GMAN: A Graph Multi-Attention Network for Traffic Prediction

GDL 2023
Group id: 3
Project id: GMAN

Ariola Leka, Elie Youssef, Francisco Amoros Cubells
{ariola.leka, elie.youssef, francisco.amoros.cubells}@usi.ch

Abstract

In this report we reproduce the following paper[1]. In it, the authors provide a graph multi-attention network implementation for traffic prediction. There are two ideas at play here. First, is that using spatio-temporal graphs will leverage the information already present in the data. Second, is that the attention mechanism will help alleviate the error propagation problem among prediction time steps. We are able to reproduce the results in the paper, even with slight adjustments to the hyperparameters. We test the GMAN model on other datasets to compare with the current state of the art. We achieve expected results, except for the fault-tolerance section.

1 Introduction

The goal of our project is to reproduce the paper[1]'s findings which tackles traffic prediction. As we all know, traffic is dynamic, being affected by a multitude of factors that are ever changing and interconnected. The paper we are reproducing focuses on **spatio-temporal** factors, and proposes a graph multi-attention network (GMAN) to build a predictor for the long-term future traffic conditions on a road network. We were able to replicate the results of the paper. We also try GMAN on two other datasets, comparing the results to the state of the art.

We can ask the following: *What problems does GMAN address?*

We identify 3 main areas:

1.1 Why are spatio-temporal elements relevant in this context?

Traffic conditions are expected to be affected both by spatial and temporal elements. For example, driving at 8AM and 5PM when it's the peak of traffic, is different to driving at the early morning hours when there is no cars around (*temporal factors*). Driving on a highway will allow for more speed than driving on a narrow, badly maintained road (*spatial factors*). Drivers also affect each other's behavior, so we need to be able to model this relation between how congested the traffic is and how fast the traffic is moving. Graph deep learning models are able to take into account these spatio-temporal features to provide a better prediction of the traffic conditions in the future.

1.2 Why is attention important here?

Since the authors wanted to deal with long-term prediction, there was an issue. Small errors in each time-step are fine when we're doing short-term prediction. However, these small errors compound when predictions are made further into the future. The attention layer is describing the relationships between the past and the future time steps, which reduces the error propagation into the future predictions.

1.3 What is the novelty that this paper presents?

The authors are attempting to predict the long-term (1 hour ahead) traffic conditions by solving two major problems that traditionally hinder this task:

First, they introduce **spatio-temporal attention** mechanisms to solve the complex spatio-temporal correlations like the **dynamic spatial** correlation and the **non-linear temporal** correlation problems.

Second, they insert a transform attention layer in between the encoder-decoder mechanism of **GMAN** which addresses the problem of error propagation on long-term time spans.

Third, they introduce a group spatial attention mechanism to deal with calculating the attention scores of large numbers of vertices. After running the model on the two datasets presented in the paper (Xiamen and Pems), they obtain a 4% improvement in MAE on the 1 hour ahead traffic prediction above the (at the time) best available model Graph WaveNet [2].

2 Related works

We want to expand on three fields where we see advancements that are relevant to this work:

1. Traffic prediction is a field where many machine learning approaches have been explored:

- Long short-term memory neural network for traffic speed prediction using remote microwave sensor data [3]: The researchers study the feasibility of an LSTM model with short-term speed traffic prediction, and compare it with an RNN based model, an SVM, an ARIMA model, and a Kalman filter. They demonstrate that an LSTM approach is feasible for traffic prediction tasks, and they achieve satisfactory results.
- ST-ResNet: Deep Spatio-Temporal Residual Networks for Citywide Crowd Flow Prediction[4]: The temporal closeness, period, and trend characteristics of crowd traffic are modeled by the authors using a residual neural network framework. They create a dedicated branch of residual convolutional units for each

property that represents a characteristic of crowd movement. Based on data, ST-ResNet learns to dynamically aggregate the output of the three residual neural networks, giving various branches and regions varied weights. To give a more accurate forecast of the flow of crowds, the aggregate is augmented with external elements, such as the weather and day of the week.

- LC-RNN: A Deep Learning Model for Traffic Speed Prediction[5]: Another proposed model to tackle the traffic prediction problem. LC-RNN is comprised of several look-up convolution layers modeling local spatial evolution, a recurrent layer modeling long temporal dependency, a periodicity extraction layer modeling periodicity factors and a context extraction layer modeling context factors. It takes advantage of both RNN and CNN models by using them in tandem, and taking a rational integration of them. The goal is to learn more salient time-series patterns that are dynamically adaptable to the traffic conditions of the testing area.

