DC-ESN: Diffusion Convolutional Echo State Network for Spatiotemporal Traffic Forecasting

Anonymous authors
Paper under double-blind review

Abstract

Traffic forecasting is a challenging spatiotemporal problem that requires capturing complex dependencies across both road networks and time. Existing deep learning approaches such as Convolutional Neural Networks (CNNs) for spatial modeling, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks for temporal modeling, as well as Graph Convolutional Networks (GCNs) for capturing topological structures have achieved notable success. But they often incur high computational cost and require large number of trainable parameters. In this work, we introduces a novel architecture named Diffusion-Convolutional Echo State Network (DC-ESN) designed for spatio-temporal forecasting, which combines diffusion convolution for spatial feature extraction with a Echo State Network (ESN) for efficient temporal modeling. This structural decoupling allows the model to learn complex spatial topologies via gradient descent while leveraging the asymptotic stability of the fixed reservoir for temporal memory, offering a robust and computationally efficient alternative to fully trainable spatiotemporal graph networks. Compared with Diffusion-Convolutional Recurrent Neural Networks (DCRNN), Diffusion-Convolutional Echo State Network (DC-ESN) achieves comparable predictive accuracy while significantly improving inference efficiency and reducing GPU memory usage. Experiments on the METR-LA and PEMS-BAY benchmark traffic datasets demonstrate that DC-ESN attains faster inference with minimal accuracy loss, making it suitable for real-time forecasting applications.

Keywords: Diffusion convolution, Echo state network, Graph neural network, Reservoir computing, Spatiotemporal forecasting, Traffic forecasting

1 Introduction

Spatiotemporal forecasting is a crucial task in many scientific and engineering domains, including weather modeling, environmental monitoring, and traffic prediction. It aims to predict future observations that depend on both spatial and temporal relationships. Applications of spatiotemporal forecasting include traffic flow prediction, energy demand forecasting and air quality estimation. The main focus of this research is traffic forecasting on road networks, which constitutes a fundamental component of Intelligent Transportation Systems (ITS). The purpose of traffic forecasting in this work is to anticipate future traffic speeds for a sensor network based on past traffic speeds and the underlying road network structure. This is a challenging task due to the inherent nonlinear nature of traffic flow, complex spatial correlations across road networks, and strong temporal dependencies over time.

Traditional machine learning methods such as Support Vector Regression (SVR), Random Forests (RF), and Autoregressive Integrated Moving Average (ARIMA) have been widely applied to this problem, but their ability to capture nonlinear and high-dimensional spatiotemporal relationships is limited. Traditional time-series models such as ARIMA and Kalman filters assume linear and stationary processes, that restricts their ability to model nonlinear dependencies (Lippi et al., 2013). Traffic time series exhibit significant temporal dynamics, and non-stationarity caused by recurring events, such as accidents or rush hours, can make long-term forecasting challenging (Cheng et al., 2021).

In many classical deep-learning tasks (e.g., image recognition), data lie on a regular Euclidean grid; pixels are arranged in rows and columns, and convolution operations assume uniform neighbourhoods (Lecun et al., 1998; Hinton et al., 2012). Traffic networks, however, do not conform to this structure. As noted in the literature (Shuman et al., 2013; Xing et al., 2023); because transportation networks are non-Euclidean in structure, dividing the network into grids for convolutional operations using CNNs destroys important structural information. This drawback is illustrated in figure 1. Recent advancements in deep learning, such as Graph Neural Networks (GNNs), have improved performance in non-Euclidean domains.

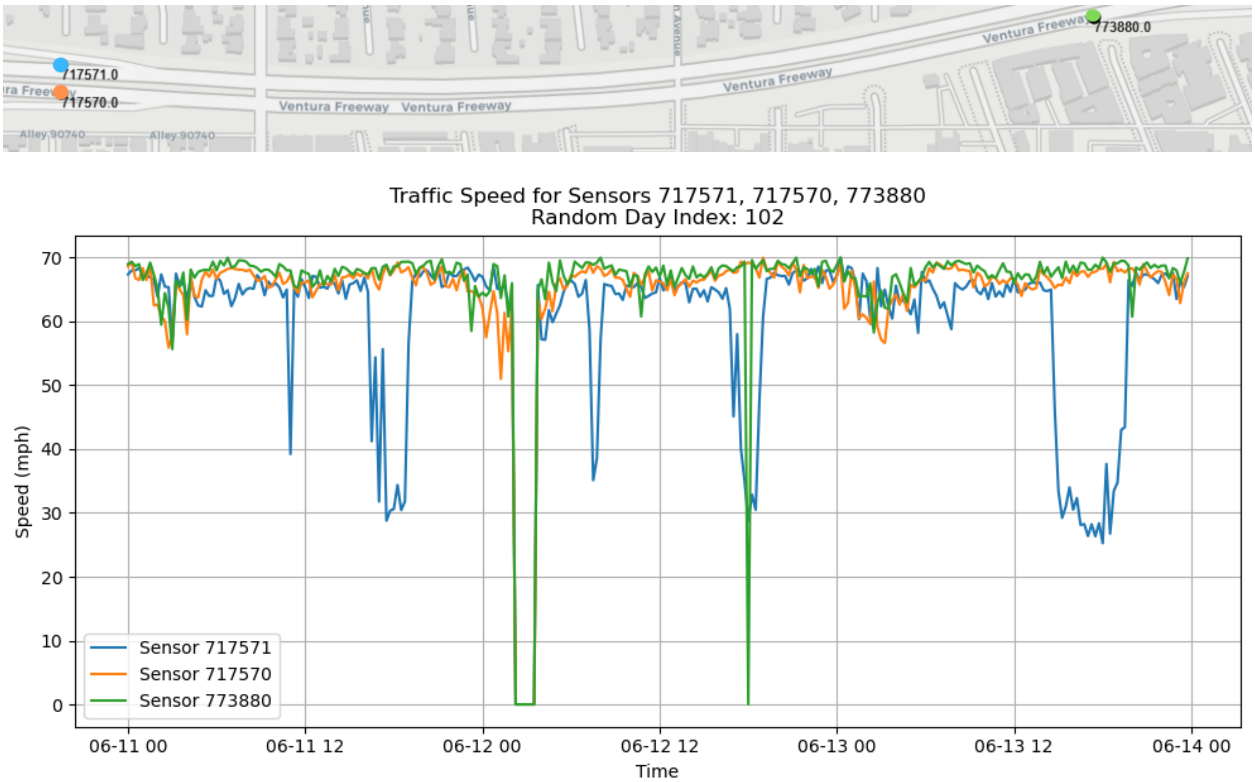


Figure 1: Speeds at sensors 717570 and 773880 are correlated because they are located on the same road, whereas the speed at sensor 717571 follows a different pattern, indicating no spatial relation between sensors that are close to each other in the Euclidean domain (sensor 717571 and 717570).

The Diffusion Convolutional Recurrent Neural Network (DCRNN) (Li et al., 2018) is a graph-based model that introduces the idea of modeling spatial dependencies using diffusion processes on directed graphs, combined with Gated Recurrent Units (GRUs) for temporal dynamics. Although GRUs and Long Short-Term Memory (LSTM) networks have demonstrated strong capabilities in capturing long-range temporal dependencies, their training involves backpropagation through time (BPTT), which is computationally expensive and prone to vanishing or exploding gradients (Bengio et al., 1994; Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013).

To address these challenges, we replace GRUs with a lightweight and efficient reservoir computing model known as the Echo State Network (ESN) (Jaeger, 2001). ESNs consist of a large, fixed recurrent reservoir with randomly initialized and untrained internal weights, while only the output (readout) layer is trained. This structure allows ESNs to retain the temporal representational power of recurrent models while significantly reducing computational cost and training time.

