Covariance-Driven Graph Embedding for Real-Time Traffic State Prediction

Sai Vamsi Alisetti
Department of Computer Science
University of California, Santa Barbara
saivamsi@ucsb.edu

Vikas Kalagi
Department of Computer Science
University of California, Santa Barbara
vikaskalagi@ucsb.edu

Abstract

Accurate traffic forecasting is essential for enhancing urban mobility and enabling intelligent transportation systems. This paper introduces a novel approach to traffic prediction by leveraging temporal graph structures to capture spatial and temporal dependencies within traffic networks. Each graph, defined by its nodes, edges, and adjacency matrix, represents the traffic conditions at a specific time step. Our method employs a feature-level analysis by computing pairwise covariances between nodes in the current graph and those in preceding time steps. To extract meaningful patterns, we perform eigenvalue decomposition to determine the principal eigenvector. Node features are then projected onto this eigenvector, and cosine similarity is utilized to identify the most relevant nodes for each feature. This process ensures a robust and adaptive feature selection mechanism, enabling our approach to effectively capture complex traffic dynamics for more accurate predictions. Please find the code here - GitHub.

1 INTRODUCTION

In the modern era of interconnected systems, traffic prediction plays a pivotal role in optimizing transportation networks, reducing congestion, and enabling smart city applications. Accurate modeling of traffic dynamics is inherently challenging due to the spatial and temporal dependencies among traffic nodes, as well as the influence of external factors. Graph-based models provide a powerful framework to capture these dependencies, where nodes represent traffic points, edges denote connections, and node features encapsulate traffic-related attributes such as speed, flow, and density. In this study, we propose a novel approach to model traffic dynamics by leveraging graph structures with adjacency matrices, where each graph represents a snapshot of the traffic network at a given time instance.

Our methodology focuses on incorporating temporal graphs, $\mathcal{G}^{(t)} = (V, E, \mathbf{A})$, where each node is associated with F features and T time instances. To enhance predictive accuracy, we compute pairwise covariance matrices between nodes in the current graph $\mathcal{G}^{(t+1)}$ and preceding graphs $\mathcal{G}^{(t)}$, perform eigenvalue decomposition, and utilize the principal eigenvectors to select the most relevant nodes based on cosine similarity. This selection process identifies key spatial relationships while dynamically adapting to temporal patterns. Additionally, a learnable attention mechanism refines the spatial weights by accounting for higher-order neighborhood effects, ensuring that our model captures both local and global correlations. The aggregated node features across S time steps are further processed through a multi-layer framework to compute the final embeddings.

By combining graph theory, statistical methods, and attention-based learning, our framework aims to provide an interpretable and robust mechanism for traffic forecasting. The results demonstrate its potential to enhance decision-making for traffic management and urban planning.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

2 LITERATURE REVIEW

GraphSAGE-based Dynamic Spatial–Temporal Graph Convolutional Networks (DST-GCN) [4] have recently gained significant attention in the domain of traffic prediction, primarily due to their ability to capture both spatial dependencies among traffic nodes (e.g., road intersections or sensors) and temporal patterns in traffic flow data. The GraphSAGE [2] framework, which utilizes inductive graph neural networks, aggregates information from neighboring nodes to learn node representations in dynamic graph structures. This is particularly useful for traffic prediction, where the traffic network is inherently dynamic, with varying traffic conditions, sensor readings, and temporal fluctuations. By integrating spatio-temporal features, DST-GCNs not only model the spatial correlations between traffic nodes at a given time but also adapt to the evolving nature of traffic flow over time, enabling accurate short-term and long-term traffic predictions. Recent studies have demonstrated the effectiveness of DST-GCNs in capturing complex interactions between spatial and temporal factors, outperforming traditional traffic forecasting methods that fail to account for dynamic changes in the network.

The Temporal Graph Convolutional Network (T-GCN) [6] has emerged as a powerful approach for traffic prediction by leveraging the temporal dependencies in dynamic traffic networks. T-GCN combines the strengths of graph convolutional networks (GCNs) [3] with temporal modeling, effectively capturing both spatial correlations between traffic nodes (e.g., road segments or intersections) and their time-varying behaviors. Unlike traditional methods that treat traffic flow as static or independent at each time step, T-GCN dynamically learns the temporal evolution of traffic patterns by applying graph convolutions along with recurrent layers or temporal convolutions to model long-range dependencies. This allows T-GCN to handle irregularities in traffic data, such as peak hours or unexpected events, by learning from historical traffic conditions. Several studies have shown that T-GCN significantly improves traffic forecasting accuracy, particularly in complex urban environments, by providing a more holistic representation of traffic dynamics that traditional methods cannot capture.