2. Deep Learning on Graphs: Graph deep learning is a field that is seeing a lot of innovation on many different fronts. We refer to this survey [6] to showcase some of the latest advancements. Due to the particular properties of graphs, applying deep learning to the frequently encountered graph data is challenging. This is why researching dedicated graph deep learning methods is needed. Papers like the one we're reproducing is one example of such dive into this field, where graph data is front and center. The data comes from a sensor network, representing traffic speed. It is subject to spatio-temporal factors. Since we're using a graph deep learning approach, the model is able to take into consideration the relation between the different sensors. This relation is affected by the distance between them, and the time difference.

3. Attention Mechanisms: The attention mechanism became a *de facto* in deep learning models that use an end-to-end approach, due to its high efficiency and flexibility. Vaswani et al., (2017)[7] achieve this by using a scaled dot product and a multi-head attention mechanism. They develop an end-to-end architecture that can compute a score function that values certain parts of the input. The authors of "Graph Attention Networks"[8] added attention to obtain the features of the neighbouring nodes in the graph, without needing to perform costly compute operations, resulting in getting the spatial correlations in a graph-structured data, without needing to know the graph structure beforehand.

3 Methodology

3.1 Notations

We represent the road network with a **weighted directed graph** composed of three components: $G = (V, E, A)$.

- V is the set of all nodes, where in our case a node is a sensor on the road network.
- E is the set of all edges which are the connections between different sensors in the road network.
- A is the adjacency matrix : $A \in R^{N*N}$ where $N = |V|$.

We represent the traffic conditions at time step **t** by $X_t \in R^{N \times C}$, where C is the number of traffic features that we are considering on the road network.

When given **P** past time step observations for all N vertices, $X = (X_{t_1}, X_{t_2}, ..., X_{t_P}) \in R^{P*N*C}$, the goal is to predict the traffic conditions for the next **Q** time steps which is denoted as: $\hat{y} = (\hat{X}t_{P+1}, \hat{X}t_{P+2}, ..., \hat{X}t_{P+Q}) \in R^{Q*N*C}$.

3.2 GMAN model

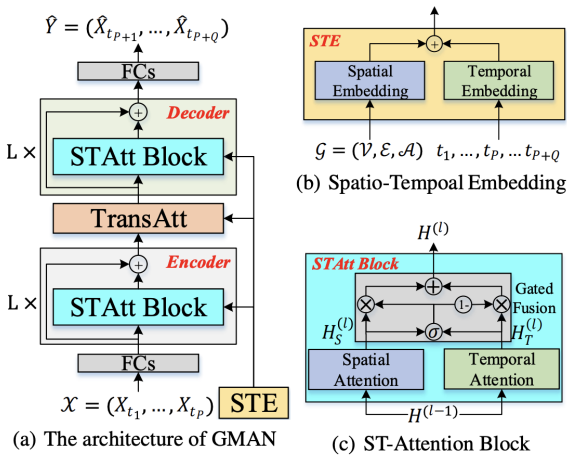

Figure 1. The architecture of GMAN with its two sub-building blocks on the right. Figure taken from the paper[1]

In the proposed graph multi-attention network (GMAN) paper, we have an encoder-decoder structure illustrated in figure 1.a. 1. Both the encoder and the decoder include **L x ST-Attention blocks** (STAtt Block) with residual connections, where L is a hyperparameter.

What are residual connections?

In the paper **Deep Residual Learning for Image Recognition**[9] the authors proposed a new architecture for deep neural networks called "ResNet". The core idea behind it is to use residual connections to address the vanishing gradient problem that can occur when training very deep neural networks. A **residual connection** is a shortcut connection that **bypasses** one or more layers of the network and allows the input to be directly added to the output of the layer(s) being bypassed. By doing so, the network can learn residual mappings, which are the differences between the input and output of a layer, rather than trying to learn the entire mapping in one step.

The ST-Attention blocks consist of spatial and temporal attention mechanisms with gated fusion. To convert the encoded traffic features to the decoder, a transform attention layer is added between the encoder and decoder. The multi-attention mechanisms incorporate the graph structure and time information through a **spatio-temporal embedding (STE)**. All layers produce outputs of **D** dimensions to facilitate the residual connection. In the following sections, we present a description of each of these modules in detail.

