
DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Multi-agent Systems

VERBAL EXPLANATIONS OF SPATIO-TEMPORAL GRAPH NEURAL NETWORKS FOR TRAFFIC FORECASTING

CANDIDATE
Riccardo Spolaor

SUPERVISOR
Prof. Roberta Calegari

CO-SUPERVISOR
Prof. Panchamy
Krishnakumari

Academic year 2022-2023
Session 3rd

Abstract

Intelligent Transportation Systems (ITS) built using Deep Neural Network (DNN) models offer an effective solution for handling short-term traffic flow, which greatly assists drivers, travellers or public security and safety in their decision-making. In particular, Spatio-Temporal Graph Neural Networks (STGNNs) have gained popularity as a powerful tool for effectively modelling spatio-temporal dependencies in diverse real-world urban applications, including intelligent transportation and public safety. However, the black-box nature of these models prevents a true understanding of the results and a trustworthy adoption by their users.

The research field of eXplainable Artificial Intelligence (XAI) addresses this concern by developing systems that help users trust non-transparent AI. Non-expert ITS users are primarily interested in the non-technical reasons behind model predictions. Hence, leveraging Natural Language Generation (NLG), verbal descriptions of the reasons behind model outcomes are a peculiar tool to provide an easy and clear illustration of the process.

This work focuses on developing an XAI system to explain short-term speed forecasts in traffic networks obtained from STGNNs. The primary emphasis lies in explaining the reasons behind predicted traffic congestions or free flows. Key information justifying these predictions is extracted from the input traffic network in the form of a significant subgraph. The information of the subgraph is finally summarized and it is then converted into text using a template-based approach.

This thesis makes a dual contribution. First, it delves into explaining short-term traffic predictions by STGNNs, an underexplored area in current literature. Second, it leverages NLG techniques uniquely, using them to verbally translate XAI explanations into a coherent narrative, following a data-to-sequence template-based approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Traffic forecasting plays a fundamental role within the realm of *Intelligent Transportation Systems (ITS)*. Precise predictions on traffic have the chance to improve results on several problems related to transportation, such as route optimization, efficient dispatch of vehicles, and handling traffic congestion. In recent years, this domain has observed a notable increase in research outputs, with a particular emphasis on employing *Deep Learning* techniques, which have shown notable progress in predicting and managing traffic more effectively [1].

The accuracy of predictions on traffic lowers as the forecasting time frame extends. This happens because forecasts rely on external factors, which become increasingly uncertain in the distant future. Moreover, these forecasts are dependent on the relationships between travel patterns and these external factors, which can evolve over time. [2]. On the other hand, *short-term traffic forecasting* is a thoroughly explored area of study and it is, thus, more reliable for developing a more advanced transportation system to help end-users control traffic and avoid congestion. Recent studies focused on examining past traffic data and forecasting traffic patterns for the subsequent minutes or next few hours, reaching reliable levels of accuracy [3, 4, 5].

Among different *Deep Neural Network (DNN)* architectures employed in short-term traffic forecasting, *Graph Neural Networks (GNNs)* have been largely adopted due to their ability to properly encode the graph structure of a traffic network along with the spatial dependencies between its nodes, becoming state-of-the-art models applied in the field. In order to account for temporal relations in the traffic networks, *Spatial-Temporal GNNs (STGNNs)* are the de facto standard solution adopted for this subject area [6, 7, 8]. They combine the advantages of GNNs and *Recurrent Neural Networks (RNNs)* to model both the spatial and local temporal dependencies in traffic data. Furthermore, global temporal dependencies can be handled by the use of *transformers*.

Such architectures, along with other DNNs, are highly successful in making accurate predictions, although their computational procedure lacks transparency and poses a significant challenge in understanding the rationale behind their decisions. Consequently, the explanations for the models' predictions are considered opaque or "black", leading to their classification as *black box* models. This has led to a growing interest in the field of *eXplainable Artificial Intelligence (XAI)*, which aims to make the reasoning of black box models more accessible and interpretable [9, 10].

In the realm of ITS, retrieving knowledge from traffic forecasting models can enhance their design and yield insights. This has implications for crucial processes like road design, proactive traffic control systems, route planning, and other aspects of the ITS ecosystem impacted by traffic-related knowledge [11].

XAI is divided into two primary categories: *Intrinsic* and *Post-hoc* explanations [12]. Intrinsic methods involve constructing inherently explanatory models (such as *decision trees* or *rule-based models*), while Post-hoc approaches involve developing separate models to explain the functioning of existing models. The latter methodology aligns with the investigation into the interpretability of black box models like STGNNs.

Post hoc explanations can be performed through two main techniques:

model explanation and *outcome explanation* [13, 14]. Model explanation techniques aim to globally understand the behaviour of a black box model by providing insights into why it generates particular predictions, drawing from its structure and parameters. On the other hand, outcome explanation methods provide local explanations for specific instances, unveiling the link between particular inputs and their predicted outcome. In addition, a technique defined as *model inspection* that derives from both model and outcome explanations can be applied. This method uses visual or textual representations to explain specific properties or decisions of the black box model. For instance, it enables sensitivity analysis by visually demonstrating how varying inputs affect predictions.

When it comes to explaining complex concepts to non-technical users, outcome explanations or model inspections tend to resonate more effectively than model explanations. This is because the former methods more easily find roots in the *cognitive and social processes of understanding*, by highlighting the link between specific inputs and the resulting predictions or outcomes. The cognitive process of understanding identifies specific relations, while the social aspect focuses on verbal exchange between explainers and explainees [15]. In particular, the verbal delineation of the explanation is argued to be more impactful than visual representations. However, visuals can still play a supportive role, offering benefits in guiding comprehension.

This thesis aims to develop a new XAI system tailored for non-technical users, explaining short-term speed predictions in traffic networks generated by a STGNN. It focuses on clarifying the reasons behind both traffic congestion and free flow. To achieve this goal, the system uses a transparent and easily comprehensible post-hoc model inspection technique to provide explanations which can be more easily conveyed to the target users and which consider both the local and global scope of explanations. These explanations take the visual form of an *important subgraph* within the input traffic network data, highlighting information crucial for the forecasts. These visual explanations are then enriched by their translation into verbal descriptions to better align with the social process of understanding. The system summarizes essential data from the extracted subgraph into clear, straightforward text, using a simple *data-to-sequence template-based* method to ensure easy comprehension. The code of the experiment is available publicly [16].

1.2 Contribution

This thesis contributes to XAI applied to ITS for two key reasons. First, it aims to generate easily understandable, clear and concise explanations for predictions made by STGNNs regarding short-term traffic forecasting. These kind of black box models have not received much attention in current academic literature. By focusing on this underexplored domain, the thesis proposes a novel pathway for understanding how STGNNs can be trusted in the prediction of future traffic patterns whether they determine a congestion or a free flow.

Secondly, the thesis introduces an innovative use of *Natural Language Generation (NLG)* techniques. It transforms visual explanations of traffic network predictions into verbal descriptions, aligning better with how people naturally understand information. This method uses a data-to-sequence template-based approach to convert complex visual social traffic network data into narratives that anyone can grasp easily. By bridging the gap between technical insights and comprehensible narratives, this application of NLG enriches the accessibility of AI-generated insights related to transportation, making them more user-friendly and applicable in real-world scenarios.

1.3 Structure of the thesis

The thesis structure is divided in the following chapters:

Chapter 2 delivers an exploration of the literature regarding the practical applications of GNNs, with an emphasis on STGNNs. Then, it delves into the background of techniques designed to enhance the interpretability and explainability of these models. Finally, it conducts an analysis of previously employed data-to-text methodologies.

Chapter 3 provides an overview of the chosen experimental architecture. Specifically, details of the adopted STGNN model will be illustrated, offering an understanding of its design. Following this, the explanation pipeline implemented in this study will be presented, sorting out the reasons behind its use and its implementation. Lastly, the chapter will uncover the template approach utilized to transform the results of the explanation pipeline into coherent verbal narratives.

Chapter 4 highlights the experimental setup adopted for the thesis. It begins by outlining the technology stack employed in the experiment. Subsequently, a comprehensive analysis and exploration of the datasets utilized are presented. Finally, the training process of the adopted STGNN model is described, along with the fine-tuning procedure of the explanation pipeline and the rule set adopted in the template-based verbal translation.

Chapter 5 will present the quantitative results obtained and showcase the generated outputs. In detail, it will start by outlining the error metrics for the STGNN prediction. Following this, a comprehensive analysis of the errors in the explanation pipeline results will be conducted, emphasizing its strengths, weaknesses, and time taken. Lastly, it will illustrate examples of verbal translations of the explanations supported by their visual representations in the input traffic network, showcasing their effectiveness.

Chapter 6 will finally recap the results obtained in the thesis work and it will outline the explainer limitations and potential directions for future work.

Chapter 2

Background

The work of the thesis focuses on creating an *eXplainable AI (XAI)* pipeline specifically designed for users without technical expertise. The system aims to explain forecasts of short-term speeds in static spatio-temporal traffic networks produced by a *Spatio-Temporal Graph Neural Network (STGNN)*. STGNNs are built upon *Graph Neural Networks (GNNs)*, a class of Deep Learning models designed to operate on static graph-structured data. In the rest of this chapter, all the background notions needed to understand the work are provided. Related works are discussed as well.

2.1 Graph Neural Networks

Static graphs, illustrated in Figure 2.1, are a data structure which defines a set of objects, or nodes, and their relationships, or edges, and are a powerful tool to represent data with interconnected relations, such as social networks, knowledge graphs, and traffic networks [17].

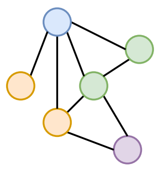
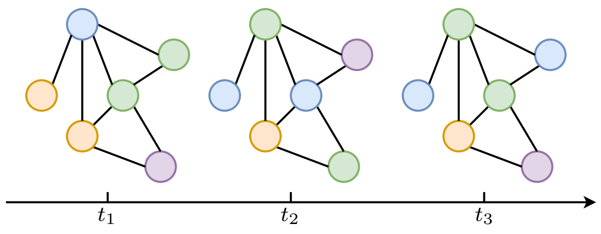


Figure 2.1: Example of a static graph.

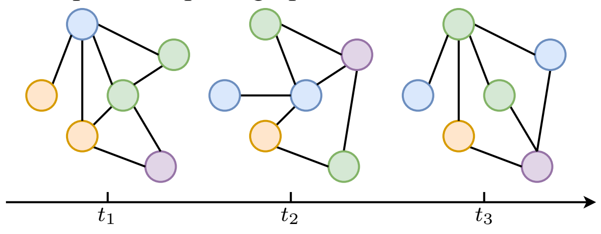
GNNs are able to capture these relationships and use them to make predictions and solve related problems. GNNs can be applied for a variety of tasks [18], such as:

- **Node prediction:** Predicting the category of a node, such as the scientific domain of an article in a citation network.
- **Link prediction:** Predicting whether an edge exists between two nodes. For instance, whether an article quotes another in a citation network.
- **Graph classification:** Predicting the category of a graph. As an example, whether a social network is a business or a traffic network.

The most adopted GNN architectures include *Graph Convolutional Networks (GCNs)*, *Graph Attention Networks (GATs)* and *Message Passing Neural Networks (MPNNs)*. *Spatio-Temporal GNNs (STGNNs)* extend these architectures to account for static or dynamic spatio-temporal graphs (shown in Figure 2.2).



(a) Example of static spatio-temporal graph where node attributes change over time.



(b) Example of dynamic spatio-temporal graph where node attributes and links change over time.

Figure 2.2: Example of static and dynamic spatio-temporal graphs.

Static spatio-temporal graphs include temporal dependencies between network nodes that change their attribute values over time, while their spatial relationships defined in their edge set remain fixed. In contrast, dynamic spatio-temporal graphs capture evolving relationships among nodes, where connections and attributes change frequently over time intervals.

2.1.1 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) [19] stand out as the most referenced models in the field of GNNs, widely adopted as the predominant architecture in practical applications. Their main functionality lies behind the concept of *spectral graph convolution* which enables the model to learn the features of neighbouring nodes. This operation is defined as the multiplication of a signal $x \in \mathbb{R}^N$ (a scalar for every node) with a filter $g_\theta = diag(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain:

$$g_\theta * x = U g_\theta U^T x \tag{2.1}$$

U is a matrix of the eigenvectors of the normalized graph Laplacian $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. I is the identity matrix, A is the adjacency matrix of the nodes of the graph and D is the degree matrix of the nodes of the graph. L can be decomposed into $L = U \Lambda U^T$. Λ is a diagonal matrix with the eigenvalues of the graph and $U^T x$ is the Fourier transform of x.

Since the computation of the normalized graph Laplacian L could lead to the problem of vanishing gradient, GCNs adopt the following renormalization trick in their calculation:

$$I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \tag{2.2}$$

with $\tilde{A} = A + I$ and $\tilde{D_{ii}} = \sum_j \tilde{A}_{ij}$.

Given a matrix of N nodes and C features $X \in \mathbb{R}^{N \times C}$ and given a trainable parameters matrix of C features and F filters $W \in \mathbb{R}^{C \times F}$, the GCN operation can be illustrated as:

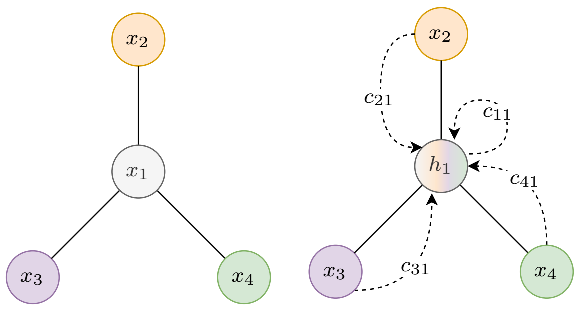
$$H = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X W) \tag{2.3}$$

with σ being a selected non-linear activation function. The operation can be repeated several times in order to encode a deeper representation of the network.

This definition can be expressed from a node-wise perspective as seen in Figure 2.3. Given a node i, its updated representation h_i through the GCN operation is seen as:

$$h_i = \sigma(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} x_j W) \tag{2.4}$$

Where N_i is the neighbourhood of i, $c_{ij} = \sqrt{d_i d_j}$ and $d_i = |N_i|$ denotes the degree of node i.



(a) Example graph. (b) GCN operation. $h_1 = \sigma(\sum_{j \in \mathcal{N}_1} \frac{1}{c_{1j}} x_j W)$

Figure 2.3: Illustration of the GCN mechanism.

2.1.2 Graph Attention Networks

Graph Attention Networks (GATs) [20] revisit the node-wise update rule of GCNs expressed in Equation 2.4, by presenting an attention-driven design for classifying nodes within graph-based data. The concept involves generating hidden representations for each node in the graph by focusing on its neighbouring nodes, employing a self-attention approach.

Considering a set of input node features $X = \{x_1, ..., x_N\}$, with $x_i \in \mathbb{R}^F$, where N is the number of nodes and F is the number of features in each node, a common linear transformation is employed as an initial stage. This is characterized by a shared weight matrix $W \in \mathbb{R}^{F' \times F}$ transforming the node features X in a hidden representation with F' features. Following this, self-attention is executed among the nodes using a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ between couples of neighbour nodes i, j to calculate attention coefficients e_{ij} that indicate the importance of node j's features to node i:

$$e_{ij} = a(Wx_i, Wx_j) \tag{2.5}$$

Attention coefficients are then normalized across the neighbourhood N_i of each node i following the function:

$$\alpha_{ij} = \text{softmax}(\frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}) \tag{2.6}$$

The updated representation of node i is finally obtained by updating the GCN operation in Equation 2.4:

$$h_i = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} x_j W) \tag{2.7}$$

The attention coefficient computation process is observed in Figure 2.4.

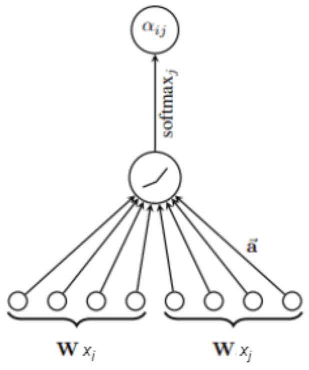
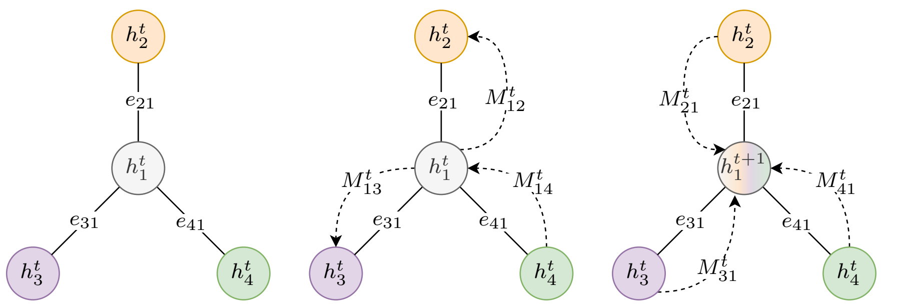


Figure 2.4: Illustration of the Graph Attention Network attention coefficient computation.

2.1.3 Message Passing Neural Networks

Message Passing Neural Networks (MPNNs) [21] are based on the *message passing* mechanism, which is illustrated in Figure 2.5. The representation of the graph nodes is iteratively updated by the encoded information obtained from their neighbouring nodes. This process is repeated multiple times until the nodes converge to a representation that captures the global structure of the graph.



(a) Example graph. (b) Message generation. (c) Message aggregation and representation update.

Figure 2.5: Illustration of the message passing mechanism.

In detail, the process at each iteration t can be divided into the following steps:

1. **Message generation:** Every node i in the graph generates a message M_{ij}^t for each of its neighbors j. Messages are defined by a learnable function ϕ of the node features h_i^t, the neighbour features h_j^t, and the eventual edge weight among them e_{ij}:

$$M_{ij}^t = \phi(h_i^t, h_j^t, e_{ij}) \tag{2.8}$$

2. **Message aggregation:** Each node i aggregates the received messages by its neighbours ($N(i)$), using a permutation-invariant function θ, meaning the order of message reception doesn't affect the result, such as a

summing or an averaging:

$$m_i^{t+1} = \theta_{j \in N(i)}(M_{ij}^t) \tag{2.9}$$

3. **Representation update:** Finally, each node i updates its representation h_i^{t+1} through a learnable function U_t depending on its current attributes h_i^t and the aggregated information of its neighbours m_i^{t+1}:

$$h_i^{t+1} = U_t(m_i^{t+1}, h_i^t) \tag{2.10}$$

2.1.4 Spatio-Temporal Graph Neural Networks

Spatio-Temporal Graph Neural Networks (STGNNs) [22] are extensions of GNNs that are designed to work with spatio-temporal graph data. Spatio-temporal graphs are data structures where the nodes have both spatial and temporal information associated with them. This type of data is common in applications such as traffic forecasting, weather forecasting, and disaster management.

STGNN methods fall into two main categories, namely: *Sequential STGNNs*, where patterns in space and time are learned separately and then combined to understand their relationship, and *Simultaneous STGNNs* which are models that simultaneously handle spatial connections and temporal data to create unified spatial-temporal embeddings. To simplify, the Sequential approach deals with time step by step, while the Simultaneous processes space and time together within one layer. The Sequential method is adept at building time series models for graph data, excelling in tasks involving time considerations but they demand substantial computation for each time step. On the other hand, the Simultaneous approach efficiently manages spatial and temporal data within a single layer, yet it might not suit all tasks, especially those emphasizing time series analysis.

Among different Successive STGNN applications, *RNN-based methods* and *Transformer-based methods* stand out in order to respectively capture short-term and long-term temporal dependencies in the temporal network.

RNN-based methods They integrate *Recurrent Neural Networks (RNNs)* with GNNs to model spatio-temporal data. This integration can be performed by either modifying the inner operations of the RNN cells through Graph Neural Network methodologies aimed at extracting the spatial correlations or by performing recursively spatial and temporal encodings in sequence.

The *Graph-Convolutional Long-Short Term Memory network (GC-LSTM)* [23], for instance, applies a modification to the classical *Long-Short Term Memory (LSTM)* cell. In its original definition, the cell state and the output are updated recursively from the input sequence using gating operations involving matrix multiplications. In GC-LSTM cells, the cell state and output are updated by replacing the matrix multiplications with graph convolutions.

Another example of a RNN-based method is *T-GCN* [24]. This model uses a recursive structure to encode the spatial and temporal correlation. In detail, at each time step, GCN and *Gated Recurrent Unit (GRU)* layers work in sequence to learn spatial and temporal relationships from graph data as illustrated in Figure 2.6.

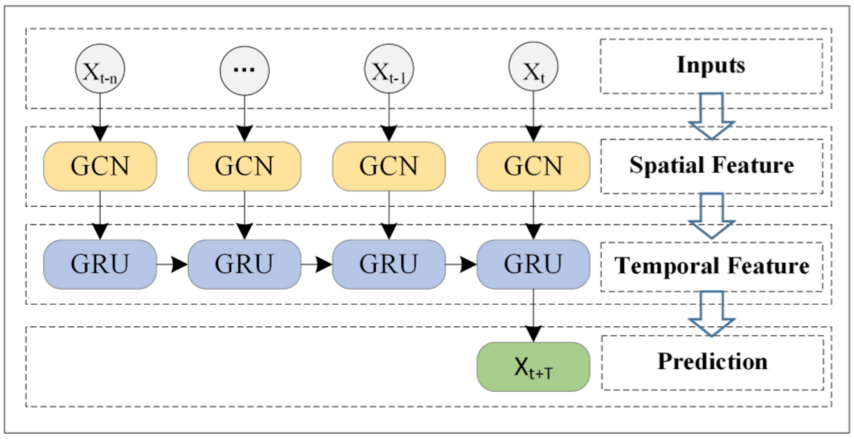


Figure 2.6: Illustration of the process of T-GCN.

Given $f(A, X_t)$ the GCN operation applied on the graph X at timestep t

using the adjacency matrix A and W_u, W_r and W_c learnable parameter matrices, the T-GCN operation can be decomposed in the following steps. Firstly, the update gate result u_t is obtained from the GCN operation on the network at time t and the concatenation with the hidden state h at time $t-1$:

$$u_t = \sigma(W_u[f(A, X_t), h_{t-1}]) \tag{2.11}$$

Then, the reset gate result r_t is obtained in a similar fashion:

$$r_t = \sigma(W_r[f(A, X_t), h_{t-1}]) \tag{2.12}$$

Next, a concatenation of the GNN operation on the network at time t and the elment-wise product of the result of the reset gate and the previous hidden state is computed:

$$c_t = tanh(W_c[f(A, X_t), (r_t * h_{t-1})]) \tag{2.13}$$

Finally, the hidden state h at time t is obtained as follows:

$$h_t = u_t * h_{t-1} + (1 - u_t) * c_t \tag{2.14}$$

Transformer-based methods They connect the spatial-capturing properties of GNNs with the power of *transformer* architectures in order to encode long-term temporal relationships in the data. In contrast to conventional RNNs, transformers handle time series data comprehensively rather than sequentially, through recursive processes. Specifically, a GNN derives spatial embeddings of the graph nodes at every time point, which are then input into a transformer employing self-attention mechanisms to calculate similarity scores a-cross the entire time sequence. Transformers operate without relying on past hidden states to capture dependencies on previous timestamps. Instead, each step directly accesses all other steps, mitigating the risk of losing past information.

As an example, *Spatio-Temporal Attention Graph Isomorphism Network (STAGIN)* [25] in order to extract temporal information on the spatio-temporal graph G employs a single-headed transformer encoder upon the sequence of graph features $(\tilde{h}_{G(1)}, ..., \tilde{h}_{G(t)})$ obtained through spatial encodings of the graph at each timestep $T \in [1, t]$. The resulting temporal encodings $(h_{G(1)}, ..., h_{G(t)})$ are then averaged to obtain the final graph representation $h_{G_{dyn}} = \sum_{t \in T} \frac{h_{G(t)}}{|T|}$. The process is observed in Figure 2.7.

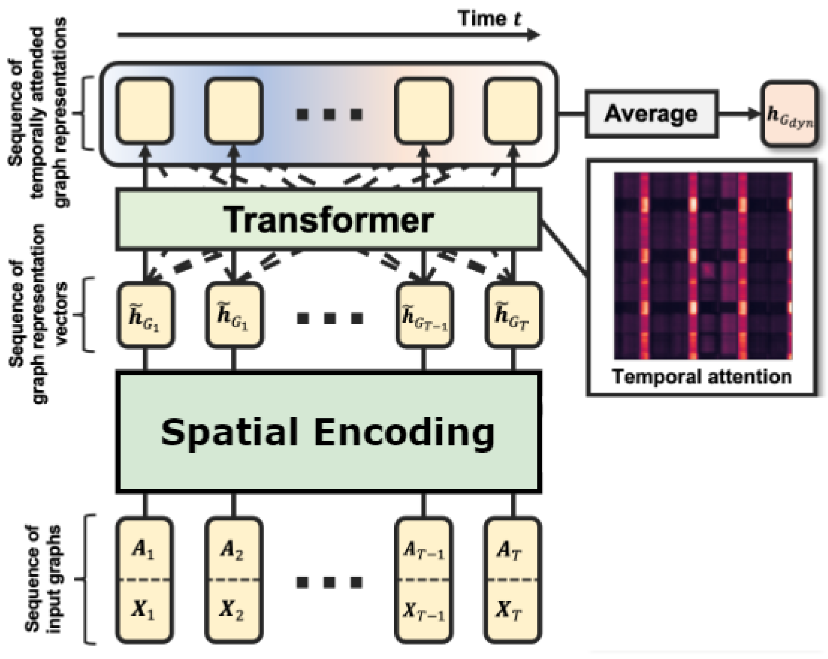


Figure 2.7: Illustration of STAGIN spatio-temporal encoding mechanism.

Simultaneous STGNNs include *Spatial Attention-based methods* and *Temporal Attention-based methods*.

Spatial Attention-based methods They gauge input similarities and blend them based on softmax-calculated similarities. Unlike standard transformer-like models that derive similarity from transformed features, the crucial point of employing attention, as proposed in *(2.5+1)D Spatio-Temporal Scene Graphs for Video Question Answering* [26], lies in using a similarity defined by the spatio-temporal adjacency of graph nodes within a $(2.5{+}1)$D scene graph. For any two nodes $v_1, v_2 \in V$, a similarity kernel k is defined to capture spatial-temporal proximity between v_1, v_2 at scales δ_S and δ_T for spatial and temporal signals respectively. The (2.5+1)D-Transformer can then be expressed as:

$$F = \text{softmax}(K(V, V \mid \delta_S, \delta_T))V_F \tag{2.15}$$

Here, K represents the spatio-temporal kernel matrix formed on V for each node pair, and V_F denotes the Value matrix transformed from node embeddings. This kernel merges features from nearby nodes in the graph, considering both spatial and temporal proximity.

Temporal Attention-based methods They employ the attention mechanism to assign weights to graph edges based on temporal distance. These weights are determined by assessing similarity between node features connected by each edge across different time steps. This approach enables the model to concentrate on neighboring nodes that offer the most relevant temporal information for the ongoing prediction task. For instance, *Graph Multi-Attention Network (GMAN)* [27] defines temporal attention between two time steps of a particular node as the dot product of spatial-temporal embedding concatenation and the node's hidden embedding at those time steps.

2.2 GNNs & Explainability

GNNs generally operate as *black-box* models due to their non-transparent nature, which obscures the reasoning behind their decisions. Hence, *Post hoc* explanations become essential to highlight their decision-making processes. These explanations are pursued after the models predictions, aiming to clarify in retrospect the reasoning behind specific decisions. Post-hoc explanation techniques comprise two main categories: *model explanation* and *outcome explanation*. Model explanation methods aim to comprehensively understand the behavior of a black box model by seeking why it generates specific predictions, developing insights from its structure and parameters. Conversely, outcome explanation techniques offer localized explanations highlighting the causal relationship between particular inputs and their predicted outcomes. A hybrid technique, termed *model inspection*, emerges from both model and outcome explanations, employing visual or textual representations to elucidate specific properties or decisions of the black box model [13, 14].

For what concerns GNNs, explanations often follow an outcome explanation or model inspection methodology. These methods are broadly divided into two groups: *non-GNN-originated methods*, which stem from other Deep Learning domains, and *GNN-originated methods*, specifically tailored for GNNs. Recent attention has primarily focused on GNN-originated methods due to their novelty. In understanding GNNs, their explanations often manifest as important subgraphs within their input graphs data. While some explanation methods directly output these subgraphs, most methods assign importance scores to each edge in the graph [28].

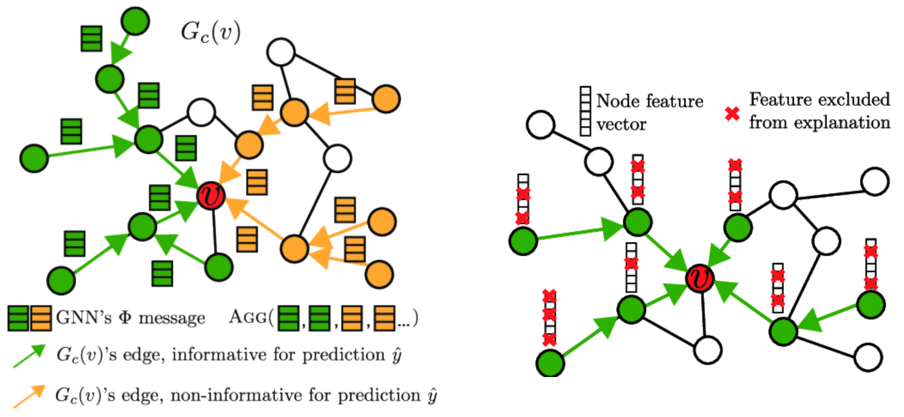
As these networks process graph-structured data, understanding the precise logic leading to their outputs becomes challenging. Firstly, due to graphs combinatorial nature, seeking important sub-structures involves assessing numerous combinations to optimize specific predictions, presenting a major obstacle. Additionally, graphs may include node attributes and edge connectivity, both of which can impact predictions and require their joint consideration in the explanation process. Moreover, these explanations must be flexibile in order to accommodate different GNN architectures [29].

When it comes to STGNNs, the challenge in explaining their behavior intensifies due to the reason that they incorporate the temporal dimension on top of the spatial one. These models not only deal with spatial relationships among elements of static graphs but also account for how these relationships evolve over time. The growing interest in explaining GNN models, coupled with the unique challenges posed by unfolding the behavior of STGNNs, has resulted in a scarce academic literature addressing the explainability of STGNNs.

This section offers an overview of various GNN-originated explainability methods and their limited current applications for STGNNs. These methods have been explored to construct the explanation pipeline for the STGNN utilized in this thesis. This pipeline aims to provide an easily understandable and transparent model inspection approach by extracting a significant subgraph from the input traffic network data. This subgraph serves to explain the predictions made by the STGNN.

2.2.1 GNNExplainer

GNNExplainer [30] is the first general, model-agnostic outcome explanation approach adopted to explain GNN-based models on any static graph-based task. It aims to produce an explanation by obtaining a subsection of the computation graph and specific node features that significantly impact the model Φ's predictions Y (Figure 2.8).


$G_c(v)$
v
GNN's Φ message
AGG(, , , ...)
$G_c(v)$'s edge, informative for prediction $\hat{y}$
$G_c(v)$'s edge, non-informative for prediction $\hat{y}$
Node feature vector
Feature excluded from explanation

(a) GNNExplainer identifies a small set of nodes and edges (green) that are crucial for the prediction in v.

(b) In addition, GNNExplainer identifies what feature dimensions of the selected nodes are important for the prediction of v.

Figure 2.8: Example of the GNNExplainer approach for node classification.

The objective with a given input graph G is to obtain a sub-graph $G_S \subseteq G$ and the corresponding features $X_S = \{x_j | v_j \in G_S\}$ through a learned mask that significantly influence Φ's predictions. To quantify importance, mutual information MI is employed following the given optimization framework that aims at maximizing MI between the explanations and the original model's predictions:

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y \mid G_S, X_S) \tag{2.16}$$

Where the mutual information MI quantifies the change in the conditional entropy H between $Y = \Phi(G, X)$ and $Y = \Phi(G_S, X_S)$.

This objective is translated in practice by finding the edge mask M that minimizes the following computation:

$$\min_{M} H(Y \mid G = A \odot \sigma(M), X = X_S) \tag{2.17}$$

With σ being the sigmoid function, A the adyacency matrix of G and $\odot$ the element-wise matrix multiplication operation.

GNNExplainer faces scalability issues due to parameter size scaling with the input graph size. Moreover, it offers only individual-level explanations, lacking a comprehensive understanding of predictions on a global scale. Another drawback is its inability to take into account the temporal dimension of spatio-temporal Graph, rendering it unusable to explain STGNNs.