Towards Dynamic Spatial-Temporal Graph Learning: A Decoupled Perspective [5] presents an innovative approach to modeling dynamic graph structures by separately addressing the spatial and temporal aspects of graph learning. This decoupled perspective aims to overcome the limitations of traditional models that treat spatial and temporal dependencies as a single, unified entity. By decoupling the spatial and temporal components, the approach allows for more flexible and efficient learning of dynamic graphs, particularly in scenarios where spatial and temporal dynamics evolve at different rates. This method enhances the model's ability to capture the complex interactions within dynamic systems, such as traffic flow or social networks, where spatial patterns (e.g., geographical proximity) and temporal patterns (e.g., time-dependent behavior) evolve independently. The proposed framework has been shown to improve prediction accuracy in tasks involving dynamic graph data by providing a more modular and interpretable approach to learning these dual dependencies, offering new insights into handling real-time dynamic systems with greater precision.

3 PROBLEM FORMULATION

In the context of traffic modeling, we represent the traffic network as a sequence of temporal graphs $\mathcal{G}^{(t)} = (V, E, \mathbf{A})$, where V is the set of nodes, E is the set of edges, and $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$ is the adjacency matrix. Each graph captures the state of the traffic network at a specific time step t, and we have T instances of such graphs representing the temporal evolution of the network. Each node $v_i \in V$ is associated with a feature vector $\mathbf{x}_i^{(t)} \in \mathbb{R}^F$, where F represents the number of features encapsulating traffic-related characteristics such as speed, flow, and density.

For each node $v_i^{(t+1)}$ in the current graph $\mathcal{G}^{(t+1)}$, we compute the covariance matrix $\mathbf{C}_i^{(t+1)} \in \mathbb{R}^{F\times F}$ with respect to the features of all nodes $v_j^{(t)} \in \mathcal{G}^{(t)}$. Mathematically, this is expressed as:

$$\mathbf{C}_i^{(t+1)} = \frac{1}{|V^{(t)}|} \sum_{j\in V^{(t)}} \left(\mathbf{x}_j^{(t)} - \bar{\mathbf{x}}_i^{(t+1)}\right) \left(\mathbf{x}_j^{(t)} - \bar{\mathbf{x}}_i^{(t+1)}\right)^\top,$$

where $\bar{\mathbf{x}}_i^{(t+1)}$ is the mean feature vector for node $v_i^{(t+1)}$ across all time steps t. Using $\mathbf{C}_i^{(t+1)}$, eigenvalue decomposition is performed:

$$\mathbf{C}_i^{(t+1)}\mathbf{u}_k = \lambda_k \mathbf{u}_k, \quad k = 1, 2, \ldots, F,$$

where λ_k are the eigenvalues (sorted in descending order) and $\mathbf{u}_k$ are the corresponding eigenvectors. The K largest eigenvalues and their eigenvectors are selected, and each node's feature vector is projected onto these eigenvectors:

$$\mathbf{p}_i^{(t+1)} = \sum_{k=1}^{K} (\mathbf{x}_i^{(t+1)} \cdot \mathbf{u}_k)\mathbf{u}_k,$$

where $\mathbf{p}_i^{(t+1)}$ represents the projected features for node $v_i^{(t+1)}$. Similarly, projections $\mathbf{p}_j^{(t)}$ are computed for all nodes $v_j^{(t)} \in \mathcal{G}^{(t)}$. The similarity between the current node and previous nodes is measured using cosine similarity:

$$\mathrm{sim}(\mathbf{p}_i^{(t+1)}, \mathbf{p}_j^{(t)}) = \frac{\mathbf{p}_i^{(t+1)} \cdot \mathbf{p}_j^{(t)}}{\|\mathbf{p}_i^{(t+1)}\|\|\mathbf{p}_j^{(t)}\|}.$$