Spatio-Temporal Embedding: Taking into account that the traffic condition is constrained by the road network, it is

important to incorporate the road network information into the prediction models. There are two pieces that form the Spatio-Temporal embedding:

- **Spatial Embedding**: Encodes the nodes into vector while preserving the graph structure. This is done by using the node2vec algorithm.
  - **Node2vec[10]**: The authors introduce a sampling strategy. To build the context for each node, Node2Vec employs a sampling strategy that involves conducting multiple random walks starting from each node. These random walks traverse the graph and capture the network neighborhood of each node. The exploration of the context is parameterized to balance between Breadth First Search (BFS) to obtain direct connections and Depth First Search (DFS) to uncover more in-depth relationships. Once the contexts, denoted as Ns(u), are obtained for all nodes u in the graph, the next step is to find vector representations for the nodes. A mapping function, f(u), is used to map each node u to a corresponding context vector of d dimensions. These vectors are then used to calculate the dot product, which represents the similarity between two nodes. Smaller angles between vectors indicate higher similarity. To quantify the similarity scores, Node2Vec employs **softmax** to transform them into probabilities. The dot product can be interpreted as a conditional probability, $p(v|u)$, and the goal is to maximize this value. The process is applied to all nodes in the context vector of u, $Ns(u)$, and the probabilities for all nodes in the context of u, $p(Ns(u)|u)$, are calculated. Assuming node independence, the probabilities for all nodes are multiplied $\prod_{v_i \in Ns(u)} p(v_i|u)$. This process is performed for all nodes in the graph simultaneously, resulting in the final equation $\sum_{u \in v} log(p(Ns(u)|u))$, which helps find the graph embedding.

  Afterwards, the embedding of node2vec is fed into a 2 layer fully connected (FC) network. The output of the spatial embedding will be of the form $e^S_{v_i} \in R^D$ where $v_i \in V$, and S is referring to spatial.
- **Temporal Embedding**: It is the encoding of each time step into a vector representation.
  If we assume that one day has **T** time steps, we encode **the day of the week** and **time of the day** in R^7 and R^T by using one hot encoding and this way from the concatenation we get the vector R^{T+7}.
  To map the time feature into a R^D vector we feed the R^{T+7} into a 2 layer FC. The output of the temporal embedding is going to be of the form: $e^T_{t_j} \in R^D$, where T stands for temporal. t_j is now both the historical and the future time step ($t_j = t_1, ..., t_P, ..., t_{P+Q}$). As a final operation, the spatial embedding and the temporal embedding are combined as in figure: 1.b. The shape of the spatio-temporal embedding for N nodes in P+Q time steps, will result in $R^{(P+Q)*N*D}$.

ST-Attention Block: The ST-Attention Block contains three components: **Spatial Attention**, **Temporal Attention** and **Gated Fusion**.
Notations:
$H^{(l-1)}$ is the input of the l^{th} block
$h^{(l-1)}_{v_i,t_j}$ is a hidden state of the vertex v_i at the time t_j
$H^{(l)}_S$ is the Spatial attention output of the l^{th} block
$H^{(l)}_T$ is the Temporal attention output of the l^{th} block
$ht^{(l)}_{v_i,t_j}$, $hs^{(l)}_{v_i,t_j}$ are the hidden states for $H^{(l)}_T$ and $H^{(l)}_S$, respectively, at time t_j.
$H^{(l)}$ is the output of the gated fusion of the l^{th} block.
$f(x) = ReLU(x * \mathbf{W} + \mathbf{b})$ is the non linear transformation where W and b are learnable parameters.

- ***Spatial attention***: To solve the dynamic spatial correlation problem a spatial attention mechanism is designed and included. Different weights are assigned dynamically to different vertices, at different time steps. We calculate the weighted sum of all vertices:

$$hs^{(l)}_{v_i,t_j} = \sum_{v \in V} \alpha_{v_i,v} * h^{(l-1)}_{v,t_j} \quad (1)$$

  Where $\alpha_{v_i,v}$ is the attention score. The attention scores should sum up to one.
  For analyzing sensor correlations, it is important to consider both the influence of current traffic conditions, as well as the underlying road network structure. To give a solution, both traffic features and the graph structure are incorporated into the attention scoring mechanism. To accomplish this, we combine the hidden state (using the concatenation operator $||$) with the spatio-temporal embedding and employ the scaled dot-product ($\langle \bullet, \bullet \rangle$) approach from (Vaswani et al., 2017)[7] to calculate the relevance between two vertices, vi and v:

$$s_{v_i,v} = \frac{\langle h^{(l-1)}_{v_i,t_j} \| e_{v_i,t_j} \ , \ h^{(l-1)}_{v,t_j} \| e_{v,t_j} \rangle}{\sqrt{2D}} \quad (2)$$

  where 2D is the dimension of the concatenation. After, softmax is applied at the output:

$$\alpha_{v_i,v} = \frac{\exp(s_{v_i,v})}{\sum_{v_r \in V} \exp(s_{v_i,v_r})} \quad (3)$$