Recent developments in Echo State Networks (ESNs) (Gallicchio et al., 2018; Gallicchio & Micheli, 2017), further extend ESNs by stacking multiple reservoirs hierarchically to capture multi-scale temporal dependencies. Leveraging these properties, we propose the Diffusion Convolutional Echo State Network (DC-ESN), which integrates diffusion convolution for spatial feature extraction with ESNs for efficient and scalable

temporal modeling. This hybrid design preserves DCRNN's ability to model spatiotemporal correlations while achieving faster training and inference due to the non-trainable recurrent dynamics of the ESN. The proposed architecture achieves up to 5.5x faster training time and 64% lower GPU usage than DCRNN, with only 3-6% higher MAE, making it suitable for real-time deployment.

2 Related Work

Traffic forecasting has been an active research topic in both transportation engineering and machine learning, focusing on modeling complex spatiotemporal dependencies in road networks. In general, existing methods can be categorized into three main groups: Statistical approaches, Classical machine learning approaches, and Deep learning approaches. Each category contributes unique modeling perspectives but also faces inherent limitations when dealing with highly nonlinear and complex spatiotemporal dependencies. Early approaches were dominated by statistical and classical machine learning methods such as Autoregressive Integrated Moving Average (ARIMA), Vector Autoregression (VAR), Kalman filtering, Support Vector Regression (SVR), Random Forests (RF), and k-Nearest Neighbors (kNN). Although these techniques achieved moderate success, they relied heavily on handcrafted features and were limited in handling nonlinear relationships and large-scale, high-dimensional data.

With the advent of deep learning, models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) have been employed to capture temporal dependencies, while Convolutional Neural Networks (CNNs) have been utilized to model spatial correlations by treating traffic data as grid-like images. However, CNN-based approaches struggle with non-Euclidean road network structures. To address this, Graph Neural Networks (GNNs) have emerged as powerful tools for modeling graph-structured data. In particular, Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016) and Chebyshev Networks (ChebNet) (Defferrard et al., 2016) extended CNN operations to irregular graph domains, while Diffusion Convolutional Recurrent Neural Networks (DCRNN) (Li et al., 2018) modeled traffic flow as a diffusion process on directed graphs to capture asymmetric spatial dependencies.

Despite their effectiveness, these deep architectures often involve high computational costs and a large number of trainable parameters ,a trend frequently characterized as "Red AI" (Schwartz et al., 2020) This reliance on massive computational power not only limits scalability in real-time applications but also raises significant barriers to research inclusivity (Schwartz et al., 2020). Recently, reservoir computing frameworks such as the Echo State Network (ESN) (Jaeger, 2001) and its deep variants (Gallicchio et al., 2018) (Gallicchio & Micheli, 2017) have gained attention for their ability to model temporal dynamics efficiently using fixed recurrent reservoirs and fast training. However, ESNs alone cannot capture spatial dependencies inherent in graph-structured traffic data. Our proposed Diffusion Convolutional Echo State Network (DC-ESN) bridges these paradigms by combining diffusion-based graph convolution with a ESN architecture to jointly learn spatial and temporal dependencies. This hybrid design provides a favorable balance between predictive accuracy, computational efficiency, and parameter economy for large-scale spatiotemporal forecasting tasks which following the "Green AI" directive to prioritize efficiency alongside accuracy as a primary evaluation criterion (Schwartz et al., 2020).

3 Methodology

In this study, we propose the Diffusion Convolutional Echo State Network (DC-ESN), a hybrid deep learning framework designed for spatiotemporal traffic forecasting. Our approach synergizes the spatial feature extraction capabilities of Graph Neural Networks (GNNs), specifically Diffusion Convolution (Li et al., 2018), with the computational efficiency of Reservoir Computing (Jaeger, 2001). Unlike the standard DCRNN, which relies on fully trainable Gated Recurrent Units (GRUs), the DC-ESN employs a large, fixed sparse reservoir to model temporal dynamics, restricting gradient-based training to the spatial diffusion and readout modules.

3.1 Problem Formulation

Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{W})$ with $|\mathcal{V}| = N$ nodes, where $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ denotes a weighted adjacency matrix describing pairwise relationships between nodes. A graph signal at time t is represented as $\boldsymbol{X}^{(t)} \in \mathbb{R}^{N \times P}$, where each node is associated with P features. In the context of traffic forecasting, nodes correspond to traffic sensors distributed across a road network, and the graph structure encodes spatial connectivity between sensors (e.g., road distance or topology) (Li et al., 2018). The graph signal $\boldsymbol{X}^{(t)}$ therefore represents traffic measurements such as speed, flow, or occupancy collected at time t.

Given a sequence of historical graph signals

$$\{\boldsymbol{X}^{(t-T'+1)}, \ldots, \boldsymbol{X}^{(t)}\},$$

the objective is to learn a predictive mapping

$$h : (\boldsymbol{X}^{(t-T'+1)}, \ldots, \boldsymbol{X}^{(t)}, \mathcal{G}) \rightarrow (\boldsymbol{X}^{(t+1)}, \ldots, \boldsymbol{X}^{(t+T)}),$$

which forecasts the future evolution of the graph signal over the next T time steps.

This formulation captures the spatiotemporal structure of traffic data, where spatial dependencies are governed by the graph topology and temporal dependencies arise from the dynamic evolution of traffic conditions.

3.2 Spatial Modeling via Graph Diffusion

To characterize the non-Euclidean spatial dependencies inherent in traffic sensor networks, we adopt a graph-based diffusion framework (Li et al., 2018). By representing the sensor topology as a directed graph $\mathcal{G}$, we can effectively model the asymmetric propagation of traffic flow. We define this spatial influence through a random walk with a restart probability $\alpha \in [0, 1]$, where the transition dynamics are governed by the out-degree normalized adjacency matrix $\boldsymbol{D}_O^{-1}\boldsymbol{W}$. Here, $\boldsymbol{D}_O = \mathrm{diag}(\boldsymbol{W}\mathbf{1})$ serves as the out-degree diagonal matrix, with $\mathbf{1} \in \mathbb{R}^N$ representing an all-one vector.

As this Markovian process reaches a steady state, it converges to a stationary distribution $\boldsymbol{P} \in \mathbb{R}^{N \times N}$. Each row $\boldsymbol{P}_{i,:}$ in this distribution quantifies the "reachability" or spatial proximity originating from node $v_i \in \mathcal{V}$. Following the theoretical foundations of random walks on graphs (Teng, 2016), this process can be expressed in the following closed form:

$$\boldsymbol{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (\boldsymbol{D}_O^{-1}\boldsymbol{W})^k \tag{1}$$

In practice, the infinite series is approximated by truncating the diffusion process to K steps. To ensure the model remains sensitive to both upstream and downstream dependencies, we incorporate a reciprocal diffusion component defined by the in-degree transition matrix $\boldsymbol{D}_I^{-1}\boldsymbol{W}^\top$, where $\boldsymbol{D}_I = \mathrm{diag}(\boldsymbol{W}^\top\mathbf{1})$ denotes the in-degree diagonal matrix. By integrating both forward and backward diffusion processes, the architecture effectively captures the bidirectional nature of traffic information flow within the network (Li et al., 2018).

3.3 Diffusion Convolution Implementation

To model spatial relationships, we employ a diffusion convolution operation over graph signals (Kipf & Welling, 2016). While the theoretical formulation involves summing diffusion steps, our implementation utilizes a Projection-Concatenation-Projection (PCP) architecture to enhance spatial expressiveness and computational efficiency on GPUs.