Nodes $v_j^{(t)}$ with similarity greater than a threshold τ are selected, forming the set $S(v_i)$:

$$S(v_i) = \{v_j^{(t)} \in \mathcal{G}^{(t)} : \mathrm{sim}(\mathbf{p}_i^{(t+1)}, \mathbf{p}_j^{(t)}) > \tau\}.$$

This set of selected nodes is used to define spatial weights $W_{i,j}$, which will further influence the aggregation of features. The final weights and feature updates will be detailed in subsequent sections. Repeat the process for all nodes $v \in V$ in G_{t+1}. For each node v, identify its corresponding set of relevant nodes $S(v)$ from the previous graph G_t. Refer to 1 algorithm. Here's the enhanced problem statement incorporating the weights learning based on attention mechanism section into the earlier description:

3.1 Weights Learning Based on Attention Mechanism

For each node $v_i \in V$ in G_{t+1}, compute weights W_{ij} for all nodes $v_j \in S(v_i)$ (i.e., the selected relevant nodes from G_t) using an **attention mechanism**. The attention weights are computed as follows:

3.2 Spatial Correlation via Attention

The spatial correlation between the features of node v_i in G_{t+1} and node v_j in G_t is given by:

$$\rho_{ij}^{(k)} = \mathrm{LeakyReLU}\left(\mathbf{a}^\top \cdot \mathrm{CONCAT}(\mathbf{U}\mathbf{x}_i, \mathbf{U}\mathbf{x}_j)\right)$$

where: - $\mathbf{U}$ is a transformation matrix applied to the feature vectors, - $\mathrm{CONCAT}(\cdot,\cdot)$ denotes the concatenation of transformed feature vectors, - $\mathbf{a}$ is a learnable attention vector, - LeakyReLU is the activation function applied to the attention computation.

3.3 Weight Coefficient Normalization

Normalize the attention scores using the softmax function:

$$W_{ij}^{(k)} = \frac{\exp(\rho_{ij}^{(k)})}{\sum_{u \in S(v_i)} \exp(\rho_{iu}^{(k)})}$$

This normalization ensures that the weights sum to 1 across the selected nodes. 2 algorithm

Algorithm 1 Feature Embedding and Node Selection for Traffic Graphs

Input: Sequence of temporal graphs $\{\mathcal{G}^{(t)} = (V, E, \mathbf{A})\}_{t=0}^{T-1}$, Node features $\mathbf{x}_i^{(t)} \in \mathbb{R}^F$, Threshold τ, Number of top eigenvectors K.
Output: Set of selected nodes $S(v_i)$ for each node v_i in $\mathcal{G}^{(t+1)}$.
for each node $v_i \in V^{(t+1)}$ **do**
  Initialize $S(v_i) = \emptyset$
  Compute the mean feature vector:

$$\bar{\mathbf{x}}_i^{(t+1)} = \frac{1}{|V^{(t)}|} \sum_{j \in V^{(t)}} \mathbf{x}_j^{(t)}$$

  Compute the covariance matrix:

$$\mathbf{C}_i^{(t+1)} = \frac{1}{|V^{(t)}|} \sum_{j \in V^{(t)}} \left(\mathbf{x}_j^{(t)} - \bar{\mathbf{x}}_i^{(t+1)}\right) \left(\mathbf{x}_j^{(t)} - \bar{\mathbf{x}}_i^{(t+1)}\right)^\top$$

  Perform eigenvalue decomposition:

$$\mathbf{C}_i^{(t+1)} \mathbf{u}_k = \lambda_k \mathbf{u}_k, \quad k = 1, 2, \ldots, F$$

  Sort eigenvalues λ_k in descending order and select the top K eigenvectors $\{\mathbf{u}_k\}_{k=1}^K$
  Project current node features onto eigenvectors:

$$\mathbf{p}_i^{(t+1)} = \sum_{k=1}^{K} (\mathbf{x}_i^{(t+1)} \cdot \mathbf{u}_k) \mathbf{u}_k$$

  for each node $v_j \in V^{(t)}$ **do**
    Project features of v_j onto eigenvectors:

$$\mathbf{p}_j^{(t)} = \sum_{k=1}^{K} (\mathbf{x}_j^{(t)} \cdot \mathbf{u}_k) \mathbf{u}_k$$