  At this point the hidden states can be updated.
  To ensure a more stable learning process, we enhance the spatial attention mechanism by including multiple attention heads, following the concept introduced by Vaswani et al. (2017)[7]. This involves concatenating **K** parallel attention mechanisms, each having distinct learnable projections, where **K** is a hyperparameter.

$$s^{(k)}_{v_i,v} = \frac{\langle f^{(k)}_{s,1}(h^{(l-1)}_{v_i,t_j} \| e_{v_i,t_j}), f^{(k)}_{s,2}(h^{(l-1)}_{v,t_j} \| e_{v,t_j}) \rangle}{\sqrt{d}} \quad (4)$$

$$\alpha^{(k)}_{v_i,v} = \frac{\exp(s^{(k)}_{v_i,v})}{\sum_{v_r \in V} exp(s^{(k)}_{v_i,v_r})} \quad (5)$$

$$hs^{(l)}_{v_i,t_j} = \|^K_{k=1} \left\{ \sum_{v \in V} \alpha^{(k)}_{v_i,v} * f^{(k)}_{s,3}(h^{(l-1)}_{v,t_j}) \right\} \quad (6)$$

It is important to denote that the $f_{s,n}$, n={1,2,3}, are three non-linear transformations which produce outputs in the dimensions of $d = D/K$.
To overcome the computational and memory burden associated with a large number of vertices N, a solution is introduced, called **group spatial attention**. This approach consists of two components: **intra-group spatial attention** and **inter-group spatial attention**. By utilizing these mechanisms, N^2 attention scores can be computed more efficiently.
N vertices are divided into G groups where $M = N/G$ is the number of nodes each group has. By employing equations 4, 5, and 6, we compute the intra-group attention within each group to capture local spatial correlations. Importantly, the learnable parameters are shared across all groups. To generate a representative feature for each group, we utilize a max-pooling technique. Subsequently, we calculate the inter-group spatial attention, which captures correlations between different groups, resulting in a global feature for each group. Finally, we combine the local feature with its corresponding global feature, creating a comprehensive output representation. At each time step, we have this many attention scores computed:
$GM^2 + G^2 = NM + (N/M)^2$
The minimum number of attention scores is achieved when the gradient is zero. This happens when $M = \sqrt[3]{2N}$.
Substituting into the previous equation, we get: $2^{-1/3}N^{4/3} \ll N^2$.

- ***Temporal Attention***: To solve the non-linear temporal correlation problem, a temporal attention mechanism is designed. In order to measure the inter-connectivity between different time steps, it is crucial to take both the traffic features and the time into consideration. More concretely, the hidden state is concatenated with the STE *(spatial temporal embedding)* in a similar fashion to the multi head approach of the spatial attention.

$$u_{t_j,v}^{(k)} = \frac{\langle f_{t,1}^{(k)}(h_{v_i,t_j}^{(l-1)} \| e_{v_i,t}), f_{t,2}^{(k)}(h_{v_i,t}^{(l-1)} \| e_{v_i,t}) \rangle}{\sqrt{d}} \quad (7)$$

$$\beta_{t_j,t}^{(k)} = \frac{\exp(u_{t_j,t}^{(k)})}{\sum_{t_r \in N_{t_j}} \exp(u_{t_j}^{(k)}, t_r)} \quad (8)$$

Equation (7) denotes the relevance between the two time steps t_j and t. Equation (8) gives the attention score and the two functions $f_{t,1}^{(k)}$ and $f_{t,2}^{(k)}$ are the learnable transforms. To conclude, the hidden state after calculating the attention score is going to be:

$$ht_{v_i,t_j}^{(l)} = \|_{k=1}^{K} \left\{ \sum_{t \in N_{t_j}} \beta_{t_j,t}^{(k)} * f_{t,3}^{(k)}(h_{v_i,t}^{(l-1)}) \right\} \quad (9)$$

- ***Gated fusion***: Taking into account that the road conditions in time step t are correlated with historical values and other traffic conditions of the road, a gated fusion is added in order to mix the spatial and temporal representations. After getting $H_S^{(l)}$ and $H_T^{(l)}$, they get fused as:

$$H^{(l)} = z \odot H_S^{(l)} + (1 - z) \odot H_T^{(l)} \quad (10)$$

where,

$$z = \sigma(H_S^{(l)} W_{z,1} + H_T^{(l)} W_{z,2} + b_z) \quad (11)$$

$W_{z,1}$, $W_{z,2}$,and b_z are learnable parameters, $\odot$ is the element wise product, σ is the sigmoid activation function and z is the gate.