2.2.2 GraphMask

GraphMask [31] is a post-hoc model inspection GNN explanation technique, that derives a global understanding of the model behavior. It detects which edges (u, v) at each layer k of the GNN can be removed without affecting model outcomes.

The elimination of an edge (u, v) at layer k containing message $m_{u,v}^{(k)}$ is performed by its replacement by a baseline $b^{(k)}$ guided by a single layer neural network binary classifier z trained on the whole training data, which performs a binary decision $z_{u,v}^{(K)} \in \{0,1\}$ such that:

$$\tilde{m}_{u,v}^{(k)} = z_{u,v}^{(k)} \cdot m_{u,v}^{(k)} + b^{(k)} \cdot (1 - z_{u,v}^{(k)}) \tag{2.18}$$

Given a GNN Φ of L layers, a graph G, with edge set $\mathcal{E}$, and input embeddings X the task of GraphMask is to identify a set $G_S = \{G_S^{(1)}, ..., G_S^{(L)}\}$ of informative sub-graphs with minumn number of edges at every layer. Minumum divergence between the predictions of Φ obtained with the original graph G and the subgraph G_S is computed through a divergence function $D_*[f(G,X)|| f(G_S,X)]$. The objective is defined over a dataset $\mathcal{D}$ as follows:

$$\max_{\lambda} \min_{\pi,b} \sum_{G,X\in\mathcal{D}} (\sum_{k=1}^{L} \sum_{(u,v)\in\mathcal{E}} \mathbb{1}_{[\mathbb{R}\neq 0]}(z_{u,v}^{(K)})) + \lambda(D_*[\Phi(G,X)||\Phi(G_S,X)] - \beta) \tag{2.19}$$

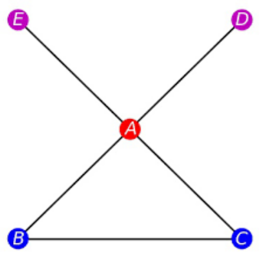
Where π is the learned parameter of z, $\lambda \in \mathbb{R} \geq 0$ denotes the Lagrange multiplier, $\mathbb{1}$ is the indicator function and β a set tolerance level.

GraphMask offers scalability and considers the dataset at a global level. However, it requires knowledge of the GNN's internal structure and is not suitable to explain STGNNs as it is not developed in order to take into account temporal dependencies. Furthermore, it falls in the paradox of explaining a black-box model using another black-box model z in its subgraph extraction process.

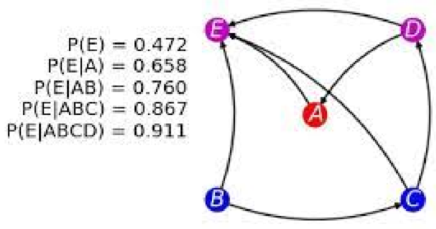
2.2.3 PGM-Explainer

Probabilistic Graphical Model Explainer (PGM-Explainer) [32] employs the Probabilistic Graphical Model (PGM) to construct model-agnostic outcome explanations for GNNs. PGMs are statistical models that encode intricate distributions in a multi-dimensional space using graph-based representations. In detail, it leverages *Bayesian networks*, a popular PGM type which depict conditional dependencies among variables through directed acyclic graphs which are known for their intuitive representations. An illustrative example of the

PGM-Explainer process is seen in Figure 2.9.



(a) A graph containing the target prediction E to be explained.



(b) A PGM-explanation in the form of Bayesian network. It estimates the conditional probability that node E has the predicted role given the other nodes.

Figure 2.9: Depiction of an example of the PGM-Explainer process.

PGM-Explainer involves finding the optimal Bayesian network, $\mathcal{B}$, through an optimization process considering the target prediction obtained by the GNN Φ:

$$\text{argmax}_{\mathcal{B}} R_{\Phi,t}(\mathcal{B}), \ \text{s.t.} |\mathcal{V}(\mathcal{B})| \leq M, t \in \mathcal{V}(\mathcal{B}) \tag{2.20}$$

where $\mathcal{V}(\mathcal{B})$ is the set of random variables in the Bayesian network $\mathcal{B}$, t is the random variable corresponding to the target prediction and R is a fitness function. The first constraint limits the number of variables in $\mathcal{B}$ by a given constant M to encourage a compact solution, while the second guarantees the target prediction is included in the explanation.

The cons of this architecture are that the explanations are provided solely at instance level and that the learning process of Bayesian Networks is very computationally expensive. Similar to the previous models, PGM-Explainer is not aimed to explain STGNN architectures.

2.2.4 SubgraphX

SubgraphX [33] adopts a different outcome explanation approach. Given Φ as a GNN model and G an input graph, the aim in the explanation task is to identify the most influential sub-graph for the prediction of Φ. To ensure comprehensibility, it focus solely on connected sub-graphs, as disconnected nodes pose challenges in interpretation. The collection of connected sub-graphs in G is denoted as $\{G_1, ..., G_n\}$, where n signifies the various connected sub-graphs within G. The explanation for the prediction concerning the input graph G can be defined as follows:

$$G^* = \text{argmax}_{|G_i| \leq N_{min}} \text{Score}(\Phi, G, G_i) \tag{2.21}$$

where Score is a scoring function for evaluating the importance of a subgraph given the trained GNNs and the input graph and N_{min} is an upper bound on the size of subgraphs so that the explanations are succinct enough.

In order to obtain candidate sub-graphs, the model generates a search tree with the input network as the root node and connected sub-graphs as children nodes. Each edge in the search tree denotes that the graph associated with a child node can be obtained by performing node-pruning from the graph described by its parent. A *Monte Carlo Tree Search (MCTS)* method explores the tree, with statistics guiding the selection of informative subgraphs for GNN predictions. Subgraphs are scored using the Shapley value, and the most relevant one is retained for explanation.

Notably, this algorithm does not require prior knowledge of the GNN's structure. However, it operates at instance level may face scalability challenges when dealing with large instances due to the need for constructing extensive search trees. It also shares with the previous illustrated models, the lack of adoption to STGNN structures.

2.2.5 PGM-Explainer for STGNNs

The approach used in *An Explainer for Temporal Graph Neural Networks* [34] builds upon the *PGM-Explainer* framework illustrated in chapter 2.2.3, aiming to provide explanations specifically tailored for STGNNs. In addition, among the experimental setups employed in the paper, the methodology has been applied to explain traffic forecasting predictions, aligning with the task in the thesis. This paper stands out as one of the rare contributions in the literature that delves into addressing the explainability of STGNNs. Its explanation framework can be observed in Figure 2.10.

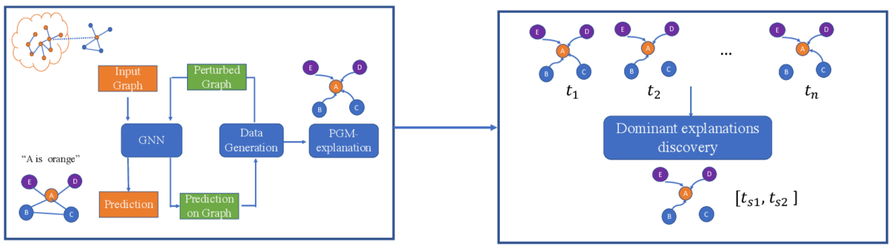


Figure 2.10: Explanation framework of PGM-Explainer for STGNNs.

In order to simplify interpretation, the STGNN model is reformulated. This reformulation allows for independent explanations at various temporal snapshots, exploiting the capabilities of the static graph PGM-Explainer. In detail, given a STGNN Φ, a time window T, a graph G and its feature matrices X the original predictions:

$$Y = \Phi(G, [X_t, ..., X_{t+T-1}]) \tag{2.22}$$

are decomposed into a sequential temporal implementation:

$$\forall i \in [t, t+T-2], \; H_{i+1} = \hat{\Phi}(G, X_i, H_1) \tag{2.23}$$

where H_i is the hidden feature configuration computed by the STGNN at timestep i and $H_{T+t-1} = Y$.

At each timestep t, the PGM-Explainer finds the optimal Bayesian network, $\mathcal{B}_t$, through the optimization process seen in Equation 2.20. Through the use of a score function F_D the concept of *interesting Bayesian network* and *temporal dominant interesting Bayesian network* are formulated. Specifically, a Bayesian network $\mathcal{B}$ qualifies as an interesting Bayesian network within a designated temporal window $[t_s, t_e]$ if $F_D(\mathcal{B})$ surpasses a set threshold. Moreover, if the temporal window $[t_s, t_e]$ doesn't fall within any other interesting Bayesian network's temporal scope, $\mathcal{B}$ is identified as a temporal dominant interesting Bayesian network within that time frame. The dominant Bayesian network defines a concise rationale of the predictions.

The search of dominant Bayesian graph is obtained through a pruning approach. In essence, when a temporal dominant Bayesian network is identified within a time frame $[t_1, t_2]$, it implies the absence of any other dominant Bayesian graph within a smaller time frame $[t_1', t_2'] \subset [t_1, t_2]$. This characteristic guides a search method starting from the largest time frame $[t_1, t_2]$. The algorithm computes the interest level of each Bayesian graph across this full time frame and recursively shrinks the time frame if no dominant Bayesian network is found. Conversely, upon discovering a dominant Bayesian network within $[t_1, t_2]$, it prunes the subset of this time frame, eliminating the need to compute interest measures for other Bayesian networks inside that subset.

The drawbacks of this architecture are the same as PGM-Explainer. In particular, explanations lack a global view as they are assessed at instance level. Moreover, computing Bayesian Networks requires high computation resources. One of its advantages is that the model is tailored specifically for explaining STGNNs.

2.2.6 TGNN-Explainer

Another methodology introduces *Temporal GNN-Explainer (TGNN-Explainer)* [35], a model inspection method applied to STGNNs node classification architectures. This approach defines an input spatio-temporal graph as a sequence of events $\mathcal{S} = \{e_1, e_2, ...\}$. Each $e_i = \{n_{u_i}, n_{v_i}, t_i, att_i\}$ means that the node

n_{u_i} and n_{v_i} have an edge at timestamp t_i with weight att_i.

Given the sequence of events and a STGNN Φ, TGNN-Explainer elucidates why the model predicts an event e_k would occur or not. Specifically, it finds a subset of events $\mathcal{R}^k$ from all the previous events G^k to maximize the mutual information $MI(Y_k, \mathcal{R}^k)$. Y_k is the original prediction by $\Phi(G^k)[e_k]$. Maximizing the mutual information $MI(Y_k, \mathcal{R}^k)$ consists in minimizing conditional entropy $H(Y_k \mid \mathcal{R}^k)$ such that:

$$\min_{\mathcal{R}^k} - \sum_{c=0,1} \mathbf{1}(Y_k = c) \log P(Y_{new} = c \mid \mathcal{R}^k) \tag{2.24}$$

where $P(Y_{new} \mid \mathcal{R}^k)$ is computed by $\Phi(\mathcal{R}^k)[e_k]$ and c indicates whether the event occurred or not.

Identifying an important subset of events $\mathcal{R}^k$ for the target event e_k is a combinatorial optimization problem where any subset of G^k could be the explanation. In order to limit the combinatorial search space, TGNN-Explainer proposes to compute a heuristic correlation score among events through a *navigator*, providing thus a global understanding of the correlation patterns in the data. Specifically, the navigator constitutes a feed-forward neural network denoted as $h_\theta(e_j, e_k)$. Its role involves deducing the correlation score of an event e_j concerning a specific target event e_k.

The navigator is concretely adopted in a *Monte Carlo Tree Search (MCTS)* employed to seek the subset of events $\mathcal{R}^k$ that explains the target event e_k. Initially, the root node of the MCT represents a collection of all potential events. Multiple rounds, or rollouts, expand nodes in the search tree, where each node indicates a viable subset of events within the search space. Each round involves:

1. Selecting a path from the root to a leaf node which meets a certain sparsity threshold.

2. Creating new nodes by eliminating less important events based on the

navigator h_θ correlation scores.

3. simulating rewards for these new nodes using Φ.

4. Updating statistical information in the path nodes by backpropagating the leaf node's reward.

Finally, the explanation result is the node with the best reward among the multiple rounds.

This methodology offers a global understanding of the explanations of a STGNN as its search is guided by a correlation score learned from a training dataset. Unfortunately, its application is solely applicable to node classification and, similarly to GraphMask, it becomes paradoxical as it explains a black-box model using another black-box model h_θ to compute navigation heuristics.

2.2.7 STExplainer

STExplainer [36] introduces a method called *structure-distilled Graph Information Bottleneck (structure-distilled GIB)* for enhancing the inherent interpretability of STGNNs. In the related paper, the architecture has been tested in explaining STGNN predictions in different domains, including the traffic forecasting one. Its goal is to distill valuable information from complex spatio-temporal graph data by the acquisition of a small subset of spatio-temporal graph structures according to the objective:

$$\min_{\mathbb{P}(\mathcal{G}_S|\mathcal{G})} -MI(Y, \mathcal{G}_S) + \beta \cdot MI(\mathcal{G}, \mathcal{G}_S) \tag{2.25}$$

where The subgraph $\mathcal{G}_S$ represents the distilled subgraph obtained from the conditional probability distribution $\mathbb{P}(\mathcal{G}_S \mid \mathcal{G})$ given the original graph $\mathcal{G}$. Moreover, Y are the predictions by the STGNN, $MI(Y, \mathcal{G}_S)$ measures the mutual information between the predictions and the distilled subgraph, while

$MI(\mathcal{G}, \mathcal{G}_S)$ is the mutual information between the original spatio-temporal graph and the distilled subgraph and β a weight parameter.

STExplainer, firstly produces a spatial-temporal encoding of the input spatio-temporal graph through spatial and temporal attention neural modules. Then, the *Gumbel-Softmax reparameterization trick* is adopted to compute the spatio-temporal probabilities $p_{uv}^{(s)}$ and $p_{uv}^{(t)}$ for each edge of the encoded graph uv in a differentiable manner. Next the adjacency matrix structures $A_S^{(s)}$ and $A_S^{(t)}$ defining the spatial and temporal subgraph $\mathcal{G}_S = (\mathcal{G}_S^{(s)}, \mathcal{G}_S^{(t)})$ are obtained such that:

$$\begin{aligned} A_S^{(s)} &= \alpha^{(s)} \odot A^{(s)},\ \alpha_{uv}^{(s)} \sim \text{Bern}(p_{uv}^{(s)}) \\ A_S^{(t)} &= \alpha^{(t)} \odot A^{(t)},\ \alpha_{uv}^{(t)} \sim \text{Bern}(p_{uv}^{(t)}) \end{aligned} \tag{2.26}$$

where $A^{(s)}$ and $A^{(t)}$ are the original spatial and temporal adjacency matrices and $\odot$ the element-wise matrix product. $\alpha^{(s)}$ and $\alpha^{(t)}$ are the spatio-temporal matrices of selected edges in the input subgraph sampled on a Bernoulli distribution considering their probabilities in $p^{(s)}$ and $p^{(t)}$. Finally, the predictions of the STExplainer, Y, are obtained by encoding the spatio-temporal features on the distilled subgraph $\mathcal{G}_S$ followed by a prediction layer. In practice, this is achieved by performing the message passing or the GCN operation (both described in section 2.1) just on the selected spatial edges in $A_S^{(s)}$ and by performing temporal attention just on the node and edges of $A^{(t)}$.

The training of STExplainer is based on a triad loss function:

$$\mathcal{L} = \mathcal{L}_0 + \lambda_1 \mathcal{L}_{\text{S-GIB}} + \lambda_2 \mathcal{L}_{\text{T-GIB}} \tag{2.27}$$

where $\mathcal{L}_0$ maximizes the mutual information $MI(Y, \mathcal{G}_S)$ of Equation 2.25, by minimizing the error between the ground truth and the predictions of the model Y using the information in the distilled subgraph $\mathcal{G}_S$. $\mathcal{L}_{\text{S-GIB}}$ and $\mathcal{L}_{\text{T-GIB}}$ are the losses for the spatial-temporal subgraph weighted by parameters λ_1 and λ_2 respectively. They maximize the mutual information $MI(\mathcal{G}, \mathcal{G}_S)$ in Equation

2.25, by minimizing the errors:

$$\begin{aligned}\mathcal{L}_{\text{S-GIB}} &= \mathbb{E}_{\mathcal{G}^{(s)}}[\text{KL}(\mathbb{P}(\mathcal{G}_S^{(s)} \mid \mathcal{G}^{(s)})||\mathbb{Q}_2(\mathcal{G}_S^{(s)}))] \\ \mathcal{L}_{\text{T-GIB}} &= \mathbb{E}_{\mathcal{G}^{(t)}}[\text{KL}(\mathbb{P}(\mathcal{G}_S^{(t)} \mid \mathcal{G}^{(t)})||\mathbb{Q}_2(\mathcal{G}_S^{(t)}))]\end{aligned} \tag{2.28}$$

where $\mathbb{P}(\mathcal{G}_S^{(s)} \mid \mathcal{G}^{(s)})$ and $\mathbb{P}(\mathcal{G}_S^{(t)} \mid \mathcal{G}^{(t)})$ are the conditional spatial and temporal probability distributions computed, considering the spatial and temporal node sets with $\mathcal{V}^{(s)}$ and $\mathcal{V}^{(t)}$, as:

$$\begin{aligned}\mathbb{P}(\mathcal{G}_S^{(s)} \mid \mathcal{G}^{(s)}) &= \textstyle\prod_{u,v\in\mathcal{V}^{(s)}} \mathbb{P}(\alpha_{uv}^{(s)} \mid p_{uv}^{(s)}) \\ \mathbb{P}(\mathcal{G}_S^{(t)} \mid \mathcal{G}^{(t)}) &= \textstyle\prod_{u,v\in\mathcal{V}^{(t)}} \mathbb{P}(\alpha_{uv}^{(t)} \mid p_{uv}^{(t)})\end{aligned} \tag{2.29}$$

while $\mathbb{Q}_2(\mathcal{G}_S^{(s)})$ and $\mathbb{Q}_2(\mathcal{G}_S^{(t)})$ are the prior spatial and temporal distributions of $\mathcal{G}_S$ computed as:

$$\begin{aligned}\mathbb{Q}_2(\mathcal{G}_S^{(s)}) &= \mathbb{P}(n^{(s)}) \textstyle\prod_{u,v=1}^{n} \mathbb{P}(\alpha_{uv}^{'(s)}) \\ \mathbb{Q}_2(\mathcal{G}_S^{(t)}) &= \mathbb{P}(n^{(t)}) \textstyle\prod_{u,v=1}^{n} \mathbb{P}(\alpha_{uv}^{'(t)})\end{aligned} \tag{2.30}$$

where $\alpha^{'(s)} \sim \text{Bern}(r^{(s)})$ $\alpha^{'(t)} \sim \text{Bern}(r^{(t)})$ are selected spatial and temporal edges through a Bernoulli sampling on hyperparameters $r^{(s)}$ and $r^{(t)}$, while $n^{(s)}$ and $n^{(t)}$ are the spatial and temporal edges.

This methodology differs from the previously introduced as it offers an inherently interpretable STGNN model which outputs both predictions and the most important input data leading to them. Nonetheless, the subgraph extraction based on the edge sampling methodology is obtained on an encoded version of the original input graph through a black box architecture. Although the edge sampling method is transparent it is unclear which are the patterns that lead to the edge probabilities used for their selection and it is hence obscure why the subgraph attributes are important to the prediction.

2.2.8 Related Work

The thesis work focuses on explaining STGNN predictions in traffic forecasting. Previously introduced methods like *GNNExplainer* (section 2.2.1, *GraphMask* (section 2.2.2), *PGM-Explainer* (section 2.2.3), and *SubgraphX* (section 2.2.4) address explainability in static GNNs by extrapolating the input subgraphs that lead to the predictions, but they cannot derive explanations that consider the temporal and spatial component mutually. In existing studies, such as [37], explainers designed for static GNNs have been employed to explain temporal graph sequences predicted by STGNNs by treating each timestep separately. However, this approach has a limitation as it overlooks the temporal correlation in the explanation process.

STGNN explainability methods, similarly to the static ones, aim at outputting the rationale of the predictions of an STGNN by a subset of the input data, although they are specifically designed to work on STGNNs architectures while considering the temporal patterns in the data on top of the spatial ones. In academic literature, the methods designed to explain STGNNs are *PGM-Explainer for STGNNs* (section 2.2.5), *TGN-Explainer* (section 2.2.6) and *STExplainer* (section 2.2.7). In particular, *PGM-Explainer for STGNNs* and *STExplainer* have also been applied to explain forecasts within traffic networks, the same task of the thesis.

PGM-Explainer for STGNNs (section 2.2.5), similarly to the explainer used in the thesis work, describes the rationale of the predictions by the input data in a model-agnostic manner. Its rationale is a dominant Bayesian Networks that contains the input nodes with the higher conditional probability with respect to the predictions it has to explain. Its limitation lay in the fact that it is applied at instance level and it thus lacks a global view of the explanations. Moreover, it requires expensive computations to find the dominant Bayesian Networks. Countrarily, the explainer employed in the thesis generates explanations through both a global perspective and a local inspection of each instance. Furthermore, the rationale is not given by a Bayesian Network, but as a subgraph of the input network. The computation of the explainer used in the thesis work is not extensive as the search space is cut by a global heuristic.

TGN-Explainer (section 2.2.6) presents a large amount of similarities with the thesis work, as both provide a global and local understanding of the explanations in a model-agnostic manner by a rationale given as a subset of the input data. Both models use MCTS as a search methodology to extract the explanations. TGN-Explainer differs from the employed explainer because it is applied to node classification, a different task than the one described in the thesis. Its limitation lays in the fact that it uses a black-box model in its architecture to compute a global correlation score among input nodes and predicted nodes to explain. These scores are then used to guide the MCTS. The thesis work addresses this limitation by computing the global heuristic score between input and output nodes in a transparent manner, using traffic-domain related laws.

STExplainer (section 2.2.7) provides a framework to build inherently explainable STGNNs models. On the other hand, the thesis work focuses in building a post-hoc explainer that can be applied to any STGNN in a model-agnostic manner. The similarity of the architectures is that they both find explanations from subsets of the input network. A limitation of STExplainer lays in its subgraph extraction methodology, even though transparent in the process, it operates on an encoded version of the input graph via a black-box architecture, making the interpretability of the patterns on which the extraction is performed uncertain. The thesis overcomes this by conducting all search operations directly on the input network without encoding it. This approach ensures that the explainability methodology operates directly on human-interpretable data rather than on encoded or cryptic representations.

To sum up, the explainability methodology envisioned in the thesis aims to find the rationale of traffic forecasts by an STGNN through an important subgraph of the input network. It proposes an explainer able to capture both temporal and spatial correlations between input data and forecasted predictions transparently exploiting traffic-domain laws. Moreover, it deals with generating explanations through both a global perspective and a local inspection of each instance. Furthermore, its nature is post-hoc and model-agnostic as the explainer does not need to know the specifics of the implementation of the STGNN used for the predictions.

Its limitation lies in its domain-specific application of forecasts applied on traffic networks since it exploit traffic laws as prior knowledge. However, it intends to demonstrate that leveraging domain-specific knowledge in the explanations of black-box architectures enhances transparency and global insights, showcasing the potential for clear and effective explanations when domain expertise is utilized.

2.3 Verbal Explanations

Data-to-sequence systems offer a potent capability of generating verbal summarizations automatically from data sources, simplifying the communication of intricate information. Instead of relying on visualization methods, these systems utilize natural human language, the primary mode of human-to-human interaction. Moreover, they possess the ability to address their output content to suit users' preferences, backgrounds, or interests, making interactions more engaging for users. Content selection stands as a critical component within data-to-sequence systems, as it dictates which information among the available data should be conveyed to the user [38].

In addition, it is argued that describing the explanation in words has a greater effect than using visual representations as it is in line with the concept of the *social processes of understanding*. This involves an interactive process between the explainer and explainee, aiming to equip the explainee with adequate information to grasp the causes behind an event [15].

This section outlines different ways to turn structured data into natural language explanations. All these methods stand as potential choices for achieving the thesis goal: transforming pertinent sub-graphs, acquired through the explanation pipeline, into coherent and understandable narratives that elucidate the outputs of the STGNN model.

2.3.1 Template-Based Approaches

Historically, many data-to-sequence systems have relied on rule-based methods, employing template selection and completion to generate natural language text [39]. These approaches remain prevalent in practical applications due to their robustness and ability to produce high-quality output given sufficient time and resources. Furthermore, human control over the output ensures accuracy in representing the data.

However, while effective for simpler scenarios, rule-based systems often fall short when describing complex problems, requiring an extensive and time-consuming set of rules. Thus, the laborious nature of rule creation and the challenge of considering all possible variations make them impractical for many domains.

2.3.2 Scalable Micro-Planned Generation

Another interpretable solution for data-to-sequence translation is inspired by the pipeline presented in the paper *Scalable Micro-planned Generation of Discourse from Structured Data* [40]. While the primary focus of this model is generating natural language descriptions from tabular data, it has also been applied to knowledge graphs and JSON structures. The proposed approach can be observed in Figure 2.11 and it involves the following steps

1. Conversion of data into a canonical triple format $< e_1, r_{12}, e_2 >$, where two data entities e_1 and e_2 are linked by a relation r_{12}. The entities e_1 and

e_2 are tagged using a *Named Entity Recognizer (NER)*, which assigns domain-independent place-holder tags to them.

2. Generation of simple sentences from each canonical triple $< e_1, r_{12}, e_2 >$. A preprocessing step firstly transforms the relation term in the triple r_{12} into a verb phrase. Sentence generation is next obtained by feeding the triples to different *NLG* neural architectures. The resulting sentences of each model are then scored in terms of fluency and adequancy and the best results are kept for the successive steps.

3. Application of sentence compounding and co-reference substitution for improved fluency. Sentence compounding consists in taking pairs of sentences obtained at the previous steps and produce a compound complex sentence. Every sentence is split into a $< e_1, r_{vp}, e_2 >$ form where e_1 and e_2 are entities that appear in the input and r_{vp} is the relation verb phrase. If two sentences have either the same subject e_1 or the same object e_2, they can be combined using ‘AND’ to link their relationship phrases. When one sentence’s object matches the following sentence’s subject, a clause can be formed by introducing “who” or “which”. Otherwise, if neither e_1 nor e_2 match between sentences, they can be merged using ‘AND’. Co-reference substitution is finally applied to enhance paragraph coherence. It consists in a heuristic that replaces repeating entities with pronominal anaphora.

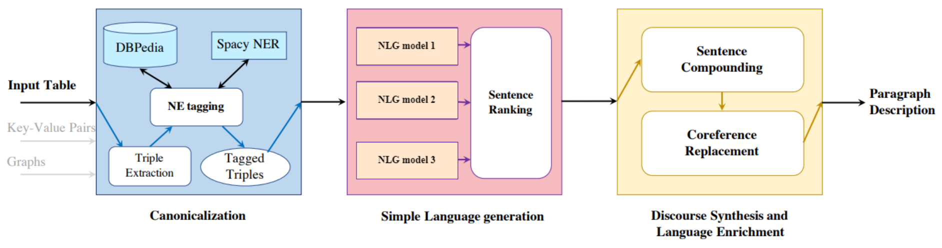


Figure 2.11: Scalable Micro-Planned Generation framework.

This approach is domain-generic and largely interpretable, although it incorporates non-transparent architectures such as neural models and pre-trained NERs.

2.3.3 CycleGT

CycleGT [41] involves the use of a self-supervised generative model, which includes both a *graph-to-sequence* and *sequence-to-graph* component and leverages backtranslation techniques. Its framework is illustrated in Figure 2.12.

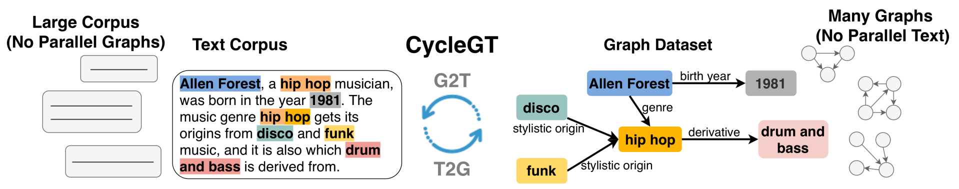


Figure 2.12: Illustration of CycleGT.

It considers two separate sets of data:

- A collection of N text sequences $D_T = \{t_i\}_{i=1}^N$
- A set of M graphs $D_G = \{g_j\}_{j=1}^M$

Both sets share a common latent content distribution denoted as z, yet they manifest in different forms, namely text and graphs. The likelihood of these sets can be expressed through the shared content z:

$$\log p(g) = \log \int_z p(g \mid z)p(z)dz \tag{2.31}$$

$$\log p(t) = \log \int_z p(t \mid z)p(z)dz \tag{2.32}$$

The objective involves training two models without supervision: a graph-to-sequence (G2S) neural model to generate text from graphs, and a sequence-to-graph (S2G) neural architecture which produces graphs from text. This assumes an implicit correspondence, although not strictly one-to-one, between text and graphs. The parameters of G2S are defined as θ and the ones of S2G as ϕ. Considering an unseen ground truth distribution, D_{Pair} (e.g., a test set), where each paired text and graph (t, g) shares identical content. The optimal goal is to maximize the log-likelihood of θ and ϕ across text and graph pairs (t, g) sampled from D_{Pair}:

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{(t,g)} \sim D_{\text{Pair}}[\log p(t \mid g, \theta) + \log p(g \mid t, \phi)] \tag{2.33}$$

The training procedure is based on the concept of back translation. This process ensures that a variable x and its bijective mapping function $f(\cdot)$ adhere to the equation $x = f^{-1}(f(x))$, where f^{-1} represents the inverse function of f. In CycleGT scenario, G2S and S2G act as inverse functions since one changes a graph into text while the other transforms text into a graph. Each text is hence matched with its back-translated form and each graph with its corresponding back-translated version through the objectives:

$$\mathcal{L}(\theta) = \mathbb{E}_{(t,\hat{g}) \in \hat{D}_{\text{Pair}}}[-\log p(t \mid \hat{g}, \theta)] \tag{2.34}$$

$$\mathcal{L}(\phi) = \mathbb{E}_{(\hat{t},g) \in \hat{D}_{\text{Pair}}}[-\log p(g \mid \hat{t}, \phi)] \tag{2.35}$$

Where $\hat{D}_{\text{Pair}}$ is a synthetic approximation of D_{Pair} generated by the two models $(t, \hat{g} = S2G_{\phi}(t))$ and $(\hat{t} = G2S_{\theta}(g), g)$ for all $t \in D_T, g \in D_G$. The sum of these 2 objectives reasonably approximates the log-likelihood expressed in Equation 2.33.

This solution limitation lies in the lack of interpretability of the applied neural models and its reliance on an additional text dataset, D_T.

Chapter 3

Design

3.1 Problem Statement

The thesis aims to create an *eXplainable Artificial Intelligence (XAI)* framework specifically designed for individuals without technical expertise. This system will explain short-term traffic speed forecasts within static spatio-temporal traffic networks, produced by a STGNN. The emphasis is on providing understandable explanations for both traffic jams and unrestricted traffic movements in the form of coherent narratives supported by graphical explanations. A static spatio-temporal traffic network is a data structure defined by $\mathcal{G} = \{\mathcal{G}_0, ..., \mathcal{G}_T\}$ which covers a discretized time window of T timesteps $\mathcal{T} = [0, ..., T]$. Each $\mathcal{G}_t = (\mathcal{V}, \mathcal{X}_t, \mathcal{E}, A)$, with $t \in \mathcal{T}$, is a static network representing the state of the nodes of $\mathcal{G}$ during timestep t where:

- $\mathcal{V} = \{v_1, ..., v_N\}$ is the static set of nodes of the network. This set is the same at each timestep. Moreover, each node v_i in the traffic network represents a location in the physical space (e.g.: a loop detector of a highway).

- $\mathcal{X}_t = \{X_t^{v_i} \in \mathbb{R}^{N \times F} \mid v_i \in \mathcal{V}\}$ defines the set of node attributes in the traffic network at timestep t, where N is the number of nodes and F the number of features. In the traffic forecasting domain, without any loss of generality, it is assumed that at every time step t, the data collected contains the average speed recorded within the discretized period. Specifically, within the traffic network for each node $v_i \in \mathcal{V}$ and each timestep $t \in \mathcal{T}$, $s_t^{v_i} \in \mathbb{R}$ describes the average speed measurement for v_i collected in the discrete time interval t, where $s_t^{v_i} \in X_t^{v_i}$.

- $\mathcal{E} = \{(v_i, v_j) \mid v_i, v_j \in \mathcal{V}\}$ is the set of edges connecting the nodes, fixed for the whole time period.