    Compute cosine similarity:

$$\text{sim}(\mathbf{p}_i^{(t+1)}, \mathbf{p}_j^{(t)}) = \frac{\mathbf{p}_i^{(t+1)} \cdot \mathbf{p}_j^{(t)}}{\|\mathbf{p}_i^{(t+1)}\| \|\mathbf{p}_j^{(t)}\|}$$

    if sim $> \tau$ **then**
      Add v_j to $S(v_i)$
    end if
  end for
end for
return $\{S(v_i) : v_i \in V^{(t+1)}\}$

3.4 Global Spatial Weight Matrix

Combine weights across all feature dimensions k to construct a global weight matrix:

$$W = \sum_{k=1}^{F} W^{(k)}$$

3.5 Multi-head Attention for Robustness

In practice, use multi-head attention to improve the stability and robustness of weight learning. Let M denote the number of attention heads. The final weight coefficients are averaged across all attention heads:

$$W_{ij} = \frac{1}{M} \sum_{m=1}^{M} W_{ij,m}$$

where $W_{ij,m}$ is the weight computed by the m-th attention module. 3 algorithm

Given consecutive input feature matrices $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(S-1)}$, where $\mathbf{X}^{(t)}$ represents the feature vectors of all nodes at time t, and the spatial weights $\mathbf{W}$ obtained using the attention mechanism, we aim to generate embeddings $\widetilde{\mathbf{X}}^{(S)} = \{\widetilde{\mathbf{x}}_i^{(S)} \mid \forall i \in V\}$ that summarize the temporal and spatial relationships. The process begins with feature aggregation at each time step t. For each node $v_i \in V$, features from the selected nodes $v_j \in S(v_i)$, as determined in the earlier steps, are aggregated using the spatial weight matrix $\mathbf{W}$. The aggregated feature vector $\mathbf{h}_i$ is computed as:

$$\mathbf{h}_i = \sum_{j \in S(v_i)} W_{i,j} \mathbf{x}_j^{(t)},$$

where $W_{i,j}$ represents the spatial weight between node v_i and a selected node $v_j \in S(v_i)$, and $\mathbf{x}_j^{(t)}$ is the feature vector of node v_j at time t. 4 algorithm. For each node v_i, the current feature $\mathbf{x}_i^{(t)}$ is concatenated with the aggregated features $\mathbf{h}_i$, resulting in the concatenated feature vector:

$$\mathbf{h}_i^{\text{cat}} = \text{CONCAT}\left(\mathbf{x}_i^{(t)}, \mathbf{h}_i\right).$$

The concatenated feature vector $\mathbf{h}_i^{\text{cat}} \in \mathbb{R}^{2F}$ (since it combines $\mathbf{x}_i^{(t)} \in \mathbb{R}^F$ and $\mathbf{h}_i \in \mathbb{R}^F$) is then passed through a fully connected layer to produce a transformed feature representation for node v_i. This transformation is given by:

$$\mathbf{h}_i^{(t)} = \sigma\left(\mathbf{h}_i^{\text{cat}} \mathbf{U}_t + \mathbf{b}_t\right),$$

where $\mathbf{U}_t \in \mathbb{R}^{2F \times F'}$ is a learnable weight matrix, $\mathbf{b}_t \in \mathbb{R}^{F'}$ is a learnable bias vector, and $\sigma(\cdot)$ is an activation function (e.g., ReLU). The transformed feature vector $\mathbf{h}_i^{(t)} \in \mathbb{R}^{F'}$ represents the embedding of node v_i at time t. This process is repeated for S consecutive time steps, 5 algorithm and the final temporal-spatial embedding $\widetilde{\mathbf{x}}_i^{(S)}$ for node v_i is computed by summing the transformed embeddings over all S time steps:

$$\widetilde{\mathbf{x}}_i^{(S)} = \sum_{t=0}^{S-1} \mathbf{h}_i^{(t)}.$$

Refer to 6 algorithm.The final embedding matrix $\widetilde{\mathbf{X}}^{(S)} \in \mathbb{R}^{N \times F'}$ is obtained by aggregating the embeddings of all nodes in the graph, where each row corresponds to the embedding $\widetilde{\mathbf{x}}_i^{(S)}$ of a node $v_i \in V$. By restricting the aggregation to nodes in $S(v_i)$, this approach incorporates only the most relevant spatial and temporal relationships for each node, as determined by the cosine similarity thresholding in the earlier steps. This ensures that the final embeddings capture meaningful patterns in both space and time, making them suitable for downstream tasks such as prediction, clustering, or anomaly detection.