Transform Attention: As we have said in the introduction, a transform attention layer is added between the encoder and the decoder to lower the error propagation problem. The main idea is to have a direct correlation between each future time step and each historical time step. By doing so, we can convert the encoded traffic features into future representations, which then serve as input for the decoder.

Given a node v_i, the relevance ($\lambda_{t_j,t}$) between the time step $t_j = t_{P+1}, \ldots, t_{P+Q}$ and the historical time step $t = t_1, \ldots, t_P$, is calculated using the STE:

$$\lambda_{t_j,t}^{(k)} = \frac{\langle f_{t_r,1}^{(k)}(e_{v_i,t_j}), f_{t_r,2}^{(k)}(e_{v,t_j}) \rangle}{\sqrt{d}} \quad (12)$$

$$\gamma_{t_j,t}^{(k)} = \frac{\exp(\lambda_{t_j,t}^{(k)})}{\sum_{t_r=t_1}^{t_P} exp(\lambda_{t_j,t_r}^{(k)})} \quad (13)$$

$$h_{v_i,t_j}^{(l)} = \|_{k=1}^{K} \left\{ \sum_{t=t_1}^{t=t_P} \gamma_{t_j,t}^{(k)} * f_{tr,3}^{(k)}(h_{v_i,t}^{(l-1)}) \right\} \quad (14)$$

Where $\gamma_{t_j,t}^{(k)}$ is the attention score and $h_{v_i,t_j}^{(l)}$ is the updated hidden state for k attention heads.

Big picture Encoder-Decoder: As it can be seen in figure 1.a. 1, the historical data are fed initially in a FC layer and the shapes are transformed from R^{P*N*C} to R^{P*N*D}. Afterwards, this output passes through the encoder with L layers to produce $H^{(L)} \in R^{P*N*D}$. After passing through the transformer attention, the encoded feature $H^{(L)}$ is converted in order to produce the future representations $H^{(L+1)}$. Finally, the output of the transformer enters through the decoder to produce $H^{(2L+1)} \in R^{Q*N*D}$ where afterwards the FC layers transform it into the prediction $\hat{Y} \in R^{Q*N*C}$.

By using back-propagation the model gets trained end-to-end by minimising the mean absolute error (MAE):

$$Loss(\theta) = \frac{1}{Q} \sum_{t=t_{P+1}}^{t_{P+Q}} |Y_t - \hat{Y}_t| \quad (15)$$

where θ are all the hyperparameters of the model.

4 Implementation

4.1 Datasets

We performed our experiments on the following datasets:

- **PEMS-BAY**: is a public traffic speed dataset, collected from California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). It contains six months of data of 325 road traffic sensors (from 1-st January 2016 to 30 of June 2016).[11]
- **METR-LA**: is a public traffic speed dataset collected from loop-detectors located on the LA County road network. It contains 4 months of data of 207 sensors (from Mar 1st 2012 to Jun 30th 2012). [11]
- **AirQuality**: is a dataset that contains information about air pollutants, collected by 437 air quality monitoring stations spread across 43 Chinese cities from May 2014 to April 2015. The dataset also contains a smaller version with only the subset of nodes containing the 36 sensors in Beijing.

The PEMS-BAY and METR-LA datasets were downloaded from the Google drive folder provided by the authors of the paper, even though they do not use the METR-LA dataset in their results. The AirQuality dataset was downloaded from torch-spatiotemporal library.[12]

4.2 Hyperparameters

The hyper-parameters are:

- Time step duration:
  - Experiment A, B, C, D: 5 minutes
  - Experiment E: 60 minutes
- History steps (how many time steps to look into the past):
  - Experiment A: 1, 4, 12 (5, 20, 60 minutes into the past)
  - Experiment B, C: 12
  - Experiment D: 1, 4, 8, 12
  - Experiment E: 6, 12 (hours into the past)
- Prediction steps (how many time steps to look into the future):
  - Experiment A: 1, 4, 12 (5, 20, 60 minutes into the future)
  - Experiment B: 3, 6, 12
  - Experiment C: 12
  - Experiment D: 1, 4, 8, 12
  - Experiment E: 6, 12 (hours into the future)
- Number of Spatio-Temporal Attention Blocks (**L**):
  - Experiment A, C, D: 3
  - Experiment B: 1
  - Experiment E: 2
- Number of attention heads: 8
- Dimension of each attention head output: 8

4.3 Experimental setup

We used the code provided by the authors of the paper. There was a 3rd party Pytorch implementation but it had some bugs. We managed to make it work, but the results reported are using the original tensorflow version.