- $A = \{a_{v_i v_j} \mid v_i, v_j \in \mathcal{V}\}$ is the adjacency matrix that measures the spatial proximity $a_{v_i v_j}$ of pair of nodes v_i, v_j in the traffic network and that is also unchanged for the whole period.

The first phase in the framework pipeline, denoted as *prediction phase*, consists of the forecast of upcoming node speeds in the traffic network through a well-trained STGNN architecture denoted as Φ. This model serves as the predictor for the state of the traffic network nodes in the successive $\mathcal{T}' = [T+1, ..., T']$ discretized timesteps such that:

$$\hat{\mathcal{Y}} = \Phi(\mathcal{G}) \tag{3.1}$$

where $\hat{\mathcal{Y}} = \{\hat{\mathcal{Y}}_{T+1}, ..., \hat{\mathcal{Y}}_{T'}\}$ and $\hat{\mathcal{Y}}_{t'} = \{\hat{s}_{t'}^{v_i} \in \mathbb{R} \mid v_i \in \mathcal{V}\}$ define the set of predicted average node speeds in the traffic network at the discrete timestep $t' \in \mathcal{T}'$. The predicted traffic network in the successive timesteps can be thus defined as $\hat{\mathcal{G}} = \{\hat{\mathcal{G}}_{T+1}, ..., \hat{\mathcal{G}}_{T'}\}$, with $\hat{\mathcal{G}}_{t'} = (\mathcal{V}, \hat{\mathcal{Y}}_{t'}, \mathcal{E}, A)$ and $t' \in \mathcal{T}'$.

Within the forecasted network, there exist events that could be of particular interest to users, namely congestion or instances of free-flowing traffic. These occurrences stand out due to their direct impact on daily commutes and travel experiences. Congestions and free-flowing traffic greatly influence travel time, route planning, and overall convenience for individuals navigating through road networks. These events are denoted by subgraphs of the forecasted traffic network $\hat{\mathcal{G}}^* \subseteq \hat{\mathcal{G}}$, where $\hat{\mathcal{G}}^* = \{\hat{\mathcal{G}}^*_{T+1} \subseteq \hat{\mathcal{G}}_{T+1}, ..., \hat{\mathcal{G}}^*_{T'} \subseteq \hat{\mathcal{G}}_{T'}\}$.

The subgraphs $\hat{\mathcal{G}}^*$ comprise spatio-temporally connected nodes which speed attributes $\hat{\mathcal{Y}}^* = \{\hat{\mathcal{Y}}^*_{T+1} \subseteq \hat{\mathcal{Y}}_{T+1}, ..., \hat{\mathcal{Y}}^*_{T'} \subseteq \hat{\mathcal{Y}}_{T'}\}$ define a noteworthy predicted event, such as traffic congestion or free-flowing traffic. The set of nodes and edges of $\hat{\mathcal{G}}^*$ is not static over the time period as it may change at each timestep. In detail, Each $\hat{\mathcal{G}}^*_{t'}$ is defined as $\hat{\mathcal{G}}^*_{t'} = (\hat{\mathcal{V}}^*_{t'}, \hat{\mathcal{Y}}^*_{t'}, \hat{\mathcal{E}}^*_{t'}, A)$, with $t' \in \mathcal{T}'$ with:

- $\hat{\mathcal{V}}^*_{t'}$ being the set of nodes of $\hat{\mathcal{G}}^*$ considered at timestep t'. This set is defined as a different subset of $\mathcal{V}$ at each timestep $\hat{\mathcal{V}}^*_{t'} \subseteq \mathcal{V}$.

- $\hat{\mathcal{Y}}^*_{t'}$ defining a set that explains the average speeds of the nodes in $\hat{\mathcal{G}}^*$ at timestep t'. In detail, the set of speeds are obtained among the forecasted ones in $\hat{\mathcal{Y}}_{t'}$ considering just the set of nodes in $\hat{\mathcal{V}}^*_{t'}$ as $\hat{\mathcal{Y}}^*_{t'} = \{\hat{s}^{v_i}_{t'} \in \mathbb{R} \mid v_i \in \hat{\mathcal{V}}^*_{t'}\}$.

- $\hat{\mathcal{E}}^*_{t'}$ being the set of edges connecting the nodes at timestep t' of $\hat{\mathcal{G}}^*$. This set changes at each timestep t' and is defined as the subset of $\mathcal{E}$ limited to the nodes in $\hat{\mathcal{V}}^*_{t'}$ such that $\hat{\mathcal{E}}^*_{t'} = \{(v_i, v_j) \mid (v_i, v_j) \in \mathcal{E} \wedge v_i, v_j \in \hat{\mathcal{V}}^*_{t'}\}$.

The second phase is the *explanation phase* that aims at understanding the causes behind an event described by $\hat{\mathcal{G}}^*$. Extrapolating the causes leading to congestion or a free-flow could help users anticipate potential delays or smoother travel, enabling them to make informed decisions about their routes and travel plans. The factors that cause $\hat{\mathcal{G}}^*$ are extracted through a transparent model inspection and a model-agnostic explainer called ψ. This process aims to reveal the primary factors driving the predictions within $\hat{\mathcal{G}}^*$ generated by the predictive model Φ. The explainer ψ achieves this by identifying an important subgraph, denoted as $\tilde{\mathcal{G}} \subseteq \mathcal{G}$, within the input network $\mathcal{G}$. This subgraph encapsulates the most critical attributes that influence Φ in forecasting the event $\hat{\mathcal{G}}^*$. ψ operates independently of the specific predictive model as it conducts an outcome explanation process, shedding light on the rationale behind the forecasts. In detail, ψ returns the important subgraph $\tilde{\mathcal{G}}$ by means of the speed attributes of the event $\hat{\mathcal{Y}}^*$, the STGNN Φ and the input traffic network $\mathcal{G}$:

$$\tilde{\mathcal{G}} = \psi(\hat{\mathcal{Y}}^*, \Phi, \mathcal{G}) \tag{3.2}$$

The important subgraph is structured as $\tilde{\mathcal{G}} = \{\tilde{\mathcal{G}}_0 \subseteq \mathcal{G}_0, ..., \tilde{\mathcal{G}}_T \subseteq \mathcal{G}_T\}$. Similarly to $\hat{\mathcal{G}}^*$, the nodes and edges comprising $\tilde{\mathcal{G}}$ fluctuate over time, potentially changing at each timestep. Specifically, each $\tilde{\mathcal{G}}_t$ is characterized as $\tilde{\mathcal{G}}_t = (\tilde{\mathcal{V}}_t, \tilde{\mathcal{X}}_t, \tilde{\mathcal{E}}_t, A)$, where t belongs to the set $\mathcal{T}$ where:

- $\tilde{\mathcal{V}}_t$ is the set representing the nodes within $\tilde{\mathcal{G}}$ at timestep t and it is delineated as a distinct subset $\tilde{\mathcal{V}}_t \subseteq \mathcal{V}$ for each timestep.

- $\tilde{\mathcal{X}}_t$ is a set that defines the node attributes within $\tilde{\mathcal{G}}$ at timestep t. Specifically, it comprises the node attributes in $\mathcal{X}_t$, considering solely the traffic nodes in $\tilde{\mathcal{V}}_t$, expressed as $\tilde{\mathcal{X}}_t = \{X_t^{v_i} \mid v_i \in \tilde{\mathcal{V}}_t\}$.

- $\tilde{\mathcal{E}}_t$ is the set of edges linking nodes of $\tilde{\mathcal{G}}$ at timestep t, altering at each timestep t. It is defined as a subset of $\mathcal{E}$ constrained to nodes in $\tilde{\mathcal{V}}_t$, denoted as $\tilde{\mathcal{E}}_t = \{(v_i, v_j) \mid (v_i, v_j) \in \mathcal{E} \wedge v_i, v_j \in \tilde{\mathcal{V}}_t\ \}$.

The aim of the explainer ψ is to find $\tilde{\mathcal{G}}$ such that it maximizes the mutual information MI:

$$MI(\hat{\mathcal{Y}}^*, \tilde{\mathcal{G}}) = H(\hat{\mathcal{Y}}^*) - H(\hat{\mathcal{Y}}^* \mid \tilde{\mathcal{G}}) \tag{3.3}$$

Maximizing this term is the equivalent of finding $\tilde{\mathcal{G}}$ such that it minimizes the error $\mathcal{L}$ between the original forecasted event speeds $\hat{\mathcal{Y}}^*$ and the ones re-predicted by Φ, denoted as $\tilde{\mathcal{Y}}^*$, using just the information present in the important subgraph $\tilde{\mathcal{G}}$:

$$\min_{\tilde{\mathcal{G}}} \mathcal{L}(\hat{\mathcal{Y}}^*, \tilde{\mathcal{Y}}^* \mid \tilde{\mathcal{Y}} = \Phi(\tilde{\mathcal{G}})) \tag{3.4}$$

Where $\tilde{\mathcal{Y}} = \{\tilde{\mathcal{Y}}_{T+1} = \{\tilde{s}_{T+1}^{v_i} \mid v_i \in \mathcal{V}\}, ..., \tilde{\mathcal{Y}}_{T'} = \{\tilde{s}_{T'}^{v_i} \mid v_i \in \mathcal{V}\}\}$ are the re-predicted future node speeds by Φ given the limited information of the important subgraph $\tilde{\mathcal{G}}$, while $\tilde{\mathcal{Y}}^* = \{\tilde{\mathcal{Y}}_{T+1}^* = \{\tilde{s}_{T+1}^{v_i} \mid v_i \in \hat{\mathcal{V}}_{T+1}^*\}, ..., \tilde{\mathcal{Y}}_{T'}^* =$

$\{\tilde{s}^{v_i}_{T'} \mid v_i \in \hat{\mathcal{V}}^*_{T'}\}\}$ are the re-predicted speed attributes of the event described by $\hat{\mathcal{G}}^*$ according to the information in $\tilde{\mathcal{Y}}$. Additionally, the size of $\tilde{\mathcal{G}}$ should be concise, as the total number of nodes considering each timestep should respect a sparsity threshold $\tilde{N}$ such that:

$$\sum_{t \in \mathcal{T}} |\tilde{\mathcal{V}}_t| \leq \tilde{N} \tag{3.5}$$

with $\tilde{\mathcal{V}}_t$ being the set of nodes at timestep t.

It is worth mentioning that the important subgraph obtained in the input to explain the selected event is not straightforward to interpret. It comprises a set of nodes, yet the descriptive context remains inaccessible. Therefore, by clustering the input subgraph into sets of subgraphs, a more comprehensible interpretation of the important subgraph can be facilitated. Each cluster, representing a subgraph of the important graph, will clarify whether the cluster represents congestion or free flow. Without clustering, the explanation would be: the event in the output is the consequence of this set of nodes in the input. Through clustering, the explanation transforms into: the event in the output is due to this congestion, this free flow, and this other congestion in the input.

Formally, the important subgraph $\tilde{\mathcal{G}}$ can be dissected into the sequence of C distinct events it encapsulates. Each event is denoted as $\tilde{\mathcal{G}}^c \subseteq \tilde{\mathcal{G}}$, with $c \in [1, ..., C]$, thus forming the set $\tilde{\mathcal{G}} = \{\tilde{\mathcal{G}}^1, ..., \tilde{\mathcal{G}}^C\}$. The traffic nodes of these events should be both spatially and temporally interconnected while presenting similar values of speed and they aim at showing how prior traffic conditions within historical data contribute to shape the predicted outcome $\hat{\mathcal{G}}^*$.

The final phase consists in the *verbal explanation* of the factors that influenced the event, represented by the subgraphs $\tilde{\mathcal{G}}^c$, that lead to the prediction of the event $\hat{\mathcal{G}}^*$ by the STGNN Φ. Firstly, important content is extracted for each event $\tilde{\mathcal{G}}^c$ and for the predicted event $\hat{\mathcal{G}}^*$. Namely, for each event it selected is the kind of occurrence it describes (e.g.: congestion or free flow), the location where the event occurred in the physical space according to the coordinate of its traffic nodes, the average speed of the nodes registered and the span of time in which the event occurred. Each piece of content k is extracted by a function $k = f(\mathcal{G}^c)$, with $\mathcal{G}^c$ being either one of the subgraphs $\tilde{\mathcal{G}}^c$ or the event to explain $\tilde{\mathcal{G}}^*$. To conclude, the obtained content is translated into text through a template-based approach.

The obtained verbal narratives are formed as follows. Firstly, the predicted event $\hat{\mathcal{G}}^*$ is introduced, by indicating its measured average speed, whether it is congestion or a free flow, where it happened and at what time. Next, it is expressed that the reasons behind the prediction of $\hat{\mathcal{G}}^*$ lay on previously observed events in the traffic network. These events are none other than the ones obtained from the important subgraph $\tilde{\mathcal{G}}^c$. Finally, each event $\tilde{\mathcal{G}}^c$ described in detail similarly to $\hat{\mathcal{G}}^*$. Figure 3.1 sums up the pipeline adopted.

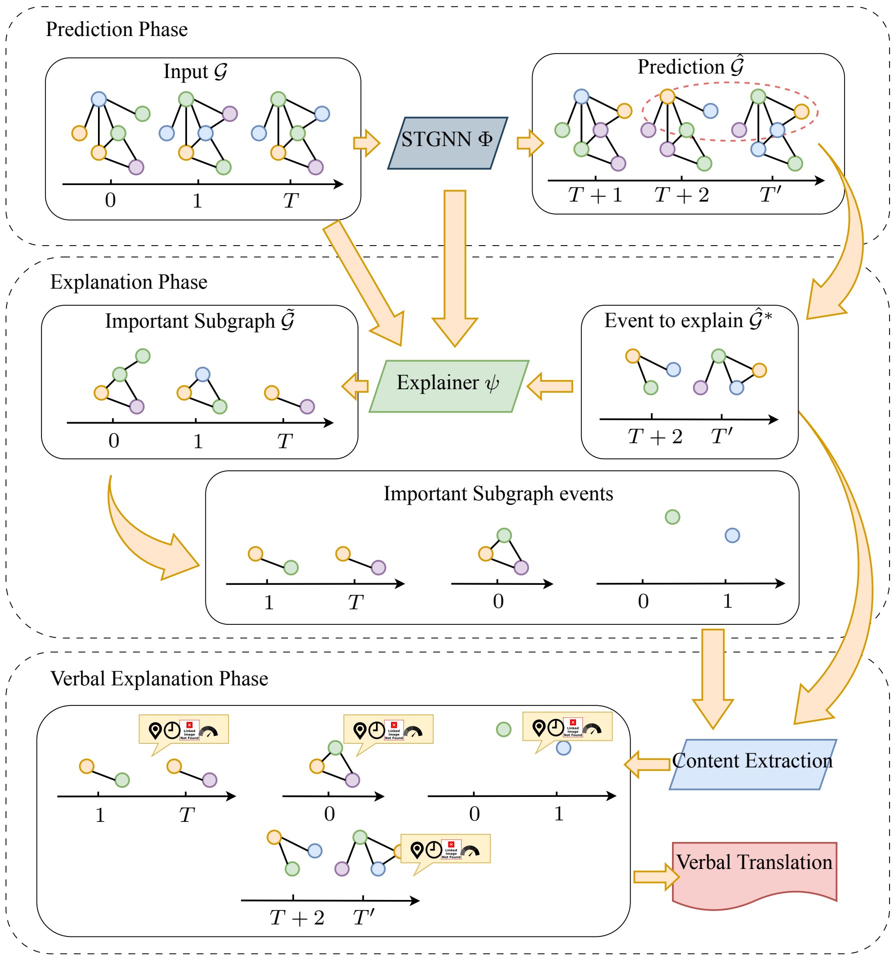


Figure 3.1: Experiment pipeline.

3.2 STGNN

The *Spatio-Temporal Graph Neural Network (STGNN)*, referred to as Φ in this study, aligns with the model introduced in the paper titled *Traffic Flow Prediction via Spatial Temporal Graph Neural Network* [42]. As previously illustrated in Equation 3.1, Φ produces future speed forecasts $\hat{\mathcal{Y}}$ in a traffic network given historical information $\mathcal{G}$. The decision to utilize this model stems from its primary focus on forecasting traffic speed. Moreover, this architecture incorporates a dynamic positional attention mechanism, enabling efficient information gathered from nearby roads. Simultaneously, it integrates a sequential component designed to capture traffic flow dynamics, effectively leveraging both local and global temporal relationships.

The framework is depicted in Figure 3.2 and it comprises three key components: the *Spatial Graph Neural Network (S-GNN)* layers, the *Gated Recurrent Unit (GRU)* layers and the *transformer* layer. The S-GNN layers are responsible for capturing spatial relationships among roads within the traffic network. The GRU layers are designed to sequentially capture local temporal dependencies, while the transformer layer is dedicated to directly capturing long-range or global temporal dependencies within the sequence.

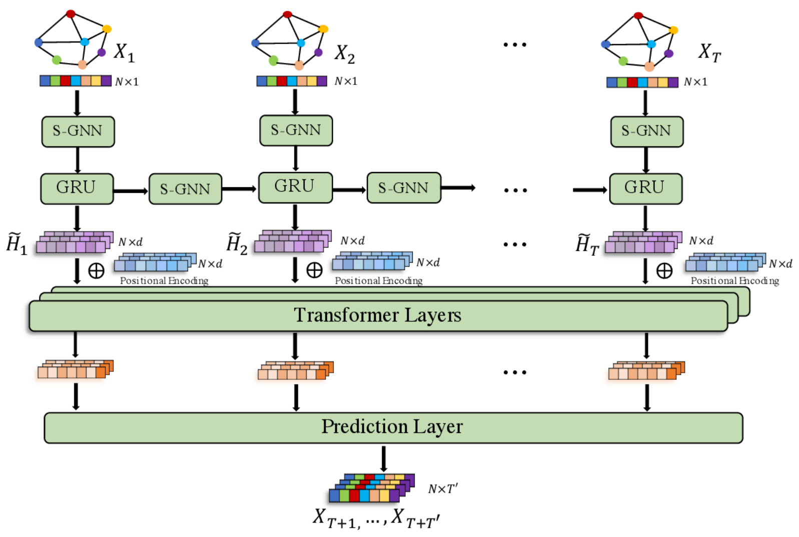


Figure 3.2: Illustration of the applied STGNN architecture.

In detail, given an input static spatio-temporal graph $\mathcal{G} = \{\mathcal{G}_t\}_{t \in \mathcal{T}}$, the STGNN firstly extract the hidden patterns between node atributes through a linear layer, then encodes through the S-GNN modules the spatial relationships of the input static network at each timestep $\mathcal{G}_t$ using the hidden node patterns at time t, $\mathcal{X}_t$, and the adjacency matrix A and obtaining their hidden representation $\tilde{\mathcal{X}_t}$. Next, the local temporal dependency among the hidden representations of each successive input graph $(\tilde{\mathcal{X}}_{t-1}, \tilde{\mathcal{X}_t})$ is captured by a series of GRU layers, returning hidden representation $\tilde{H}_t$. Then, the transformer layer is applied to each node individually in order to encode its global temporal dependencies. Finally, the set of outputs of the transformer layer is taken as input of a multi-layer feed-forward network utilized to forecast the traffic speed for forthcoming periods $\hat{\mathcal{Y}}$ using the found spatio-temporal patterns.

It's important to note that the S-GNN layers, employed to model spatial relationships between nodes at each time step, are applied to both the input and hidden representations obtained by the GRU unit. The GRU and transformer layers function to capture temporal dependencies for each node individually, offering distinct perspectives: local and global temporal connections respectively. Following, each module comprising the STGNN is described in detail.

3.2.1 S-GNN Layers

The S-GNN layers aim at grasping spatial connections within the input traffic network, through a *Graph Convolutional Network (GCN)* model. This model is utilized to convert and distribute spatial information across the network at each timestep by the adoption of the GCN operation, described in Equation 2.3. Given the traffic network node attributes $\mathcal{X}_{in}$, and the adjacency matrix A, the operation is described as:

$$\mathcal{X}_{out} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \mathcal{X}_{in} W) \tag{3.6}$$

where σ is a non-linear activation function, $\tilde{A} = A + I$ is the refined adjacency matrix with I the identity matrix, $\tilde{D}$ is the refined degree matrix and W are the learnable parameters.

The original GCN operation relies solely on network information constructed from the proximity of sensors along roads. Nevertheless, the relationship between roads can be considerably deeper. Various factors, including the number of lanes connecting roads, road conditions, vehicle and population density, as well as unforeseen events, significantly impact traffic flow. Consequently, when executing the aggregation by GCN, the information from neighboring nodes should not be uniformly combined for a specific central node. S-GNN layers propose, thus, a modified GCN operation to capture these factors.

Firstly, a latent positional representation is derived from every node in order to encompass these influences. More precisely, for each node v_i a latent positional representation p_{v_i} of its attributes X_t^{vi} is obtained by a sequential feed-forward simple neural architecture. Then, the pair-wise relations R between each couple of road nodes v_i and v_j are modeled as:

$$R_{v_i v_j} = \frac{\exp(p_{v_i}^T \cdot p_{v_j})}{\sum_{v_k \in \mathcal{V}} \exp(p_{v_i}^T \cdot p_{v_k})} \tag{3.7}$$

The relation matrix R is sparsified as R_M through a masking process adept at reducing the computational complexity:

$$R_M = \begin{cases} R_{v_i v_j}, & \text{if } \hat{A}_{v_i v_j} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

Finally, the GCN operation in Equation 3.6, is modified to take into account the newly computed sparsified relation matrix R_M:

$$\mathcal{X}_{out} = \sigma(\tilde{D}_R^{-\frac{1}{2}} \tilde{R} \tilde{D}_R^{-\frac{1}{2}} \mathcal{X}_{in} W) \tag{3.9}$$

Where $\tilde{R} = R_M + I$, $\tilde{D}_R$ is the degree matrix of $\tilde{R}$ and σ is the $ReLU$ non-linear transformation. For simplicity, the operation is sum-up through the following formula:

$$\mathcal{X}_{out} = f(A, \mathcal{X}_{in}) \tag{3.10}$$

3.2.2 GRU Layers

To model short-term temporal relationships, the GRU is employed to handle sequential data. A hidden representation $\tilde{H}_t$ is obtained for each time step t, regulating information flow to successive steps and serving as the current step's output. To incorporate spatial relationships into the sequence processing, the modified GCN operation described in Equation 3.10 is applied to both the input and hidden representations of the GRU. Specifically, at time step t, given the input node features $\mathcal{X}_t$ and previous step's hidden representations $\tilde{H}_{t-1}$, the modified GCN operation is applied to both as follows:

$$\begin{aligned} \tilde{\mathcal{X}}_t &= f(A, \mathcal{X}_t) \\ H_t &= f(A, \tilde{H}_{t-1}) \end{aligned} \tag{3.11}$$

Afterwards, the operation of the GRU at timestep t can be expressed as follows:

$$\begin{aligned} u_t &= \sigma(W_u \tilde{\mathcal{X}}_t + U_u H_{t-1}) \\ r_t &= \sigma(W_r \tilde{\mathcal{X}}_t + U_r H_{t-1}) \\ c_t &= tanh(W_c \tilde{\mathcal{X}}_t + U_c(r_t \odot U_c H_{t1})) \\ \tilde{H}_t &= (1 - u_t)c_t + u_t H_{t-1} \end{aligned} \tag{3.12}$$

where u_t is the result of the update gate, r_t the outcome of the reset gate and c_t the output of the state gate. $\odot$ is the element-wise multiplication, $W_u, W_r, W_c, U_u, U_r, U_c$ are the learnable parameters and σ is the sigmoid function. Moreover, H_t is the output of the current time step after the application of the GCN operation, which also serves as the input to the next time step. The initial hidden state H_0 is set to a matrix of 0 values.

3.2.3 Transformer Layer

GRU is useful for capturing sequential short-term time-based data, but in traffic forecasting, time patterns might not just follow a sequence. To accurately predict traffic speed, it's crucial to analyse overall time patterns. To achieve this, a *transformer* layer is placed after the GRU, aiming to directly capture long-term time dependencies. The transformer layer is applied to each node independently. For a specific node, v_i, the output sequence $\{\tilde{H}_0^{v_i}, ..., \tilde{H}_T^{v_i})$ obtained by GRU is the input for the transformer. This layer, depicted in Figure 3.3, includes a multi-head attention layer, a shared feed-forward neural network layer, and batch normalization layers positioned between them.

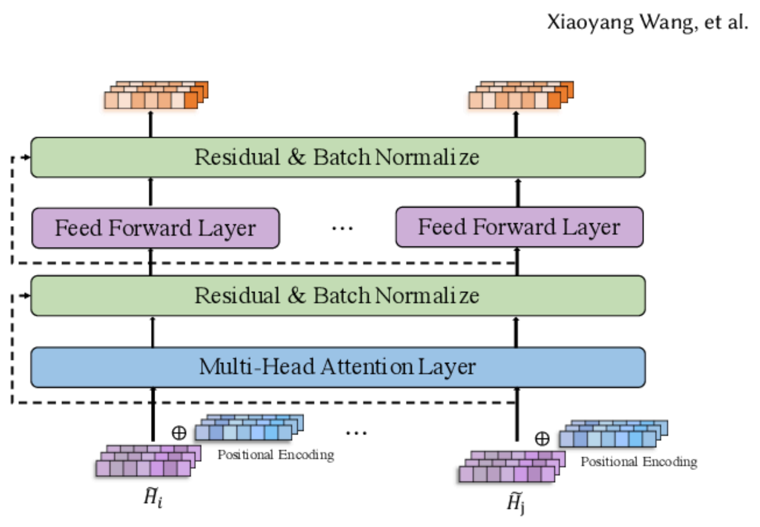


Figure 3.3: The transformer layer of the STGNN.

Initially, single-head attention is introduced, followed by an explanation of its extension to multi-head attention. The attention function involves computing for each node v_i queries Q^{v_i} and keys K^{v_i} with a dimension of d_k and values V^{v_i} with a dimension of d_v across all the temporal sequence positions. By computing the dot product between queries and keys, dividing them by $\sqrt{d_K}$, applying a softmax function and multiplying by their values, attention scores for each position are obtained for node v_i as:

$$\text{attention}^{v_i}(Q^{v_i}, K^{v_i}, V^{v_i}) = \text{softmax}\left(\frac{Q^{v_i}(K^{v_i})^T}{\sqrt{d_K}}\right) V^{v_i} \tag{3.13}$$

Q^{v_i}, K^{v_i} and V^{v_i} are derived from the output $\{\tilde{H}_0^{v_i}, ..., \tilde{H}_T^{v_i})$ of the GRU layer, organized into a matrix $\tilde{H}^{v_i} \in \mathbb{R}^{T\times d}$, where v_i indicates the corresponding node, T the time window size and d their hidden representation size. This matrix $\tilde{H}^{v_i}$ is obtained by stacking the output sequence from the GRU layer row-wise in sequential order and then linearly projecting it into:

$$Q^{v_i} = \tilde{H}^{v_i} W_Q,\ K^{v_i} = \tilde{H}^{v_i} W_K,\ V^{v_i} = \tilde{H}^{v_i} W_V \tag{3.14}$$

where $W_Q, W_K \in \mathbb{R}^{d\times d_k}$ and $W_V \in \mathbb{R}^{d\times d_v}$ are learnable parameters shared for all the nodes.

Multi-head attention is preferred over a single attention function because it combines information from various representation subspaces, enhancing the model's learning abilities. This method employs S sets of projection matrices to project $\tilde{H}^{v_i}$ into different sets of queries, keys, and values $Q_s^{v_i}, K_s^{v_i}$, and $V_s^{v_i}$, for each head s. The output of multi-head attention for node v_i is a concatenation of individual attention function outputs, resulting in an augmented representation:

$$\text{multihead}^{v_i}(\tilde{H}^{v_i}) = \text{concatenate}(\text{head}_1^{v_i}, ..., \text{head}_S^{v_i}) W_S \tag{3.15}$$

with W_S being a learnable parameter matrix and each $\text{head}_s^{v_i}$ is obtained by:

$$\text{head}_s^{v_i} = \text{attention}_s^{v_i}(Q_s^{v_i}, K_s^{v_i}, V_s^{v_i}) = \text{softmax}\left(\frac{Q_s^{v_i}(K_s^{v_i})^T}{\sqrt{d_K}}\right) V_s^{v_i} \tag{3.16}$$

where queries, keys and values for each attention head s are given by:

$$Q_s^{v_i} = \tilde{H}^{v_i} W_Q^s,\ K_s^{v_i} = \tilde{H}^{v_i} W_K^s,\ V_s^{v_i} = \tilde{H}^{v_i} W_V^s \tag{3.17}$$

with W_Q^s, W_K^s and W_V^s being learnable parameters shared between all nodes at head s.

To enable the transformer layer to consider the relative position of $\tilde{H}_t^{v_i}$

within the complete sequence, position encoding e_t is employed for each position. New representations $\tilde{H}_t^{\prime v_i}$ are generated by combining $\tilde{H}_t^{v_i}$ with the respective position encoding e_t:

$$\tilde{H}_t^{\prime v_i} = \tilde{H}_t^{v_i} + e_t \tag{3.18}$$

with e_t computed by:

$$e_t = \begin{cases} sin(\frac{t}{10{,}000^{2i/d}}), & \text{if } t = 0, 2, 4, ... \\ cos(\frac{t}{10{,}000^{2i/d}}), & \text{otherwise} \end{cases} \tag{3.19}$$

with i being the i^{th} dimension of the embedding.

In conclusion, instead of the sequence $\tilde{H}^{v_i} = (\tilde{H}_0^{v_i}, ..., \tilde{H}_T^{v_i})$ obtained directly from the GRU layer, the one with positional encoding $\tilde{H}^{\prime v_i} = (\tilde{H}_0^{\prime v_i}, ...,$ $\tilde{H}_T^{\prime v_i})$ is passed to the transformer layer. The results are stacked node-wise and passed to the prediction layer to obtain the future node speed information in the traffic network $\hat{\mathcal{Y}}$.

3.3 Explainer

The obtained speed predictions $\hat{\mathcal{Y}}$, describe the traffic network $\hat{\mathcal{G}}$ at future timesteps $\mathcal{T}' = [T+1, ..., T']$. Within the predicted network, there exist events of congestions or free flow, $\hat{\mathcal{G}}^* \subseteq \hat{\mathcal{G}}$, which might draw users' attention. The process to extract them is described in section 3.3.1. An explainer ψ is employed to uncover the primary factors that drive predictions within $\hat{\mathcal{G}}^*$, generated by the predictive model Φ. The explainer ψ process consists in identifying an important subgraph, denoted as $\tilde{\mathcal{G}} \subseteq \mathcal{G}$, within the input network $\mathcal{G}$, which serves as the explanation behind the forecasts $\hat{\mathcal{Y}}^*$ by the STGNN Φ as seen in equation 3.2. Moreover, the error between the original predicted events $\hat{\mathcal{Y}}^*$ and the re-predicted events $\tilde{\mathcal{Y}}^*$ by Φ using as input the limited information in $\tilde{\mathcal{G}}$ should be minimum, as expressed in Equation 3.4. In addition, it's essential for the size of $\tilde{\mathcal{G}}$ to remain compact. This means that the overall count of nodes considered across each timestep should adhere to a predefined sparsity threshold $\tilde{N}$ as per Equation 3.5.

The explainer ψ is designed as a post-hoc model-agnostic architecture, which explains the causes of the predictions of the model Φ according to its input data, while being unaware of its internal structure. ψ roots its core idea in the *TGNN-Explainer* [35] described in section 2.2.6, but deeply modifies its design to account for the task of the problem and its domain while replacing its opaque components with a transparent model. In particular:

- It updates the reward mechanism used in the *Monte Carlo Tree Search (MCTS)*, initially tailored for node classification, to suit the current problem context, i.e., forecasting future speed in a spatio-temporal graph (regression problem).
- It substitutes the *navigator* model, a black-box trained neural architecture aimed at computing broad correlations between input and output data to be explained, with a global heuristic. This heuristic is designed for high comprehensibility and transparency and it integrates domain-specific elements to refine the solution search space, aligning with the pattern-extraction approach pursued by the STGNN Φ.

The explainer ψ is a MCTS algorithm that performs a localized search on the input graph $\mathcal{G}$ of the instance to be explained. In particular, it returns the important subgraph $\tilde{\mathcal{G}}$ which serves as the explanation of the predicted *selected event* $\hat{\mathcal{G}}^*$ outputed by Φ. The search space is cut by a transparent *global heuristic* (section 3.3.3) that exploits domain-specific characteristics and limits the search on a subgraph of the input graph $\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$ composed of the top scoring N^* nodes according to the global heuristic. The utilization of both global and local aspects in the explanations, presented visually through the defining structure of the important subgraph $\tilde{\mathcal{G}}$ of the input traffic network $\mathcal{G}$, places ψ within the realm of model inspection techniques.

