COMP 4222 Project Report

Spatio-Temporal Attention Graph Network (STAGN) for Traffic Prediction

Authors

November 21, 2025

Contents

1 Introduction

Urban traffic systems have become increasingly complex, prompting the need for innovative traffic prediction methods. Traditional approaches often rely on historical data and fixed sensor placements, which can limit their effectiveness.

While GMAN: A Graph Multi-Attention Network for Traffic Prediction marks a substantial advancement in this field, there remain critical areas for enhancement. This proposal seeks to refine GMAN by addressing these limitations, thereby amplifying its efficacy in traffic forecasting. Enhancing GMAN is vital for smart city initiatives, advanced traffic management systems, and comprehensive urban planning. By improving the precision of traffic predictions, we can significantly alleviate congestion, optimize transportation resources, and elevate overall urban mobility.

2 Background

GMAN's architecture incorporates a sophisticated multi-attention mechanism that lets the model focus on various dimensions of the input data simultaneously. This capability is crucial for comprehending the intricate interactions within a road network over time. Its encoder-decoder structure processes input traffic features through multiple attention blocks, effectively capturing both local and global dependencies. The encoder assimilates pertinent historical information, while the decoder forecasts future traffic states. [1]

2.1 Spatio-temportal Embeddings

The Spatio-Temporal (ST) Embedding in GMAN effectively captures both spatial and temporal dynamics in traffic prediction. It integrates information from the road network structure and time features to enhance the model's predictive capabilities.

This component encodes the graph structure of the road network. It uses the node2vec approach to learn vertex representations, which are then passed through a two-layer fully-connected neural network to generate spatial embeddings. The resulting spatial embedding for each vertex v_i is represented as $e_{S_{v_i}} \in \mathbb{R}^D$, capturing static relationships among traffic sensors.

To account for time-dependent variations in traffic conditions, the temporal embedding encodes each time step into a vector. It employs one-hot encoding for the day of the week and time of day, concatenating these into a single vector. This vector is also transformed using a two-layer fully-connected neural network, yielding $e_{T_{t_j}} \in \mathbb{R}^D$ for each time step t_j.

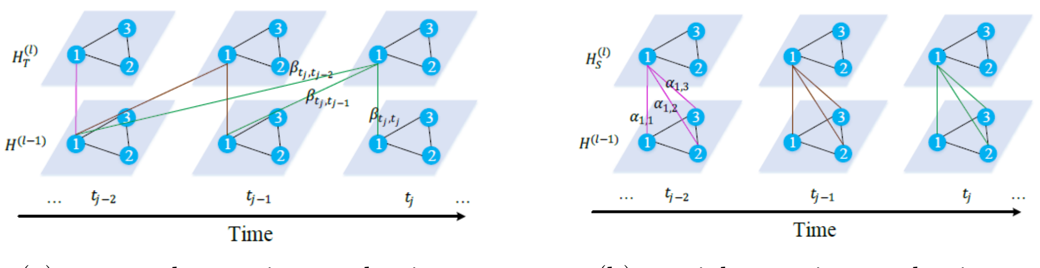
The final STE combines both spatial and temporal embeddings for each vertex at a given time step. For vertex v_i at time step t_j, it is defined as:

$$e_{v_i,t_j} = e_{S_{v_i}} + e_{T_{t_j}}$$

2.2 Spatio-Temporal Attention

The model also features a transformative attention layer that bridges the encoder and decoder, enhancing prediction accuracy and mitigating error propagation. This innovative approach empowers GMAN to adapt seamlessly to fluctuating traffic conditions influenced by diverse factors such as weather variations, special events, and road closures.

The attention layer accounts for both spatial and temporal attention of the traffic graph. For temporal attention, since the traffic condition depends on the previous time step observation of the same place, the temporal attention mechanism is designed to capture non-linear relation between the traffic condition between different time steps of the same place. For spatial attention, the model learns the spatial dependencies between different places in the traffic across the same time step.



(a) temporal attention mechanism (b) spatial attention mechanism

Figure 1: Spatio-temporal attention mechanism in GMAN

The encoder and the decoder of GMAN consists of spatio-temporal attention blocks, which consists of the spatial attention mechanism and the temporal attention mechanism. The two attentino mechanisms are combined with gated-fusion mechanism to produce the final output a block. Each encoder and decoder consists of L such blocks, and each block receives input from the previous one.