Experiment A : This was the code provided by the authors of the paper. We used the tensorflow implementation. The code was ran on **Lambda Labs** https://lambdalabs.com/ in a GPU server. We used L = 3. **Source Code**

Experiment B : We used the same code as experiment A. The code was ran on Google Colab in a GPU Runtime, **Source Code**. We used L=1, to explore how using only 1 ST-Attention block affects the results.

Experiment C : We try with the same setup that the paper presents. We run two variations of the experiment:

- **Dropping** the data by deleting the row
- Setting the value to **zero**

We run with nine different ratios: from 0.1 to 0.9, where that percentage of the data gets randomly dropped/zeroed. The rest of the model remains unchanged. **Source Code**

Experiment D In this experiment, we perform multiple runs of the model, with each run having one component removed from the model, to check their individual effects. **Source Code**

a. GMAN-NS: We remove the **spatial attention** by feeding the input from the previous layer only into the **temporal attention** block. Afterwards, the output $H_T^{(l)}$ is handled like so:

$$H^{(L)} = (1 - \sigma(H_T^{(l)})) * H_T^{(l)} \quad (16)$$

With $H^{(L)}$being the output of the ST-Attention block. We can see how the ST-Attention Block looks like after removing the spatial attention in figure 2.

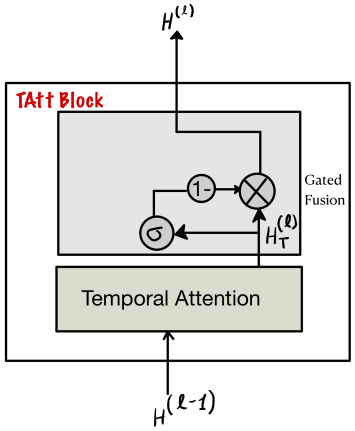

Figure 2. GMAN without spatial attention

b. GMAN-NT: The same technique is also followed for the removal of the **temporal attention**. The only difference is that now we remove the other part of the block, ending up with the output:

$$H^{(L)} = \sigma(H_T^{(l)}) * H_T^{(l)} \quad (17)$$

We can see how the ST-Attention Block looks like after removing the temporal attention in figure 3.

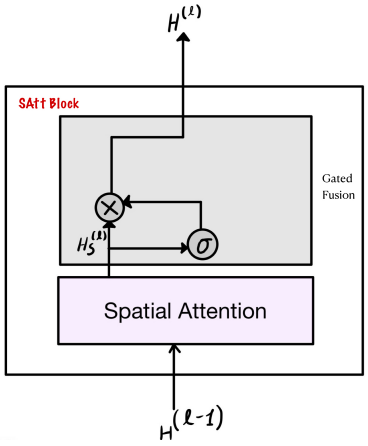

Figure 3. GMAN without temporal attention

c. GMAN-NG: For the removal of the **gated fusion**, we exclude the block where the activation function, multiplication and adding happens. The output of the ST-Attention block is now only the outputs coming from the individual **Spatial Attention** and **Temporal Attention**. We can see how the ST-Attention Block looks like after removing the gated fusion in figure 4.

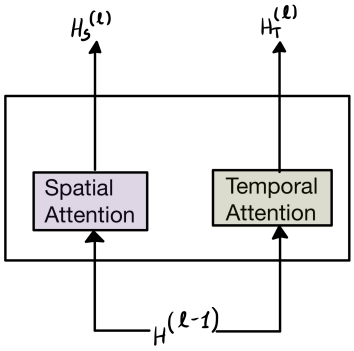

Figure 4. GMAN without Gated Fusion

d. GMAN-NTr: Lastly, we only remove the middle step of the **transformer attention** between the encoder and the decoder, by directly inputting the output of the encoder into the decoder.

Experiment E We start with extracting the adjacency matrix from the Air quality dataset. We choose the **pm2.5** air pollutant since the paper we are comparing with uses the same. We use a node2vec implementation provided by the following library. This way we extract the **Spatial Embedding** of the graph. We set the time step to 60 minutes to align with the dataset since the aggregation of the data is in one hour intervals. Due to the fact that we have **437 sensors**, the size of the model was big, and we ran into some out-of-memory issues. This led us to use L = 2 to compensate. **Source Code**

4.4 Computational requirements

Experiment A : This experiment was run on the PEMS-BAY dataset. It took approximately 23gb of GPU Ram. The runtime information is provided:

Table 1. Experiment A training time

Models	Total (min)	Per epoch (sec)	Epochs
GMAN-1	24.5	101	13/50
GMAN-4	207.4	324	36/50
GMAN-12	433	898	27/50

Experiment B : Ran for 5 epochs for each prediction length on 2 datasets: PEMS (Same as the paper) and METR (independent benchmark). It took approximately 14gb of GPU Ram. The total training time for each prediction horizon, in minutes, was:

Table 2. Experiment B training time

	15 min	30 min	1 hour
PEMS	42.7	49.9	68.5
METR	14.6	17.7	23.4

Experiment C : This experiment was run on the PEMS-BAY dataset for 1 hour prediction. It took approximately 23gb of GPU Ram.