3.3.1 Output Events Selection

The subgraph $\hat{\mathcal{G}}^*$ extracted from the predicted output graph $\hat{\mathcal{G}}$ should consist of a series of spatially and temporally connected nodes that define an important event as the causes of the prediction. The speed similarity of the nodes of the subgraph is crucial for identifying notable traffic events like congestion or free-flow, as it signifies either widespread slowdown (congestion) or consistent smooth traffic flow (free-flow) across the interconnected nodes within that particular timeframe. To perform the extraction of these events from $\hat{\mathcal{G}}$, while guaranteeing the spatio-temporal and speed similarity requirement is met, the methodology explained in *Revealing the day-to-day regularity of urban congestion patterns with 3D speed maps* [43] is adapted for the problem. The paper concentrates on investigating macroscopic mobility patterns by analyzing the congestion dynamics within the transportation network and its methodology applies to the problem of the thesis. In detail, a *distance matrix* $M \in \mathbb{R}^{(N \cdot T') \times (N \cdot T')}$ between each output traffic node at each timestep is defined, with N being the number of nodes and T' the number of output timesteps. Following that each node of $\hat{\mathcal{G}}$ is clustered by a clustering algorithm while considering the distance matrix as a measure of dissimilarity.

The distance matrix M is constructed as the composition of three different matrices, namely a *spatial distance matrix* $M_d \in \mathbb{R}^{(N \cdot T') \times (N \cdot T')}$, a *temporal distance matrix* $M_t \in \mathbb{R}^{(N \cdot T') \times (N \cdot T')}$ and a *speed distance matrix* $M_s \in \mathbb{R}^{(N \cdot T') \times (N \cdot T')}$. In each of these matrices, the first N row and column indices define the measured distances of the N nodes at the first time step $t = 0$, the second N row and column indices are the distances of the N nodes at the second time step $t = 1$ and so on. In particular:

- The **spatial distance matrix** M_d is an $(N \cdot T') \times (N \cdot T')$ matrix derived from the adjacency matrix of the traffic network $A \in \mathbb{R}^{N \times N}$ and describing the spatial distance of each nodes regardless of their timestep. For each single node v_i and each pair of timesteps t_1 and t_2, $M_d^{(v_i \cdot t_1)(v_i \cdot t_2)}$

is 0 because the same nodes at different timesteps have spatial distance 0. For each pair of nodes v_i and v_j where $v_i \neq v_j$ and each timestep t_1 and t_2, $M_d^{(v_i \cdot t_1)(v_j \cdot t_2)} = 1 - A_{v_i v_j}$, as it defines the spatial distance among the two nodes regardless of the timestep. Note that the spatial distance is the opposite of the spatial adjacency expressed in A.

- The **temporal distance matrix** M_t is an $(N \cdot T') \times (N \cdot T')$ matrix describing the temporal distance of the nodes at each timestep. For each pair of nodes v_i and v_j and each couple of timesteps t_1 and t_2, $M_t^{(v_i \cdot t_1)(v_j \cdot t_2)}$ is equal to the absolute difference between t_1 and t_2.

- The **speed distance matrix** M_s is an $(N \cdot T') \times (N \cdot T')$ matrix describing the speed distance of the nodes at each timestep. For each pair of nodes v_i and v_j and each pair of timesteps t_1 and t_2, $M_s^{(v_i \cdot t_1)(v_j \cdot t_2)} = |s_{t_1}^{v_i} - s_{t_2}^{v_j}|$ is equal to the absolute difference between the discretized speed $s_{t_1}^{v_i}$ of node v_i at time t_1 and the discretized speed $s_{t_2}^{v_j}$ of node v_j at time t_2.

The matrices M_d, M_t and M_s are each scaled through *min-max scaling* between 0 and 1 and summed together in order to obtain M as follows:

$$M = W \cdot M_s + M_d + M_t \tag{3.20}$$

where the speed distance is overweighted by multiplying M_s by a factor $W \geq 1$ because the speed variable is expected to play a predominant role during the clustering process. M is next normalized again between 0 and 1 through min-max scaling. The clustering algorithm should generate groups of nodes in which each node is connected by a spatial path to every other node of the group. To guarantee it, the matrix M should reflect whether pairs of nodes are spatially unreachable. Hence, for each couple of nodes v_i and v_j and for each pair of timesteps t_1 and t_2, if in the original adjacency matrix A expresses that v_i and v_j are unconnected, $A_{v_i v_j} = 0$, then an "out of bound" distance value $1,000$ is expressed in the matrix M such that $M^{(v_i \cdot t_1)(v_j \cdot t_2)} = 1,000$.

Next, the clustering technique *DBSCAN* is used on the distance matrix M. It identifies whether each node of the predicted traffic network $\hat{\mathcal{G}}$ at a given time, specified by an index in M, is part of a distinct traffic cluster $\hat{\mathcal{G}}_c \subseteq \hat{\mathcal{G}}$ or categorized as noise as seen in Figure 3.4. DBSCAN is capable of handling irregularly shaped clusters and detecting outliers without prior knowledge of the cluster count. This method is chosen because the resulting traffic network might feature numerous unspecified traffic clusters along oddly shaped road sections, where some nodes might not significantly contribute to a specific cluster and are better classified as noise. Among the obtained clusters of the predicted traffic network $\hat{\mathcal{G}}_c$, each is classified as describing congestion or a free flow based on its node speeds and is considered as a potential event to explain $\hat{\mathcal{G}}^*$.

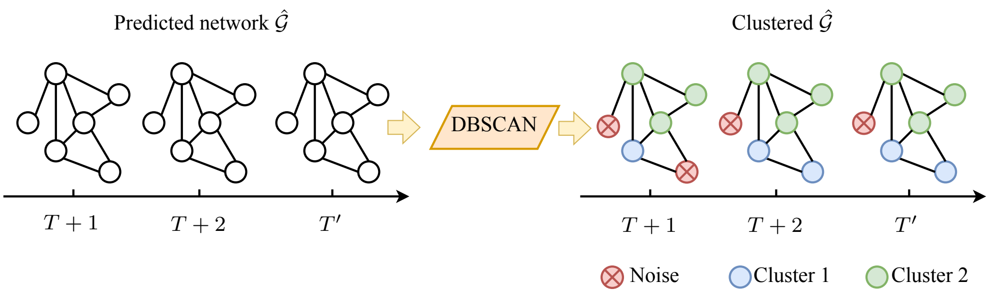

Predicted network $\hat{\mathcal{G}}$
Clustered $\hat{\mathcal{G}}$
DBSCAN
$T+1$
$T+2$
T'
$T+1$
$T+2$
T'
Noise
Cluster 1
Cluster 2

Figure 3.4: Illustration of DBSCAN clustering on the predicted traffic network.

3.3.2 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) [44] is employed as the localized search algorithm that outputs the important subgraph $\tilde{\mathcal{G}}$. MCTS is an algorithm for decision-making that involves exploring tree structures representing combinatorial spaces. These trees consist of nodes representing states or configurations of the problem, and edges representing transitions or actions between these states. The approach utilizes a smart tree search strategy, maintaining a balance between exploration and exploitation. MCTS conducts random sampling through simulations, storing action statistics to guarantee more informed choices in each following iteration. The method performs random sampling in the form of simulations and has become a state-of-the-art technique for combinatorial games. It has also found applications in practical domains such as transportation, scheduling, and security, where efficient problem-dependent modifications are often required.

The MCTS algorithm used in the thesis is listed in Algorithm 3.1 and visually illustrated in Figure 3.5.

Algorithm 3.1 Monte Carlo Tree Search

Input: The output event to explain $\hat{\mathcal{Y}}^*$; The STGNN Φ; The input traffic network $\mathcal{G}$; the sparsity threshold $\tilde{N}$.
Output: The important subgraph $\tilde{\mathcal{G}}$ serving as an explanation of the event $\hat{\mathcal{Y}}^*$.

```
 1: $\mathcal{G}_{\text{root}} \leftarrow$ subgraph of $\mathcal{G}$ by the global heuristic
 2: $\mathcal{N}_{\text{root}}[G] \leftarrow \mathcal{G}_{\text{root}}$
 3: $r^* \leftarrow -\infty$
 4: $\tilde{\mathcal{G}} \leftarrow$ NULL
 5: **for** $R$ rollouts **do**
 6:     path $= [\mathcal{N}_{\text{root}}]$
 7:     $\mathcal{N}_i \leftarrow \mathcal{N}_{\text{root}}$
 8:     **while** number of nodes in $\mathcal{N}_i > \tilde{N}$ **do**
 9:         **do** node expansion
10:         $\mathcal{N}_c \leftarrow$ **do** node selection
11:         $v_i^t \leftarrow$ traffic node removed in the node selection
12:         $\mathcal{G}_{\mathcal{N}_c} \leftarrow \mathcal{G}_{\mathcal{N}_i} \setminus \{v_i^t\}$
13:         $\mathcal{N}_c[G] \leftarrow \mathcal{G}_{\mathcal{N}_c}$
14:         $\mathcal{N}_i \leftarrow \mathcal{N}_c$
15:         append(path, $\mathcal{N}_i$)
16:     **end while**
17:     $\mathcal{N}_{\text{leaf}} \leftarrow \mathcal{N}_i$
18:     append(path, $\mathcal{N}_{\text{leaf}}$)
19:     $\mathcal{G}_{\text{leaf}} = \mathcal{N}_{\text{leaf}}[G]$
20:     $r(\mathcal{N}_{\text{leaf}}) \leftarrow 1/\mathcal{L}(\hat{\mathcal{Y}}^*, \tilde{\mathcal{Y}}^* \mid \Phi(\mathcal{G}_{\text{leaf}}))$        ▷ reward of $\mathcal{N}_{\text{leaf}}$
21:     **if** $r(\mathcal{N}_{\text{leaf}}) > r^*$ **then**
22:         $r^* \leftarrow r(\mathcal{N}_{\text{leaf}})$
23:         $\tilde{\mathcal{G}} \leftarrow \mathcal{G}_{\text{leaf}}$
24:     **end if**
25:     **do** backpropagation of $r(\mathcal{N}_{\text{leaf}})$ on path
26: **end for**
27: **return** $\tilde{\mathcal{G}}$
```

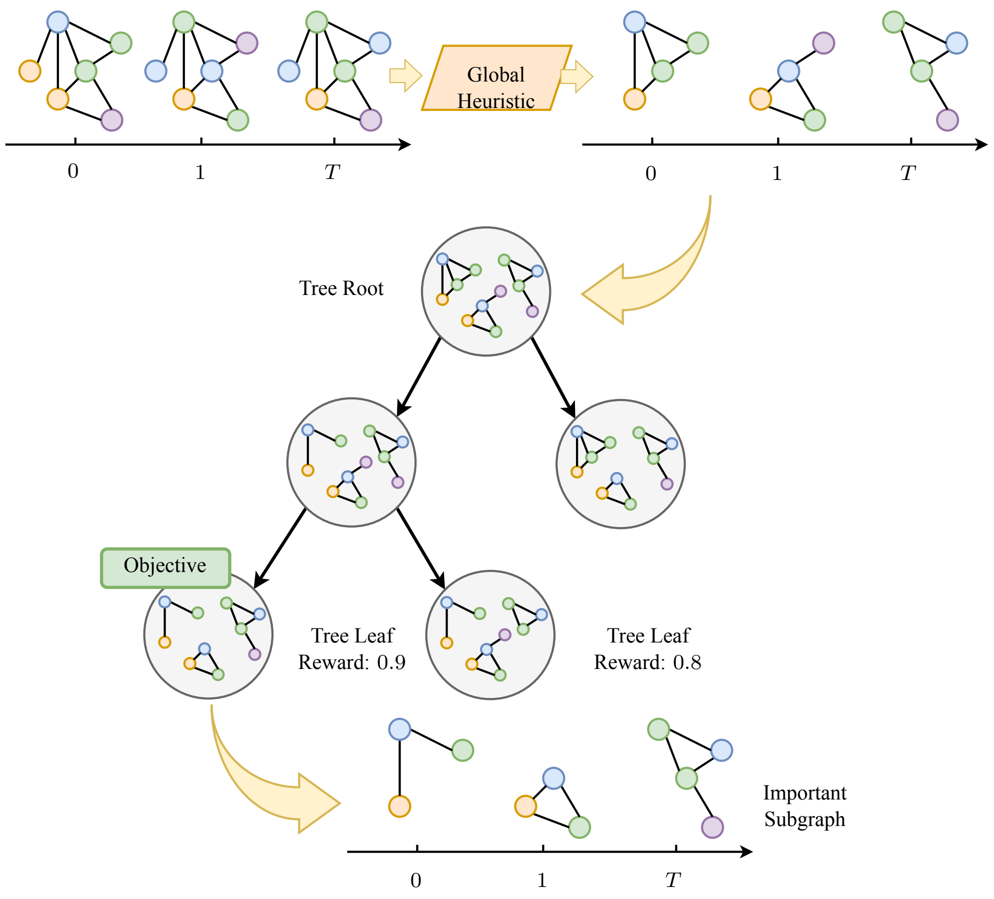


Figure 3.5: Illustration of the Monte Carlo Tree Search.

The root node $\mathcal{N}_{\text{root}}$ is initially established as a selected subgraph within the input traffic graph $\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$ according to a global heuristic defined in section 3.3.3. A subsequent number of R iterations, known as rollouts, aim to expand nodes within the search tree and each node in this structure represents a workable subset of $\mathcal{G}_{\text{root}}$ within the search area. The rollout ends as a leaf node $\mathcal{N}_{\text{leaf}}$, representing a candidate important subgraph $\mathcal{G}_{\text{leaf}} \subseteq \mathcal{G}_{\text{root}}$, is reached. Moreover, $\mathcal{G}_{\text{leaf}}$ should respect the sparsity threshold $\tilde{N}$. These iterations involve choosing a path from the root $\mathcal{N}_{\text{root}}$ to a leaf node $\mathcal{N}_{\text{leaf}}$ where for each tree node $\mathcal{N}_i$ in the path describing the traffic network $\mathcal{G}_{\mathcal{N}_i} \subset \mathcal{G}_{\text{root}}$:

1. A new tree node is expanded from $\mathcal{N}_i$ according to an action. The action consists in removing a specific traffic node in $\mathcal{G}_{\mathcal{N}_i}$.

2. A child node is selected from $\mathcal{N}_i$ among the expanded ones. In particular, it is selected the child node that maximizes exploration and exploitation.

3. If $\mathcal{N}_i$ is a leaf node, its reward is computed using the STGNN Φ.

4. If $\mathcal{N}_i$ is a leaf node, its reward is backpropagated from it up to the path nodes until $\mathcal{N}_{\text{root}}$ is reached to update information useful for exploration and exploitation in the tree search.

Ultimately, the final explanation outcome defined as the important subgraph $\tilde{\mathcal{G}}$ is determined by the subgraph $\mathcal{G}_{\text{leaf}}$ of the leaf node $\mathcal{N}_{\text{leaf}}$ that attains the highest reward and which meets the specified sparsity threshold $\tilde{N}$.

Initialization The root node $\mathcal{N}_{\text{root}}$ of the MCT is set as a subgraph $\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$ of the input traffic network as seen in Figure 3.6. This subgraph contains candidate input traffic nodes that are considered important to explain the predicted output event $\hat{\mathcal{G}}^*$ by Φ according to a global perspective. The subgraph $\mathcal{G}_{\text{root}}$ is extracted according to the global heuristic explained in section 3.3.3.

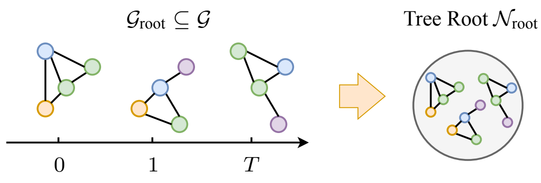

$\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$
Tree Root $\mathcal{N}_{\text{root}}$
0
1
T

Figure 3.6: Initialization of the MCTS.

Node Selection A node in the search tree denoted by $\mathcal{N}_i$ defines a subgraph $\mathcal{G}_{\mathcal{N}_i} \subset \mathcal{G}_{\text{root}}$ of the root node $\mathcal{N}_{\text{root}}$ containing the traffic network subgraph $\mathcal{G}_{\text{root}}$. The action of discarding a traffic node v_j from $\mathcal{G}_{\mathcal{N}_i}$ at timestep t is represented by $a_{v_j t}$. Performing action $a_{v_j t}$ in the tree search results in the selection of the tree node $\mathcal{N}_c$ in the rollout path. $\mathcal{N}_c$ encloses a traffic network subgraph $\mathcal{G}_{\mathcal{N}_c} \subset \mathcal{G}_{N_i}$ such that the traffic node v_j at timestep t is removed ($\mathcal{G}_{\mathcal{N}_c} = \mathcal{G}_{N_i} \setminus \mathcal{V}_t^{v_j}$). The *Upper Confidence bound applied to Trees (UCT)* formula is utilized to ensure a balance between exploitation and exploration during node selection. When assessing node $\mathcal{N}_i$, the criteria for action selection are based on:

$$a^* = \text{argmax}_{a_{v_j t} \in C(\mathcal{N}_i)} \left(\frac{c(\mathcal{N}_i, a_{v_j t})}{n(\mathcal{N}_i, a_{v_j t})} + \frac{\lambda}{r^*} \frac{\sqrt{\sum_{a_{v_l t} \in C(\mathcal{N}_i)} n(\mathcal{N}_i, a_{v_l t})}}{n(\mathcal{N}_i, a_{v_j t})} \right) \tag{3.21}$$

where $C(\mathcal{N}_i)$ represents the actions of nodes already expanded from $\mathcal{N}_i$, while $n(\mathcal{N}_i, a_{v_j t})$ signifies the number of times action $a_{v_j t}$ has been selected in previous rollouts at node $\mathcal{N}_i$ and $c(\mathcal{N}_i, a_{v_j t})$ represents the cumulative reward obtained by choosing action $a_{v_j t}$ at tree node $\mathcal{N}_i$ in previous rollouts. Moreover, λ is a constant parameter which weights the exploration in the search tree. The parameter is divided by the best reward obtained so far r^*, in order to push further exploration for low reward while limiting it if the best reward is large.

In summation, the former component focuses on exploitation, favouring nodes with higher cumulative rewards and that, hence, reached high rewarding leaf nodes in previous rollouts. Contrarily, the latter relates to exploration, favouring nodes which have been selected fewer times in the path of previous rollouts. The transition from node $\mathcal{N}_i$ to his child $\mathcal{N}_c$ in the path is performed by discarding from $\mathcal{G}_{\mathcal{N}_i}$ the node according to the best action a^* as seen in Figure 3.7.

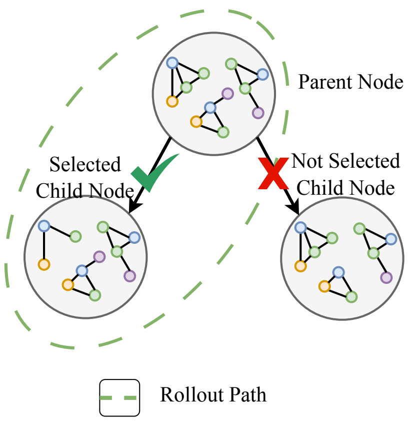


Figure 3.7: Illustration of Node Selection.

Node Expansion The strategy of node expansion is the focus of the localized aspect of the search, that seeks to expand nodes for paths leading to leaf nodes containing important subgraph $\tilde{\mathcal{G}}$. The expansion discards the global aspect of the search, in order to find a local explanation more suited for the instance to explain. Assuming the selected node $\mathcal{N}_i$ is expandable (i.e., the set of already considered actions $C(\mathcal{N}_i)$ is not complete as there are possible other actions leading to different child nodes), a random action a_{v_jt} is selected such that $a_{v_jt} \notin C(\mathcal{N}_i)$ and the set of expanded actions of $\mathcal{N}_i$ is then updated by including a_{v_jt}, $C(\mathcal{N}_i) = C(\mathcal{N}_i) \bigcup \{a_{v_jt}\}$. Figure 3.8 illustrates the process.

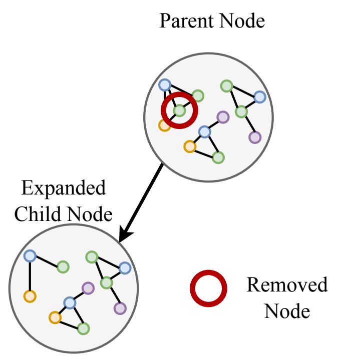


Figure 3.8: Illustration of Node Expansion.

Node selection and expansion follow an alternating pattern. Starting from the root node $\mathcal{N}_{\text{root}}$, a new child node is expanded by a random action a_{v_jt} and a_{v_jt} is then added to its set of expanded actions $C(\mathcal{N}_{\text{root}})$. Subsequently, the action with the greatest value, as per Equation 3.21, is chosen among the root's expanded actions $C(\mathcal{N}_{\text{root}})$ and the corresponding child node $\mathcal{N}_c$ is selected. This process iterates, conducting expansion and selection from the new node $\mathcal{N}_c$, until the current node qualifies as a leaf node, precisely when its subgraph contains fewer or equal than $\tilde{N}$ traffic nodes.

Reward Simulation The reward is simulated by the STGNN Φ. In detail, the reward of a leaf node $r(\mathcal{N}_{\text{leaf}})$, describing a candidate important subgraph of the input traffic network $\mathcal{G}_{\text{leaf}} \subseteq \mathcal{G}$, is computed as the reciprocal of the error $\mathcal{L}$ between the original predicted event speed features $\hat{\mathcal{Y}}^*$ and the re-predicted event speeds $\tilde{\mathcal{Y}}^*$ computed by Φ, while considering the information limited by the subgraph $\mathcal{G}_{\text{leaf}}$, following the optimization method observed in Equation 3.4:

$$r(\mathcal{N}_{\text{leaf}}) = \frac{1}{\mathcal{L}(\hat{\mathcal{Y}}^*, \tilde{\mathcal{Y}}^* = \Phi(\tilde{\mathcal{G}}))} \tag{3.22}$$

This assures that the lower the error between the original and re-predicted event features, the higher the reward of the leaf node $r(\mathcal{N}_{\text{leaf}})$. The process is illustrated in Figure 3.9.

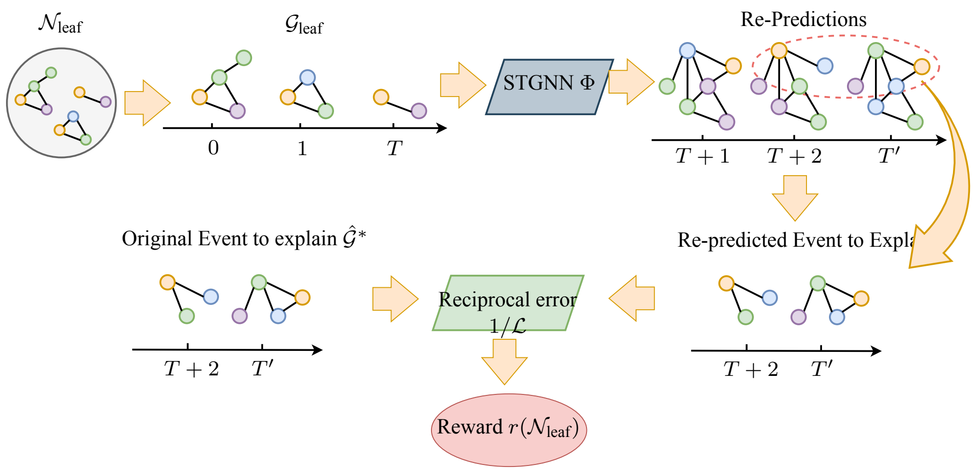

$\mathcal{N}_{\text{leaf}}$
$\mathcal{G}_{\text{leaf}}$
0
1
T
STGNN Φ
Re-Predictions
$T+1$
$T+2$
T'
Original Event to explain $\hat{\mathcal{G}}^*$
$T+2$
T'
Reciprocal error $1/\mathcal{L}$
Re-predicted Event to Expla
$T+2$
T'
Reward $r(\mathcal{N}_{\text{leaf}})$

Figure 3.9: Illustration of the reward mechanism.

In practice, the STGNN Φ cannot use the subset of input features described by $\mathcal{G}_{\text{leaf}}$ as its input layer consists in a feature matrix of predetermined size $\mathcal{X}_{\text{in}} \in \mathbb{R}^{T \times N \times F}$, where T is the number of timesteps, N is the number of nodes and F the number of features. Thus, the absence of traffic nodes in the input traffic network $\mathcal{G}$ described by the subgraph $\mathcal{G}_{\text{leaf}}$ is obtained by substituting in $\mathcal{G}$ the speed features of the missing traffic nodes by a predetermined missing value flag $\tilde{s}$ such that:

$$\forall\, v_i \in \mathcal{V} \wedge t \in \mathcal{T} \mid v_i \notin \mathcal{V}^t_{\text{leaf}},\ s^{v_i}_t = \tilde{s} \tag{3.23}$$

with $\mathcal{V}^t_{\text{leaf}}$ being the set of nodes of $\mathcal{G}_{\text{leaf}}$ at timestep t and $s^{v_i}_t \in X^{v_i}_t$ being the speed feature of node v_i at timestep t in the input graph $\mathcal{G}$. The modified graph

$\mathcal{G}$ incorporating the adjusted speeds as per the equation is utilized to obtain the re-predictions $\tilde{\mathcal{Y}}^*$ by feeding it to the STGNN Φ.

Backpropagation Backpropagation, is applied after the reward simulation of a leaf node $\mathcal{N}_{\text{leaf}}$ in order to update the statistics of the nodes in the roll-out path for balancing exploration and exploitation in the successive rollouts. During backpropagation, each node $\mathcal{N}_i$ selected by action $a_{v_j t}$ in the rollout path spanning from the root $\mathcal{N}_{\text{root}}$ to the leaf node $\mathcal{N}_{\text{leaf}}$ will update the number of selection of the action $a_{v_j t}$ from $\mathcal{N}_i$ ($n(\mathcal{N}_i, a_{v_j t})$) and the cumulative reward of selecting the action $a_{v_j t}$ from $\mathcal{N}_i$ ($c(\mathcal{N}_i, a_{v_j t})$). In detail, as in Figure 3.10, $n(\mathcal{N}_i, a_{v_j t})$ is updated by one $n(\mathcal{N}_i, a_{v_j t}) = n(\mathcal{N}_i, a_{v_j t}) + 1$ and the $c(\mathcal{N}_i, a_{v_j t})$ by the leaf node's reward $c(\mathcal{N}_i, a_{v_j t}) = c(\mathcal{N}_i, a_{v_j t}) + r(\mathcal{N}_{\text{leaf}})$.

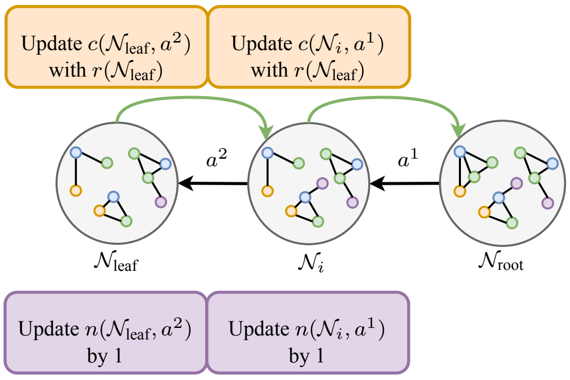

Update $c(\mathcal{N}_{\text{leaf}}, a^2)$ with $r(\mathcal{N}_{\text{leaf}})$
Update $c(\mathcal{N}_i, a^1)$ with $r(\mathcal{N}_{\text{leaf}})$
a^2
a^1
$\mathcal{N}_{\text{leaf}}$
$\mathcal{N}_i$
$\mathcal{N}_{\text{root}}$
Update $n(\mathcal{N}_{\text{leaf}}, a^2)$ by 1
Update $n(\mathcal{N}_i, a^1)$ by 1

Figure 3.10: Illustration of the backpropagation mechanism.

3.3.3 Global Heuristic

Discovering all potential subgraphs of $\mathcal{G}$ (composed of $N \cdot T$ nodes), meeting a sparsity threshold of $\tilde{N}$, involves a polynomial time complexity of $O((N \cdot T)^{\tilde{N}})$, rendering it computationally intractable. To cut down the search space a global heuristic is introduced, crafted to be easily understood and transparent, merging domain-specific components to reduce the solution search space, in line with the STGNN Φ's method of extracting patterns. Furthermore, the heuristic offers a comprehensive overview of the model predictions, as it is applied universally across instances by utilizing traffic-specific knowledge. Subsequently, the MCTS utilizes this insight to reduce the search space to a subgraph $\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$ of the input traffic network $\mathcal{G}$ and to conduct a targeted exploration for each individual instance.

The heuristic leverages two main aspects of the traffic forecasting domain, namely the *fundamental equation of traffic flow* and the *assumption of localized spatial correlation*. These two concepts are combined to compute the *global heuristic correlation scores* between each node of the input network and the nodes of the event to explain.

Fundamental equation of traffic flow The fundamental equation of traffic flow [45] is an essential concept in traffic theory and it is used to analyze and model the behaviour of traffic in various conditions. It expresses the relationship between the flow Q of traffic, the speed of vehicles V, and the density of vehicles D on a roadway as:

$$Q = D \cdot V \tag{3.24}$$

The diagram observed in figure 3.11 denotes that small changes in flow Q are propagated back through the stream of vehicles in the roadway and that the speed of the vehicles V is related to the density of the traffic D. Intuitively, traffic nodes that are close in space and time and exhibit similar speeds are likely to be part of the same flow on a roadway as they demonstrate stronger influences dictated by the fundamental equation. In contrast, if traffic nodes are close in both time and space but have different speeds, it suggests they operate independently, not adhering to the flow conservation defined by the fundamental equation. Consequently, these nodes are likely to belong to distinct traffic streams, potentially representing different lanes on a road or even entirely separate, unconnected roads. As nodes become more distant in either time or space, the importance of flow conservation decreases. Consequently, the relevance of speed similarity between two nodes diminishes, as they are less likely to belong to the same traffic stream governed by flow conservation. In practice, nodes distant in space are more likely to represent distinct unconnected road segments, while nodes distant in time are more likely to characterize unrelated traffic streams.

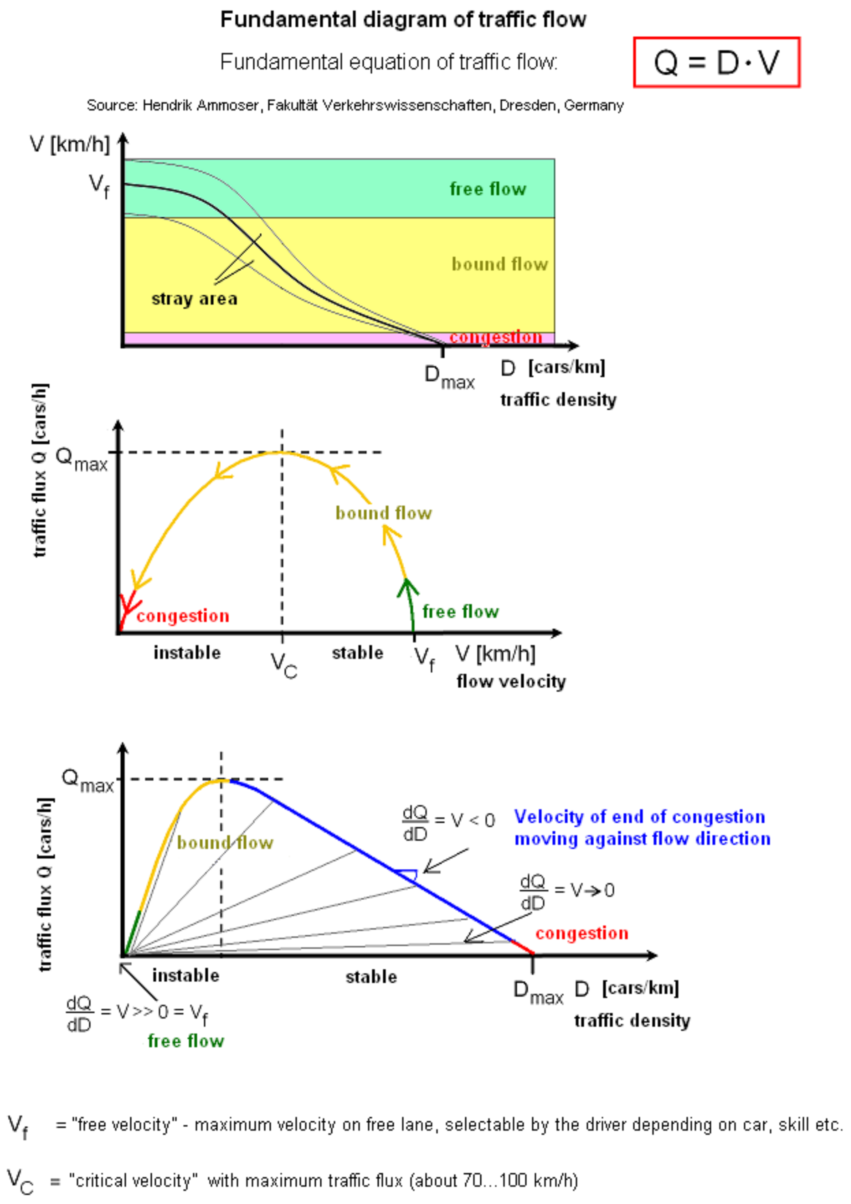

Fundamental diagram of traffic flow
Fundamental equation of traffic flow:
Q = D·V
Source: Hendrik Ammoser, Fakultät Verkehrswissenschaften, Dresden, Germany
V [km/h]
Vf
free flow
bound flow
stray area
congestion
Dmax
D [cars/km]
traffic density
traffic flux Q [cars/h]
Qmax
bound flow
congestion
free flow
instable
VC
stable
Vf
V [km/h]
flow velocity
traffic flux Q [cars/h]
Qmax
bound flow
dQ/dD = V < 0
Velocity of end of congestion moving against flow direction
dQ/dD = V→0
congestion
instable
stable
Dmax
D [cars/km]
traffic density
dQ/dD = V >> 0 = Vf
free flow
Vf = "free velocity" - maximum velocity on free lane, selectable by the driver depending on car, skill etc.
VC = "critical velocity" with maximum traffic flux (about 70...100 km/h)

Figure 3.11: Diagram of the fundamental equation of traffic flow

To capture this dynamic, a flow correlation coefficient $cf_{tt'}^{v_i v_j}$ is introduced.