Let $\boldsymbol{Z}^{(t)} = [\boldsymbol{X}^{(t)} \| \boldsymbol{H}^{(t-1)}] \in \mathbb{R}^{N \times (P+H)}$ be the concatenation of the current input and the previous recurrent state. The spatial processing proceeds in three stages:

1. Input Projection (Dimensionality Alignment): First, we project the high-dimensional input into the hidden dimension H using a trainable weight matrix $\boldsymbol{W}_{\text{pre}} \in \mathbb{R}^{(P+H)\times H}$. This bottleneck step significantly reduces the parameter count for subsequent operations:

$$\tilde{\boldsymbol{Z}}^{(t)} = \boldsymbol{Z}^{(t)}\boldsymbol{W}_{\text{pre}} \tag{2}$$

2. Multi-Hop Diffusion and Concatenation: Instead of summing the diffusion steps, we explicitly compute the diffusion states for K hops in both forward and backward directions and concatenate them into a comprehensive spatial vector$\boldsymbol{U}^{(t)}$ as shown in figure 2. This allows the model to learn distinct weights for different diffusion distances (Abu-El-Haija et al., 2019) (e.g., immediate vs. distant neighbors):

$$\boldsymbol{U}^{(t)} = \left[\tilde{\boldsymbol{Z}}^{(t)} \Big\Vert_{k=1}^{K} (\boldsymbol{D}_O^{-1}\boldsymbol{W})^k \tilde{\boldsymbol{Z}}^{(t)} \Big\Vert_{k=1}^{K} (\boldsymbol{D}_I^{-1}\boldsymbol{W}^\top)^k \tilde{\boldsymbol{Z}}^{(t)}\right] \tag{3}$$

where $\Vert$ denotes concatenation along the feature dimension. The resulting vector $\boldsymbol{U}^{(t)} \in \mathbb{R}^{N\times(H\cdot(2K+1))}$ preserves multi-scale spatial information.

3. Spatial Mixing: Finally, the spatial output $\boldsymbol{S}^{(t)}$ is obtained by mapping the concatenated features back

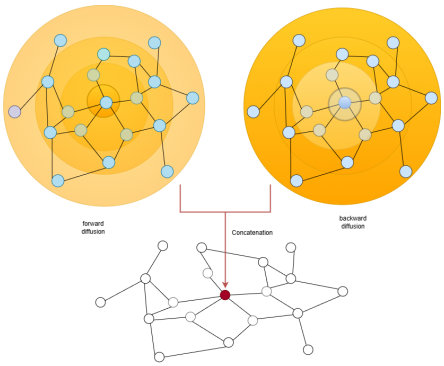


Figure 2: Diffusion convolution using MixHop to aggregate multi-hop forward and reverse random walks.

to the hidden dimension using a trainable post-weight matrix $\boldsymbol{W}_{\text{post}} \in \mathbb{R}^{(H\cdot(2K+1))\times H}$:

$$\boldsymbol{S}^{(t)} = \boldsymbol{U}^{(t)}\boldsymbol{W}_{\text{post}} \tag{4}$$

3.4 Temporal Dependency Modeling: The DC-ESN Cell

To capture the temporal dynamics of traffic flow, we replace the fully trainable Gated Recurrent Unit (GRU) used in prior models (e.g., DCRNN) with an Echo State Network (ESN) reservoir architecture as shown in figure 3. Unlike traditional RNNs, ESNs maintain a large, fixed recurrent reservoir with randomly initialized connections. Only the spatial projection weights ($\boldsymbol{W}_{\text{pre}}, \boldsymbol{W}_{\text{post}}$) and the readout layer are trained, enabling highly efficient learning while preserving strong nonlinear temporal representation capabilities.

3.4.1 Reservoir Dynamics

Let $\boldsymbol{X}^{(t)}$ denote the input at time t. The update mechanism for the hidden state $\boldsymbol{H}^{(t)}$ combines the trainable spatial features with the fixed temporal dynamics. The state update equation is given by:

$$\boldsymbol{H}^{(t)} = (1-\alpha)\boldsymbol{H}^{(t-1)} + \alpha \tanh\left(\boldsymbol{S}^{(t)} + \boldsymbol{T}^{(t)}\right) \tag{5}$$

where:

- $\alpha \in (0, 1]$ is the leaking rate, acting as a low-pass filter for memory retention.

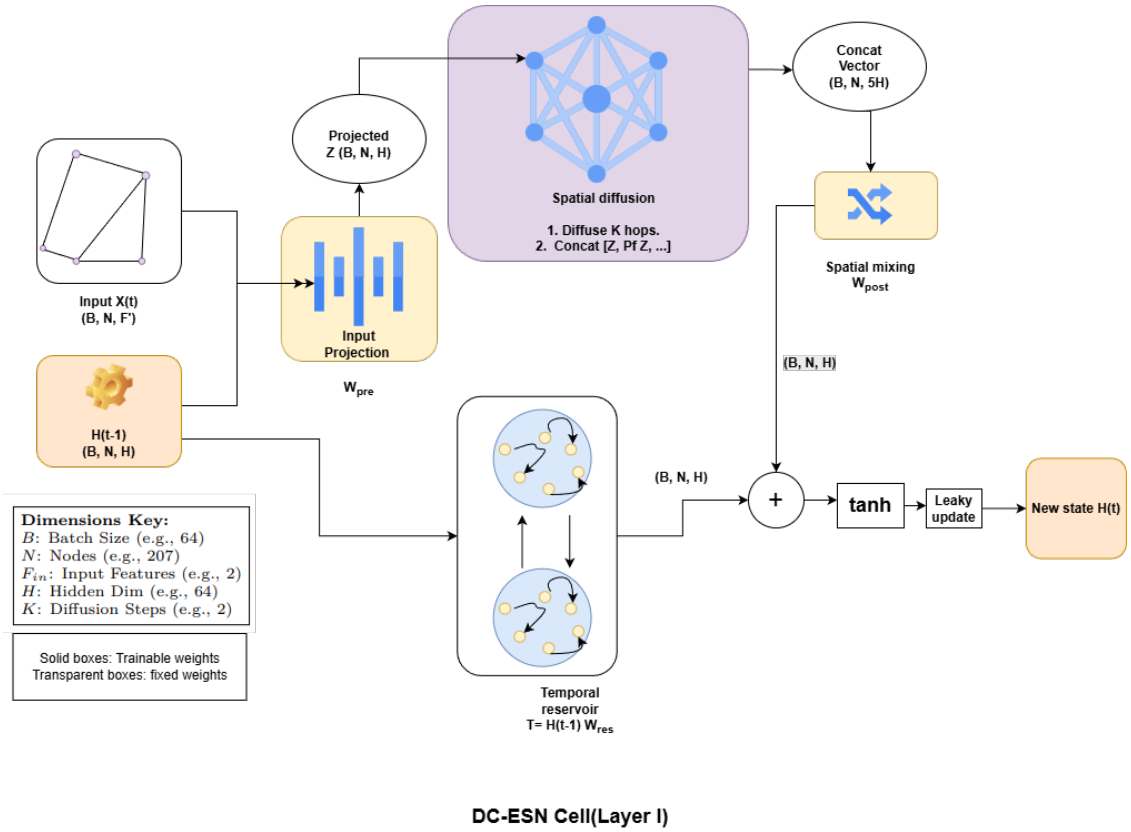


Figure 3: The DC-ESN Cell architecture decoupling spatial and temporal modeling. The cell integrates a trainable diffusion convolution module (top) to capture spatial dependencies via multi-hop concatenation with a fixed, sparse reservoir (bottom) for efficient temporal memory.

- $\boldsymbol{S}^{(t)}$ is the Spatial Component derived from the trainable Diffusion Convolution (Eq. 6).
- $\boldsymbol{T}^{(t)}$ is the Temporal Component. To allow for heterogeneous temporal dynamics across the network, we utilize a node-specific reservoir operation:

$$\boldsymbol{T}_i^{(t)} = \boldsymbol{H}_i^{(t-1)} \mathbf{W}_{\text{res},i} \quad \forall i \in \{1, \ldots, N\} \tag{6}$$

Here, $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N \times H \times H}$ is a tensor containing independent reservoir weights for each node. These weights are initialized sparsely with spectral radius $\rho < 1$ and are frozen (non-trainable), ensuring the "Echo State Property" and avoiding the vanishing gradient problem.

3.5 Encoder–Decoder Architecture

We adopt a Sequence-to-Sequence (Seq2Seq) framework to handle multi-step forecasting. Both the encoder and decoder are constructed using stacked DC-ESN cells as depicted in figure 4.

3.5.1 Encoder

The encoder processes the historical sequence $\boldsymbol{X} = [\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(T')}]$. At each step t, the hidden state evolves according to the DC-ESN dynamics described above. The encoder summarizes the temporal evolution of the input sequence into the final hidden states $\boldsymbol{H}^{(T')}$, which compactly represent the past spatiotemporal context.