Table 3. Experiment C training time

Ratio	Total (min)	Per epoch (sec)	Epochs
0.1 to 0.9 zeroed	87.2	902	5/5
0.1 removed	814.2	817	56/100
0.5 removed	428.9	452	51/100
0.9 removed	241.6	90	100/100

Experiment D : This experiment was run on the PEMS-BAY dataset. It took approximately 23gb of GPU Ram.

Table 4. Experiment D training time

Models	Total (min)	Per epoch (sec)	Epochs
NS-4	52.7	118	25/100
NS-8	109	211	29/100
NS-12	129	310	23/100
NG-1	28.5	97	16/100
NG-4	110.5	307	20/100
NG-8	336.4	513	33/100
NG-12	511.9	846	34/100
NT-8	289.7	453	36/100
NT-12	326.1	666	30/100
NTr-1	67.7	99	38/100
NTr-4	250.3	314	45/100
NTr-8	262.4	587	25/100
NTr-12	798.2	867	52/100

Where -1,-4,-8,-12, refer to the number of prediction steps.

Experiment E : This experiment was run on the Air Quality dataset. It took approximately 23gb of GPU Ram.

Table 5. Experiment E training time

Prediction step	Total (min)	Per epoch (sec)	Epochs
6	30.0	87.5	19/50
12	49.3	160.1	17/50

5 Results

Table 6. Experimental results on PeMS dataset

Methods	Time Frame	MAE	RMSE	MAPE
GMAN Paper	15 mins	1.34	2.82	2.81%
Experiment B	15 mins	1.35	2.91	2.98%
Experiment A	20 mins	1.26	2.68	2.68%
GMAN Paper	30 mins	1.62	3.72	3.63%
Experiment B	30 mins	1.58	3.46	3.67%
GMAN Paper	1 hour	1.86	4.32	4.31%
Experiment A	1 hour	1.79	4.11	4.17%
Experiment B	1 hour	1.88	4.22	4.47%
Best	1 hour	1.77	N/A	N/A

As we can see in table 6, we achieve very similar results to the paper. We are satisfied with the results of Experiments A and B, as they are even slightly better than expected.

Experiment B yielding similar results despite having only 1 spatio-temporal block, suggests that the spatio-temporal correlations are successfully captured even with 1 block.

Table 7. Experimental results on METR dataset

Methods	Time Frame	MAE	RMSE	MAPE
Experiment B	15 mins	2.81	5.40	7.20%
Experiment B	30 mins	3.11	6.13	8.82%
Experiment B	1 hour	3.51	7.03	10.33%
Best	1 hour	3.28	N/A	N/A

Where Best is Traffic Transformer[13]. The results come from a 2020 paper which is currently ranked #1 globally on both pems-bay and metr-la datasets.

The encoder-decoder architecture with attention in between, is the same one used in the current best model TrafficTransformer[13]. This further proves that this architecture is the proper way to model the spatio-temporal relationships, and that this modeling will provide accurate long-term predictions.

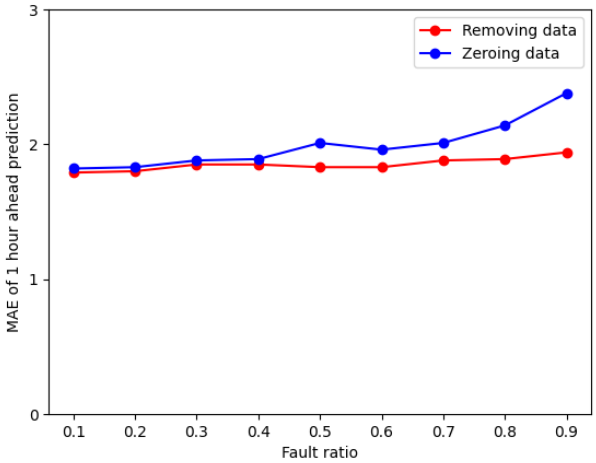

Figure 5. Experiment C

In Experiment C, we were surprised by the results. We expected the error to increase linearly with the fault-ratio. Instead, we got an almost constant error. We tried both setting the data values to zero, or dropping the row altogether, and this still happened. In figure 5, we can see that both variations exhibit this peculiar behavior. We remark that zeroing the data instead of dropping the rows provides a more expected behavior, to a minor extent.

