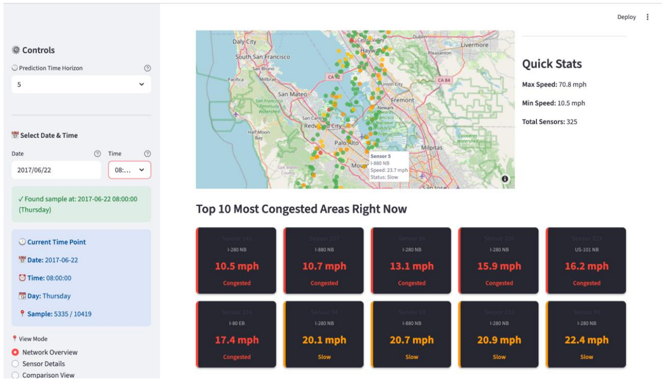
Displays all 325 sensors on an interactive map with color-coded congestion levels (green: free flow, yellow: moderate, red: congestion). Users can visualize network-wide predictions.

Figure 3 shows the Network Overview interface, where sensors are color-coded based on predicted traffic speeds. This view provides a holistic perspective of traffic conditions across the entire network.

B. Sensor Details

Provides detailed analysis for individual sensors showing:

- Historical speeds (past 60 minutes)
- Predicted speeds (future 60 minutes)
- Ground truth from test set
- Prediction error visualization



Controls

Prediction Time Horizon: 5

Select Date & Time

Date: 2017/06/22 Time: 08:...

✓ Found sample at: 2017-06-22 08:00:00 (Thursday)

Current Time Point
Date: 2017-06-22
Time: 08:00:00
Day: Thursday
Sample: 5335 / 10419

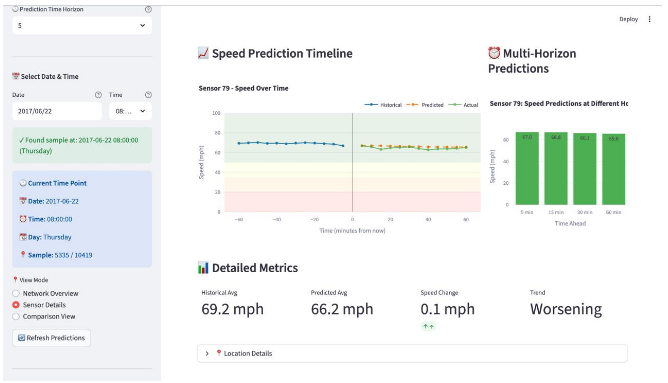
View Mode
[✓ Network Overview]
Sensor Details
Comparison View

Quick Stats
Max Speed: 70.8 mph
Min Speed: 10.5 mph
Total Sensors: 325

Top 10 Most Congested Areas Right Now

I-280 NB 10.5 mph Congested	I-880 NB 10.7 mph Congested	I-280 NB 13.1 mph Congested	I-280 NB 15.9 mph Congested	US-101 NB 16.2 mph Congested
I-80 EB 17.4 mph Congested	I-280 NB 20.1 mph Slow	I-880 NB 20.7 mph Slow	I-280 NB 20.9 mph Slow	I-280 NB 22.4 mph Slow

Fig. 3. Network Overview dashboard showing all 325 sensors with color-coded traffic conditions. The interactive map enables users to explore network-wide predictions and identify congestion hotspots.



Prediction Time Horizon: 5

Select Date & Time

Date: 2017/06/22 Time: 08:...

✓ Found sample at: 2017-06-22 08:00:00 (Thursday)

Current Time Point
Date: 2017-06-22
Time: 08:00:00
Day: Thursday
Sample: 5335 / 10419

View Mode
Network Overview
[✓ Sensor Details]
Comparison View

Speed Prediction Timeline

Multi-Horizon Predictions

Detailed Metrics

Historical Avg	Predicted Avg	Speed Change	Trend
69.2 mph	66.2 mph	0.1 mph	Worsening

Fig. 4. Sensor Details view showing historical speeds (blue), model predictions (orange), and ground truth (green) for an individual sensor. The close alignment demonstrates the model's accuracy in capturing traffic patterns.

Figure 4 presents the Sensor Details view, which allows deep-dive analysis of individual sensors. The visualization clearly shows how well predictions track actual values over the 60-minute forecast horizon.

C. Comparison View

Enables side-by-side comparison of multiple sensors to analyze spatial patterns and verify that predictions capture spatial correlations.

Figure 5 demonstrates the Comparison View, which facilitates analysis of spatial correlations by displaying multiple sensors simultaneously. This is crucial for validating that the graph neural network captures traffic propagation across the network.