3.5.2 Decoder

The decoder generates forecasts $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{X}}^{(T'+1)}, \ldots, \hat{\boldsymbol{X}}^{(T'+T)}]$ autoregressively. Initialized with the encoder's final state, the decoder predicts the next time step. The output is projected to the target dimension via a trainable linear readout:

$$\hat{\boldsymbol{X}}^{(t)} = \boldsymbol{H}^{(t)} \boldsymbol{W}_{\text{out}} + \boldsymbol{b} \tag{7}$$

where, $\boldsymbol{W}_{\text{out}}$ and $\boldsymbol{b}$ are learnable output weights and bias.

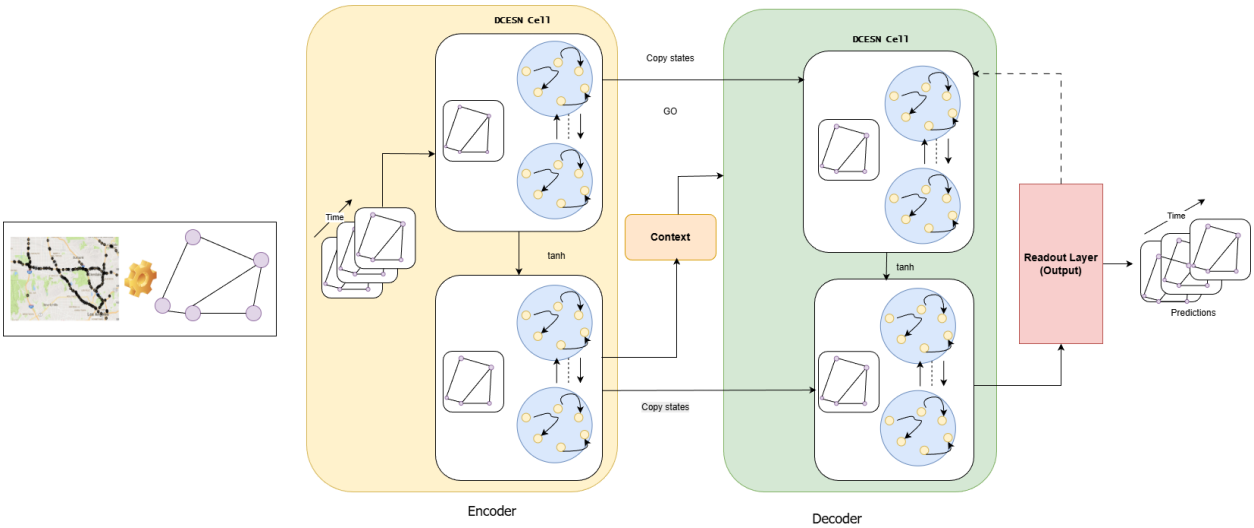


Figure 4: Model Architecture of DC-ESN

3.6 Hybrid Optimization Strategy

The training process involves a hybrid optimization strategy where parameters are partitioned into fixed and trainable sets:

- Fixed: $\Theta_{\text{fix}} = \{\mathbf{W}_{\text{res}}\}$ (Reservoir weights).
- Trainable: $\Theta_{\text{train}} = \{\boldsymbol{W}_{\text{pre}}, \boldsymbol{W}_{\text{post}}, \boldsymbol{W}_{\text{out}}\}$ (Spatial and Readout weights).

We minimize the forecasting loss (e.g., Mean Absolute Error(MAE)) using backpropagation through time with respect to the trainable parameters. Gradients flow through the reservoir states to update the spatial parameters Θ_{train}, but do not update $\mathbf{W}_{\text{res}}$. To mitigate exposure bias, the discrepancy between training (using ground truth) and inference (using model predictions), we employ scheduled sampling (teacher forcing). At iteration i, the decoder receives the ground truth with probability ϵ_i or its own prediction with probability $(1 - \epsilon_i)$. The probability ϵ_i decays from 1 to 0 during training, gradually adapting the model to the inference distribution.

4 Datasets

4.1 Datasets and Preprocessing

The DC-ESN framework was evaluated on two large-scale, real-world traffic benchmarks: METR-LA and PEMS-BAY (Li et al., 2018). The METR-LA data set consists of traffic speed data captured by 207 loop detector sensors on Los Angeles County highways, spanning a four-month period from March 1, 2012, to

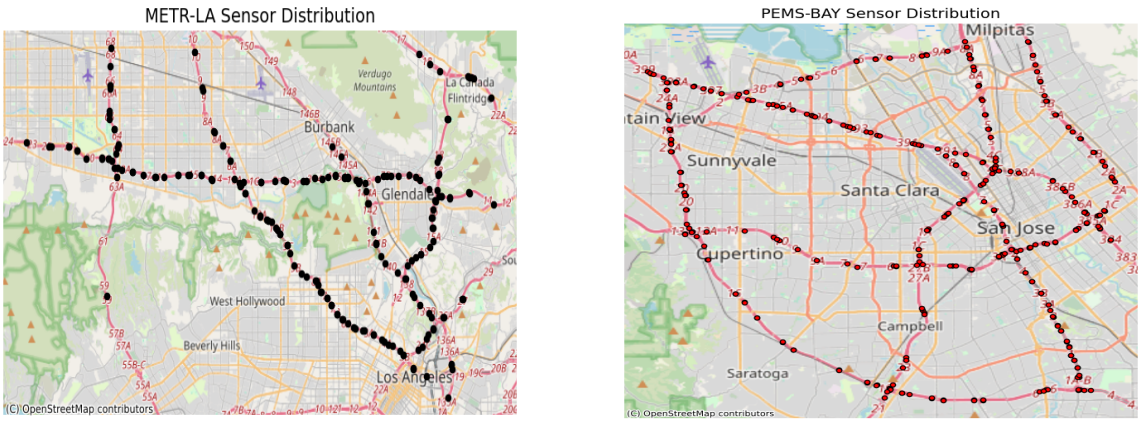


Figure 5: Sensor network of METR-LA(left) and PEMS-BAY(right) datasets.

June 30, 2012 (Jagadish et al., 2014). The PEMS-BAY dataset comprises traffic speed data from 325 sensors in the Bay Area, collected by the California Department of Transportation (CalTrans) over six months, from January 1, 2017, to May 31, 2017. For both datasets, traffic readings were aggregated into 5-minute windows and normalized using Z-Score normalization, with a chronological data split of 70% for training, 10% for validation and 20% for testing. Table 1 summarizes the datasets used in the experiments and figure 5 shows the sensor distribution of both the datasets.

Table 1: Summary of datasets used in experiments

Dataset	Sensors	Time Steps	Time Range	Sample Rate	Features
METR-LA	207	$\approx$ 34,272	Mar 1, 2012 – Jun 30, 2012	5 min	1 (Speed)
PEMS-BAY	325	$\approx$ 52,116	Jan 1, 2017 – May 31, 2017	5 min	1 (Speed)

4.2 Graph Construction

To incorporate the spatial topology of the road network, the sensor network was modeled as a weighted directed graph. The adjacency matrix weights, W_{ij}, were calculated using a thresholded Gaussian kernel based on the road network distance between sensors (Shuman et al., 2013). The weight calculation is defined as:

$$W_{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\right) & \text{if dist}(v_i, v_j) \leq \kappa, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where W_{ij} represents the edge weight between sensors v_i and $v_j \in \mathcal{V}$, $\text{dist}(v_i, v_j)$ is the road network distance, σ is the standard deviation of distances, and κ is the threshold. To compute $\text{dist}(v_i, v_j)$ for each node pair v_i , v_j of the sensor network for METR-LA and PEMS-BAY datasets, each traffic sensor is provided with fixed geographic coordinates (latitude and longitude). These coordinates were first standardized and converted into a geospatial representation. To obtain realistic spatial relationships, the drivable road networks for Los Angeles (METR-LA) and the San Francisco Bay Area (PEMS-BAY) were extracted from OpenStreetMap using the OSMnx library. The resulting transportation networks were represented as directed, weighted graphs in which edges correspond to road segments and weights denote their physical lengths. Each sensor was mapped to its nearest node in the corresponding road graph using a spatial nearest-neighbor search. Road-network distances between sensors were then computed using Dijkstra's shortest-path algorithm, ensuring that inter-sensor distances reflect true drivable routes rather than Euclidean separation. These distances were subsequently used to construct the adjacency matrix for graph-based traffic forecasting models. This

preprocessing step ensures that the spatial structure accurately captures the real transportation network in both regions.