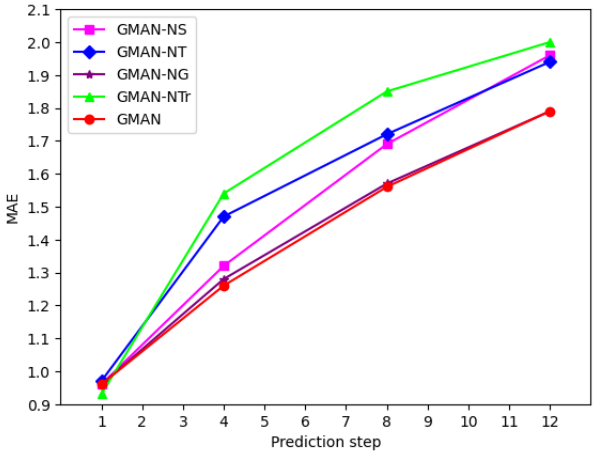

Figure 6. Experiment D

In Experiment D, we got the expected results. Figure 6 shows that the full GMAN model gives the best results. As we see, whenever we remove either the spatial, or the temporal or the transform attention layer, the performance drops, emphasizing the importance of each of these components.

We remark that the GMAN-NG model, with no gated fusion, has almost the exact same MAE as the full one, leading us to think that the gated fusion has a minor effect.

Table 8. Experiment E

Models	MAE (1-6 hr)	MAE (7-12 hr)
GMAN Paper	20.22	26.98
AirQuality	23.7	52.4

In Experiment E, we had positive results. GMAN worked well in predicting air quality conditions on the **p2.5** feature of the dataset. In table 8 we see the results, and how GMAN is outperforming the model presented in the following paper [14].

6 Discussion and conclusion

In this report we saw how GMAN performs well on the long term traffic prediction task that it was created for.

We also show a good performance on air quality prediction, further showing the robustness of the encoder decoder architecture.

Using 1 Spatio-Temporal Attention Block worked well, however, having 3 blocks still provided better results.

We also show the importance of each individual component of the model. Removing any one component decrements the performance, with the exception of the gated fusion, which we did not find to have a significant detriment when not used.

Finally, we encountered some surprising results when checking the fault tolerance of the model. The MAE didn't seem to be drastically affected when the ratio of the missing data increased, as opposed to the findings of the paper.

7 Google Drive Folder

All the code, as well the data and figures can be found here, with the exception of Experiment B, which can be found here.

References

[1] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction, 2019. URL https://arxiv.org/abs/1911.08415.

[2] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1907–1913. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/264. URL https://doi.org/10.24963/ijcai.2019/264.

[3] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54: 187–197, 2015. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2015.03.014. URL https://www.sciencedirect.com/science/article/pii/S0968090X15000935.

[4] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. doi: 10.1609/aaai.v31i1.10735. URL https://ojs.aaai.org/index.php/AAAI/article/view/10735.

[5] Zhongjian Lv, Jiajie Xu, Kai Zheng, Hongzhi Yin, Pengpeng Zhao, and Xiaofang Zhou. Lc-rnn: A deep learning model for traffic speed prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3470–3476. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/482. URL https://doi.org/10.24963/ijcai.2018/482.

[6] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2022. doi: 10.1109/TKDE.2020.2981333. URL https://ieeexplore.ieee.org/document/9039675.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[10] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL http://arxiv.org/abs/1607.00653.

[11] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S. Jensen. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting, 2022. URL https://arxiv.org/abs/2206.09112.

[12] Ivan Marisca Andrea Cini. Torch spatiotemporal air quality. URL https://torch-spatiotemporal.readthedocs.io/en/latest/modules/datasets_in_tsl.html#tsl.datasets.AirQuality.

[13] Ling Cai, Krzysztof Janowicz, Gengchen Mai, Bo Yan, and Rui Zhu. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3): 736–755, 2020. doi: https://doi.org/10.1111/tgis.12644. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12644.

[14] Yu Zheng, Xiuwen Yi, Ming Li, Ruiyuan Li, Zhangqing Shan, Eric Chang, and Tianrui Li. Forecasting fine-grained air quality based on big data. In *Proceedings of the 21th SIGKDD conference on Knowledge Discovery and Data Mining*. KDD 2015, August 2015. URL https://www.microsoft.com/en-us/research/publication/forecasting-fine-grained-air-quality-based-on