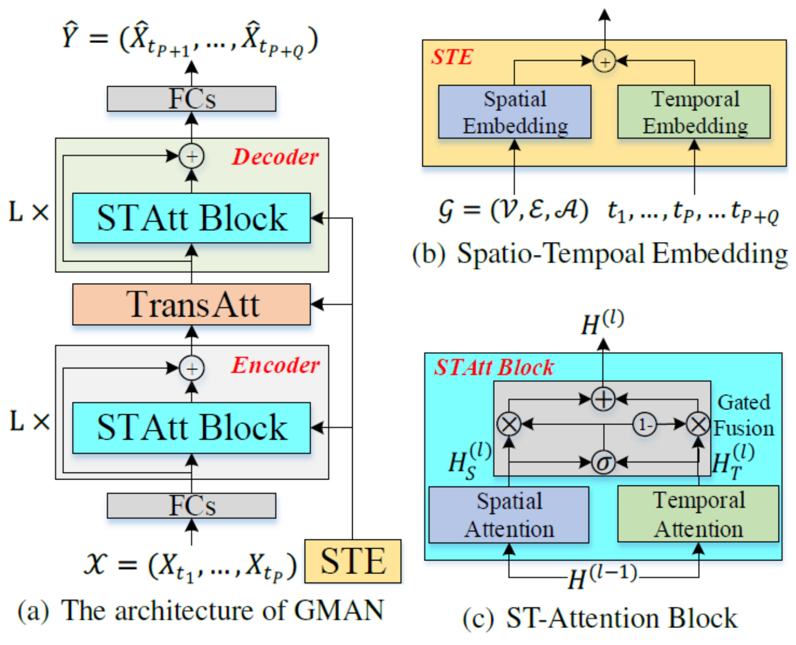

$\hat{Y} = (\hat{X}_{t_{P+1}}, \dots, \hat{X}_{t_{P+Q}})$
FCs
Decoder
L ×
STAtt Block
TransAtt
Encoder
L ×
STAtt Block
FCs
$\mathcal{X} = (X_{t_1}, \dots, X_{t_P})$
STE
(a) The architecture of GMAN
STE
Spatial Embedding
Temporal Embedding
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ $t_1, \dots, t_P, \dots t_{P+Q}$
(b) Spatio-Tempoal Embedding
$H^{(l)}$
STAtt Block
Gated Fusion
$H_S^{(l)}$
$H_T^{(l)}$
Spatial Attention
Temporal Attention
$H^{(l-1)}$
(c) ST-Attention Block

Figure 2: GMAN model

3 Dataset

We will use the PeMS datasets, which is focused on traffic speed prediction. The PeMS dataset contains 6 months of data from 325 traffic sensors, recorded from January 1 to June 30, 2017, in the Bay Area. It is structured as a weighted directed graph, where each vertex represents a traffic sensor, and edges indicate connectivity between sensors.

4 Methodology

4.1 Custom Node Embeddings

To capture the dynamics between spatial and temporal dependencies, we introduce our own embeddings and attention systems. GMAN generates 64-dimensional spatial and temporal embeddings and adds them together. This approach assumes that each timestamp uniformly affects all nodes. For instance, on a Monday morning, all regions may experience similar traffic increases, making vector addition seem consistent.

However, this is not true in reality. Some regions, like those near schools and workplaces, may potentially be more crowded on Mondays, while areas leading out of the city could have less traffic, possibly even less than on weekends.

To avoid losing information, we generate our own 52-dimensional spatial embeddings using Node2Vec. We then generate 12-dimensional temporal embeddings by passing 295 time features (288 5-minute intervals and 7 days of the week) to a fully connected layer. We then concatenate them together to get a 64-dimensional spatial-temporal embedding.

We also experimented using PCA for dimensionality reduction, but this led to worse results, presumably because the spatial and temporal embeddings are distinct enough, so PCA will only result in a loss of information.

4.2 Balanced Spectral Clustering

We run spectral clustering on our 52-dimensional node embeddings by getting their eigenpairs and considering the matrix formed by the first k eigenvectors. We then run normal K-means on the eigenvectors.

This will result in imbalanced clusters, which we do not want. To achieve balanced clustering, we offload excess nodes from overcrowded clusters $C_+^{(k)}$ to those lacking nodes using an iterative greedy algorithm. In step k, we greedily select excess nodes $N_+^{(k)} \in C_+^{(k)}$ that minimize similarity in their corresponding groups. We also keep track of clusters $C_-^{(k)}$ lacking nodes in step k, then greedily offload nodes $N \in N_+^{(k)}$ to the cluster in $C_-^{(k)}$ with the closest centroid. This offloading process may result in some new overcrowded clusters $C_+^{(k+1)}$, so we repeat this process until all clusters are balanced.

To prove this eventually converges, consider the monovariant defined by the number of excess nodes. This is monotonically decreasing across timesteps, so the process necessarily terminates.