This coefficient measures, in accordance with the fundamental law of traffic flow, the potential influence of a node v_i at time t in the input graph $\mathcal{G}$ on predicting the speed of a node v_j at time t' within the forecasted traffic event $\hat{\mathcal{G}}^*$:

$$cf_{tt'}^{v_i v_j} = \frac{\exp(-(\Delta t + \Delta d))}{\Delta s} \tag{3.25}$$

Where Δt is the time distance between timesteps t' and t, while Δd is the spatial distance between traffic nodes v_i and v_j and Δs is their speed absolute difference at the given timesteps. The higher $cf_{tt'}^{v_i v_j}$, the higher the flow correlation between nodes v_i at timestep t and node v_j at timestep t'.

Assumption of localized spatial correlation The assumption of localized spatial correlation [46], is based on the concept that not all the traffic nodes in the input network $\mathcal{G}$ can contribute to the predictions in the output network subgraph $\hat{\mathcal{G}}^*$ defining the traffic event to explain. Given a node v_j of $\hat{\mathcal{G}}^*$ and its last timestep t' a spreading cone from v_j at time t' can be drawn in the input network. The angle between the time and space axes of the spreading cone is defined as θ. Given a node v_i at a previous timestep t of the input network $\mathcal{G}$, v_i is part of that cone if it satisfies the semi vertex angle $tan(\theta) = c_r$:

$$\frac{\Delta d}{\Delta t} \leq c_r \tag{3.26}$$

where Δd is the spatial distance between v_i and v_j and Δt the temporal distance $t' - t$. All points outside the spreading cone are independent of v_j because their impact cannot reach location v_j in the next $t' - t$ steps. Figure 3.12 illustrates the process.

The lower the value $\Delta d / \Delta t$ or, equivalently, the higher $\Delta t / \Delta d$ between two nodes at two given timesteps, the higher their localized spatial correlation. To capture this property, a localized spatial correlation coefficient $cs_{tt'}^{v_i v_j}$ is thus computed as follows:

$$cs_{tt'}^{v_i v_j} = \frac{\Delta t}{\Delta d} \tag{3.27}$$

The higher $cs_{tt'}^{v_i v_j}$, the higher the localized spatial correlation between nodes v_i at timestep t and node v_j at timestep t'.

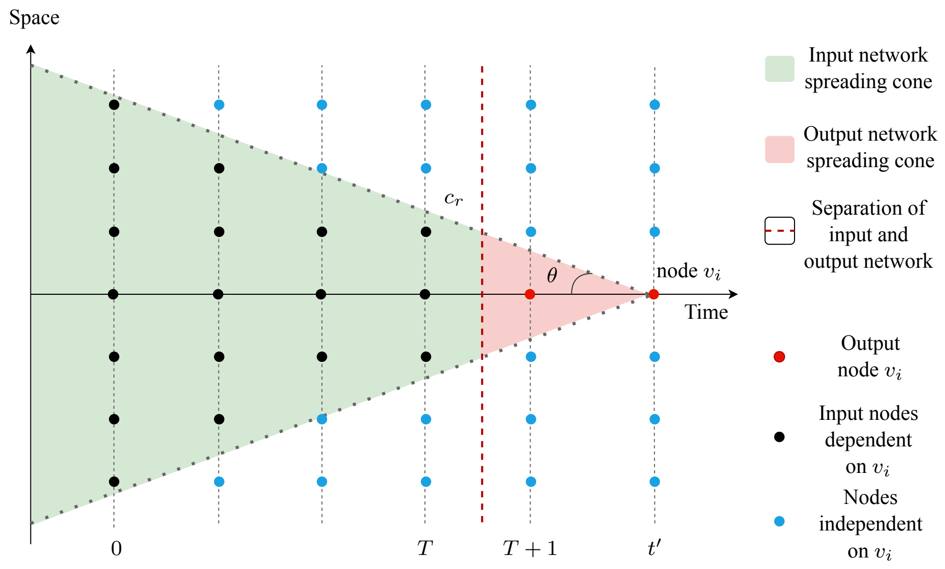

Space
c_r
θ
node v_i
Time
0
T
$T+1$
t'
Input network spreading cone
Output network spreading cone
Separation of input and output network
Output node v_i
Input nodes dependent on v_i
Nodes independent on v_i

Figure 3.12: Diagram of the spreading cone from output node v_i with last timestep t'.

Global heuristic correlation scores Given the output subgraph of the event to explain $\hat{\mathcal{G}}^*$, the correlation heuristic score among an input graph node v_i at time t and the set of nodes in $\hat{\mathcal{G}}^*$ is given by:

$$c_t^{v_i} = \sum_{v_j \in \bigcup \hat{\mathcal{V}}^*} \frac{cs_{tt'}^{v_i v_j} + cf_{tt'}^{v_i v_j}}{|\bigcup \hat{\mathcal{V}}^*|} \tag{3.28}$$

with t' being the last timestep of v_j in $\hat{\mathcal{G}}^*$ and $\bigcup \hat{\mathcal{V}}^*$ is the set of distinct indices of the nodes present in $\hat{\mathcal{G}}^*$ considering all timesteps. Note that for the experiment, Δs in $cf_{tt'}^{v_i v_j}$ was actually computed considering the speed difference among the input traffic node v_i at timestep t and the average speed of the output node v_j across all the considered output timesteps. The search space is cut by considering just the top N^* nodes in $\mathcal{G}$ reaching the highest correlation heuristic scores. The network $\mathcal{G}$ limited as the top scoring nodes is defined as $\mathcal{G}_{\text{root}} \subseteq \mathcal{G}$ and is employed in the MCTS illustrated in section 3.3.2.

3.3.4 Important Subgraph Events Extraction

The important subgraph $\tilde{\mathcal{G}}$ obtained by the explainer ψ contains the information that leads the STGNN Φ to predict the output event $\hat{\mathcal{G}}^*$. However, the specific nature of the nodes within this important subgraph $\tilde{\mathcal{G}}$ and their relationships remain ambiguous. To enhance clarity, it becomes necessary to cluster these nodes into events, such as congestions or free-flowing traffic, that are spatially and temporally connected. These events offer clearer explanations of the output event $\hat{\mathcal{G}}^*$, since they illustrate how previous traffic conditions in the historical data contribute to shaping the predicted outcome. By discerning these interconnected events, it becomes possible to unveil the sequential progression of events that ultimately lead to the forecasted traffic state, enriching the understanding of influences within the traffic network. These events consist in subgraphs $\tilde{\mathcal{G}}_c \subseteq \tilde{\mathcal{G}}$ of the important subgraph $\tilde{\mathcal{G}}$ that define clusters of spatially and temporally connected nodes defining a congestion or a free flow. In each cluster $\tilde{\mathcal{G}}_c$, it is necessary for their node speeds to be close to characterizing one of these noteworthy traffic events.

To extract these events, a similar process to the one in section 3.3.1 is adopted, although quietly modified, to account for the specificity of the data that needs to be clustered. In detail, a *distance matrix* $M \in \mathbb{R}^{(N_0+\ldots+N_T)\times(N_0+\ldots+N_T)}$ between each important subgraph node at each timestep is defined, with N_i being the number of nodes at timestep t and T the number of input timesteps. The distance matrix M is constructed as the composition of three different matrices, namely a *spatial distance matrix* $M_d \in \mathbb{R}^{(N_0+\ldots+N_T)\times(N_0+\ldots+N_T)}$, a *temporal distance matrix* $M_t \in \mathbb{R}^{(N_0+\ldots+N_T)\times(N_0+\ldots+N_T)}$ and a *speed distance matrix* $M_s \in \mathbb{R}^{(N_0+\ldots+N_T)\times(N_0+\ldots+N_T)}$. In each of these matrices, the first row and column indices in $[1, N_0]$ define the measured distances of the nodes in the important subgraph at the first time step $t = 0$, the second row and column indices in $[N_0 + 1, N_1]$ are the distances of the nodes in the important subgraph at the second time step $t = 1$ and so on. These matrices and their composition in M are computed with the same method as section 3.3.1.

Following that, a clustering algorithm is applied considering the distance matrix M as a measure of dissimilarity between traffic nodes. DBSCAN, which was used to select the events to explain in the predicted network, has been discarded in favour of *Agglomerative clustering*. This is a hierarchical clustering technique which begins by assigning each traffic node in $\tilde{\mathcal{G}}$ to its own cluster. Then, it iteratively merges the closest clusters based on the distance measures obtained by M, continuing until a specified number C of clusters is achieved and $\tilde{\mathcal{G}}$ is entirely decomposed in C clusters each defining congestions or free flows $\tilde{\mathcal{G}} = \{\tilde{\mathcal{G}}^1, ..., \tilde{\mathcal{G}}^C\}$ as seen in Figure 3.13. In this approach, the determination of the number of clusters (C) is conducted separately for each instance, considering a spectrum of potential values for C. The ideal value of C for each instance is selected to ensure that the speeds of nodes within each cluster are maximally similar, while the speeds of nodes belonging to different clusters are as dissimilar as possible. This method is chosen as it clusters each traffic node of the important subgraph, without classifying any one of them as noise and excluding it from a specific cluster. Not excluding any traffic point of $\tilde{\mathcal{G}}$ is necessary, as all of them are considered important nodes that lead to the prediction of $\hat{\mathcal{G}}^*$ according to the explainer ψ.

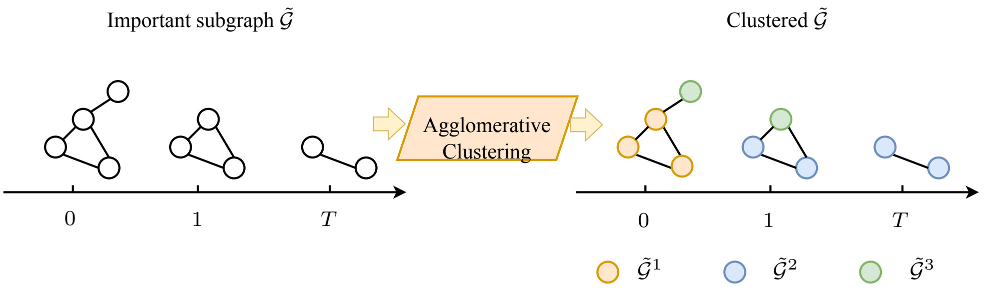


Figure 3.13: Illustration of Agglomerative clustering on the important subgraph of the input traffic network.

3.4 Verbal Explanations

Following the explanation phase driven by the explainer ψ, it is acquired, for a predicted event $\hat{\mathcal{G}}^*$, the significant subgraph $\tilde{\mathcal{G}}$ of the input traffic network. This subgraph is clustered into C distinct events $\tilde{\mathcal{G}}^c$ with $c \in [1, ..., C]$, such that $\tilde{\mathcal{G}} = \{\tilde{\mathcal{G}}^1, ..., \tilde{\mathcal{G}}^C\}$. These clusters serve as the rationale behind the predictions in $\hat{\mathcal{G}}^*$ and they function as graphical explanations for the predictions made by the model. They establish the groundwork to extract verbal narratives for the users, however, they need to be processed in two-step process: by initially *extracting significant content* from them and subsequently utilizing this content to craft a coherent *verbal translation*.

3.4.1 Content Extraction

Let's denote each cluster of the important input subgraph $\tilde{\mathcal{G}}$ and the predicted event $\hat{\mathcal{G}}^*$ as $\mathcal{G}^c$. Each event $\mathcal{G}^c = \{\mathcal{G}^c_{t_0}, ..., \mathcal{G}^c_{t_n}\}$ contains a set of traffic nodes with a certain measured average speed on each of the discretized timesteps $\mathcal{T}^c = [t_0, ..., t_n]$. If $\mathcal{G}^c$ is a cluster of the input subgraph, $\mathcal{T}^c$ is the set of input timesteps $\mathcal{T}$, while if $\mathcal{G}^c$ is the predicted event subgraph $\hat{\mathcal{G}}^*$, $\mathcal{T}^c$ is the set of output timesteps $\mathcal{T}'$. Each $\mathcal{G}^c_t$, with $t \in \mathcal{T}^c$ is defined as $\mathcal{G}^c_t = (\mathcal{V}^c_t, \mathcal{X}^c_t, \mathcal{E}^c_t, A)$, where $\mathcal{V}^c_t \subseteq \mathcal{V}$ is the set of nodes at time t, $\mathcal{X}^c_t = \{s^{v_i}_t \mid v_i \in \mathcal{V}^c_t\}$ the set of speed attributes of each node (with $s^{v_i}_t$, the speed of node v_i at time t) and $\mathcal{E}^c_t = \{(v_i, v_j) \mid (v_i, v_j) \in \mathcal{E} \land v_i, v_j \in \mathcal{V}^c_t\}$ the set of edges at timestep t.

The most important content to extract for each event $\mathcal{G}^c$ is the kind of occurrence it describes, a traffic congestion or free flow, the location where the event occurred in the physical space, the average speed of the nodes registered for the event and the span of time in which the event occurred. A series of functions aimed at extracting these pieces of information are introduced:

- speed$(\mathcal{G}^c)$ extracts the average speed $s_{\mathcal{G}^c}$ among all nodes at each timestep of the event described by $\mathcal{G}^c$ such that:

$$s_{\mathcal{G}^c} = \text{speed}(\mathcal{G}^c) = \sum_{t \in \mathcal{T}^c} \sum_{v_i \in \mathcal{V}_t^c,} \frac{s_t^{v_i}}{|\bigcup\limits_{t' \in \mathcal{T}^c} \mathcal{V}_{t'}^c|} \tag{3.29}$$

- label($\mathcal{G}^c$) defines the kind of event $k_{\mathcal{G}^c}$ described by $\mathcal{G}^c$ according to its average speed $s_{\mathcal{G}^c}$. $k_{\mathcal{G}^c}$ can have the value of *"severe congestion"*, *"congestion"* or *"free flow"* based on the speed thresholds s_{sc}, s_c as:

$$k_{\mathcal{G}^c} = \text{label}(\mathcal{G}_c) = \begin{cases} \textit{``severe congestion''}, & \text{if } s_{\mathcal{G}^c} \leq s_{sc} \\ \textit{``congestion''}, & \text{if } t_{sc} < s_{\mathcal{G}^c} \leq s_c \\ \textit{``free flow''}, & \text{otherwise} \end{cases} \tag{3.30}$$

- location($\mathcal{G}^c$) delineates the event occurrence spots. It outputs a dictionary, $l_{\mathcal{G}^c}$, encompassing streets $l_0, l_1, ...$ where the nodes of $\mathcal{G}^c$ are situated. These streets are matched with sets of kilometrage $\{km_0, km_1, ...\}$, specifying the precise location of nodes along each street such that:

$$l_{\mathcal{G}^c} = \text{location}(\mathcal{G}^c) = \{l_0 : \{km_0, ...\}, l_1 : \{km_0, ...\}\} \tag{3.31}$$

Algorithm 3.2 Location Extraction Algorithm: location($\cdot$)

Input: A traffic event part of the explanation $\mathcal{G}^c$.

Output: The location dictionary $l_{\mathcal{G}^c}$ with the streets as keys and the list of kilometres at which the nodes are present as values.

```
1:  $l_{\mathcal{G}^c} \leftarrow \emptyset$
2:  for $v_i \in \bigcup_{t \in \mathcal{T}^c} \mathcal{V}_t^c$ do
3:      $l_{v_i} \leftarrow$ street of $v_i$
4:      $km_{v_i} \leftarrow$ kilometrage of $v_i$ in $l_{v_i}$
5:      if $l_{v_i} \in l_{\mathcal{G}^c}$ then
6:          append($l_{\mathcal{G}^c}[l_{v_i}], km_{v_i}$)
7:      else
8:          $l_{\mathcal{G}^c}[l_{v_i}] \leftarrow \{km_{v_i}\}$
9:      end if
10: end for
11: return $l_{\mathcal{G}^c}$
```

As seen in Algorithm 3.2, for each node v_i in $\mathcal{G}^c$, defined by the union of the set of nodes at each timestep $\bigcup_{t \in \mathcal{T}^c} \mathcal{V}_t^c$, the street l_{v_i} where v_i is located is extracted. If l_{v_i} is already a key of the dictionary $l_{\mathcal{G}^c}$, then the kilometrage km_{v_i} of the node on street l_{v_i} is added to its value set, otherwise the key l_{v_i} is added to the dictionary $l_{\mathcal{G}^c}$ and its value is initiated with a set containing its kilometrage km_{v_i}.

- days($\mathcal{G}^c$) returns the set of days $d_{\mathcal{G}^c} = \{d_0, d_1, ...\}$ in which the event occurs, since the traffic forecasting obtained by the STGNN Φ is based on a short term period, the days on which each event spans is usually just one, although, it may happen that the event starts at around midnight and crosses over the next day. The set of days is obtained by the information of the timesteps t where there are traffic nodes present in $\mathcal{G}^c$ ($\mathcal{V}_t^c \neq \emptyset$), such that:

$$d_{\mathcal{G}^c} = \text{days}(\mathcal{G}^c) = \{\text{day of } t \mid t \in \mathcal{T}^c \wedge \mathcal{V}_t^c \neq \emptyset\} \tag{3.32}$$

- time($\mathcal{G}^c$) outputs a pair $t_{\mathcal{G}^c} = (t_{\text{start}}, t_{\text{end}})$ defining the starting time of the event t_{start} and the end time t_{end} in the format hh:mm, where hh defines the hour and mm the minute in each time of the pair:

$$t_{\mathcal{G}^c} = \text{time}(\mathcal{G}^c) = (t_{\text{start}}, t_{\text{end}}) \tag{3.33}$$

As seen in Algorithm 3.3, the starting time t_{start} of the event is obtained as the time of the first timestep t where the event $\mathcal{G}^c$ presents a non-empty set of traffic nodes, $\mathcal{V}_t^c \neq \emptyset$, while the end time t_{end} is the time of the last timestep t' where traffic nodes are present in $\mathcal{G}^c$, $\mathcal{V}_{t'}^c \neq \emptyset$.

Algorithm 3.3 Time Extraction Algorithm: time($\cdot$)

Input: A traffic event part of the explanation $\mathcal{G}^c$.

Output: The pair of timesteps $(t_{\text{start}}, t_{\text{end}})$ defining the time when the traffic event started and ended.

```
 1: $t_{\text{start}} \leftarrow$ NULL
 2: $t_{\text{end}} \leftarrow$ NULL
 3: for $t \in \mathcal{T}^c$ do
 4:     if $\mathcal{V}_t^c \neq \emptyset$ then
 5:         $t_{\text{end}} \leftarrow$ time of $t$
 6:         if $t_{\text{start}} =$ NULL then
 7:             $t_{\text{start}} \leftarrow$ time of $t$
 8:         end if
 9:     end if
10: end for
11: return $(t_{\text{start}}, t_{\text{end}})$
```

3.4.2 Verbal Translation

Following content selection, the verbal translation occurs through a template-based approach. This method involves substituting placeholders in textual templates with the chosen content to form coherent narratives. This approach is preferred over complex, highly automated, and domain-generic operations due to its adherence to the principle of *Occam's razor*. This is a principle in problem-solving and philosophy that advocates for selecting the simplest solution when faced with multiple options. It suggests that among competing hypotheses or explanations, the one that requires the fewest assumptions or entities tends to be the most likely or preferred explanation. In essence, it encourages prioritizing simplicity and minimizing complexity when making decisions or formulating explanations. Moreover, employing manual template-based approaches, despite demanding more time for their construction, is advantageous as they are less susceptible to errors.

The verbal translation consists of composing a series of paragraphs by exploiting the content extracted from the graphical explanations. The first paragraph describes the predicted event $\hat{\mathcal{G}}^*$ and briefly sums up its causes, while the second to last paragraphs illustrate in detail each cause $\tilde{\mathcal{G}}^c$ leading to the event which is each cluster of the important subgraph. The paragraphs describing the causes are sorted by the time $\tilde{\mathcal{G}}^c$ occurred, based on its temporal information $d_{\tilde{\mathcal{G}}^c}$ and $t_{\tilde{\mathcal{G}}^c}$. Each paragraph is formulated by filling a series of templates with the extracted content.

For the first paragraph three sets of templates are defined. τ_{p} which templates serve to describe the predicted event $\hat{\mathcal{G}}^*$, τ_+ which templates may be used to describe additional information of the event if needed and τ_{c} which briefly sums up the causes $\tilde{\mathcal{G}}^c$ leading to the event. In each set, its elements are templates that contain the same latent information and are thus equivalent in terms of semantics. In order to form the first paragraph a random template from each set, $T_p \in \tau_{\text{p}}$, $T_+ \in \tau_+$ and $T_c \in \tau_{\text{c}}$, is fetched and filled with the extracted content. The filled templates are concatenated ($*$) in the order:

$$\text{first paragraph} = T_p * T_+ * T_c \tag{3.34}$$

Each template $T_p \in \tau_{\text{p}}$ contains the following placeholders that need to be filled with the selected content:

- The placeholder <*speed*> needs to be filled with the average speed extracted content $s_{\hat{\mathcal{G}}^*}$ of the event to explain $\hat{\mathcal{G}}^*$.
- The placeholder <*label*> needs to be filled with the kind of event content $k_{\hat{\mathcal{G}}^*}$ defining the event to explain $\hat{\mathcal{G}}^*$.
- The placeholder <*location main*> needs to be filled with the main street information of the extracted location information $l_{\hat{\mathcal{G}}^*}$ of location$(\mathcal{G}^c)$. The main street is defined as the $l_{\text{main}} \in l_{\hat{\mathcal{G}}^*}$, comprising the larger set of nodes considering each timestep in $\hat{\mathcal{G}}^*$. In practice, l_{main} is selected as the street in the dictionary that presents the largest set of kilometres.
- The placeholder <*day*> needs to be filled with the days in which the event $\hat{\mathcal{G}}^*$ occurred defined by $d_{\hat{\mathcal{G}}^*}$.
- The placeholder <*time*> is filled with the timespan in which the event $\hat{\mathcal{G}}^*$ lasted. This information is found in $t_{\hat{\mathcal{G}}^*}$.

The template $T_+ \in \tau_+$ is filled just in the case the dictionary of location $l_{\hat{\mathcal{G}}*}$ is composed of more than one key, meaning that the event occurred in more than one single street. In this case the template T_+ presents a placeholder <*label*> that is filled as per the template T_p and a placeholder <*location secondary*> that needs to be filled with the secondary street information of the extracted location information $l_{\hat{\mathcal{G}}*}$ of location$(\mathcal{G}^c)$. The secondary streets are all the streets in $l_{\hat{\mathcal{G}}*}$ minus the main one l_main. Hence, the secondary street information is defined by the dictionary $l_{\hat{\mathcal{G}}*}$ with the key l_main excluded, $l_{\hat{\mathcal{G}}*} \setminus \{l_\text{main}\}$. In the case where the dictionary $l_{\hat{\mathcal{G}}*}$ contains just one street all the necessary location information has already been mentioned by replacing the placeholder <*location main*> in the template T_p, thus, filling T_+ is unnecessary. In this case T_+ is substituted by the empty string $\emptyset$ in the concatenation, meaning that Equation 3.34 is substituted by:

$$\text{first paragraph} = T_p * \emptyset * T_c = T_p * T_c \tag{3.35}$$

$\mathbf{T_p}$	$\mathbf{T_+}$	$\mathbf{T_c}$
"A <label> was predicted <location main> <day>, with an average speed of <speed> km/h <time>."	*"The <label> also affected <location minor>"*	*"This was caused by <causes>."*
Extracted content		
$s_{\hat{\mathcal{G}}*} = 42.47$; $k_{\hat{\mathcal{G}}*} =$ *severe congestion*; $l_{\hat{\mathcal{G}}*} = \{$*Golden State Freeway* : $\{10, 11\}\}$; $d_{\hat{\mathcal{G}}*} = \{$*Thursday, 05/06/2012*$\}$; $t_{\hat{\mathcal{G}}*} = (09{:}45, 10{:}35)$; $k_{\tilde{\mathcal{G}}1} =$ *congestion*; $k_{\tilde{\mathcal{G}}2} =$ *free flow*		
Filled $\mathbf{T_p}$	**Filled $\mathbf{T_+}$**	**Filled $\mathbf{T_c}$**
"A severe congestion was predicted on Golden State Freeway at kms 10 and 11 on Thursday, 05/06/2012, with an average speed of 42.47 km/h from 09:45 to 10:35."	$\emptyset$	*"This was caused by a congestion and a free flow."*
Paragraph Composition		
"A severe congestion was predicted on Golden State Freeway at kms 10 and 11 on Thursday, 05/06/2012, with an average speed of 42.47 km/h from 09:45 to 10:35. This was caused by a congestion and a free flow."		

Table 3.1: Example of first paragraph formation.

The template $T_\text{c} \in \tau_\text{c}$ is filled by substituting a placeholder <*causes*> by the information of the event kind $k_{\tilde{\mathcal{G}}^c}$ of each subgraph of the important subgraph $\tilde{\mathcal{G}}^c$, which serves as an explanation of $\hat{\mathcal{G}}^*$. In this template, the causes leading to predictions are briefly summed up by a sentence such as *"This was caused by a series of congestions and free-flows."* or *"This was driven by a congestion."*. An example of the formation of the first paragraph is given in Table 3.1.

For the other paragraphs, two sets of templates are used. τ_e which templates serve to describe the input event $\tilde{\mathcal{G}}^c$ and τ_+ which is the same set as the first paragraph to describe additional location information. Once again, the elements of each template are equivalent in terms of semantics. In order to form the other paragraphs a random template from each set, $T_e \in \tau_\mathrm{e}$ and $T_+ \in \tau_+$, is fetched and filled with the extracted content. Afterwards, the filled templates are concatenated ($*$) in the order:

$$\text{other paragraph} = T_e * T_+ \tag{3.36}$$

Each template $T_e \in \tau_\mathrm{e}$ contains the same placeholders as the elements of τ_c, although they are filled with extracted content of the considered input event $\tilde{\mathcal{G}}^c$, instead of the extracted content of $\hat{\mathcal{G}}^*$. An example of the formation of the other paragraphs is given in Table 3.2.

T$_\mathrm{e}$	**T$_+$**
"A contributing <label> manifested on <location main>, occurring <time> <day> with an average speed of <speed> km/h."	*"The <label> also affected <location secondary>."*
Extracted content	
$s_{\tilde{\mathcal{G}}^c} = 97.09$; $k_{\tilde{\mathcal{G}}^c} =$ *free flow*; $l_{\tilde{\mathcal{G}}^c} = \{$*Ventura Freeway* $: \{9, 10, 11, 12, 13\},$ *Golden State Freeway* $: \{10\}\}$; $d_{\tilde{\mathcal{G}}^c} = \{$*Thursday, 05/06/2012*$\}$; $t_{\tilde{\mathcal{G}}^c} = (08{:}45, 09{:}40)$	
Filled T$_\mathrm{e}$	**Filled T$_+$**
"A contributing free flow manifested on Ventura Freeway at kms 9, 10, 11, 12 and 13, occurring from 08:45 to 09:40 on Thursday, 05/06/2012 with an average speed of 97.09 km/h."	*"The free flow also affected Golden State Freeway at km 10."*
Paragraph Composition	
"A contributing free flow manifested on Ventura Freeway at kms 9, 10, 11, 12 and 13, occurring from 08:45 to 09:40 on Thursday, 05/06/2012 with an average speed of 97.09 km/h. The free flow also affected Golden State Freeway at km 10."	

Table 3.2: Example of other paragraphs formation.

The final verbal translation consists of the concatenation of the first paragraph, followed by the other paragraphs sorted by the time of the occurrence of the event they define. The ruleset governing the filling of each placeholder p within the templates is determined by a series of dedicated functions rule(p, c) utilizing the required content c. A comprehensive explanation of the adopted ruleset in the experiment is provided in section 4.5.2. In this section are also discussed additional enrichments applied to the paragraphs to enhance the natural flow and expression of the verbal narratives.

Chapter 4

Experimental Setup

4.1 Technology Stack

The experiment utilizes *Python* [47] as the primary programming language. *PyTorch* [48] version 2.0.0+cu117 was employed for building the STGNN model and defining the error and validation metrics. Clustering methods like DBSCAN and Agglomerative Clustering were selected from *scikit-learn* [49] version 1.3.2. *Numpy* [50] version 1.23.5 was utilized for matrix operations, while data analysis was performed through *pandas* [51] version 1.5.3. The visualization of the spatio-temporal traffic networks on the geographical maps was accomplished using *Kepler.gl* [52]. Furthermore, the *GeoPy* [53] Python library has been used for geolocation operations.

The experiment was conducted on a machine with an architecture comprising 16GB of RAM, an Intel Core i7 10700k 2.60 GHz 6-core CPU, and an NVIDIA GeForce RTX 2080 GPU with 8GB of RAM. The STGNN training and inference operations have been performed using GPU computations through the *Cuda* Pytorch library.

To ensure reproducibility, seeding was employed for random operations. Specifically, seed 42 was used for the random Python library, the random numpy module, as well as for Pyorch and cuda random modules. Moreover, deterministic Cuda operations were exclusively used by setting `cudnn.`

`deterministic` as `True`.

4.2 Data

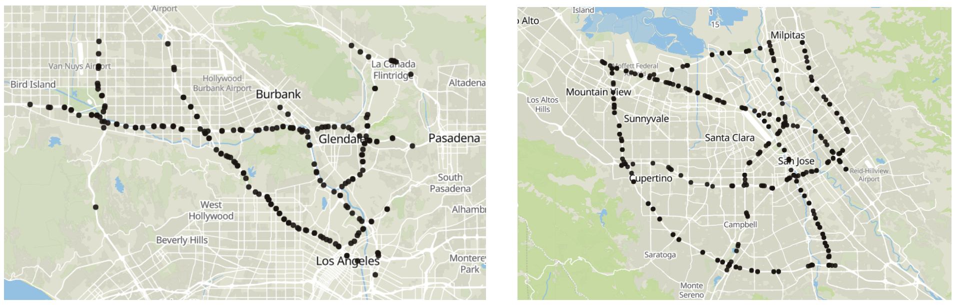
Experiments were carried out using two extensive real-world datasets: The *METR-LA* dataset, comprising traffic data gathered from loop detectors along Los Angeles County's highways [54] and the *PEMS-BAY* dataset, which loop detectors data is sourced from the *California Transportation Agencies Performance Measurement System (PeMS)*. The experiment uses the same subset of data as the one collected in the paper *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting* [55]. Specifically, the METR-LA dataset employs 207 sensors covering the area of central Los Angeles, while including the Glendale, Burbank and La Cañada Flintridge districts. On the west side, it extends up to the Sepulveda zone. Data is gathered from March 1st, 2012, to June 30th, 2012, over a 4-month period. The traffic speed readings are aggregated in 5-minute windows for a total of $6,519,002$ observed traffic data points. On the other hand, the PEMS-BAY dataset includes 325 sensors in the Bay Area mainly covering the zone of central San Jose while also including areas of Milpitas, Santa Clara, Cupertino, Sunnyvale and Mountain View. The data is collected from January 1st, 2017, to May 31st, 2017, spanning 6 months. Speed readings are aggregated once again in 5-minute windows for a total of $16,937,179$ traffic data points. These two datasets were chosen for the experiment due to their availability as open-source data and their extensive use as benchmarks in traffic forecasting experiments. Figure 4.1 illustrates the spatial distribution of the loop detectors contained in the datasets.