Algorithm 2 Part 1: Attention Weight Computation

Input: Feature matrices $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(S-1)}$ for S time steps. Each $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times F}$ represents the node features at time step t.
Output: Attention weights matrix W_{ij}.
for $t = 0$ **to** $S - 1$ **do**
    For each time step t**:**
    for each node $v_i \in V$ **do**
        Step A: *Compute attention weights for each selected node* $v_j \in S(v_i)$*:*

$$W_{ij}^{(k)} = \frac{\exp(\rho_{ij}^{(k)})}{\sum_{u \in S(v_i)} \exp(\rho_{iu}^{(k)})}$$

        where $\rho_{ij}^{(k)}$ is computed as:

$$\rho_{ij}^{(k)} = \text{LeakyReLU}\left(\mathbf{a}^{\top} \cdot \text{CONCAT}(\mathbf{U}\mathbf{x}_i^{(t)}, \mathbf{U}\mathbf{x}_j^{(t)})\right)$$

    end for
end for

Algorithm 3 Part 2: Weight Normalization and Multi-head Attention

Input: Attention weights matrix W_{ij}.
Output: Normalized weights matrix W_{ij} and multi-head attention weights.
for $t = 0$ **to** $S - 1$ **do**
    For each time step t**:**
    for each node $v_i \in V$ **do**
        Step B: *Normalize the attention scores using the softmax function:*

$$W_{ij}^{(k)} = \frac{\exp(\rho_{ij}^{(k)})}{\sum_{u \in S(v_i)} \exp(\rho_{iu}^{(k)})}$$

    end for
end for
Step C: *Apply Multi-head Attention to improve robustness:*

$$W_{ij} = \frac{1}{M} \sum_{m=1}^{M} W_{ij,m}$$

where M is the number of attention heads, and $W_{ij,m}$ is the weight computed by the m-th attention head.

4 PRELIMINARY RESULTS

In our preliminary experiments, we explored a modified version of the graph convolutional method which includes the temporal covariance of the nodes into the convolutional operation. Equation 1 shows the original formulation of graph convolutions, where the output $H^{(l+1)}$ is computed as a function of the input feature matrix $H^{(l)}$, the weight matrix $W^{(l)}$, and the normalized Laplacian matrix L_{sym}.

$$H^{(l+1)} = \sigma(L_{\text{sym}} H^{(l)} W^{(l)}) \tag{1}$$

Algorithm 4 Part 3: Feature Aggregation

Input: Feature matrix $\mathbf{X}^{(t)}$ and normalized attention weights W_{ij}.
Output: Aggregated feature vector $\mathbf{h}_i^{(t)}$.
for $t = 0$ **to** $S - 1$ **do**
  For each time step t**:**
  for each node $v_i \in V$ **do**
    Step D: *Aggregate features from selected neighbors* $v_j \in S(v_i)$ *using the attention weights:*

$$\mathbf{h}_i^{(t)} = \sum_{j \in S(v_i)} W_{ij} \mathbf{x}_j^{(t)}$$

  end for
end for

Algorithm 5 Part 4: Feature Concatenation and Transformation

Input: Original feature vector $\mathbf{x}_i^{(t)}$ and aggregated feature $\mathbf{h}_i^{(t)}$.
Output: Transformed feature vector $\mathbf{h}_i^{(t)}$.
for $t = 0$ **to** $S - 1$ **do**
  For each time step t**:**
  for each node $v_i \in V$ **do**
    Step E: *Concatenate the original feature with the aggregated feature:*

$$\mathbf{h}_i^{\text{cat}} = \text{CONCAT}(\mathbf{x}_i^{(t)}, \mathbf{h}_i^{(t)})$$

    Step F: *Transform the concatenated feature via a fully connected layer:*

$$\mathbf{h}_i^{(t)} = \sigma\left(\mathbf{h}_i^{\text{cat}} \mathbf{U}_t + \mathbf{b}_t\right)$$

    where $\mathbf{U}_t \in \mathbb{R}^{2F \times F'}$ and $\mathbf{b}_t \in \mathbb{R}^{F'}$ are learnable parameters, and $\sigma(\cdot)$ is an activation function (e.g., ReLU).
  end for
end for