5 Experimental Results and Discussions

5.1 Baseline Comparisons

The performance of DC-ESN was benchmarked against a diverse set of existing models, including traditional statistical methods such as Historical Average (HA), ARIMA with Kalman filtering, Vector Auto-Regression (VAR) (Hamilton, 1994). Additionally, deep learning baselines were employed, specifically Feed Forward Neural Networks (FNN) and Recurrent Neural Networks utilizing fully connected LSTM units (FC-LSTM) and spatiotemporal model mainly DCRNN that uses Diffusion convolution for spatial and GRU for temporal dependency modeling (Sutskever et al., 2014). Evaluation metrics included Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE) across 15-minute, 30-minute, and 1-hour forecasting horizons.

To validate the performance of our approach, we compare it against a comprehensive set of baselines ranging from statistical methods to deep learning frameworks.

5.2 Baseline Implementation Details

To validate the performance of our approach, we compare it against a comprehensive set of baselines ranging from statistical methods to deep learning frameworks. The hyperparameters were selected using the "Tree-structured Parzen Estimator" (Bergstra et al., 2011). The details of configurations for each model are as follows:

- Historical Average (HA): This method models traffic as a seasonal process. It predicts future speeds by calculating the weighted average of the same time slot from previous weeks. Specifically, we use a period of one week and average the data from the preceding four weeks (e.g., predicting a Wednesday 5:00 PM slot using the average of the last four Wednesdays at 5:00 PM).
- ARIMA_{kal}: Auto-Regressive Integrated Moving Average model with Kalman filtering, with model order $(3, 0, 1)$.
- Vector Auto-Regression (VAR): This architecture captures pairwise relationships between sensors. We set the number of time lags to 3, following standard time series analysis approaches (Hamilton, 1994).
- Feed Forward Neural Network (FNN): We employ a fully connected neural network consisting of two hidden layers with 256 units each. Following the experimental configuration in (Li et al., 2018), dropout with probability 0.5 and L2 regularization with weight decay 1×10^{-2} are applied to the hidden layers. The model is trained using the Adam optimizer (Abadi et al., 2016) with a batch size of 64 and MAE as the loss function. The learning rate is initialized at 1×10^{-3} and decays by a factor of 0.1 every 20 epochs starting from epoch 50.
- FC-LSTM: We adopt a sequence-to-sequence architecture based on Long Short-Term Memory (LSTM) units with peephole connections (Sutskever et al., 2014). Following the experimental configuration in (Li et al., 2018), both the encoder and decoder contain two recurrent layers with 256 hidden units each. L1 and L2 regularization with coefficients 2×10^{-5} and 5×10^{-4} are applied to the model parameters. Training is performed using a batch size of 64 with MAE as the loss function. The learning rate is initialized at 1×10^{-4} and decayed by a factor of 0.1 every 10 epochs starting from epoch 20.
- Diffusion Convolutional Recurrent Neural Network (DCRNN): We also include DCRNN (Li et al., 2018) as a strong baseline. The architecture uses an encoder-decoder structure with two recurrent layers, each containing 64 units. The maximum steps for the random walk in the diffusion convolution

(K) is set to 3. Training utilizes scheduled sampling (Bengio et al., 2015) with a reverse sigmoid decay $\epsilon_i = \tau/(\tau + \exp(i/\tau))$, where $\tau = 3000$. The initial learning rate is 1×10^{-2}, decaying by a factor of 0.1 every 10 epochs starting from the 20th epoch.

For all deep learning models (FNN, FC-LSTM, and DCRNN), we employ early stopping based on the validation error to prevent overfitting.

5.3 DC-ESN performance: Results and Ablation Studies

We evaluated the proposed DC-ESN on two benchmark traffic datasets: METR-LA (207 sensors) and PEMS-BAY (325 sensors). We compared the performance against the state-of-the-art DCRNN baseline in terms of prediction accuracy, convergence speed, and computational resource consumption. Experimental results demonstrated that DC-ESN consistently outperformed all baselines other then DCRRN across both datasets and all forecasting horizons, DC-ESN shows comparable results with DCRNN even though it is slightly lighter than DCRNN in terms of trainable parameters and computation with the performance advantage becoming more pronounced in long-term (1-hour) forecasting (Li et al., 2018). Table 2 shows the comparison of DC-ESN with baseline for seed 42 and Table 3 shows the comparison of DC-ESN for different seeds (42, 100, 199, 500) as reservoir initialized randomly. figure 8 and figure 9 show the performance comparison of DC-ESN (for both the datasets) against DCRNN in terms of mean ± standard deviation of metrics for MAE, RMSE, and MAPE across all forecasting horizons for different seeds 42, 100, 199, 500. figure 6 and figure 7 are forecast visualization of DC-ESN and DCRNN on both the dataset (METR-LA,PEMS-BAY) across 3-, 6-, and 12-step prediction horizons.

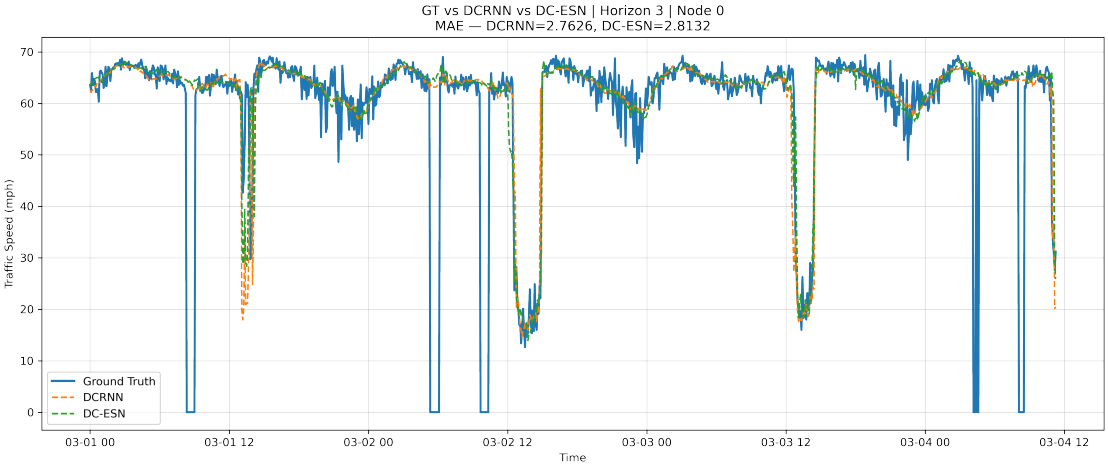
Table 2: Performance comparison of DC-ESN with different approaches for traffic speed forecasting on METR-LA and PEMS-BAY datasets for seed 42.