For both dataset the following content is provided:

- The latitude and longitude information for each loop detector.
- An adjacency matrix measuring the spatial proximity between any pair

of loop detectors normalized between 0 and 1.

- For each discretized timestep, the date and time of the measurement along with the registered average speed in miles per hour by each loop detector.



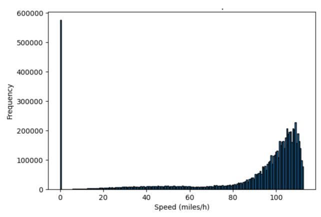
(a) METR-LA (b) PEMS-BAY

Figure 4.1: Distribution of the sensors in the employed datasets.

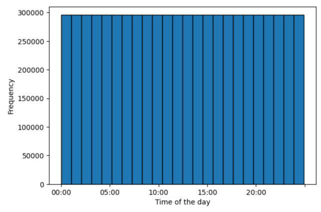
4.2.1 Analysis

In both datasets, observations with a speed value of 0 are considered as missing data. A closer look at the METR-LA dataset reveals that the instances with missing speed information amount to a significant portion, accounting for 8.1% of all observations. Conversely, in the PEMS-BAY dataset, the occurrence of missing data is notably low, constituting only 0.0031% of the total data.

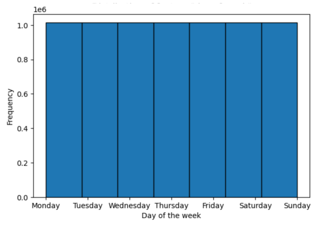
As seen in Figure 4.2, the speed distribution of METR-LA nodes displays a left-skewed pattern, suggesting a prevalence of free-flowing traffic events and rare congestions during the observed period. A prominent peak at value 0 indicates a significant number of missing speed data instances, aligning with the previously observed high percentage of missing data. Regarding the distribution of observation timestamps in METR-LA for both time of day and day of the week, each element appears well represented, displaying a uniform distribution pattern.



(a) Speed frequency.



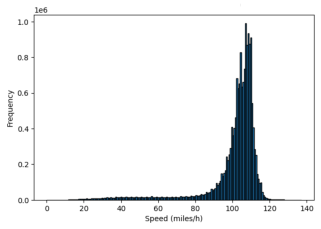
(b) Time of the day frequency.



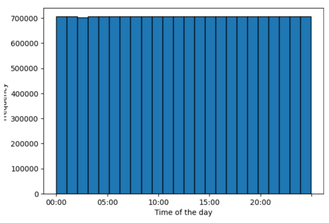
(c) Day of the week frequency.

Figure 4.2: METR-LA data distribution.

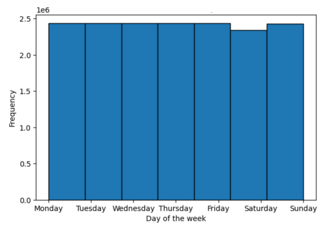
Likewise, Figure 4.3 demonstrates a left-skewed speed distribution among PEMS-BAY nodes, suggesting infrequent congestion occurrences. Unlike METR-LA, there are no peaks at value 0, aligning with the previously seen low percentage of missing data. Additionally, the distributions for time of day and day of the week exhibit uniform patterns in PEMS-BAY, indicating an equal representation for each day and time slot.



(a) Speed frequency.



(b) Time of the day frequency.



(c) Day of the week frequency.

Figure 4.3: PEMS-BAY data distribution.

The datasets are further inspected by extracting the speed distribution of the three nodes exhibiting the most significant speed variations, considered pivotal for analysis, for each day of the initial week as a sample. Additionally, an analysis is conducted on the average speed distribution of all nodes based on both the hour of the day and the day of the week. Regarding the first week of METR-LA, Figure 4.4 shows that during weekdays, the lowest speeds occur between 7:00 am and 6:00 pm. However, on certain days like Monday and Tuesday, speed picks up during off-peak hours, notably around 12:00 am and 1:00 pm. Notably, on Friday and Saturday, the trend of low speeds seems to persist until approximately 8:00 pm. This might be associated with these days being near the end of the week when people tend to be more active during later hours. This pattern of lower speeds in late hours isn't observed on Sunday. On weekends, it's clear that movement is less in the early hours, with significant speed decreases starting around 10:00 am. For Saturday, slower speeds continue until 8:00 pm, while on Sunday, the slower speed trend diminishes around 2:00 pm.

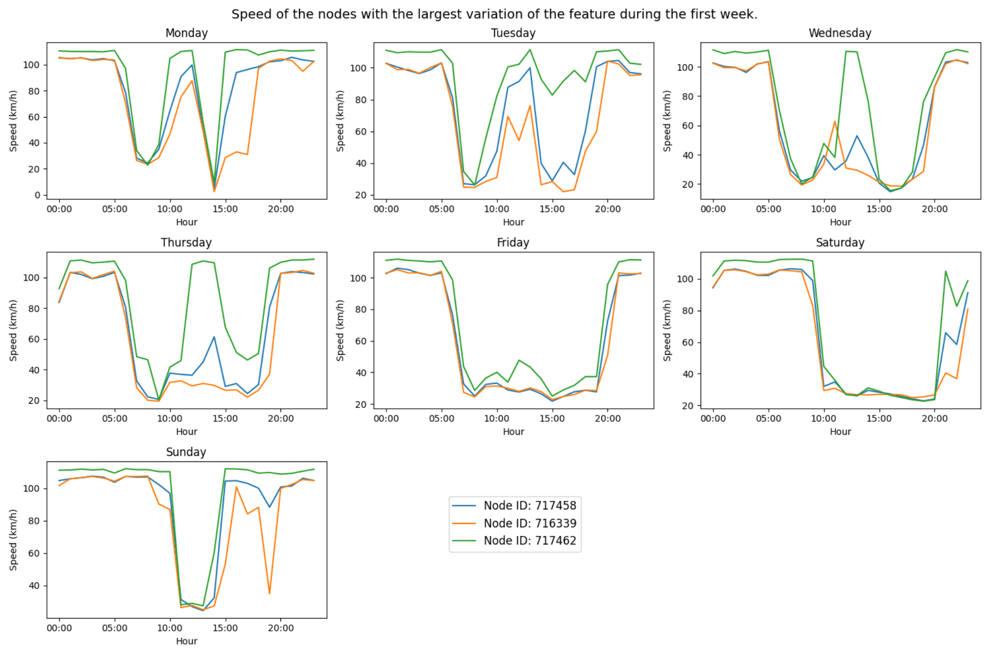
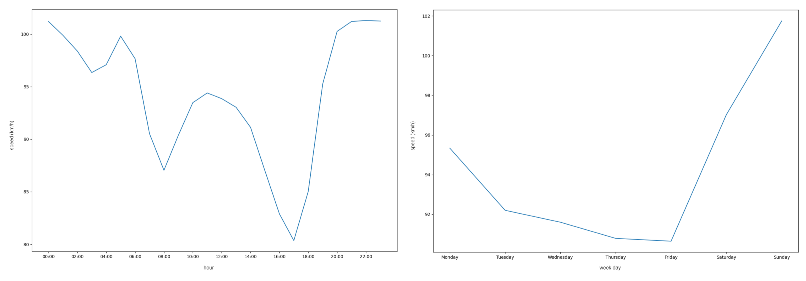


Figure 4.4: Analysis of the speed of pivotal nodes of METR-LA on the first week.

As seen in Figure 4.5, in terms of hourly METR-LA average speeds, there are two notable dips occurring during peak traffic hours at 8:00 a.m. and 5:00 p.m., corresponding to the times when people commute to and from work. These periods likely experience heavier traffic flow. Conversely, the highest speeds are unsurprisingly seen during nighttime hours when traffic flow tends to be lighter. The average speed typically remains high throughout the week, fluctuating between 90 to 100 km/h. Weekdays generally exhibit the lowest average speeds, except for Mondays. Conversely, higher average speeds are noticeable during the weekends, suggesting reduced overall traffic congestion.



(a) Average speed per hour. (b) Average speed per week

Figure 4.5: Analysis of the average speed of the METR-LA nodes for the collected period.

For what concerns PEMS-BAY, Figure 4.6 depicts that during the weekdays of the first week, the lowest speed values occur between 3:00 pm and 7:00 pm, coinciding with working hours. Saturday exhibits a pattern where speeds are high in the morning and decrease towards the night. Peaks in speed are particularly noticeable at 9:00 am, 3:00 pm, and 8:00 pm. On Sunday, speeds tend to remain high, with decreases between 12:00 am and 5:00 pm.

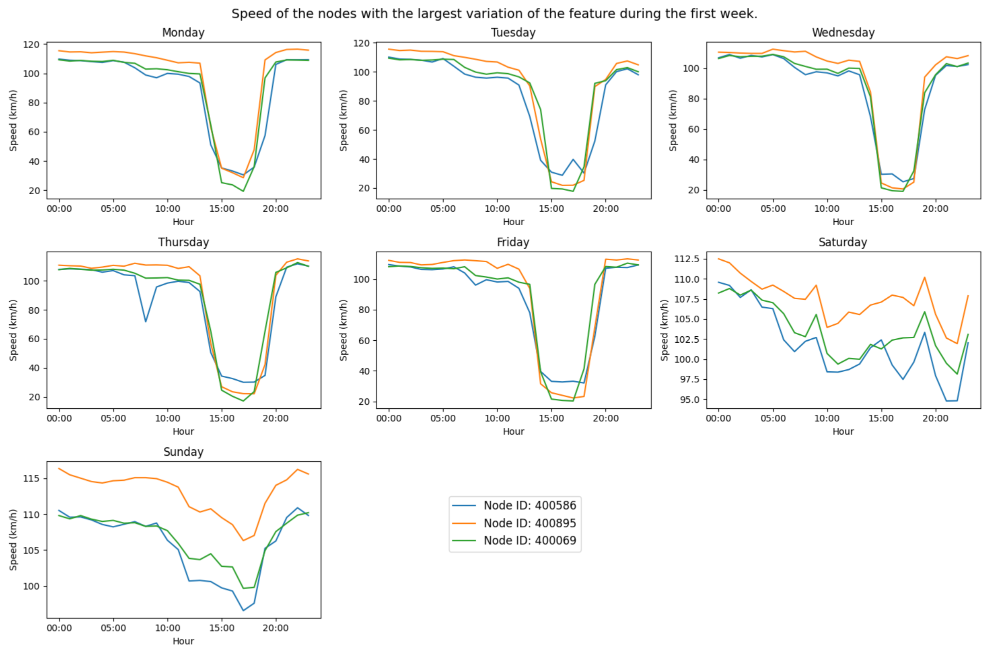
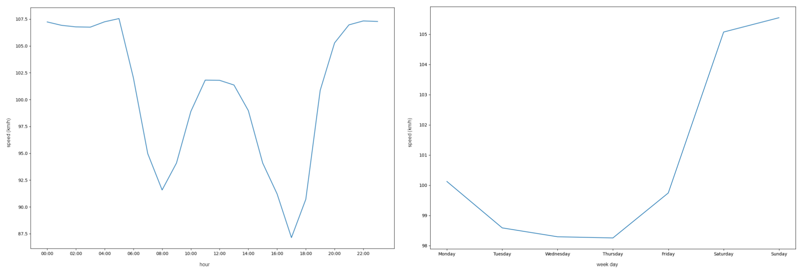


Figure 4.6: Analysis of the speed of pivotal nodes of PEMS-BAY on the first week.

As depicted in Figure 4.7, the peak velocities in PEMS-BAY occur at night, while the lowest points are evident during rush hours at 8:00 am and 5:00 pm. As anticipated, weekdays exhibit lower average speeds, typically below 100 km/h. On weekends, the average speed rises to over 105 km/h. This aligns with expectations as traffic flow is usually heavier on weekdays. However, the disparity in speed between weekdays and weekends isn't substantial.



(a) Average speed per hour. (b) Average speed per week

Figure 4.7: Analysis of the average speed of the PEMS-BAY nodes for the collected period.

4.2.2 Pre-processing

The experiment involved partitioning both datasets into three subsets: a *training set* $\mathcal{D}_{\text{train}}$, a *validation set* $\mathcal{D}_{\text{val}}$, and a *test set* $\mathcal{D}_{\text{test}}$. Given the time-series nature of the datasets, the split was performed sequentially in time. This sequential splitting ensured the preservation of temporal continuity and prevented data leakage by maintaining the order of observations across the three sets, thus avoiding the shuffling of future and past data within the subsets. The division of the datasets aligns with the methodology detailed in the paper by [42]. It involves allocating the last 20% of observations to form the test sets $\mathcal{D}_{\text{test}}$, while the train and validation sets are extracted from the initial 80% portions. More precisely, within these initial portions, the final 10% is allocated to construct the validation sets $\mathcal{D}_{\text{val}}$, leaving the initial 90% for the training sets $\mathcal{D}_{\text{train}}$.

After dividing the dataset into the three segments, sequence samples are generated by sliding a window of width $T + T'$. Each sample sequence spans time steps with 5-minute intervals. The initial T time steps are designated as input data for predictions, while the subsequent T' steps represent the ground truth. In this experiment, both T and T' are set to 12-time steps, establishing a forecasting approach that utilizes one hour of historical traffic data to predict the state of the traffic network in the subsequent hour. The resulting datasets are thus built as a set of input instances $\mathcal{X}$ and the corresponding ground truth $\mathcal{Y}$. Each input instance $\mathcal{X}_i \in \mathbb{R}^{T \times N \times F}$ is a three-dimensional matrix describing the input traffic network with $T = 12$ the number of input timesteps, N the number of traffic nodes and F the number of features. The feature set of each node at each timestep $X_t^{v_i} \in \mathbb{R}^F$, where $t \in [0, ..., 11]$ is an input timestep and v_i is a traffic node, is a vector composed of 9 features, where:

- The first element of the vector is the registered average speed $s_t^{v_i}$ of the node v_i at discretized timestep t.

- The second element is the encoding of the time of the day normalized between 0 and 1 which is extracted from the date associated at timestep t, such that:
$$X_{v_i}^t[1] = \frac{\text{hour}(t) \cdot 60 + \text{minute}(t)}{23 \cdot 60 + 59} \tag{4.1}$$
with hour(t) a function to extract the hour at the given timestep t and minute(t), the function to extract its associated minute.

- The last 7 elements are the *one-hot-encoding* of the day of the week associated with timestep t, allowing for a unique binary sequence to indicate which day is associated with that specific time step. For instance, if the sequence is `[0, 0, 0, 0, 0, 0, 1]`, it signifies that the time step t corresponds to Sunday. Each day of the week is mapped to a unique binary pattern, providing its categorical representation.

Note that no further time information, such as the month or the year, has been extracted from the timestep and added to the input feature set. This decision was deliberate, as the time span covered in both datasets was too short for the models to be trained to learn from such patterns.

On the other hand, each ground truth of an instance $\mathcal{Y}_i \in \mathbb{R}^{T' \times N \times 1}$ is a three-dimensional matrix describing the output traffic network with $T' = 12$ the number of output timesteps and N the number of traffic nodes. In this matrix, the last dimension records the registered speed $s_t^{v_i}$ of each node v_i at the respective time step t. Note that the encoding of the time information is absent in the ground truth, as the focus of the STGNN is to learn future speed information in the traffic network and the future timestamps can be deduced directly from the input time information.

Each of the train, validation and test datasets $\mathcal{D}$ is thus finally defined as a set of instances containing input data $\mathcal{X}_i$ that describes the input traffic network $\mathcal{G}_i$ at the first T timesteps and ground truth data $\mathcal{Y}_i$ defining the traffic network $\mathcal{G}_i'$ at future T' timesteps, such that:

$$\mathcal{D} = \{(\mathcal{G}_i, \mathcal{G}_i')\}_{i \in [0,\dots,N]} \tag{4.2}$$

with N being the number of instances in D.

4.3 STGNN

In this section the hyperparameters of the used STGNNs, along with the training approach are described.

4.3.1 Hyperparameters

In the employed STGNN Φ the hidden features of the initial linear layer, of the S-GNN layers and of the GRU layers are set to 64. Moreover, the latent positional representation of the nodes obtained at the beginning of the S-GNN layers is obtained by a sequential feed forward network composed as an input linear layer with the same number of input and output features followed by another linear layer with an output dimension half of the input features. The number of attention heads of the transformer layer is set to 4, while the number of hidden features used for computing the queries $Q_s^{v_i}$, keys $K_s^{v_i}$ and values $V_s^{v_i}$ of each node v_i for each attention head of the transformer s is set as 64. Finally, the output layer outputing the predictions is a feed-forward neural network composed of two layers. The former layer has a hidden dimension of 64 and is followed by *ReLU* non-linear activation function. The latter layer has output dimension 1, as the STGNN predicts an output matrix $\hat{\mathcal{Y}} \in \mathbb{R}^{T' \times N \times 1}$, containing for each node of the traffic network at each future timestep the value of its predicted average speed.

4.3.2 Training Procedure

Two different STGNNs are trained, one on the METR-LA training set and the second on the PEMS-BAY training data ($\mathcal{D}_{\text{train}}$). To train both models *Adam* is used as an optimizer with learning rate $1e-3$ and weight decay $2e-6$. The training epochs are set to 200 for both models and the training instances are shuffled at each epoch to induce a more generalized training approach and reduce overfitting. For METR-LA a batch size of 64 is used during training, while for PEMS-BAY this parameter is set as 32. The employed loss function $\mathcal{L}$ is the *Mean Absolute Error (MAE)*. The function computes the average of the absolute error in each batch between the speed predictions of the model $\hat{\mathcal{Y}}_i \in \hat{\mathcal{Y}}$, describing the predicted output traffic network, $\hat{\mathcal{G}}_i$ and the ground truth speeds $\mathcal{Y}_i \in \mathcal{Y}$, describing the actual output traffic network, $\mathcal{G}'_i$. In the computation the nodes having missing ground truth speeds are excluded ($\mathcal{Y}_i = \{s_t^{v_j} \mid s_t^{v_j} \in \mathcal{Y}_i \neq 0\}$ and $\hat{\mathcal{Y}}_i = \{\hat{s}_t^{v_j} \mid \hat{s}_t^{v_j} \in \hat{\mathcal{Y}}_i \wedge s_t^{v_j} \in \mathcal{Y} \neq 0\}$), such that:

$$\text{MAE}(\hat{\mathcal{Y}}, \mathcal{Y}) = \frac{1}{N} \sum_{i=1}^{N} |\mathcal{Y}_i - \hat{\mathcal{Y}}_i| \tag{4.3}$$

with N being the number of batch samples.

Validation metrics are computed on the respective validation sets $\mathcal{D}_{\text{val}}$ after each training epoch is completed to measure the prediction accuracy of the model on data non used for training and, thus, avoid overfitting. Validation is obtained as the average error between the speed predictions of the model $\hat{\mathcal{Y}}_i \in \hat{\mathcal{Y}}$ and the ground truth speeds $\mathcal{Y}_i \in \mathcal{Y}$ of all the instances of the validation datasets. Nodes having missing ground truth speeds are excluded in the computation as it is done for computing the loss. For all metrics, smaller values indicate better prediction performances. In particular the following validation metrics are used:

- The *Mean Absolute Error (MAE)* computed as in the loss function as per Equation 4.3. This metric is more punitive on samples that are easier to predict and it is more suitable to measure the overfitting on those.
- The *Root Mean Squared Error (RMSE)*, defined as the rooted average squared difference between the predicted values and the ground truth:

$$\text{RMSE}(\hat{\mathcal{Y}}, \mathcal{Y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathcal{Y}_i - \hat{\mathcal{Y}}_i)^2} \tag{4.4}$$

  This metric punishes more unpredictable samples and its error measures overfitting mainly on those.
- *Mean absolute percentage error (MAPE)*, which considers the percentage of errors to ground truth as:

$$\text{MAPE}(\hat{\mathcal{Y}}, \mathcal{Y}) = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\mathcal{Y}_i - \hat{\mathcal{Y}}_i}{\mathcal{Y}_i}\right| \tag{4.5}$$

A checkpoint monitor is utilized to store the best performances of the models during validation. It saves the weights of the STGNNs Φ at the epoch in which the minimal sum of validation metrics is obtained:

$$\text{argmin}_{\Phi_i \mid i \in \text{epochs}} \text{MAE}_{\Phi_i}(\mathcal{D}_{\text{val}}) + \text{RMSE}_{\Phi_i}(\mathcal{D}_{\text{val}}) + \text{MAPE}_{\Phi_i}(\mathcal{D}_{\text{val}}) \tag{4.6}$$

where MAE_{Φ_i}, RMSE_{Φ_i} and MAPE_{Φ_i} are the metrics computed from Φ at epoch i. The trained models are ultimately evaluated on the same metrics used for validation on the respective test datasets to acquire more dependable error results on data that has not been encountered during the training procedure.

Additionally, the speed feature in the input data is scaled through *standard scaling* before being fed to the STGNN. The remaining input features, namely time of the day and one-hot-encoded day of the week, are not furtherly processed as they already lay in a space between 0 and 1. Standardization aids in stabilizing the training procedure by mitigating the challenge posed when dataset features exist on varying scales. Neural networks find it challenging to learn when adjustments to weights are required to cater to each feature differently, causing instability in the training process. Standard scaling on each speed input feature s is computed as:

$$s' = \frac{s - \tilde{\mu}}{\tilde{\sigma}} \tag{4.7}$$

where $\tilde{\mu}$ is the mean value of the speed distribution estimated by averaging the speed values in the whole respective training dataset $\mathcal{D}_{\text{train}}$, while $\tilde{\sigma}$ is the estimated standard deviation of the speed distribution computed as the standard deviation of the speed values in $\mathcal{D}_{\text{train}}$. After the models inference the speed values of the predictions $\hat{s}'$ are brought back to their original scale with:

$$\hat{s} = \hat{s}' \cdot \tilde{\sigma} + \tilde{\mu} \tag{4.8}$$

This guarantees that the predictions are expressed in miles per hour and that the loss function and the validation metrics are computed between instances with features at the same scale.

4.4 Explainer

This section details the selection of the hyperparameters used in the DBSCAN algorithm in order to extract the output events $\hat{\mathcal{G}}^*$ in the predicted traffic networks $\hat{\mathcal{G}}$, while illustrating the employed evaluation metrics to evaluate the clustering results. Then it will delineate the hyperparameter selection approach for the explainer ψ along with the metrics used to evaluate the important subgraph of the input network $\tilde{\mathcal{G}}$ that ψ outputs as an explanation. Finally it will illustrate the parameters used in the Agglomerative Clustering for the content extraction of the events of the important subgraph $\tilde{\mathcal{G}}$ along with the evaluation approach used.

4.4.1 Output Events Selection Setup

Events selection is performed by dividing the traffic network $\hat{\mathcal{G}}$ predicted by the STGNN Φ into separate clusters $\hat{\mathcal{G}}^*$ which describe important events, such as congestions or free-flows. These clusters should each contain traffic nodes close in time and space and with low variances of speed. Moreover, each of them should be as dissimilar as possible to the others. The clustering is performed by applying the DBSCAN algorithm on the predictions $\hat{\mathcal{G}}$ using a distance matrix M among all nodes of $\hat{\mathcal{G}}$ as a measure of dissimilarity.

The hyperparameters of DBSCAN are `eps`, measuring the maximum distance between two samples for one to be considered as in the neighbourhood of the other, and `min samples`, namely the number of samples in a neighbourhood for a point to be considered as a core point. For what concerns the clustering of the predicted traffic networks $\hat{\mathcal{G}}$ by Φ of both METR-LA and PEMS-BAY, the selected values for `eps` is 0.35, while the number of `min samples` is set as 5. Moreover, the speed distance matrix component of the distance matrix M is overweighted by a factor W of 3 for both METR-LA and PEMS-BAY. W, `eps` and `min samples` are tuned by performing grid search on the predictions $\hat{\mathcal{G}}$ obtained by Φ on the training datasets $\mathcal{D}_{\text{train}}$.

Each combination of hyperparameters was evaluated by considering the following metrics on the clustering results:

- **Within cluster variance:** It measures the normalized variance among the features of the same clusters weighted by the size of the cluster. The lower the result the better, as it denotes that the similarity of each element within the same clusters is high. It is applied on each clustered prediction while considering the speed attributes of the nodes of the obtained clusters as:

  $$WCV = \frac{1}{\sum_{\hat{\mathcal{G}}_i \in \hat{\mathcal{G}}} N_i} \frac{\sum_{\hat{\mathcal{G}}_i \in \hat{\mathcal{G}}} N_i \sigma_i^2}{\sigma^2} \tag{4.9}$$

  with $\hat{\mathcal{G}}_i$ being the i^{th} cluster in the predicted network $\hat{\mathcal{G}}$, N_i being the number of nodes in $\hat{\mathcal{G}}_i$ and σ_i^2 the speed variance among its nodes. σ^2 is the variance among all nodes of $\hat{\mathcal{G}}$.

  The within cluster variance is averaged across all clustering results of each predicted network of the employed dataset. Moreover, nodes that are flagged as "noise" by DBSCAN are excluded from the computation.

- **Cluster Dissimilarity:** It measures the dissimilarity among the features of different clusters. The higher the result the better, as it denotes that the dissimilarity of each element within different clusters is high. It is applied on each clustered prediction while considering the speed attributes of the nodes of the obtained clusters as:

  $$CD = \frac{\sum_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2 \in \hat{\mathcal{G}} | \hat{\mathcal{G}}_1 \neq \hat{\mathcal{G}}_2} \sqrt{N_1 \cdot N_2} \cdot |\mu_1 - \mu_2|}{\sum_{\hat{\mathcal{G}}_1, \hat{\mathcal{G}}_2 \in \hat{\mathcal{G}} | \hat{\mathcal{G}}_1 \neq \hat{\mathcal{G}}_2} \sqrt{N_1 \cdot N_2}} \tag{4.10}$$

  with $\hat{\mathcal{G}}_i$ being the i^{th} cluster in the predicted network $\hat{\mathcal{G}}$, N_i being the number of nodes in $\hat{\mathcal{G}}_i$ and μ_i the mean speed among its nodes.

  The cluster dissimilarity is averaged across all clustering results of each predicted network of the employed dataset. Moreover, nodes that are flagged as "noise" by DBSCAN are excluded from the computation.

- **Noise Ratio:** It measures the ratio of the nodes classified as outliers by DBSCAN in a predicted traffic network $\hat{\mathcal{G}}_i$. The lower this value the better, as it means that more nodes are assigned to a cluster. The noise ratio is averaged across all clustering results of each predicted network of the employed dataset.

DBSCAN has been applied for both METR-LA and PEMS-BAY on the predictions of each instance of each dataset, namely the train $\mathcal{D}_{\text{train}}$, validation $\mathcal{D}_{\text{val}}$ and test sets $\mathcal{D}_{\text{test}}$. The obtained clustered events of each prediction of each set $\mathcal{D}$ have been collected into three different sets: a set describing events of *severe congestion*, one describing *congestions* and the latter containing events of *free-flows* of traffic. The rule applied to assign an event $\hat{\mathcal{G}}_i$ to a specific set considers two speed thresholds: t_1 and t_2 such that:

$$\hat{\mathcal{G}}_i = \begin{cases} \textit{severe congestion}, & \text{if } \mu_i \leq t_1 \\ \textit{congestion}, & \text{if } t_1 < \mu_i \leq t_2 \\ \textit{free-flow}, & \text{otherwise} \end{cases} \tag{4.11}$$

where μ_i is the average speed of the traffic nodes in $\hat{\mathcal{G}}_i$. In detail, the thresholds have been selected as the ones defined in the official mobility performance reports of the *Department of Transportation of California* [56, 57]. For both METR-LA and PEMS-BAY, the threshold for severe congestion t_1 is set as 35 miles per hour or ≈ 56 kms per hour, while the one for a congestion t_2 is set as 60 miles per hour, meaning ≈ 96 kms per hour.

For both METR-LA and PEMS-BAY, three datasets are built, namely $\mathcal{D}^*_{\text{train}}$, $\mathcal{D}^*_{\text{val}}$ and $\mathcal{D}^*_{\text{test}}$. Each of these sets $\mathcal{D}^*$ contains an equal number of severe congestions, congestions and free flows. In practice, each instance of each dataset $\mathcal{D}^*$ is described by a predicted event $\hat{\mathcal{G}}^*_i \subseteq \hat{\mathcal{G}}_i$ and the associated input traffic network $\mathcal{G}_i$ that led to the prediction of $\hat{\mathcal{G}}_i$ by Φ, such that:

$$\mathcal{D}^* = \{(\mathcal{G}_i, \hat{\mathcal{G}}_i^*)\}_{i \in [0,\dots,N]} \tag{4.12}$$

where N is the number of instances in $\mathcal{D}^*$. The number of events selected in each dataset has been chosen in order to reflect the sizes of the respective original datasets $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$. In particular, for both METR-LA and PEMS-BAY, 999 instances have been used for $\mathcal{D}^*_{\text{train}}$, 198 for $\mathcal{D}^*_{\text{val}}$ and 300 for $\mathcal{D}^*_{\text{test}}$. In each dataset $\mathcal{D}^*$, one third of events are severe congestions, one third are congestions and one third are free-flows. The event selection criteria has been employed in a way that the sampled events are as equidistant in time as much as possible, in order to guarantee the maximum temporal variance of the instances in the datasets.

4.4.2 Monte Carlo Tree Search Setup

The MCTS employed by the explainer ψ computes a global heuristic among the traffic nodes of the input traffic network $\mathcal{G}$ and the nodes of a predicted event to explain $\hat{\mathcal{G}}^*$. It obtains a subset of $\mathcal{G}$ defined as $\mathcal{G}_{\text{root}}$, by selecting the top N^* scoring nodes according to the global heuristic. It finally performs a localized MCTS on $\mathcal{G}_{\text{root}}$ and outputs an important subgraph $\tilde{\mathcal{G}} \subseteq \mathcal{G}_{\text{root}}$ that serves as the explanation of the event $\hat{\mathcal{G}}^*$.

The hyperparameters of ψ contain N^*, defining the size of $\mathcal{G}_{\text{root}}$ and $\tilde{N}$, the sparsity threshold describing the maximum size of the important subgraph $\tilde{\mathcal{G}}$. Other hyperparameters include the exploration weight λ used for weighting the exploration in the MCTS, the number R of rollouts to perform and the value $\tilde{s}$ that is used to substitute the speed values of the removed nodes in a leaf of the tree $\mathcal{G}_{\text{leaf}}$. For METR-LA N^* is selected as double the value of $\tilde{N}$, while this hyperparameter is set as double the size of the total nodes in the event to explain $\hat{\mathcal{G}}^*$. Additionally, λ is set as 20 and R as 50. Finally, $\tilde{s}$ is set to 0, the original value used to define missing recorded speed data of traffic nodes. This value has been adopted since the frequency of missing data in METR-LA is quite high as seen in section 4.2.1 and the STGNN Φ is well-trained in order to recognize missing data patterns and output reliable predictions despite it. For PEMS-BAY N^* is selected with the same criteria as in METR-LA, while $\tilde{N}$ is set as three times the size of the event to explain. λ is set as 10 and R, again at 50. However, $\tilde{s}$ is not set to 0, as the frequency of missing data in PEMS-BAY is insignificant as seen in section 4.2.1. Due to this reason, the STGNN Φ trained on PEMS-BAY data is expectedly not able to handle correctly missing node speed information. If the speed $\tilde{s}_t^{v_i}$ of each removed node v_i at timestep t of $\mathcal{G}_{\text{leaf}}$ is set as 0, Φ is expected to confuse it as an occurrence of severe congestion. Thus $\tilde{s}$ is set as an "out-of-bound" value of the speed distribution, namely -60. The hyperparameters are tuned by performing a grid search on the instances of the dataset $\mathcal{D}^*_{\text{train}}$. Each possible combination of hyperparameters was evaluated by considering the *Fidelity*$^-$ on the sets $\mathcal{D}^*_{\text{train}}$ along with the *average explanation time*:

- **Fidelity**$^-$ [58]: It computes the difference in absolute error between predictions obtained using the whole input graph and the ones obtained using the important substructure of the input graph that serves as explanation. The lower the metric the better, as the explanation obtained by the important subgraph is more sufficient to explain the predictions:

$$\text{Fidelity}^- = \frac{1}{N}\sum_{i=1}^{N}(|\Phi(\mathcal{G}_i) - \Phi(\tilde{\mathcal{G}}_i))|) \tag{4.13}$$

  Where Φ is the model computing the predictions, $\mathcal{G}_i$ the i^{th} input graph, $\tilde{\mathcal{G}}_i$ the i^{th} important subgraph serving as explanation and N the number of instances used to compute the metric.