Algorithm 6 Part 5: Temporal-Spatial Embedding Generation

Input: Transformed feature vectors $\mathbf{h}_i^{(t)}$ for all time steps.
Output: Final temporal-spatial embedding $\widetilde{\mathbf{x}}_i^{(S)}$.
for each node $v_i \in V$ **do**
  Step G: *Generate final temporal-spatial embedding by aggregating the transformed embeddings over all time steps:*

$$\widetilde{\mathbf{x}}_i^{(S)} = \sum_{t=0}^{S-1} \mathbf{h}_i^{(t)}$$

end for
Step H: *Construct the final embedding matrix:*

$$\widetilde{\mathbf{X}}^{(S)} = [\widetilde{\mathbf{x}}_1^{(S)}, \widetilde{\mathbf{x}}_2^{(S)}, \ldots, \widetilde{\mathbf{x}}_N^{(S)}]$$

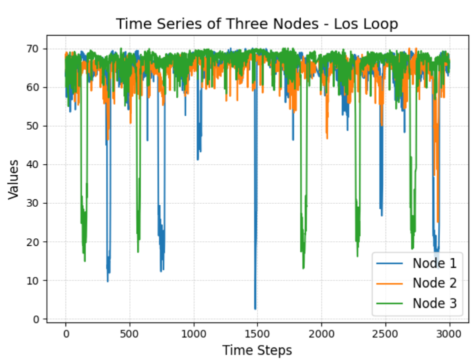
To further enhance the model's ability to capture temporal dependencies between nodes, we incorporated covariance-based embeddings, as proposed in the recent work by [1]. Equation 2 illustrates the updated approach, where we apply element-wise masks M_l and M_c to the Laplacian matrix L_{sym} and the covariance matrix C, respectively. These masks selectively weight or zero out certain contributions, allowing the model to learn more robust representations.

$$H^{(l+1)} = \sigma \left(\left| \frac{M_l \odot L_{\text{sym}} + M_c \odot C}{M_l \odot L_{\text{sym}} + M_c \odot C} \right| H^{(l)} W^{(l)} \right) \tag{2}$$

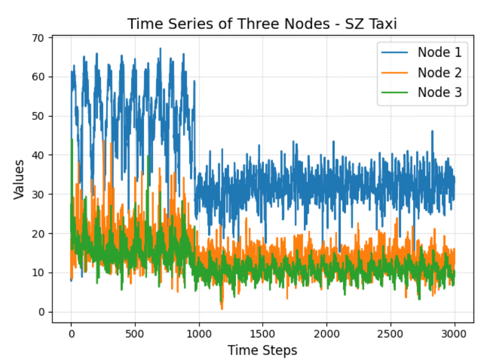
By incorporating the covariance matrix C, which models the temporal dependencies between nodes, we aim to improve the model's performance on tasks that require understanding the dynamic relationships within the data. In the following, we present our experimental results comparing the performance of the original graph convolutional method and the modified version with the covariance-based embeddings. This will allow us to assess the impact of the proposed changes and their potential to enhance the model's ability to capture complex spatio-temporal patterns in the data.

4.1 About the data

1. **SZ-taxi Dataset**: This dataset contains taxi trajectory data from Shenzhen, collected between January 1 and January 31, 2015. The study focuses on 156 major roads in the Luohu District. The dataset comprises two main components: an adjacency matrix and a feature matrix. The 156×156 adjacency matrix represents the spatial relationships between roads, where each row corresponds to a road, and the values of the matrix indicate the connectivity between them. The feature matrix captures changes in traffic speed over time for each road. Each row corresponds to a road, while each column represents the aggregated traffic speed in 15-minute intervals.
2. **Los-loop Dataset**: This dataset was gathered from real-time loop detectors on highways in Los Angeles County between March 1 and March 7, 2012. It includes data from 207 sensors and records their traffic speed, aggregated every 5 minutes. Similarly to the SZ-taxi dataset, it comprises an adjacency matrix and a feature matrix. The adjacency matrix is based on the distances between the sensors within the traffic network. Since the dataset contained missing values, a linear interpolation method was applied to fill the gaps.