4.3 Temporal Attention in the ST-Att Block

In the original GMAN model, given a node v_i at time step t, we calculate its attention score to all of the other time steps of the same node using a multi-head approach as shown.

$$u_{t_j,t}^{(k)} = \frac{f_{t,1}^{(k)}(h_{v_1,t_j}^{(l-1)}||e_{v_i,t_j}) \cdot f_{t,2}^{(k)}(h_{v_i,t}^{(l-1)}||e_{v_i,t})}{\sqrt{d}}, \quad \beta_{t_j,t}^{(k)} = \frac{\exp(u_{t_j,t}^{(k)})}{\sum\limits_{t_r} \exp(u_{t_j,t_r}^{(k)})}$$

Next, using these attentions, we calculate the temporal embedding by

$$ht_{v_i,t_j}^{(l)} = ||_{k=1}^{K} \left\{ \sum_{t \in N_{t_j}} \beta_{t_j,t}^{(k)} \cdot f_{t,3}^{(k)}(h_{v_i,t}^{(l-1)}) \right\}$$

However, in our new model, given a node v_i at time step t, we wish to calculate its attention score to all of the other time steps of each node belonging to the same cluster as v_i. Now, suppose v_k is some node in the cluster with v_i which we call C_{v_i}, then the attentions are calculated using the same multihead approach as shown:

$$u_{v_s,t_j,t}^{(k)} = \frac{f_{t,1}^{(k)}(h_{v_i,t_j}^{(l-1)}||e_{v_i,t_j}) \cdot f_{t,2}^{(k)}(h_{v_s,t}^{(l-1)}||e_{v_s,t})}{\sqrt{d}}, \quad \beta_{v_s,t_j,t}^{(k)} = \frac{\exp(u_{v_s,t_j,t}^{(k)})}{\sum\limits_{v_s \in C_{v_i}} \sum\limits_{t_r} \exp(u_{v_s,t_j,t_r}^{(k)})}$$

After calculating the attentions, we calculate the temporal embedding as shown below

$$ht_{v_i,t_j}^{(l)} = ||_{k=1}^{K} \left\{ \sum_{t \in N_{t_j}} \sum_{v_s \in C_{v_i}} \beta_{v_s,t_j,t}^{(k)} \cdot f_{t,3}^{(k)}(h_{v_s,t}^{(l-1)}) \right\}$$

To summarize, we simply added attention to the different time steps of other nodes. Then in the training process, we will attempt to optimize the attentions of not just to the previous time steps to the current node but also to previous time steps of the neighbors within the cluster.

For the implementation of this to our model, we first made use of the groupings generated from the balanced spectral clustering. We perform balanced spectral clustering on 325 nodes to generate 65 groups of 5 nodes each. We then rearrange the data so that all the nodes belonging to the same cluster are grouped together. After this, we can easily create a matrix which contains all the values of $(h_{v_i,t_j}^{(l-1)})$ passed through $f_{t,1}^{(k)}$ and another matrix which represents the embeddings passed through $f_{t,2}^{(k)}$ and $f_{t,3}^{(k)}$. Then we can use matrix multiplications to do the computations of the temporal embeddings.

5 Experimentation

We made use of two widely used metrics to evaluate the performance of STAGN. These are the Mean Absolute Error (MAE) and RMSE (Root Mean Square Error).

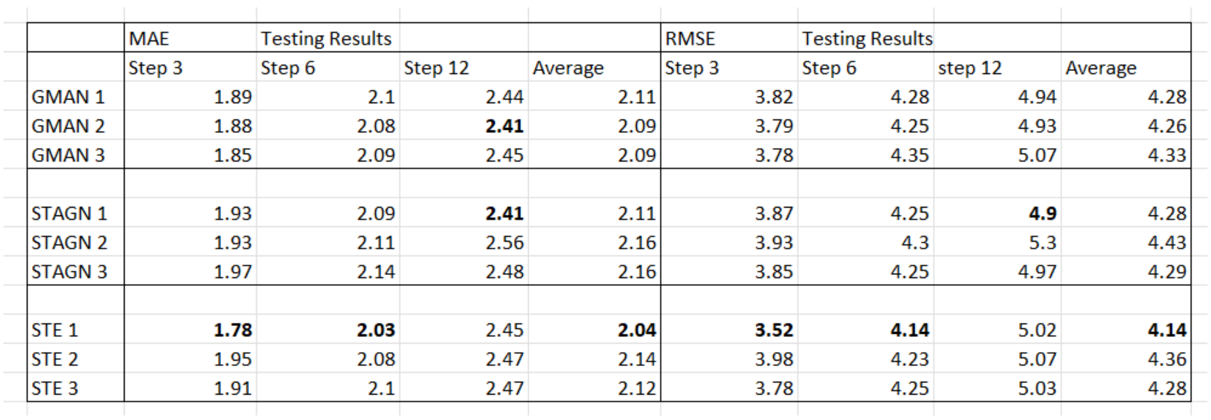
For the hyperparameters, in the multihead attention, we used $K = 8$ which was the default done in the paper. However, we lowered the number of epochs from 3 to 1 as the training time would take too long on the computers that we are running it on. We also used only a single layer of ST-Attention block or we set $L = 1$.