  In the thesis work this metric is adapted for the problem task. It computes the average error among each instance i between the original predicted event speeds $\hat{\mathcal{Y}}_i^*$, obtained by feeding the STGNN Φ with the whole input network $\mathcal{G}$, and the re-predicted event speeds $\tilde{\mathcal{Y}}_i^*$, obtained by feeding to Φ just the important subgraph $\tilde{\mathcal{G}}$. The metric is thus equivalent to the computation of the MAE between each $\hat{\mathcal{Y}}_i^*$ and $\tilde{\mathcal{Y}}_i^*$ and it is actually expressed as:

$$\text{Fidelity}^- = \text{MAE}^- = \frac{1}{N}\sum_{i=1}^{N}(|\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^*|) \tag{4.14}$$

Where N are the number of instances in $\mathcal{D}^*_{\text{train}}$, $\hat{\mathcal{Y}}_i^*$ are the original predicted speeds of the event to explain $\hat{\mathcal{G}}_i^*$ at instance i in $\mathcal{D}^*_{\text{train}}$ and $\tilde{\mathcal{Y}}_i^*$ are the re-predicted event speeds of the event at instance i using just the important subgraph $\tilde{\mathcal{G}}_i$ obtained from $\mathcal{G}_i$ by ψ.

- **Average Explanation Time:** It measures the average time employed to output the important subgraph $\tilde{\mathcal{G}}$ by ψ for each instance of $\mathcal{D}^*_{\text{train}}$.

The goodness of the results of the explanations has been evaluated on $\mathcal{D}^*_{\text{val}}$ and $\mathcal{D}^*_{\text{test}}$. The selected metrics are:

- **Fidelity$^-$:** Computed as the averaged MAE between $\hat{\mathcal{Y}}_i$ and $\tilde{\mathcal{Y}}_i$ considering all the instances i of the dataset as per Equation 4.14.

  Moreover, modifications of the classic Fidelity$^-$ have been also considered by using different error metrics, namely RMSE and MAPE. *RMSE$^-$* is the fidelity computed using the RMSE as:

$$\text{RMSE}^- = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^*)^2} \tag{4.15}$$

  MAPE$^-$ is the fidelity computed using the MAPE as:

$$\text{MAPE}^- = \frac{100}{N}\sum_{i=1}^{N}|\frac{\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^*}{\hat{\mathcal{Y}}_i^*}| \tag{4.16}$$

- **Fidelity$^+$** [58]: It computes the difference in absolute error between predictions obtained using the whole input graph and the ones obtained using the complement of the important substructure of the input graph that serves as explanation. The higher the metric the better, as the explanation described by the important subgraph is more necessary to explain the predictions:

$$\text{Fidelity}^{+} = \frac{1}{N}\sum_{i=1}^{N}(|\Phi(\mathcal{G}_i) - \Phi(\tilde{\mathcal{G}}_i^{\mathsf{c}}))|) \tag{4.17}$$

Where Φ is the model computing the predictions, $\mathcal{G}_i$ the i^{th} input graph, $\tilde{\mathcal{G}}_i^{\mathsf{c}} = \mathcal{G} \setminus \tilde{\mathcal{G}}_i$ the complement of the i^{th} important subgraph serving as explanation and N the number of instances used to compute the metric.

In the thesis, this metric calculates the average error per instance i between the originally predicted event speeds $\hat{\mathcal{Y}}_i^*$, derived from the entire input network $\mathcal{G}$ fed into the STGNN Φ, and the re-predicted event speeds $\tilde{\mathcal{Y}}_i^{*\mathsf{c}}$, obtained by feeding only the complement of the important subgraph $\tilde{\mathcal{G}}_i^{\mathsf{c}}$ to Φ. The metric computes the Mean Absolute Error (MAE) between each $\hat{\mathcal{Y}}_i^*$ and $\tilde{\mathcal{Y}}_i^{*\mathsf{c}}$, represented as:

$$\text{Fidelity}^{+} = \text{MAE}^{+} = \frac{1}{N}\sum_{i=1}^{N}(|\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^{*\mathsf{c}}|) \tag{4.18}$$

Where N are the number of instances in the dataset.

Similarly to Fidelity^{-}, modifications of the classic Fidelity^{+} have been also considered. *RMSE*$^{+}$ is the fidelity computed using the RMSE as:

$$\text{RMSE}^{+} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^{*\mathsf{c}})^2} \tag{4.19}$$

MAPE^{+} is the fidelity computed using the MAPE as:

$$\text{MAPE}^{+} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\hat{\mathcal{Y}}_i^* - \tilde{\mathcal{Y}}_i^{*\mathsf{c}}}{\hat{\mathcal{Y}}_i^*}\right| \tag{4.20}$$

- **Sparsity** [58]: It measures the average opposite of the fraction of input data that is selected as being important by the explanation method.

Note that higher sparsity values are better as they indicate that explanations are sparser, thus likely to capture only the most essential input information. Sparsity is computed as:

$$\text{Sparsity} = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{m_i}{M_i}) \tag{4.21}$$

Where N are the number of instances in the dataset, m_i the number of nodes in the important subgraph used for the explanation $\tilde{\mathcal{G}}_i$ obtained by ψ at instance i and M_i the number of nodes of the input graph $\mathcal{G}_i$ at instance i.

- **Average Explanation Time:** measuring the average time employed to output the important subgraphs across all instances.

4.4.3 Important Subgraph Events Extraction Setup

Content extraction from the important subgraph $\tilde{\mathcal{G}}$ is performed by dividing $\tilde{\mathcal{G}}$ into C separate clusters $\hat{\mathcal{G}}^c$ which describe important events, such as congestions or free-flows. These clusters should each contain traffic nodes close in time and space and with low variances of speed. Moreover, each of them should be as dissimilar as possible to the others. The clustering is performed by applying the Agglomerative Clustering algorithm using a distance matrix M among all nodes of $\tilde{\mathcal{G}}$ as a measure of dissimilarity. The parameter W weighting the speed factor in M is set to 3.

Agglomerative clustering is performed on the obtained $\tilde{\mathcal{G}}$ 5 times considering a number of clusters varying from 1 to 5. *Single linkage* is applied as a metric to agglomerate pairs of clusters at each iteration of the algorithm. In detail, it agglomerates clusters that present the minimum distances between all traffic nodes of the two sets according to M. This linkage method expects to minimize the Within Cluster Variance (Equation 4.9) among each obtained cluster $\tilde{\mathcal{G}}^c$. The number of clusters $C \in [1, ..., 5]$ is selected as the one that leads the Agglomerative Clustering results of $\tilde{\mathcal{G}}$ to the highest ratio between Cluster Dissimilarity (Equation 4.10) and Within Cluster Variance (Equation 4.9):

$$\text{argmax}_c = \frac{CD_c}{WCV_c} \tag{4.22}$$

Where CD_c and WCV_c are the Cluster Dissimilarity and Within Cluster Variance scores on Agglomerative Clustering computed using c clusters. This scoring function selects c that balances a high dissimilarity between clusters, while their within variance is low.

4.5 Verbal Explanations

This section details the specifics of the content selection functions applied to the subgraphs $\tilde{\mathcal{G}}^c$ of the important subgraph $\tilde{\mathcal{G}}$ and to the event to explain $\hat{\mathcal{G}}^*$. Then it defines the adopted ruleset to fill the placeholders of the templates used to translate into narratives the extracted content, along with additional enrichments applied to the verbal narratives to enhance their natural flow and expression.

4.5.1 Content Extraction Setup

For what concerns extracting the average speed $s_{\mathcal{G}^c}$ from an event $\mathcal{G}^c$, in order to conform to the *International System of Units* the speed measurements $s_{\mathcal{G}^c}$ are translated from miles per-hour to kilometers per hour by multiplying it by the constant 1.609344. Precisely, 1 mile is equivalent to 1.609344 kms.

The speed thresholds used to define the event kind $k_{\mathcal{G}^c}$ of the event $\mathcal{G}^c$ as per Equation 3.30 are the same as the ones used in section 4.4.1. In particular: s_{sc}, the threshold for severe congestions, is set as 35 miles per hour or ≈ 56 kms per hour, while s_c, the one for congestions, is set as 60 miles per hour, meaning ≈ 96 kms per hour.

In the dictionary describing the location of event $\mathcal{G}^c$, $l_{\mathcal{G}^c}$, The set of kilometrage of each street $l \in l_{\mathcal{G}^c}$ is computed by rounding the kilometrage of each node to the nearest integer. This is applied in order to have a more concise and descriptive set of kilometrages for the streets in the dictionary. The kilometrage for each node v_i is pre-calculated using the Haversine formula, determining the kilometers between the latitude and longitude of v_i and the starting point of the street in which v_i is situated, which is the north-westmost point. The starting point of each street has been manually fetched using *ArcGIS World Street Map* [59], while the street each node is part of has been extracted using the *GeoPy* [53] Python library.

4.5.2 Verbal Translation Setup

The placeholder of the various templates is filled using a specific ruleset, such that given a placeholder p within the templates and a specific piece of extracted content c, the placeholder is filled through a function:

$$\text{rule}(p, c) \tag{4.23}$$

Following, the ruleset for each placeholder given the content used to complete it is defined:

- The placeholder <*speed*> is simply directly filled with the average speed extracted content $s_{\mathcal{G}^c}$ of the related event $\mathcal{G}^c$ such that:

$$\text{rule}(\textit{<speed>}, s_{\mathcal{G}^c}) = \text{“}s_{\mathcal{G}^c}\text{”} \tag{4.24}$$

  As an example, the <*speed*> placeholder of the template *“A <label> was predicted <location main> <day>, with an average speed of <speed> km/h <time>.”* given $s_{\mathcal{G}^c} = 42.47$ is filled as *“A <label> was predicted <location main> <day>, with an average speed of **42.47** km/h <time>.”*

- The placeholder <*label*>, similarly for what concerns <*speed*> is directly filled with the kind of event $k_{\mathcal{G}^c}$ of the related event $\mathcal{G}^c$ such that:

$$\text{rule}(\textit{<label>}, k_{\mathcal{G}^c}) = \text{“}k_{\mathcal{G}^c}\text{”} \tag{4.25}$$

  For instance, the <*label*> placeholder of the template *“A <label> was predicted <location main> <day>, with an average speed of <speed> km/h <time>.”* given $k_{\mathcal{G}^c}$ = *severe congestion* is filled as *“A **severe congestion** was predicted <location main> <day>, with an average speed of <speed> km/h <time>.”*

- The placeholder <*location main*> is filled by firstly extracting the street $l_{\text{main}} \in l_{\mathcal{G}^c}$ with the largest list of kilometrages for event $\mathcal{G}^c$. Afterwards, the set of kilometrages $\{km_1, km_2, ...\}$ is sorted in ascending order. If the set of kilometrages contains just one element km_1, the placeholder is filled as follows:

$$\text{rule}(\textit{<location main>}, l_{\mathcal{G}^c}) = \text{“}l_{\text{main}} \text{ at km } km_1\text{”} \tag{4.26}$$

  On the other hand, If the set of kilometrages is composed of m elements $\{km_1, km_2, ..., km_m\}$, the placeholder is filled by separating the sorted kms by a comma while “AND-ing” the second last and the last as follows:

$$\text{rule}(\textit{<location main>}, l_{\mathcal{G}^c}) = \text{“}l_{\text{main}} \text{ at kms } km_1,\ km_2,\ [...] \text{ and } km_m\text{”} \tag{4.27}$$

  Furthermore, if the context of the template requires it, the preposition “on” is added at the beginning of the filled placeholder. For instance, the <*location main*> placeholder of the template *“A <label> was predicted <location main> <day>, with an average speed of <speed> km/h <time>.”* given main street l_{main} = *Golden State Freeway* with kilometrages $\{10, 11\}$ is filled as *"A <label> was predicted* ***on Golden State Freeway at kms 10 and 11*** *<day>, with an average speed of <speed> km/h <time>."*

- Placeholders *<location secondary>* are completed by getting a sentence for each street in $l_{\mathcal{G}^c} \setminus l_{\text{main}}$ excluding the main one l_{main} as per Equations 4.26 or 4.27. Afterwards, each obtained sentence $\{s_1, ...s_m\}$ is filled by separating them by a comma while "AND-ing" the second last s_{m-1} and the last s_m as follows:

$$\text{rule}(\textit{<location secondary>}, l_{\mathcal{G}^c}) = \text{“}s_1\text{, }s_2\text{, [...] and }s_m\text{”} \tag{4.28}$$

As an example, the *<location secondary>* placeholder of the template *"The <label> also affected <location secondary>."* given secondary street information $l_{\text{secondary}} = \{\textit{Golden State Freeway}\}$ with kilometrages $\{10\}$ is filled as *"The <label> also affected* ***Golden State Freeway at km 10****."*.

- The placeholder *<day>* for event $\mathcal{G}^c$ is filled through a series of processes. Firstly, the extracted days $d_{\mathcal{G}^c} = \{d_1, ..., d_m\}$ of event $\mathcal{G}^c$ are sorted chronologically. Then the date date_{d_i} and day of the week week-day_{d_i} are extracted for each day $d_i \in d_{\mathcal{G}^c}$. Lastly, if $d_{\mathcal{G}^c}$ contains just one day d_1, the placeholder is filled as follows:

$$\text{rule}(\textit{<day>}, d_{\mathcal{G}^c}) = \text{“on weekday}_{d_1}\text{, date}_{d_1}\text{”} \tag{4.29}$$

Otherwise, if $d_{\mathcal{G}^c}$ contains more days, where the first day is d_1 and the last is d_m, the placeholder is filled as:

$$\text{rule}(\textit{<day>}, d_{\mathcal{G}^c}) = \text{“from weekday}_{d_1}\text{, date}_{d_1}\text{ to weekday}_{d_m}\text{, date}_{d_m}\text{”} \tag{4.30}$$

As an example, the *<day>* placeholder of the template *"A <label> was* *predicted <location main> <day>, with an average speed of <speed> km/h <time>."* given a days content $d_{\mathcal{G}^c} = \{\{\text{weekday} : \textit{Thursday}, \text{date} : \textit{05/06/2012}\}\}$ is filled as *"A <label> was predicted <location main>* ***on Thursday, 05/06/2012****, with an average speed of <speed> km/h <time>."*

- The placeholder <*time*> of event $\mathcal{G}^c$ is filled by the times denoting the beginning t_{start} and the end t_{end} of the event $t_{\hat{\mathcal{G}}^c}$, but it also considers the information of the days of the event $d_{\mathcal{G}^c}$. In particular, if $d_{\mathcal{G}^c}$ contains just one day and $t_{\text{start}} = t_{\text{end}}$, meaning that the event started and ended at the same discretized timestep, the placeholder is filled as:

$$\text{rule}(\textit{<time>}, t_{\mathcal{G}^c}, d_{\mathcal{G}^c}) = \text{“at } t_{\text{start}}\text{”} \tag{4.31}$$

  Contrarily, if $d_{\mathcal{G}^c}$ contains more days or $t_{\text{start}} \neq t_{\text{end}}$, the event has taken place across multiple timesteps and the placeholder is filled as:

$$\text{rule}(\textit{<time>}, t_{\mathcal{G}^c}, d_{\mathcal{G}^c}) = \text{“from } t_{\text{start}} \text{ to } t_{\text{end}}\text{”} \tag{4.32}$$

  For instance, the <*time*> placeholder of the template *"A <label> was predicted <location main> <day>, with an average speed of <speed> km/h <time>."* given a time content $t_{\mathcal{G}^c} = (09{:}45, 10{:}35)$ and a days content $d_{\mathcal{G}^c} = \{\{\text{weekday} : \textit{Thursday}, \text{date} : \textit{05/06/2012}\}\}$ is filled as *"A <label> was predicted <location main> <day>, with an average speed of <speed> km/h* ***from 09:45 to 10:35****."*

- Placeholders <*causes*> given the events $\tilde{\mathcal{G}}^1, ..., \tilde{\mathcal{G}}^C$ that explain the predicted event $\hat{\mathcal{G}}^*$ are filled by extracting firstly the list of labels $k_{\tilde{\mathcal{G}}^c}$ of each input event $\tilde{\mathcal{G}}^c$, defined as K. For simplicity in the verbal translation, *severe congestions* in the extracted list K are renamed as *congestions*. Then the translation of the placeholder in the template is applied with the following rule:

$$\text{rule}(<\!\textit{causes}\!>, K) =$$
$$= \begin{cases} \textit{“an unknown reason”}, & \text{if } |K^{\mathrm{c}}| = 0 \wedge |K^{\mathrm{f}}| = 0 \\ \textit{“a free flow”}, & \text{if } |K^{\mathrm{c}}| = 0 \wedge |K^{\mathrm{f}}| = 1 \\ \textit{“a congestion”}, & \text{if } |K^{\mathrm{c}}| = 1 \wedge |K^{\mathrm{f}}| = 0 \\ \textit{“a congestion and} \\ \textit{a free flow”}, & \text{if } |K^{\mathrm{c}}| = 1 \wedge |K^{\mathrm{f}}| = 1 \\ \textit{“a series of free flows”}, & \text{if } |K^{\mathrm{c}}| = 0 \wedge |K^{\mathrm{f}}| > 1 \\ \textit{“a series of congestions”}, & \text{if } |K^{\mathrm{c}}| > 1 \wedge |K^{\mathrm{f}}| > 0 \\ \textit{“a series of free flows} \\ \textit{and a congestion”}, & \text{if } |K^{\mathrm{c}}| = 1 \wedge |K^{\mathrm{f}}| > 1 \\ \textit{“a series of congestions} \\ \textit{and a free flow”}, & \text{if } |K^{\mathrm{c}}| > 1 \wedge |K^{\mathrm{f}}| = 1 \\ \textit{“a series of congestions} \\ \textit{and free flows”}, & \text{otherwise} \end{cases} \tag{4.33}$$

where K^{f} is the sublist of K containing just free flows labels and K^{c} is the sublist containing just congestion labels. For instance, the placeholder <*causes*> in the template *“This was caused by <causes>.”* given $K = [\textit{congestion,free flow}]$ is filled as *“This was caused by **a congestion and a free flow**.”*.

Note, that an extreme scenario, in which no input contributing events are present is considered, by setting the cause of the prediction as *“an unknown reason”*. This may happen in the case that the input data from which the important subgraph is extracted presents a traffic-network composed solely of missing speed information. In this case, the important subgraph is empty $\tilde{\mathcal{G}} = \emptyset$, thus, no reason behind the prediction can be extracted.

The final verbal translation is finally enriched with additional information in order to make the narrative more fluent and coherent:

- If the event to explain is explained by a single input event $\tilde{\mathcal{G}}$, the paragraph that explains $\tilde{\mathcal{G}}$ is edited by substituting the first preposition “a” with “the” to reflect the fact that just one event led to the prediction. As an example, the paragraph *“**A** contributing free flow manifested on Ventura Freeway [...]”* is modified with *“**The** contributing free flow manifested on Ventura Freeway [...]”*
- If more than one input event $\tilde{\mathcal{G}}^1, ...\tilde{\mathcal{G}}^c$, contributed to the predicted event, additional connectors are placed at the beginning of each paragraph describing each $\tilde{\mathcal{G}}^c$. If $\tilde{\mathcal{G}}^c$ is the first described input event, the paragraph will start with a connector such as *“Firstly,”* or *“Initially,”*, while if it is the last described event connectors such as *“Finally,”* or *“Lastly,”* are added at the beginning. For the other in-between events other connectors are integrated at the beginning of the paragraph such as *“Next,”* or *“Subsequently,”*. For instance, if the paragraph *A contributing free flow manifested on Ventura Freeway [...]”* provides a description of the first contributing event, it is modified as *“**Initially,** a contributing free flow manifested on Ventura Freeway [...]”*.
- If a paragraph describes an input event $\tilde{\mathcal{G}}^c$ which label $k_{\tilde{\mathcal{G}}^c}$ has already been seen on previously described input events, a connector such as *“another”* or *“a further”* will be added before the description of the event $k_{\tilde{\mathcal{G}}^c}$ in the paragraph. This is applied to reinforce the fact that there are more than one occurring events of the same kind. For example, if the paragraph *A contributing free flow manifested on Ventura Freeway [...]”* is given and in previous paragraph free flows have already been described, it is modified as *“**A further** contributing free flow manifested on Ventura Freeway [...]”*.

- Similarly to the previous point, if a paragraph describes an event $\tilde{\mathcal{G}}^c$ that occurred in the same location of a previously described input event, a connector such as *"again"* or *"once more"* will be added before the description of the location in the paragraph. If the paragraph *A contributing free flow manifested on Ventura Freeway [...]"* is given and previous paragraph describes an event occurring again on Ventura Freeway, it is modified as *"A contributing free flow manifested on, **again,** Ventura Freeway [...]"*.

- The translation of the days information $d_{\tilde{\mathcal{G}}^c}$ of an event $\tilde{\mathcal{G}}^c$ is omitted or modified according to the day information $d_{\hat{\mathcal{G}}^*}$ of the event to explain $\hat{\mathcal{G}}^*$. If the day information in $d_{\tilde{\mathcal{G}}^c}$ is equivalent to $d_{\hat{\mathcal{G}}^*}$, meaning that the input event $\tilde{\mathcal{G}}^c$ occurred in the same days as $\tilde{\mathcal{G}}^*$, then the description of the day in the paragraph illustrating $\tilde{\mathcal{G}}^c$ is completely omitted, as it is redundant and implicit. As an example, if the paragraph *"A contributing free flow manifested [...] occurring from 08:45 to 09:40 on Thursday, 05/06/2012 with an average speed [...]"* is given and the day information is the same as the one of the event to explain, it is modified as *"A contributing free flow manifested [...] occurring from 08:45 to 09:40 with an average speed [...]"*. If the day information in $d_{\tilde{\mathcal{G}}^c}$ is composed of more days the following considerations are applied:

  - If $d_{\hat{\mathcal{G}}^*}$ has just one day and the last day of $d_{\tilde{\mathcal{G}}^c}$ is the day of $d_{\hat{\mathcal{G}}^*}$ and the first day of $d_{\tilde{\mathcal{G}}^c}$ is the day before the day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"from the previous to the same day"*
  - If $d_{\hat{\mathcal{G}}^*}$ has more days and the last day of $d_{\tilde{\mathcal{G}}^c}$ is the first day of $d_{\hat{\mathcal{G}}^*}$ and the first day of $d_{\tilde{\mathcal{G}}^c}$ is the day before the first day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"from the previous to the first day"*
  - If $d_{\hat{\mathcal{G}}^*}$ has just one day and the last day of $d_{\tilde{\mathcal{G}}^c}$ is the day of $d_{\hat{\mathcal{G}}^*}$

and the first day of $d_{\tilde{\mathcal{G}}^c}$, d_1 is not the day before the day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"from d_1 to the same day"*

– If $d_{\hat{\mathcal{G}}^*}$ has more days and the last day of $d_{\tilde{\mathcal{G}}^c}$ is the first day of $d_{\hat{\mathcal{G}}^*}$ and the first day of $d_{\tilde{\mathcal{G}}^c}$, d_1 is not the day before the first day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"from d_1 to the first day"*

– If the last day of $d_{\tilde{\mathcal{G}}^c}$ is the day d_1 before the first day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"from d_1 to the previous day"*

If $d_{\tilde{\mathcal{G}}^c}$ is composed of one day and $d_{\hat{\mathcal{G}}^*} \neq d_{\tilde{\mathcal{G}}^c}$:

– If the day of $d_{\tilde{\mathcal{G}}^c}$ is the day before the first day of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"on the previous day"*.

– If $d_{\hat{\mathcal{G}}^*}$ has more days and the day of $d_{\tilde{\mathcal{G}}^c}$ is the same as the first one of $d_{\hat{\mathcal{G}}^*}$, the day information in the paragraph describing $\tilde{\mathcal{G}}^c$ is substituted with: *"on the first day"*.

Chapter 5

Results

5.1 STGNN Results

In this section, the training and evaluation outcomes of the STGNNs Φ on the METR-LA and PEMS-BAY datasets are explored. The analysis first focuses on the training progression, showing the prediction results of the models during the various epochs. Then, the models' performance results on the test set are illustrated, focusing on their forecasting power considering different horizons of time in the future and considering different events separately. Finally, a qualitative analysis of the predictions is illustrated for both datasets.

5.1.1 Training Results

For what concerns the training outcomes for METR-LA and PEMS-BAY, as it can be seen in Figures 5.1 and 5.2, in both cases no clear signs of overfitting are observed during the whole process for both MAE, RMSE and MAPE. It is interesting to point out how the validation results on METR-LA exceed the performances of the training.

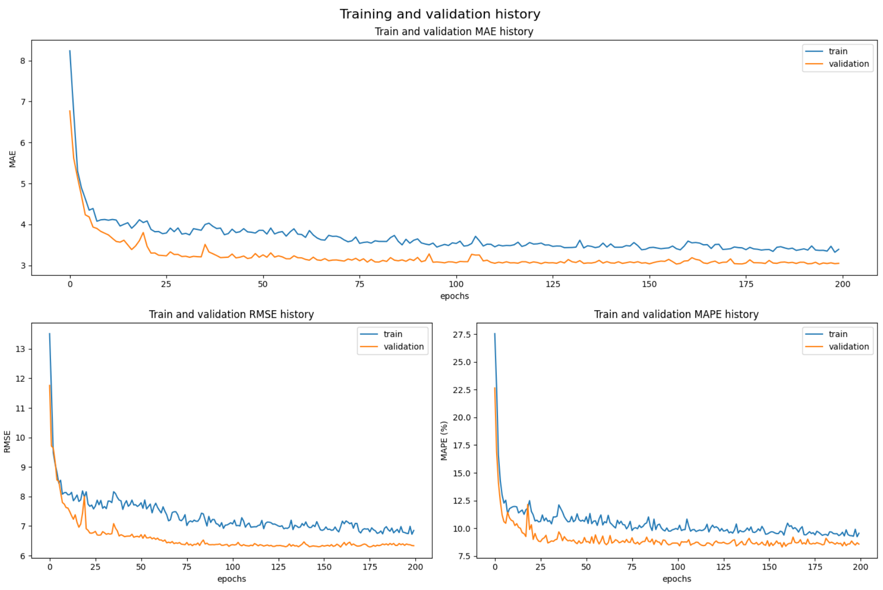


Figure 5.1: Training process outcome on the METR-LA dataset.

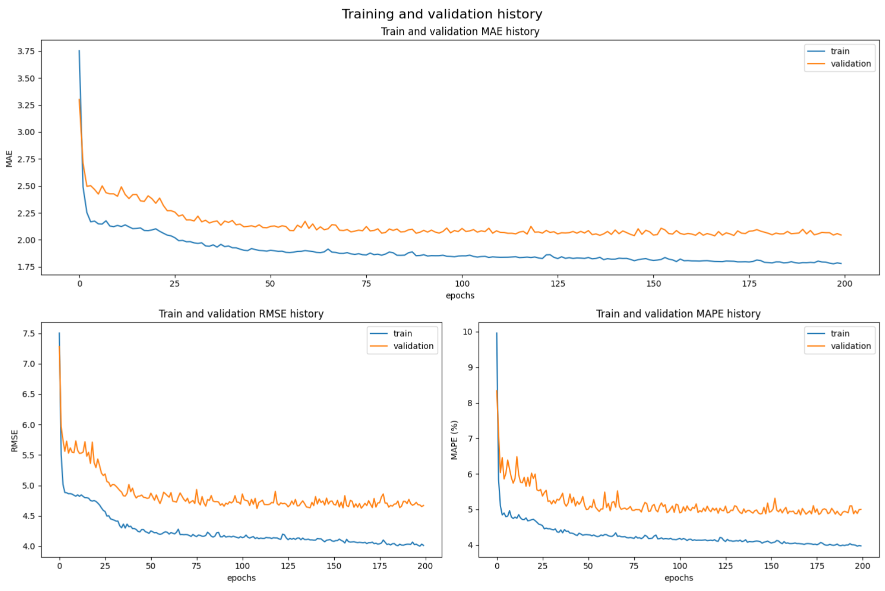


Figure 5.2: Training process outcome on the PEMS-BAY dataset.

5.1.2 Test Results

Table 5.1 presents the average prediction performance across the test sets for both METR-LA and PEMS-BAY datasets, measured in terms of *Mean Absolute Error (MAE)*, *Root Mean Squared Error (RMSE)*, and *Mean Absolute Percentage Error (MAPE)* for forecasting at intervals of 15 minutes, 30 minutes, and 60 minutes ahead.

	Next 15 minutes			Next 30 minutes			Next 60 minutes		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
METR-LA	2.92	5.55	7.8%	3.27	6.4	9.05%	3.86	7.74	11.5%
PEMS-BAY	1.18	2.48	2.43%	1.49	3.36	3.26%	1.92	4.43	4.54%

Table 5.1: STGNNs prediction results considering different time horizons.

As anticipated, the prediction accuracy tends to decrease with the increase in the forecasting horizon for both datasets. However, the errors remain within reasonable limits and affect only a small portion of the predictions as expressed by MAPE. Notably, the prediction outcomes for METR-LA display a notably inferior performance compared to PEMS-BAY. This discrepancy can be attributed to two primary factors. Firstly, METR-LA involves a notably smaller number of instances in its dataset compared to PEMS-BAY, leading to a training process with fewer observations. Secondly, and more significantly, METR-LA exhibits a substantially higher rate of missing information compared to PEMS-BAY. This higher degree of missing data contributes to increased uncertainty in the predictions, consequently impacting their accuracy.

Table 5.2 showcases the average prediction performance on the test sets for METR-LA and PEMS-BAY datasets, evaluating the MAE, RMSE, and MAPE across different speed predictions, categorized into severe congestions, congestions, and free-flow events. These event distinctions are established using the thresholds outlined in section 4.4.1.

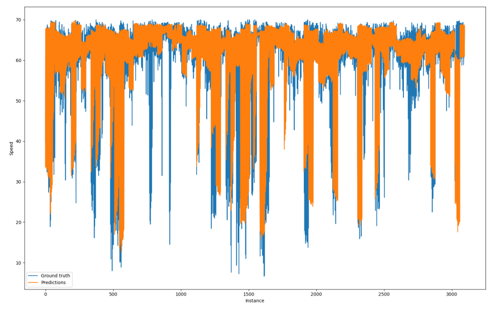
	Severe congestions			Congestions			Free flows		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
METR-LA	12.2	17.9	65.1%	5.21	7.73	10.6%	1.96	4.16	3.02%
PEMS-BAY	10.6	15.8	49.1%	4.46	7.05	9.23%	1.07	1.95	1.64%

Table 5.2: STGNNs prediction results considering different event kinds.

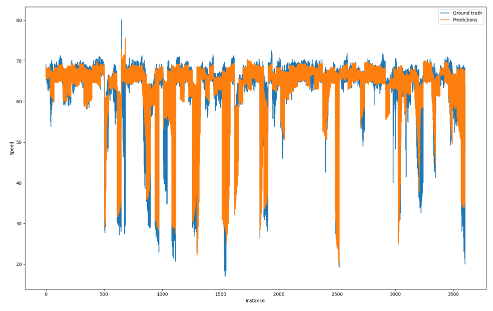
The STGNN performance reveals considerable difficulty in accurately predicting severe congestion for both datasets, outputting unreliable predictions in these instances. Predictions for congestion demonstrate acceptable outcomes, whereas the model shows notably successful performance in predicting free-flow events. This trend aligns with the data distribution noted in section 4.2.1, where the dataset exhibits an imbalance, featuring more instances of free-flow scenarios compared to congestion. The model's training methodology remains consistent with the approach detailed in the original paper [42], without specific handling of the imbalance in node speeds during the training process. Consequently, the model demonstrates inferior performance in predicting instances associated with lower speeds, describing instances of severe congestion.

5.1.3 Qualitative Analysis

Figure 5.3 displays a comparison between the ground truth speeds and the predicted speeds in the test dataset for 3 randomly selected nodes, each having 100 randomly chosen observations. The visualization highlights the challenges encountered in predicting severe congestion, consistent with the observations from Table 5.2. Despite these challenges, the model exhibits commendable performance. However, it struggles with certain speed peaks and troughs, indicating difficulty in accurate predictions for these specific instances.



(a) METR-LA predictions.



(b) PEMS-BAY predictions.

Figure 5.3: Visualization of the speed predictions by the STGNNs with respect to ground truths for randomly selected observations.