(a) **Avg Speed (Kmph) vs Time (min) Los-Loop**



(b) **Avg Speed (Kmph) vs Time (min) SZ-Taxi**

Figure 1: Time Series data of first three nodes

Both datasets exhibit distinct characteristics. The Los-Loop dataset demonstrates relatively stable patterns with minimal distributional shifts, as evident from the data of the first three nodes shown in Figure 1a. In contrast, the SZ-Taxi dataset shows noticeable distributional shifts, with a clear shift in the mean over time, as illustrated in Figure 1b.

4.2 Experiments

We leveraged the code bases of TGCN [6] and Spatio-Temporal Covariance Neural Networks [1], for reference and combined the batch covariance calculation from the latter into the former. For the experiments, we perform forecasting the average speed for 15min, 30min, 45min, 60min respectively for both the models and evaluate the model performance on certain metrics to compare the model performances. We show that the cVTGCN model which includes temporal covariance embeddings performs better than its vanilla variant.

4.2.1 Model Design

We employ the TGCN model as the foundation, enhancing it with an additional overlay layer that incorporates covariance calculations and masking operations. The key hyperparameters of the TGCN model include the learning rate, batch size, number of training epochs, and the number of hidden layers. For this experiment, these hyperparameters were fine-tuned and set to the following values: a learning rate of 0.001, a batch size of 64, and 100 training epochs.

During the training process, 80% of the overall dataset is used as the training set, while the remaining 20% is reserved for testing. The TGCN model is trained using the Adam optimizer to ensure efficient and stable convergence.

To assess the prediction performance of the TGCN model, we evaluate the difference between the actual traffic information Y_t and the predicted values $\hat{Y}_t$ using three metrics. These metrics provide a comprehensive analysis of the model's accuracy and robustness.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_t - \hat{Y}_t\right)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_t - \hat{Y}_t\right|$$

$$Accuracy = 1 - \frac{\|Y - \hat{Y}\|_F}{\|Y\|_F}$$

4.2.2 Loss Function

During training, the objective is to minimize the discrepancy between the actual and predicted traffic speeds, denoted as Y_t and $\hat{Y}_t$, respectively. The loss function for the T-GCN model is defined as:

$$\text{Loss} = \|Y_t - \hat{Y}_t\|^2 + \lambda L_{\text{reg}} + \mu L_{\text{cov}} \tag{3}$$

The first term minimizes the prediction error, ensuring the model accurately captures traffic dynamics. The second term, L_{reg}, is an L_2 regularization term to mitigate overfitting, with λ as its weight. The third term, L_{cov}, introduces covariance-based regularization to capture temporal dependencies effectively, weighted by the hyperparameter μ. The covariance-based regularization term, L_{cov}, is defined as the Frobenius norm of the difference between the predicted and observed covariance matrices:

$$L_{\text{cov}} = \|\mathbf{C}_{\text{pred}} - \mathbf{C}_{\text{obs}}\|_F^2$$

where:

- $\mathbf{C}_{\text{pred}}$ is the predicted covariance matrix of traffic speeds.
- $\mathbf{C}_{\text{obs}}$ is the observed covariance matrix of traffic speeds.
- $\|\cdot\|_F$ represents the Frobenius norm.

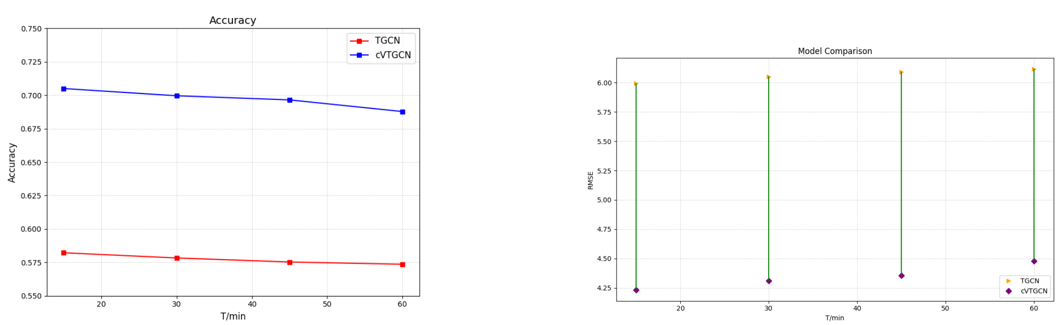
By incorporating L_{cov}, the model aligns predictions with observed temporal covariance patterns, enhancing its ability to generalize and capture complex traffic dynamics.