Lastly, for the training time, in the computers we ran the testing, it took 12700 seconds to train an entire epoch for GMAN. For the testing, it took about 7100 second. Therefore, the total train and test time about 11200 seconds or 5 and a half hours.

Next, for STAGN, the training time was a lot longer. It took about 27700 seconds to run an entire epoc. For the testing, it took about 10,100 seconds. Therefore, the train and test time took a total of about 10 and a half hours.

In addition to STAGN, we also run GMAN using our custom node embeddings, which we shall refer to as STE. The hyperparameters for STE are the same as STAGN and GMAN. We run STE three times; for the training time, on average it took only 7047 seconds to train one epoch, and for testing, it took 766 seconds.

6 Results



	MAE	Testing Results			RMSE	Testing Results		
	Step 3	Step 6	Step 12	Average	Step 3	Step 6	step 12	Average
GMAN 1	1.89	2.1	2.44	2.11	3.82	4.28	4.94	4.28
GMAN 2	1.88	2.08	**2.41**	2.09	3.79	4.25	4.93	4.26
GMAN 3	1.85	2.09	2.45	2.09	3.78	4.35	5.07	4.33
STAGN 1	1.93	2.09	**2.41**	2.11	3.87	4.25	**4.9**	4.28
STAGN 2	1.93	2.11	2.56	2.16	3.93	4.3	5.3	4.43
STAGN 3	1.97	2.14	2.48	2.16	3.85	4.25	4.97	4.29
STE 1	**1.78**	**2.03**	2.45	**2.04**	**3.52**	**4.14**	5.02	**4.14**
STE 2	1.95	2.08	2.47	2.14	3.98	4.23	5.07	4.36
STE 3	1.91	2.1	2.47	2.12	3.78	4.25	5.03	4.28

Figure 3: Testing Results, Best Results are in Bold

Now, we performed the testing of STE, STAGN and GMAN 3 times each. The reason we did this instead of training the model on 3 epochs was that we were running it on 6 computers at the same time.

First, we ran GMAN and STE to compare whether our custom node embeddings would improve the performance of GMAN. From the results, we can see that the best STE outperformed all other GMAN runs, in both MAE and RMSE. This indicates that our custom node embeddings are able to improve the performance of GMAN.

Comparing STAGN and GMAN after 1 epoch, we can see that both model tends to perform almost similarly, with GMAN having a slight edge over STAGN. However, STAGN has a bit larger spread compared to GMAN. Hence, this tells us that for GMAN, the parameters converges to a certain point pretty consistently while for STAGN it doesnt.

The reason why we believe this happens is that STAGN has a harder time to optimize the parameters for all of the attentions that we've provided. Previously, the parameters will only need to optimize for the attention between a time step t_j to t for all of the 325 nodes. However, having to also take into account the attention of the attention of a node v_i in time step t_j to a node v_s in time step t, will make it a lot more difficult to optimize the parameters. Since we ran the model for only 1 epoch, we think that if we ran it for more epochs, the model will be able to learn the attentions more efficiently and converge to a better result. In the original paper, GMAN was run for 3 epochs, and we believe that if we do the same for STAGN, the results will be better.

7 Improvements

One of the improvements that can be done to STAGN is to increase the number of parameters so that the model will have an easier time to converge. However, a possible downside of this is that the training time of the entire model will increase. We can increase the number of parameters by mapping the embeddings into a larger embedding space.

Another possible improvement is that we aggregate information from both near and farther time steps for the temporal embedding. In GMAN, we only take into account the 12 timesteps withing the past hour. However, traffic conditions may be correlated to farther timesteps. Therefore, it would be possible to take some timesteps which are farther away to produce the temporal embeddings.

8 Conclusion

In this project, we aim the improve state-of-the-art traffic prediction model GMAN by introducing custom node embeddings, balanced spectral clustering, and a new temporal attention mechanism in the ST-Attention block. We have shown that our custom node embeddings are able to improve the performance of GMAN. We also showed that STAGN is able to perform almost as well as GMAN after 1 epoch. Since we were out of time and computing resources, we were not able to run the model for more epochs. However, we believe that STAGN will be able to outperform GMAN if ran for more epochs.

References

[1] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):1234–1241, 2020.