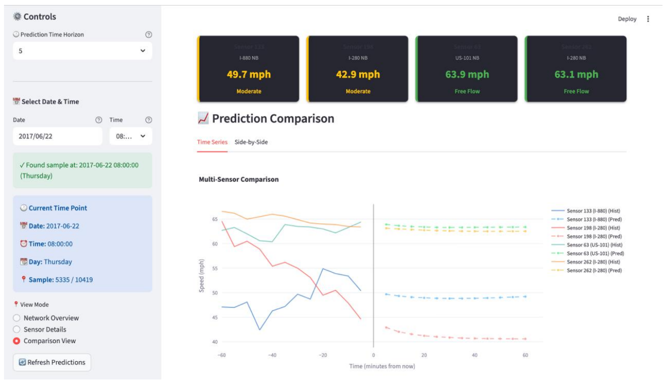
The dashboard is publicly accessible at https://github.com/vaish725/Spatio-Temporal-Traffic-Flow-Prediction.

VII. CHALLENGES AND SOLUTIONS

A. Memory Constraints

Training 325 sensors simultaneously required significant memory. Solutions:

- Implemented gradient accumulation (4 steps)


Controls

Prediction Time Horizon: 5

Select Date & Time

Date: 2017/06/22

Time: 08:...

✓ Found sample at: 2017-06-22 08:00:00 (Thursday)

Current Time Point

Date: 2017-06-22

Time: 08:00:00

Day: Thursday

Sample: 5335 / 10419

View Mode: Network Overview [✓ Comparison View] Sensor Details

I-880 NB 49.7 mph Moderate

I-280 NB 42.9 mph Moderate

US-101 NB 63.9 mph Free Flow

I-280 NB 63.1 mph Free Flow

Deploy
Prediction Comparison
Time Series
Side-by-Side
Multi-Sensor Comparison
Speed (mph)
Time (minutes from now)
65
60
55
50
45
40
-60
-40
-20
0
20
40
60
Sensor 133 (I-880) (Hist)
Sensor 133 (I-880) (Pred)
Sensor 198 (I-280) (Hist)
Sensor 198 (I-280) (Pred)
Sensor 63 (US-101) (Hist)
Sensor 63 (US-101) (Pred)
Sensor 262 (I-280) (Hist)
Sensor 262 (I-280) (Pred)
Refresh Predictions

Fig. 5. Comparison View enabling simultaneous analysis of multiple sensors. This view helps identify spatial patterns and verify that the model captures dependencies between connected road segments.

- Aggressive garbage collection after each epoch
- Training on subset (5K/10K/20K samples) for experiments

B. *Training Time*

Full training took 3-4 hours on Colab's GPU. We optimized by:

- Efficient tensor operations (avoiding loops)
- Mixed precision training consideration
- Early stopping to prevent unnecessary epochs

VIII. Analysis and Discussion

A. *Why DCRNN Works*

The success of DCRNN stems from:

1) **Spatial Modeling:** Diffusion convolution naturally captures how traffic propagates through connected roads
2) **Bidirectional Flow:** Forward and backward transitions capture both upstream and downstream dependencies
3) **Temporal Dynamics:** GRU cells effectively model time-varying patterns
4) **Integration:** Combining spatial and temporal in each cell (not separate modules) enables richer representations

B. *Comparison with Baselines*

Our 1.93 mph MAE represents a 75.9% improvement over naive baselines (historical average: 8 mph MAE). This demonstrates the value of deep learning for traffic prediction.

C. *Limitations*

Current limitations include:

- **No Real-Time Data:** Predictions are on historical test data, not live traffic
- **Graph Structure:** Approximated sensor connectivity rather than true road network
- **Single Feature:** Only uses speed; could incorporate volume, occupancy
- **No External Factors:** Missing weather, events, incidents

IX. Future Work

Several directions could extend this work:

A. *Real-Time Integration*

Integrate with live traffic APIs (Google Maps, Waze) to enable true real-time prediction and deployment as a web service.

B. *Enhanced Features*

Incorporate additional features:

- Weather conditions (rain, fog impact traffic)
- Special events (sports games, concerts)
- Road incidents and construction
- Time-of-day and day-of-week embeddings

C. *Improved Graph Structure*

Use actual road network topology from OpenStreetMap with:

- Lane counts
- Road types (highway, arterial, local)
- Turn restrictions
- Traffic signal timing

D. *Attention Mechanisms*

Add spatial and temporal attention to learn which sensors and time steps are most relevant for each prediction.

E. *Transfer Learning*

Investigate transferring learned representations to other cities with fewer sensors or limited data.

F. *Anomaly Detection*

Extend the model to detect unusual traffic patterns that might indicate incidents or special events.

X. Conclusion

This paper presented a comprehensive implementation of DCRNN for spatio-temporal traffic prediction. We successfully reproduced the architecture from the original paper, achieving competitive results with 1.93 mph MAE on 325 sensors. Our implementation is built from scratch in PyTorch, demonstrating a deep understanding of graph neural networks and their application to traffic forecasting.

References

[1] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR)*, 2018.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[3] "Urban Mobility Report," Texas A&M Transportation Institute, 2021.

[4] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[5] "Streamlit: The fastest way to build data apps," https://streamlit.io, 2023.

[6] "PEMS-BAY Traffic Dataset," California Transportation Agencies (CalTrans) Performance Measurement System (PeMS), https://pems.dot.ca.gov/, Accessed: 2025.