Horizon	Metric	HA	ARIMA_{kal}	VAR	DC-ESN	FNN (Li et al., 2018)	FC-LSTM (Li et al., 2018)	DCRNN
					METR-LA			
15 min	MAE	4.15	4.02	4.37	**2.81**	3.99	3.44	**2.76**
	RMSE	7.77	8.69	7.78	**5.35**	7.94	6.30	**5.31**
	MAPE	12.90%	9.39%	10.08%	**7.31%**	9.90%	9.60%	**7.14%**
30 min	MAE	4.15	5.09	5.40	**3.23**	4.23	3.77	**3.16**
	RMSE	7.77	11.13	9.37	**6.44**	8.17	7.23	**6.39**
	MAPE	12.90%	12.75%	12.70%	**8.84%**	12.90%	10.90%	**8.42%**
1 hour	MAE	4.15	6.79	6.50	**3.78**	4.49	4.37	**3.65**
	RMSE	7.77	14.21	10.68	**7.74**	8.69	8.69	**7.57**
	MAPE	12.90%	16.71%	15.84%	**10.86%**	14.00%	13.20%	**10.00%**
					PEMS-BAY			
15 min	MAE	2.88	1.60	1.74	**1.35**	2.20	2.05	**1.32**
	RMSE	5.59	3.43	3.09	**2.82**	4.42	4.19	**2.77**
	MAPE	6.76%	3.59%	3.60%	**2.85%**	5.20%	4.80%	**2.75%**
30 min	MAE	2.88	2.18	2.33	**1.71**	2.30	2.20	**1.65**
	RMSE	5.59	4.98	4.15	**3.87**	4.63	4.55	**3.78**
	MAPE	6.76%	4.65%	5.02%	**3.93%**	5.40%	5.20%	**3.75%**
1 hour	MAE	2.88	3.05	2.92	**2.08**	2.46	2.37	**1.97**
	RMSE	5.59	7.02	5.11	**4.79**	4.98	4.96	**4.61**
	MAPE	6.76%	6.84%	6.46%	**5.06%**	5.90%	5.70%	**4.71%**

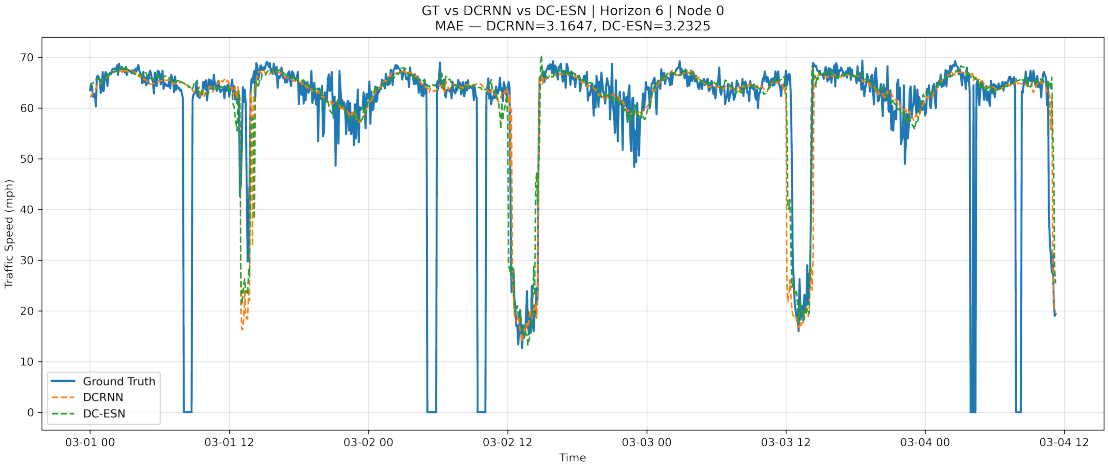
5.4 Performance Analysis

Empirical testing was conducted on an NVIDIA RTX 6000 Ada Generation GPU.

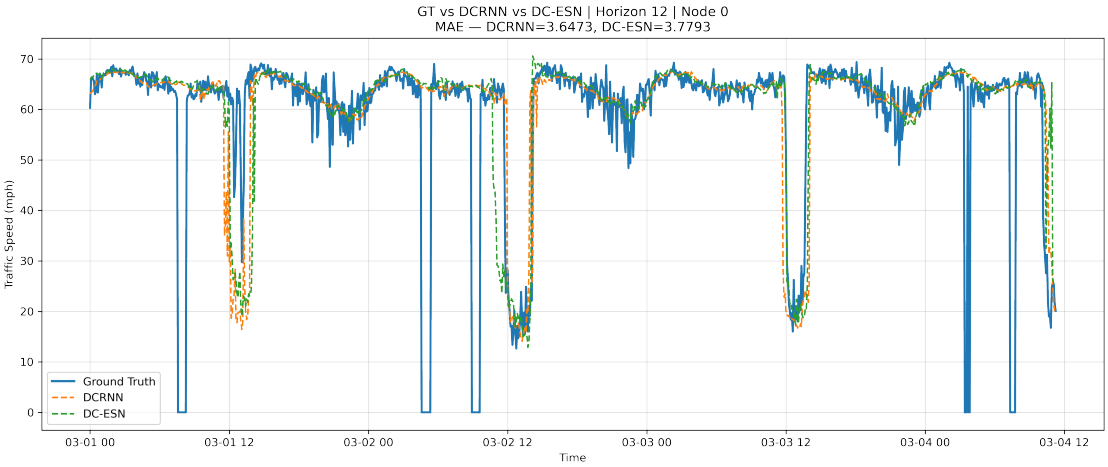
1. Computational Efficiency and Speedup: The proposed DC-ESN demonstrates a significant acceleration in training performance compared to the gated DCRNN baseline. On the METR-LA dataset, the average training latency per epoch was reduced from ∼87.4s to ∼17.8s, representing a ∼4.9× speedup. This efficiency gain is further amplified on the larger PEMS-BAY dataset, where the epoch time dropped