4.3 Analysis of results

From Table 1, it is evident that cVTGCN outperforms significantly on the SZ-Taxi dataset, while the improvement in accuracy for the Los-Loop dataset is comparatively marginal. This highlights cVTGCN's ability to effectively capture distributional shifts, as the incorporation of covariance embeddings within the convolutional operations enables the model to identify more nuanced patterns in the time series data.

Table 1: Validation Metrics at Different Time Intervals

T	Metric	Los-loop		SZ-taxi	
		TGCN	cVTGCN (Ours)	TGCN	cVTGCN (Ours)
15min	RMSE	5.3861	5.2665	5.9953	4.2328
	MAE	3.5388	3.2937	4.4205	2.9625
	Accuracy	0.9083	0.9103	0.5822	0.7050
30min	RMSE	7.6295	6.3116	6.0514	4.3108
	MAE	5.0990	3.8155	4.4882	3.0719
	Accuracy	0.8701	0.8925	0.5783	0.6996
45min	RMSE	8.3162	7.1820	6.0921	4.3536
	MAE	5.4171	4.3116	4.5216	3.1274
	Accuracy	0.8583	0.8777	0.5753	0.6965
60min	RMSE	8.8854	7.9028	6.1159	4.4775
	MAE	5.7577	4.7294	4.5468	3.2194
	Accuracy	0.8486	0.8653	0.5736	0.6878



(a) **Accuracy** (b) **RMSE**

Figure 2: Comparison of methods with SZ-Taxi

From Figure 3, it is evident that the blue line representing cVTGCN aligns more closely with the mean compared to TGCN, indicating its improved ability to capture patterns. Similarly, in Figure 4, the blue line for cVTGCN demonstrates a more pronounced shift toward the mean, further validating its superior capability in capturing distributional shifts.

Figure 3: Target vs Prediction (Los-Loop): Target 12 timesteps - Output 12 timesteps

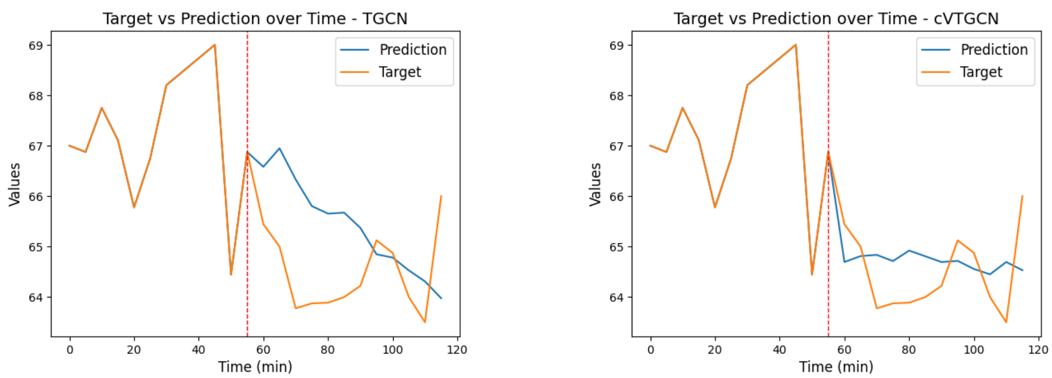


Figure 4: Input vs Prediction (SZ-Taxi): Target 12 timesteps - Output 12 timesteps

5 CONTRIBUTION STATEMENT

The contributions to this work are as follows:

- **Ideation**: Vamsi and Vikas
- **Presentation Slides**: Vamsi and Vikas
- **Literature Review:** Vamsi and Vikas
- **Code Implementation / Results:** Vamsi
- **Report Writing:** Vikas

References

[1] Andrea Cavallo, Mohammad Sabbaqi, and Elvin Isufi. Spatiotemporal covariance neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 18–34. Springer, 2024.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[3] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[4] Tao Liu, Aimin Jiang, Jia Zhou, Min Li, and Hon Keung Kwan. Graphsage-based dynamic spatial–temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):11210–11224, 2023.

[5] Binwu Wang, Pengkun Wang, Yudong Zhang, Xu Wang, Zhengyang Zhou, Lei Bai, and Yang Wang. Towards dynamic spatial-temporal graph learning: A decoupled perspective. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8):9089–9097, Mar. 2024.

[6] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9):3848–3858, 2019.