5.2 Explainer Results

In this section, the outcomes of the Explainer ψ are quantified and analysed on the test sets $\mathcal{D}^*_{\text{test}}$ of both METR-LA and PEMS-BAY in terms of *Fidelity*$^-$, *Fidelity*$^+$, *Sparsity* and *average prediction time*. Moreover, a visual example of an extracted important subgraph $\tilde{\mathcal{G}}$ by ψ divided in the event it is composed of concerning an output event $\hat{\mathcal{G}}^*$ is shown for both datasets.

5.2.1 Test Results

Table 5.3 expresses the *Fidelity*$^-$ results on the explanations of both datasets using the MAE error (*MAE*$^-$), the RMSE error (*RMSE*$^-$) and the MAPE error (*MAPE*$^-$). It illustrates the result averaged for the different kinds of predicted events and as their total average value across the whole dataset.

	Severe congestions			**Congestions**			**Free flows**		
	MAE$^-$	**RMSE$^-$**	**MAPE$^-$**	**MAE$^-$**	**RMSE$^-$**	**MAPE$^-$**	**MAE$^-$**	**RMSE$^-$**	**MAPE$^-$**
METR-LA	3.21	3.82	14.1%	1.65	2.06	3.45%	0.713	0.881	1.09%
PEMS-BAY	12.5	12.9	59.3%	2.65	3.13	5.24%	1.97	2.15	2.93%

	Total Average		
	MAE$^-$	**RMSE$^-$**	**MAPE$^-$**
METR-LA	1.84	2.24	6.14%
PEMS-BAY	5.67	6.02	22.4%

Table 5.3: Fidelity$^-$ explanation results considering different event kinds.

The explanations provided by ψ reveal the challenges the STGNNs Φ face in accurately predicting severe instances of congestion as the Fidelity$^-$ results, where higher values indicate poorer performance, are notably higher for these events in both datasets. While the Fidelity$^-$ results in METR-LA for severe congestions are still acceptable, they are significantly high for what concerns this kind of predicted event in PEMS-BAY. This implies that the explainer struggles to assemble a subgraph comprehensive enough to justify the predictions of severe congestions by Φ in this latter dataset. The results also suggest that Φ is not ideally trained to forecast severe congestion, potentially indicating its lack of internal consideration of traffic laws and patterns specific to these events. Conversely, the results of Fidelity$^-$ for other events and considered as average are generally low, indicating that the explanations are sufficient as they are enough for Φ to correctly predict the outcome event.

Table 5.4 showcases the *Fidelity*$^+$ outcomes for both datasets, encompassing as error metrics the MAE (*MAE*$^+$), RMSE (*RMSE*$^+$) and MAPE (*MAPE*$^+$). These metrics are shown as their average value for the distinct predicted events and their collective average across the entire dataset.

	Severe congestions			**Congestions**			**Free flows**		
	MAE$^+$	**RMSE$^+$**	**MAPE$^+$**	**MAE$^+$**	**RMSE$^+$**	**MAPE$^+$**	**MAE$^+$**	**RMSE$^+$**	**MAPE$^+$**
METR-LA	19.1	20.9	86.2%	9.68	10.6	20.4%	2.96	3.5	4.54%
PEMS-BAY	17.2	19.1	73.1%	9.6	10.7	19.2%	41.8	46	63.2%

	Total Average		
	MAE$^+$	**RMSE$^+$**	**MAPE$^+$**
METR-LA	10.6	11.7	37.1%
PEMS-BAY	22.8	25.3	51.8%

Table 5.4: Fidelity$^+$ explanation results considering different event kinds.

For both datasets, the Fidelity$^+$ scores are consistently higher than Fidelity$^-$ across all event types. This indicates that the extracted important subgraphs play a necessary role in explaining the predictions made by the STGNNs (Φ). In the case of METR-LA, Fidelity$^+$ echoes the observations from Fidelity$^-$: predicting severe congestions poses a greater challenge, suggesting that the STGNNs are better attuned to predicting free-flows or moderate congestions than severe congestions. Consequently, the subgraphs associated with severe congestions achieve higher Fidelity$^+$ scores, underscoring their necessity for the model to accurately predict these events. However, these subgraphs appear to be less critical when predicting congestions or free-flows, potentially due to the model's inherent bias toward these events.

Regarding PEMS-BAY, two notable observations emerge. Firstly, Fidelity$^+$ closely aligns with Fidelity$^-$ in explaining instances of severe congestion. This indicates that the explanations for these events might not be significantly more necessary than sufficient, lacking a comprehensive representation of the model's predictions. Secondly, the explanations seem crucial for predicting free-flow events, reflected in their highest Fidelity$^+$ among all event types. This could be attributed to the dataset's larger spatial node representation than METR-LA, potentially leading to confusion in predictions if the essential information provided by the important subgraph is excluded from the explanation set generated by ψ.

Table 5.5 illustrates consistently high sparsity average results for both METR-LA and PEMS-BAY, indicating that the explanations derived by ψ are concise. This conciseness is anticipated given that, in both datasets, the size $\tilde{N}$ of the important subgraphs used for explanations is relatively small, being respectively twice the size of the predicted events in METR-LA and three times the predicted events in PEMS-BAY.

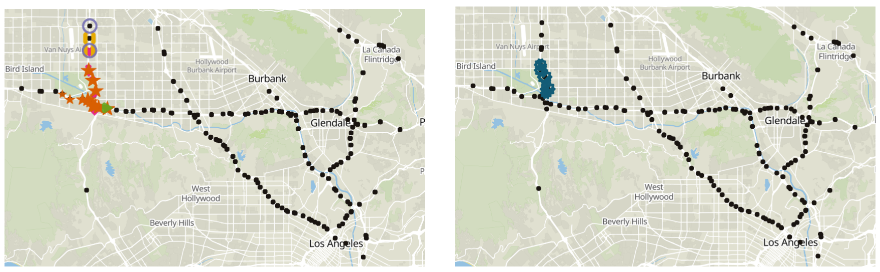
	Sparsity
METR-LA	0.985
PEMS-BAY	0.985

Table 5.5: Sparsity of the explanations.

Finally, the average time taken to compute the explanations is relatively low for both datasets: 8.88 seconds for METR-LA and 11.6 seconds for PEMS-BAY. The increased time in PEMS-BAY is attributed to the larger subgraph used for the localized search conducted by ψ. However, despite this variation, the processing time remains within acceptable limits, ensuring that users can obtain explanations within a reasonable timeframe.

5.2.2 Visual Examples of the Results

In Figure 5.4, an illustrative example showcases the extracted important subgraph, segmented into its cluster events, explaining a severe congestion occurrence near the intersection of two streets in METR-LA. The visualization highlights the succinct nature of the explanation, as it involves only a small subset of nodes compared to the entire spatial network. The explanation aligns with the notion of *localized spatial correlation* introduced in section 3.3.3, as the nodes considered in the explanation are visually spatially close to the ones of the event to explain.



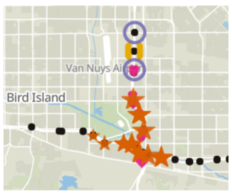
(a) Important subgraph. (b) Event to explain.

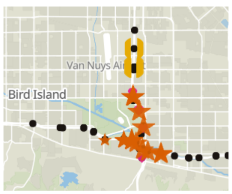
● Nodes excluded in the explanation ‖♥★○▶ Nodes of Clusters of the important subgraph ✓ Node of the event to explain

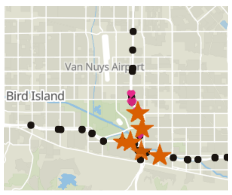
Figure 5.4: Overall visualization of an explanation of a prediction in METR-LA.

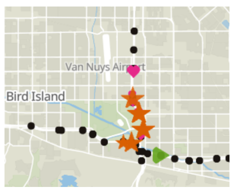
Moreover, Figure 5.5 provides a detailed depiction of the clusters within the important subgraph across time, specifically focused on explaining the event highlighted in Figure 5.4. This visualization significantly reinforces the concept of *localized spatial correlation*. As time progresses, there's a noticeable reduction in spatial distance between nodes influencing the explanation and those representing the output event being explained. This is expected, since as time approaches the event to be explained, the relevance of nearby input nodes in influencing the prediction increases. Conversely, nodes located farther away gradually contribute less to the explanation of the event.

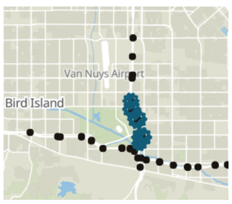
The graphical representation also illustrates the evolving nature of clusters associated with the event. Two primary clusters, namely a congestion (★) and a severe congestion (♥), consistently contribute across the entire observed period within the central region of the important subgraph. Conversely, a smaller cluster denoting a free-flow in the northern zone (○) contributes to explaining the prediction just in the first three timesteps. Similarly, another congestion in the northernmost area (‖), evolving within the first 6 timesteps, demonstrates contributions to predict the event. Additionally, a small congestion (▶) emerges in the last three timesteps, situated in the eastern area of the subgraph. For what concerns the event to explain, it is a severe congestion that develops constantly along the output period, involving the same set of nodes (✸) just above the intersection of the two streets.

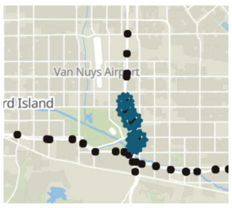

(a) Input timesteps 0-2.

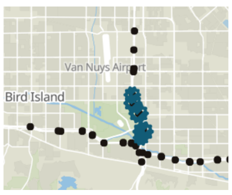

(b) Input timesteps 3-5.

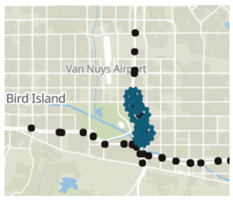

(c) Input timesteps 6-8.


(d) Input timesteps 9-11.


(e) Output timesteps 0-2.


(f) Output timesteps 3-5.


(g) Output timesteps 6-8.

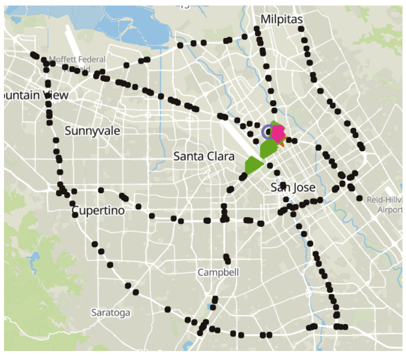

(h) Output timesteps 9-11.

● Nodes excluded in the explanation
▮♥★○▶ Nodes of Clusters of the important subgraph
✸ Node of the event to explain

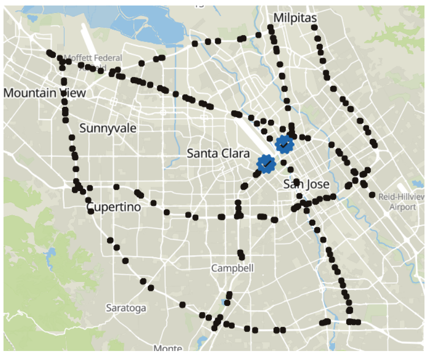
Figure 5.5: Detailed visualization of an explanation of a prediction in METR-LA.

In Figure 5.6, an example illustrates the extracted important subgraph divided into cluster events explaining congestion on a major highway in PEMS-BAY. Similarly to the previously observed explanation, this visualization emphasizes the concise nature of the subgraph relative to the complete spatial network.

The detailed explanation depicted in Figure 5.7 reveals that major congestion (▶) developing for the whole period along the street where the congested output event took place was the primary influence on the prediction. Additionally, other factors contributing to this prediction include minor congestion (○), a small free-flow (★), and a limited severe congestion (♥) occurring in the northern section of the important subgraph across the period.



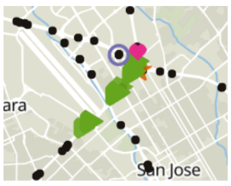
(a) Important subgraph.



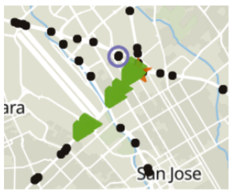
(b) Event to explain.

● Nodes excluded in the explanation ★ Nodes of Clusters of the important subgraph ✓ Node of the event to explain

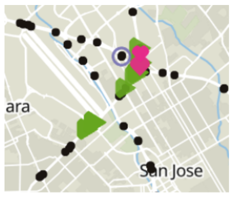
Figure 5.6: Overall visualization of an explanation of a prediction in PEMS-BAY.



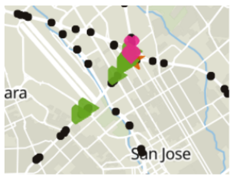
(a) Input timesteps 0-2.



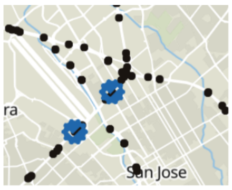
(b) Input timesteps 3-5.



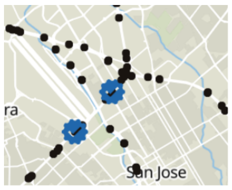
(c) Input timesteps 6-8.



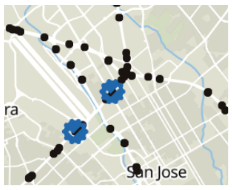
(d) Input timesteps 9-11.



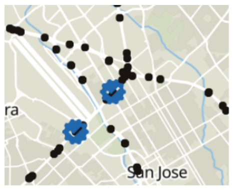
(e) Output timesteps 0-2.



(f) Output timesteps 3-5.



(g) Output timesteps 6-8.



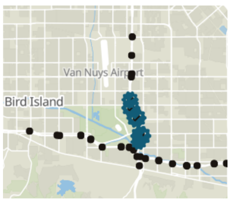
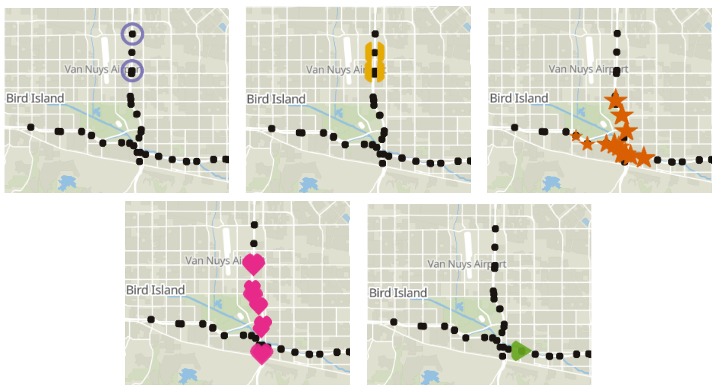
(h) Output timesteps 9-11.

● Nodes excluded in the explanation ★ Nodes of Clusters of the important subgraph ✓ Node of the event to explain

Figure 5.7: Detailed visualization of an explanation of a prediction in PEMS-BAY.

5.3 Verbal Translation Examples

Table 5.6 illustrates the verbal translation result for the explanation of the prediction in the METR-LA dataset shown in Figure 5.4 of section 5.2.2. The verbal description aims at explaining the causes that led to the prediction of severe congestion in San Diego Freeway on Wednesday, 04/06/2012, occurring from 07:45 to 08:40. Images between brackets (·) in the verbal narrative are added to guide the user to visualize the event in the graphical support of the translation.

Event to explain	Causes
	

Verbal translation

"San Diego Freeway at kms 11, 12, 13 and 14 was forecasted to see severe congestion (✱) on Wednesday, 04/06/2012, with an average speed of 24.04 km/h from 07:45 to 08:40. This occurred because of a series of congestions (⏸, ★, ♥, ▶) and a free flow (○).

To start, a contributing free flow (○) materialized, at 107.75 km/h, on San Diego Freeway at kms 8 and 10 from 06:45 to 06:50.

Following this, contributing congestion (⏸) occurred, averaging at a speed of 82.83 km/h, on, another time, San Diego Freeway at kms 9 and 10 from 06:45 to 07:10.

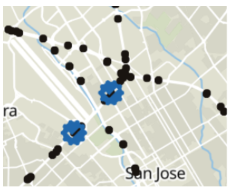
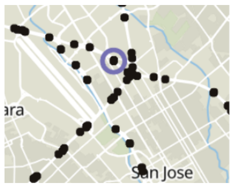
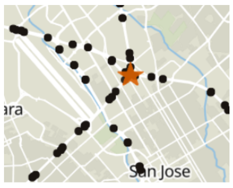
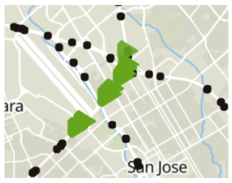
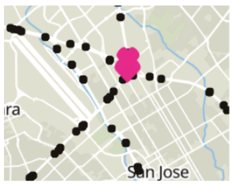
Afterwards, a new contributing congestion (★) took place, with an average speed of 88.65 km/h, on Ventura Freeway at kms 27, 28, 29, 30 and 31 occurring from 06:45 to 07:40. The congestion also took place on, once again, San Diego Freeway at kms 11, 12 and 13.

After this, a contributing severe congestion (♥) happened on, yet again, San Diego Freeway at kms 10, 11, 12, 13, 14 and 15 from 06:45 to 07:40 with an average speed of 19.70 km/h.

To conclude, yet a further contributing congestion (▶) occurred at 07:30 on, again, Ventura Freeway at km 27 with an average speed of 72.62 km/h."

Table 5.6: Verbal translation of an explanation of a prediction in METR-LA aided by graphical support.

Similarly, Table 5.7 presents the verbal translation outcome for the prediction explanation in the PEMS-BAY dataset, specifically for the congestion event illustrated in Figure 5.6 of section 5.2.2. This verbal narrative aims to show the factors contributing to the forecasted congestion on I 880, occurring on Saturday, June 21st, 2017, from 16:20 to 17:15. Supporting images enclosed in parentheses (·) have been included in the verbal description to aid users in visualizing the event in the graphical context of the translation.

Event to explain	Causes
	   

Verbal translation
"A congestion (✱) was predicted on I 880 at km 61 on Saturday, 21/06/2017, with an average speed of 83.19 km/h from 16:20 to 17:15. This was a result of a series of congestions (○,) and a free flow (★, ▶, ♥).
Firstly, contributing congestion (○) took place on Bayshore Freeway at km 62, with an average speed of 68.93 km/h, from 15:20 to 15:55.
Next, a contributing free flow (★) happened, at 112.06 km/h, on, again, Bayshore Freeway at km 63 from 15:20 to 16:10.
After this, an extra contributing congestion (▶) materialized, with an average speed of 91.86 km/h, on I 880 at kms 60 and 61 occurring from 15:20 to 16:15.
Lastly, a contributing severe congestion (♥) happened on, another time, I 880 at kms 60 and 61, occurring from 15:25 to 16:05 with an average speed of 52.03 km/h."

Table 5.7: Verbal translation of an explanation of a prediction in PEMS-BAY aided by graphical support.

Chapter 6

Conclusions

This thesis aimed to create an eXplainable AI (XAI) system tailored for non-technical users to interpret short-term speed predictions in traffic networks generated by a Spatio-Temporal Graph Neural Network (STGNN). Its primary focus was on extrapolating reasons behind traffic congestion and free flow by extracting a significant subgraph from the input network and converting these explanations into user-friendly verbal narratives.

To achieve this, the system utilized an explainer designed as a transparent, easily understandable post-hoc model inspection technique. Initially, the explainer employs a global heuristic based on traffic-domain laws to select a subset of highly correlated input data related to the predicted event. It then refined this subset using a localized Monte Carlo Tree Search (MCTS) to identify instance-specific input data influencing the prediction, culminating in narrative explanations through a template-based approach.

The results showcased the explainer's capacity to capture concise, meaningful explanations, particularly effective in describing events aligning with the STGNN's training, such as moderate congestions and free-flow scenarios. The verbal narratives provided users with a comprehensible guide to understanding the sequence of events leading to the STGNN predictions. This underscored the potential of integrating domain-related laws in constructing explanations for opaque models, hinting at their internal reasoning reflecting these fundamental laws.

However, some limitations surfaced primarily in the explainer's efficacy in explaining rare events, notably severe congestions, especially concerning the PEMS-BAY dataset. This observation suggests that the STGNNs are not properly well-trained in predicting severe congestion. Repurposing the explainer for models more aligned with data distribution and capable of handling such events could yield a more comprehensive understanding of their explanatory power. The explainer's dependence on domain-related laws presents another limitation in the fact that these rules need manual adjustments when new attributes are added to the input data or if there are alterations in the prediction task. This confines the explainer's expertise to the realm of traffic prediction, making it less adaptable to diverse scenarios and impeding its generalizability beyond this specific task.

Looking ahead, future work could explore the explainer's broader applicability. This includes extending its usage beyond traffic speed prediction to encompass various traffic attributes and characteristics. Investigating how the explainer performs across diverse domains utilizing different heuristic laws would enrich its applicability and reliability. Additionally, further assessing the explainer's model-agnostic nature by applying its explanation process to models beyond STGNNs is crucial for understanding its generalizability.

Acknowledgements

I am deeply grateful to my supervisors, Roberta Calegari and Panchamy Krishnakumari, for their support and guidance throughout the development of my work. Additionally, I extend my heartfelt thanks to TU Delft, along with the staff, researchers and friends I met there, for granting me the invaluable opportunity to pursue my thesis studies abroad and giving me diverse research and life perspectives. To my friends and parents, I am profoundly thankful for their support throughout my master's studies.

Bibliography

[1] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin. Deep learning on traffic prediction: methods, analysis, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4927–4943, June 2022. DOI: 10.1109/tits.2021.3054840. URL: https://doi.org/10.1109%2Ftits.2021.3054840.

[2] J. Hoque, G. Erhardt, D. Schmitt, M. Chen, A. Chaudhary, M. Wachs, and R. Souleyrette. The changing accuracy of traffic forecasts. *Transportation*, 49:1–22, April 2022. DOI: 10.1007/s11116-021-10182-8.

[3] K. L. Soon, R. K. C. Chan, J. M.-Y. Lim, and R. Parthiban. Short-term traffic forecasting model: prevailing trends and guidelines. *Transportation Safety and Environment*, 5(3):tdac058, December 2022. ISSN: 2631-4428. DOI: 10.1093/tse/tdac058. eprint: https://academic.oup.com/tse/article-pdf/5/3/tdac058/50695687/tdac058.pdf. URL: https://doi.org/10.1093/tse/tdac058.

[4] H. Huang, J. Chen, R. Sun, and S. Wang. Short-term traffic prediction based on time series decomposition. *Physica A: Statistical Mechanics and its Applications*, 585:126441, 2022. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2021.126441. URL: https://www.sciencedirect.com/science/article/pii/S0378437121007147.

[5] B. Sharma, S. Kumar, P. Tiwari, P. Yadav, and M. I. Nezhurina. Ann based short-term traffic flow forecasting in undivided two lane highway. *Journal of Big Data*, 5, 2018. URL: `https://api.semanticscholar.org/CorpusID:256400170`.

[6] W. Jiang and J. Luo. Graph neural network for traffic forecasting: a survey. *Expert Systems with Applications*, 207:117921, November 2022. DOI: `10.1016/j.eswa.2022.117921`. URL: `https://doi.org/10.1016%2Fj.eswa.2022.117921`.

[7] J. Tang, J. Liang, F. Liu, J. Hao, and Y. Wang. Multi-community passenger demand prediction at region level based on spatio-temporal graph convolutional network. *Transportation Research Part C: Emerging Technologies*, 124:102951, 2021. ISSN: 0968-090X. DOI: `https://doi.org/10.1016/j.trc.2020.102951`. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X20308482`.

[8] X. Zhang, C. Huang, Y. Xu, and L. Xia. Spatial-temporal convolutional graph attention networks for citywide traffic flow forecasting. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM ’20, pages 1853–1862, Virtual Event, Ireland. Association for Computing Machinery, 2020. ISBN: 9781450368599. DOI: `10.1145/3340531.3411941`. URL: `https://doi.org/10.1145/3340531.3411941`.

[9] V. Hassija, V. Chamola, A. Mahapatra, A. Singal, D. Goel, K. Huang, S. Scardapane, I. Spinelli, M. Mahmud, and A. Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, August 2023. DOI: `10.1007/s12559-023-10179-8`.

[10] S. A. and S. R. A systematic review of explainable artificial intelligence models and applications: recent developments and future trends.

Decision Analytics Journal, 7:100230, 2023. ISSN: 2772-6622. DOI: `https://doi.org/10.1016/j.dajour.2023.100230`. URL: `https://www.sciencedirect.com/science/article/pii/S277266222300070X`.

[11] A. Barredo-Arrieta, I. Laña, and J. Del Ser. What lies beneath: a note on the explainability of black-box machine learning models for road traffic forecasting. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2232–2237, 2019. DOI: `10.1109/ITSC.2019.8916985`.

[12] M. Du, N. Liu, and X. Hu. Techniques for interpretable machine learning, 2019. arXiv: `1808.00033 [cs.LG]`.

[13] N. Fouladgar and K. Främling. Xai-p-t: a brief review of explainable artificial intelligence from practice to theory, 2020. arXiv: `2012.09636 [cs.AI]`.

[14] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera. Explainable artificial intelligence (xai): what we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99:101805, 2023. ISSN: 1566-2535. DOI: `https://doi.org/10.1016/j.inffus.2023.101805`. URL: `https://www.sciencedirect.com/science/article/pii/S1566253523001148`.

[15] T. Miller. Explanation in artificial intelligence: insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2018.07.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0004370218305988`.

[16] R. Spolaor. Verbal Explanations of Spatio-Temporal Graph Neural Networks for Traffic Forecasting, version 1.0.0, February 2024.

URL: https://github.com/RiccardoSpolaor/Verbal-Explanations-of-Spatio-Temporal-Graph-Neural-Networks-for-Traffic-Forecasting.

[17] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: a review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[18] X. Liu, J. Chen, and Q. Wen. A survey on graph classification and link prediction based on gnn, 2023. arXiv: 2307.00865 [cs.LG].

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017. arXiv: 1609.02907 [cs.LG].

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018. arXiv: 1710.10903 [stat.ML].

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry, 2017. arXiv: 1704.01212 [cs.LG].

[22] Y. Li, D. Yu, Z. Liu, M. Zhang, X. Gong, and L. Zhao. Graph neural network for spatiotemporal data: methods and applications, 2023. arXiv: 2306.00012 [cs.LG].

[23] J. Chen, X. Xu, Y. Wu, and H. Zheng. GC-LSTM: graph convolution embedded LSTM for dynamic link prediction. *CoRR*, abs/1812.04206, 2018. arXiv: 1812.04206. URL: http://arxiv.org/abs/1812.04206.

[24] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li. T-gcn: a temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2020. DOI: 10.1109/TITS.2019.2935152.

[25] B. Kim, J. C. Ye, and J. Kim. Learning dynamic graph representation of brain connectome with spatio-temporal attention. *CoRR*, abs/2105.13495, 2021. arXiv: 2105.13495. URL: https://arxiv.org/abs/2105.13495.

[26] A. Cherian, C. Hori, T. K. Marks, and J. L. Roux. (2.5+1)d spatio-temporal scene graphs for video question answering, 2022. arXiv: 2202.09277 [cs.CV].

[27] C. Zheng, X. Fan, C. Wang, and J. Qi. Gman: a graph multi-attention network for traffic prediction, 2019. arXiv: 1911.08415 [eess.SP].

[28] P. Li, Y. Yang, M. Pagnucco, and Y. Song. Explainability in graph neural networks: an experimental survey, 2022. arXiv: 2203.09258 [cs.LG].

[29] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, and S. Medya. A survey on explainability of graph neural networks, 2023. arXiv: 2306.01958 [cs.LG].

[30] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: generating explanations for graph neural networks, 2019. arXiv: 1903.03894 [cs.LG].

[31] M. S. Schlichtkrull, N. D. Cao, and I. Titov. Interpreting graph neural networks for nlp with differentiable edge masking, 2022. arXiv: 2010.00577 [cs.CL].

[32] M. N. Vu and M. T. Thai. Pgm-explainer: probabilistic graphical model explanations for graph neural networks, 2020. arXiv: 2010.05788 [cs.LG].

[33] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji. On explainability of graph neural networks via subgraph explorations, 2021. arXiv: 2102.05152 [cs.LG].

[34] W. He, M. N. Vu, Z. Jiang, and M. T. Thai. An explainer for temporal graph neural networks, 2022. arXiv: 2209.00807 [cs.LG].

[35] W. Xia, M. Lai, C. Shan, Y. Zhang, X. Dai, X. Li, and D. Li. Explaining temporal graph models through an explorer-navigator framework. In *The Eleventh International Conference on Learning Representations*, 2023. URL: https://openreview.net/forum?id=BR_ZhvcYbGJ.

[36] J. Tang, L. Xia, and C. Huang. Explainable spatio-temporal graph neural networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pages 2432–2441, , Birmingham, United Kingdom, Association for Computing Machinery, 2023. ISBN: 9798400701245. DOI: 10.1145/3583780.3614871. URL: https://doi.org/10.1145/3583780.3614871.

[37] A. Verdone, S. Scardapane, and M. Panella. Explainable spatio-temporal graph neural networks for multi-site photovoltaic energy production. *Applied Energy*, 353:122151, 2024. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2023.122151. URL: https://www.sciencedirect.com/science/article/pii/S0306261923015155.

[38] D. Gkatzia. Content selection in data-to-text systems: a survey, 2016. arXiv: 1610.08375 [cs.CL].

[39] C. van der Lee, E. Krahmer, and S. Wubben. Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. In E. Krahmer, A. Gatt, and M. Goudbeek, editors, *Proceedings of the 11th International Conference on Natural Language*

Generation, pages 35–45, Tilburg University, The Netherlands. Association for Computational Linguistics, November 2018. DOI: 10.18653/v1/W18-6504. URL: https://aclanthology.org/W18-6504.

[40] A. Laha, P. Jain, A. Mishra, and K. Sankaranarayanan. Scalable micro-planned generation of discourse from structured data. *Computational Linguistics*, 45(4):737–763, December 2019. DOI: 10.1162/coli_a_00363. URL: https://aclanthology.org/J19-4005.

[41] Q. Guo, Z. Jin, X. Qiu, W. Zhang, D. Wipf, and Z. Zhang. Cyclegt: unsupervised graph-to-text and text-to-graph generation via cycle training, 2020. arXiv: 2006.04702 [cs.CL].

[42] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*, WWW '20, pages 1082–1092, Taipei, Taiwan. Association for Computing Machinery, 2020. ISBN: 9781450370233. DOI: 10.1145/3366423.3380186. URL: https://doi.org/10.1145/3366423.3380186.

[43] C. Lopez, L. Leclercq, P. Krishnakumari, N. Chiabaut, and H. van Lint. Revealing the day-to-day regularity of urban congestion patterns with 3d speed maps. *Scientific Reports*, 7(1):14029, September 2017. ISSN: 2045-2322. DOI: 10.1038/s41598-017-14237-8. URL: https://doi.org/10.1038/s41598-017-14237-8.

[44] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk. Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, July 2022. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10228-y. URL: http://dx.doi.org/10.1007/s10462-022-10228-y.

[45] D. L. GERLOUGH and M. J. HUBER, 1975. URL: https://onlinepubs.trb.org/onlinepubs/sr/sr165/165.pdf.

[46] G. Li, V. L. Knoop, and H. van Lint. Estimate the limit of predictability in short-term traffic forecasting: an entropy-based approach. *Transportation Research Part C: Emerging Technologies*, 138:103607, 2022. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2022.103607. URL: https://www.sciencedirect.com/science/article/pii/S0968090X22000535.

[47] G. van Rossum. Python tutorial. January 1995.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: an imperative style, high-performance deep learning library, 2019. arXiv: 1912.01703 [cs.LG].

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[51] T. pandas development team. Pandas-dev/pandas: pandas, version latest, February 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[52] Kepler.gl: make an impact with your location data. URL: `https://kepler.gl/`.

[53] URL: `https://geopy.readthedocs.io/en/stable/`.

[54] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014. ISSN: 0001-0782. DOI: `10.1145/2611567`. URL: `https://doi.org/10.1145/2611567`.

[55] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: data-driven traffic forecasting, 2017. arXiv: `1707.01926 [cs.LG]`.

[56] A. Armanious. District 07 mobility performance report: 2022 second quarter, July 2022. URL: `https://dot.ca.gov/-/media/dot-media/programs/traffic-operations/documents/mpr/2022/q2/2022-d7-q2.pdf`.

[57] D. Gallego. District 05 mobility performance report: 2023 second quarter, July 2023. URL: `https://dot.ca.gov/-/media/dot-media/programs/traffic-operations/documents/mpr/2023/q2/2023-d5-q2.pdf`.

[58] H. Yuan, H. Yu, S. Gui, and S. Ji. Explainability in graph neural networks: a taxonomic survey, 2022. arXiv: `2012.15445 [cs.LG]`.

[59] Arcgis: world street map. URL: `https://www.arcgis.com/apps/mapviewer/index.html`.