(a) 3-step forecast

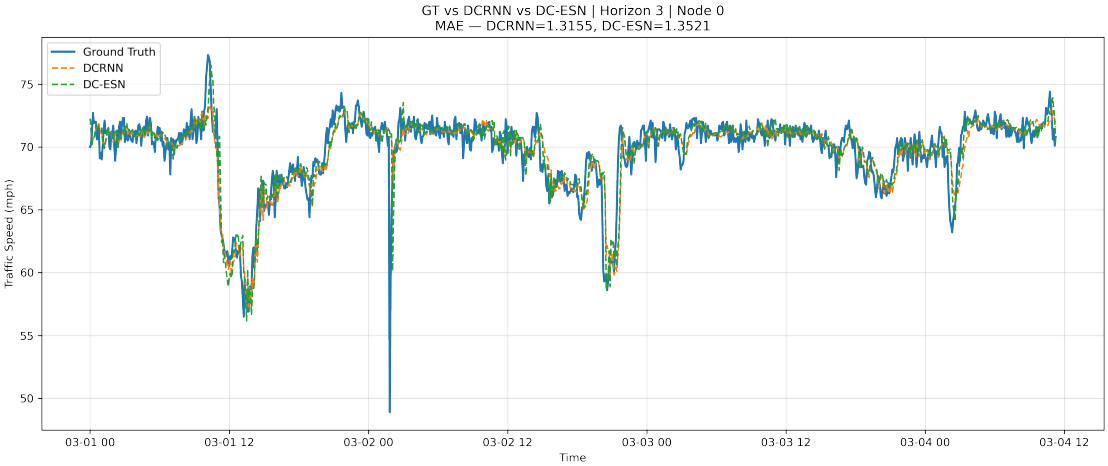


(b) 6-step forecast

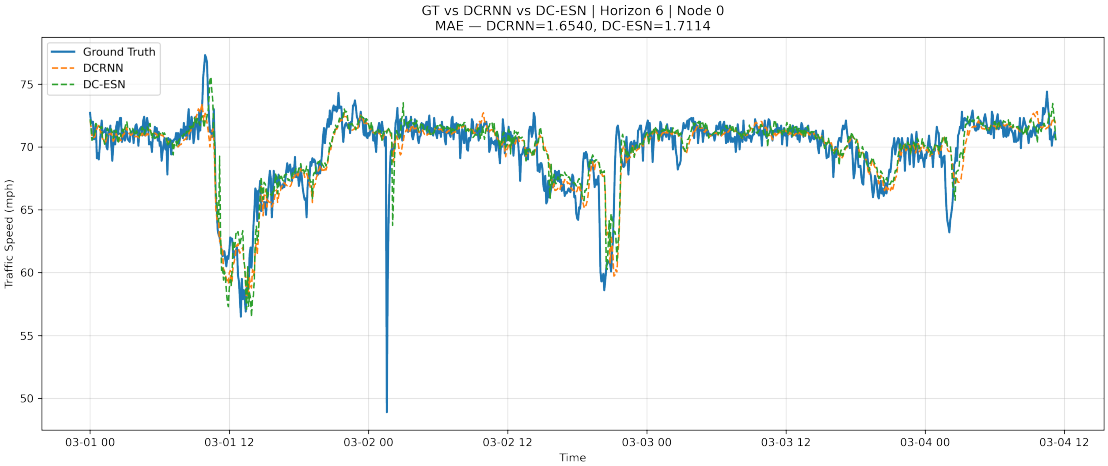


(c) 12-step forecast

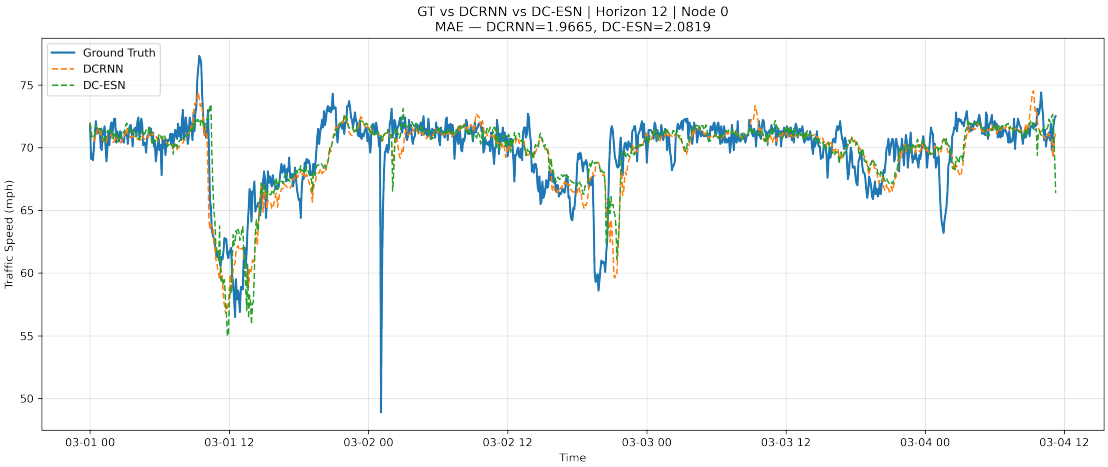
Figure 6: Forecast visualization of DC-ESN and DCRNN on the METR-LA dataset across 3-, 6-, and 12-step prediction horizons.



(a) 3-step forecast



(b) 6-step forecast

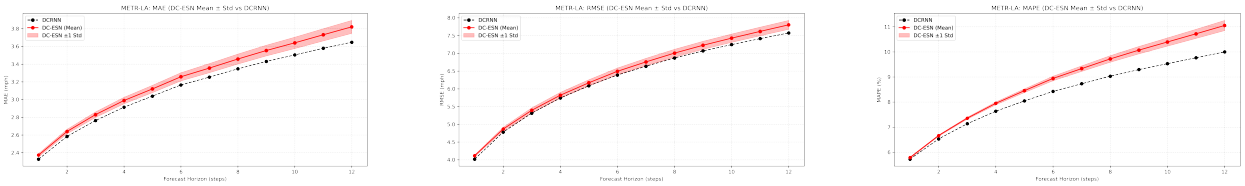


(c) 12-step forecast

Figure 7: Forecast visualization of DC-ESN and DCRNN on the PEMS-BAY dataset across 3-, 6-, and 12-step prediction horizons. Ground truth and predicted traffic speed trajectories are shown.

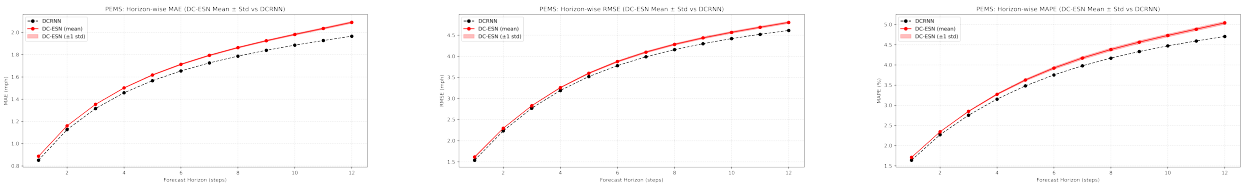
Table 3: Performance comparison of DC-ESN with different approaches. DC-ESN results are reported as mean ± standard deviation across seeds 42, 100, 199, 500.

Horizon	Metric	HA	ARIMA$_{kal}$	VAR	DC-ESN	FNN (Li et al., 2018)	FC-LSTM (Li et al., 2018)	DCRNN
					METR-LA			
15 min	MAE	4.15	4.02	4.37	2.830 ± 0.033	3.99	3.44	**2.763**
	RMSE	7.77	8.69	7.78	5.395 ± 0.075	7.94	6.30	**5.313**
	MAPE	12.90%	9.39%	10.08%	7.358 ± 0.043%	9.9%	9.6%	**7.14%**
30 min	MAE	4.15	5.09	5.40	3.257 ± 0.048	4.23	3.77	**3.165**
	RMSE	7.77	11.13	9.37	6.493 ± 0.097	8.17	7.23	**6.388**
	MAPE	12.90%	12.75%	12.7%	8.942 ± 0.095%	12.9%	10.9%	**8.42%**
1 hour	MAE	4.15	6.79	6.50	3.821 ± 0.075	4.49	4.37	**3.647**
	RMSE	7.77	14.21	10.68	7.803 ± 0.129	8.69	8.69	**7.574**
	MAPE	12.90%	16.71%	15.84%	11.051 ± 0.208%	14.0%	13.2%	**10.00%**
					PEMS-BAY			
15 min	MAE	2.88	1.60	1.74	1.354 ± 0.002	2.20	2.05	**1.316**
	RMSE	5.59	3.43	3.09	2.830 ± 0.006	4.42	4.19	**2.77**
	MAPE	6.76%	3.59%	3.6%	2.852 ± 0.011%	5.2%	4.8%	**2.75%**
30 min	MAE	2.88	2.18	2.33	1.714 ± 0.006	2.30	2.20	**1.654**
	RMSE	5.59	4.98	4.15	3.874 ± 0.021	4.63	4.55	**3.778**
	MAPE	6.76%	4.65%	5.02%	3.925 ± 0.034%	5.4%	5.2%	**3.75%**
1 hour	MAE	2.88	3.05	2.92	2.091 ± 0.010	2.46	2.37	**1.966**
	RMSE	5.59	7.02	5.11	4.801 ± 0.028	4.98	4.96	**4.611**
	MAPE	6.76%	6.84%	6.46%	5.044 ± 0.040%	5.9%	5.7%	**4.71%**



(a) MAE Comparison (b) RMSE Comparison (c) MAPE Comparison

Figure 8: Performance comparison of DC-ESN (for METR-LA dataset) against DCRNN in terms of mean ± standard deviation of metrics for MAE, RMSE, and MAPE across all forecasting horizons for different seeds 42, 100, 199, 500.



(a) MAE Comparison (b) RMSE Comparison (c) MAPE Comparison

Figure 9: Performance comparison of DC-ESN (for PEMS-BAY dataset) against DCRNN in terms of mean ± standard deviation of metrics for MAE, RMSE, and MAPE across all forecasting horizons for different seeds 42, 100, 199, 500.

from ∼164.1s to ∼ 27.8s this shows almost 5.9× improvement. This time complexity reduction comes from the structural design of the Echo State Network; while the model utilizes Backpropagation Through Time (BPTT) for optimization, the recurrent reservoir weights ($\mathbf{W}_{res}$) remain frozen. By restricting gradient computation to the spatial diffusion and output projection layers, this architecture bypasses the complex derivative chains required to train internal gates, significantly reducing the backpropagation overhead..

Table 4: Comparison of Computational Efficiency and Structural Scalability

Metric	**Dataset**	**DCRNN**	**DC-ESN (Ours)**	**Improvement**
Trainable Params	Both	372,353	**106,753**	**71.3% Reduc.**
Peak GPU VRAM	METR-LA	3,545 MB	**1,273 MB**	**64.1% Lower**
	PEMS-BAY	5,554 MB	**1,997 MB**	**64.0% Lower**
Time per Epoch	METR-LA	∼88 s	**∼18 s**	**∼4.9× Faster**
	PEMS-BAY	∼164 s	**∼27 s**	**∼5.9× Faster**

2. Memory Footprint and Edge Viability: The DC-ESN architecture exhibits a substantially lower VRAM requirement, aligning with the "Green AI" objective of reducing the computational price tag of deep learning research (Schwartz et al., 2020). On the PeMS-BAY dataset ($N = 325$), the peak GPU memory usage dropped from 5.55 GB to 1.99 GB, representing a 64% reduction (see Table 4). Such a reduced footprint makes it feasible to deploy and retrain the model on resource-constrained edge devices or consumer-grade GPUs, where the standard DCRNN would otherwise necessitate significantly smaller batch sizes or fail due to memory overflow.

3. Predictive Performance and Stability: A definitive advantage of the Reservoir Computing paradigm is the stability of temporal dynamics. DC-ESN achieves a stable plateau in all experiments. As summarized in Table 5, the DC-ESN retains over 97% of the predictive accuracy of the gated baseline while ensuring a robust near-zero generalization gap.

Table 5: Predictive Performance Comparison on METR-LA and PeMS-BAY

Dataset	**Model**	**Nodes**	**Best Val. MAE**	**Final Val. MAE**	**Stability**
METR-LA	DCRNN	207	**2.822**	4.105	
	DC-ESN	207	2.897	**3.106**	**Stable**
PeMS-BAY	DCRNN	325	**1.628**	2.425	
	DC-ESN	325	1.706	**1.660**	**Stable**

4. Accuracy vs. Efficiency Trade-off: The results suggest that the fixed reservoir serves as a robust "pre-wired" temporal feature extractor. While individual runs may exhibit slight variations based on reservoir initialization, the consistent performance confirms the model's robust stability. The slight MAE trade-off (approx. 2.3%) is accompanied by an order of magnitude increase in training speed, proving the DC-ESN to be a suitable candidate for real-time applications and sustainable AI deployment in urban mobility networks.

6 Conclusion and Future Scope

The experiments conducted in this study confirm that the proposed Diffusion Convolutional Echo State Network (DC-ESN) is a lightweight, high-speed alternative to the standard Diffusion Convolutional Recurrent Neural Network (DCRNN). The model successfully captures complex spatiotemporal dependencies with competitive accuracy, making it a strong candidate for large-scale traffic network monitoring tasks where computational resources are limited.

The proposed architecture provides the distinct advantage of combining rigorous graph-based spatial modeling with lightweight, reservoir-based temporal dynamics. By decoupling the spatial learning from the temporal recurrence, we achieve significant gains in training efficiency without sacrificing the ability to model network topology.

Future Directions

Future work can expand on this framework in three key areas:

- **Scalability :** Testing the scalability of the proposed framework on larger sensor networks with tens of thousands of sensors without slowing down.
- **Graph Echo State Networks (GESN):** Instead of separating spatial and temporal modules, future iterations could integrate network topology directly into the reservoir construction, allowing the reservoir structure to mirror the road network.
- **Symmetry-Aware Modeling :** Investigating ESNs that explicitly model the inherent symmetries in traffic data (e.g., periodicities and reversible flows) could enhance both predictive robustness and interpretability.
- **TinyML Deployment :** The sparse, fixed nature of the DC-ESN makes it ideal for ultra-low-power edge devices. Future research will explore deploying quantized models on microcontrollers and enabling efficient on-device adaptation by retraining only the lightweight readout layer to handle local concept drift.

Broader Impact Statement

This work aims to improve computational efficiency in spatiotemporal forecasting, enabling sustainable AI deployment in intelligent transportation systems. Reduced GPU usage and faster training may lower energy consumption in large-scale urban monitoring systems. However, forecasting systems must be deployed carefully to avoid reinforcing biased mobility policies or inequitable infrastructure planning.

References

Martin Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016. URL `https://arxiv.org/abs/1603.04467`.

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. URL `http://proceedings.mlr.press/v97/abu-el-haija19a/abu-el-haija19a.pdf`.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 28, pp. 1171–1179, 2015.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pp. 2546–2554, 2011.

Shifen Cheng, Feng Lu, and Peng Peng. Short-term traffic forecasting by mining the non-stationarity of spatiotemporal patterns. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6365–6383, 2021. doi: 10.1109/TITS.2020.2991781.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems*, 2016. URL https://api.semanticscholar.org/CorpusID:3016223.

Claudio Gallicchio and Alessio Micheli. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9:337 – 350, 2017. URL https://api.semanticscholar.org/CorpusID:1077549.

Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Design of deep echo state networks. *Neural networks : the official journal of the International Neural Network Society*, 108:33–47, 2018. URL https://api.semanticscholar.org/CorpusID:52075702.

James Douglas Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, NJ, 1994.

Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Yoesoep Rachmad. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pp. 1097–1105, 01 2012. doi: 10.1145/3065386.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997. doi: 10.1162/neco.1997.9.8.1735.

Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks. *GMD Report*, 148, 2001.

H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57:86–94, 2014. URL https://api.semanticscholar.org/CorpusID:2836512.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016. URL https://api.semanticscholar.org/CorpusID:3144218.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1707.01926.

Marco Lippi, Matteo Bertini, and Paolo Frasconi. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):871–882, 2013. doi: 10.1109/TITS.2013.2247040.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

Roy Schwartz, Jesse Dodge, Noah Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63: 54–63, 11 2020. doi: 10.1145/3381831.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. doi: 10.1109/MSP.2012.2235192.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, pp. 3104–3112, 2014.

Shang-Hua Teng. Scalable algorithms for data and network analysis. *Foundations and Trends in Theoretical Computer Science*, 12(1-2):1–274, 12 2016. ISSN 1551-305X. doi: 10.1561/0400000051. URL https://doi.org/10.1561/0400000051.

Zhibo Xing, Mingxia Huang, and Dan Peng. Overview of machine learning-based traffic flow prediction. *Digital Transportation and Safety*, 2:164, 2023. ISSN 2837-7842. doi: 10.48130/DTS-2023-0013. URL http://www.maxapress.com/article/doi/10.48130/DTS-2023-0013.

A Appendix

B Evaluation Metrics and Loss Formulation

To ensure a rigorous and unbiased comparison, the proposed DC-ESN is evaluated using the same methodological framework as the baseline DCRNN. We assess predictive accuracy across a 12-step temporal horizon (5 to 60 minutes) using three industry-standard metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

B.0.1 Standard Masking for Missing Values

Traffic sensor datasets often contain missing entries or recording artifacts, typically represented as zero values. To prevent these null entries from skewing the results—particularly for the MAPE calculation, where a zero value would lead to numerical divergence—we employ the standard masking strategy used in the DCRNN framework. We define the set of observed entries Ω as:

$$\Omega = \{i \mid y_i \neq 0\} \tag{9}$$

. This ensures that only valid, physically recorded traffic states contribute to both the model's optimization and the final error reporting.

B.0.2 Masked Objective Function

The model is optimized using a masked Mean Absolute Error (MAE) loss function, defined as:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} |y_i - \hat{y}_i| \tag{10}$$

By applying the mask Ω during the backward pass of the Backpropagation Through Time (BPTT) process, we ensure that gradients are computed only from valid data points. While this loss formulation is identical to the baseline, the DC-ESN demonstrates good training stability, avoiding the catastrophic gradient failures observed in gated recurrent architectures.

B.0.3 Mathematical Formulations

for given ground truth y_i and predicted state $\hat{y}_i$, the metrics are formulated as:

1. **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{|\Omega|} \sum_{i \in \Omega} |y_i - \hat{y}_i| \tag{11}$$

2. **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{|\Omega|} \sum_{i \in \Omega} (y_i - \hat{y}_i)^2} \tag{12}$$

3. **Mean Absolute Percentage Error (MAPE):**

$$\text{MAPE} = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \tag{13}$$