Information Sciences 645 (2023) 119320

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Explainability techniques applied to road traffic forecasting using Graph Neural Network models

Javier García-Sigüenza, Faraon Llorens-Largo, Leandro Tortosa, Jose F. Vicent [*]

Department of Computer Science and Artificial Intelligence, University of Alicante, Campus de San Vicente del Raspeig, Ap. Correos 99, E-03080, Alicante, Spain

A R T I C L E I N F O

Keywords:
Graph neural networks
Deep learning
Data analysis
Explainability
Traffic flow

A B S T R A C T

In recent years, several new Artificial Intelligence methods have been developed to make models more explainable and interpretable. The techniques essentially deal with the implementation of transparency and traceability of black box machine learning methods. Black box refers to the inability to explain why the model turns the input into the output, which may be problematic in some fields. To overcome this problem, our approach provides a comprehensive combination of predictive and explainability techniques. Firstly, we compared statistical regression, classic machine learning and deep learning models, reaching the conclusion that models based on deep learning exhibit greater accuracy. Of the great variety of deep learning models, the best predictive model in spatio-temporal traffic datasets was found to be the Adaptive Graph Convolutional Recurrent Network. Regarding the explainability technique, GraphMask shows a notably higher fidelity metric than other methods. The integration of both techniques was tested by means of experimental results, concluding that our approach improves deep learning model accuracy, making such models more transparent and interpretable. It allows us to discard up to 95% of the nodes used, facilitating an analysis of its behavior and thus improving the understanding of the model.

1. Introduction

The problem of road traffic, both inside and outside cities, has a great impact on people's lives, affecting both population and freight transport. Currently, it is particularly relevant to the development of smart cities, which allow traffic data to be obtained through road sensors, public transport monitoring or data obtained through mobile applications. All this digital data represents an opportunity for traffic prediction models, which can make use of a greater amount of information, with this hence being an opportunity to obtain more complex and precise models, while allowing for better validation of their predictions.

At a technical level, it is a problem that combines both spatial and temporal components, which is why some models only work with the temporal elements and others attempt to combine both the spatial and temporal components of the problem. This aspect is key when choosing the model used to seek to predict traffic, since models that only use temporary information lose much of the data available to improve their predictions.

The importance of the traffic problem has led to it being extensively studied, and different methods have been used to attempt to solve it over the years. The problem of traffic predictions has essentially been addressed via two approaches: using Statistical

* Corresponding author.
E-mail addresses: javierg.siguenza@ua.es (J. García-Sigüenza), faraon.llorens@ua.es (F. Llorens-Largo), tortosa@ua.es (L. Tortosa), jvicent@ua.es (J.F. Vicent).

https://doi.org/10.1016/j.ins.2023.119320
Received 16 January 2023; Received in revised form 26 May 2023; Accepted 10 June 2023

Available online 16 June 2023

Models or Machine-Learning-based frameworks. Traditional models often use statistical methods, with the Auto-Regressive Integrated Moving Average (ARIMA) being a reference framework for short-term traffic flow prediction [1]. Since ARIMA, improvements have been proposed, such as models with a Kalman filter [2], or models with seasonal processes or using ARIMA subsets. However, due to the stochastic nature of traffic flow, these statistical models do not typically yield accurate results [3].

Furthermore, non-statistical models, based on neural networks, have also been widely used for traffic flow prediction. In [4], the authors proposed a surface back propagation neural network (BPNN) that, despite obtaining relatively good results, did not work for large volumes of data. To solve the treatment of large amounts of information, deep learning emerged. Based on multiple layers, this allows a large number of characteristics to be progressively extracted from the input data. Thus, a large number of deep learning architectures, such as Recurrent Neural Networks (RNN) [5], Deep Belief Network (DBN) [6], Convolutional Neural Network (CNN) [7] or Restricted Boltzmann Machines (RBM) [8], have been used to study and analyze the proposed traffic problem. More specifically, in the field of Machine and Deep Learning traffic prediction, we find Linear Support Vector Regression (SVR) [9], a general end-to-end approach to sequence learning (FC-LSTM) [10], Graph Convolutional Network (GCN) [11] and Graph Convolutional Recurrent Neural Network (DCRNN) [12], among others. These last three models show the diversity of deep learning architectures, with FC-LSTM being a Recurrent Neural Network and DCRNN being based on the combination of Graph Convolutional Network and Recurrent Neuronal Network [13,14].

Thus, one of the motivations of the present work is to explore the best models for traffic forecasting. For this purpose, we compare prediction methods, taking into account the metrics of absolute error, mean square error and mean absolute percentage error, verifying that the models from the field of deep learning present better results than the others. Of the models with the best results, the one combining Graph Convolutional Network (GCN) and Recurrent Neural Network (RNN) architectures was chosen, since it enables the use of both spatial and temporal information in the problem and a more accurate prediction is obtained.

Furthermore, in some problems, the lack of understanding of the final solution is not a critical element (e.g. facial recognition), while, in other cases, the explainability factor may be crucial, for instance, in the decision to grant a loan to a bank client.

The use of predictive techniques based on machine learning (ML) has increased in many areas of our lives. However, due to their complexity, the predictions made are often difficult to validate and, above all, to explain. Even if ML frameworks are available for exploration, their complexity makes it difficult to explain how they work. This difficulty is known as the "black box" problem applied to ML and seeks to answer the question of "What else can the model tell me?" [15].

Within the field of deep learning, techniques based on Graph Neural Networks (GNNs) have been shown to be highly effective for many applications, such as computer vision [16], audiorecognition [17], video recognition [18], selecting similar neighbors [19] or community discovery [20] among others. However, when a graph neural network made a prediction, a question of explainability arose: "What fraction of the input graph has the greatest influence on the model's decision?" Finding an answer requires understanding the model's inner workings in general and emphasizing the insights on the decision for the instance at hand.

The non-linear, multi-layered structure of deep learning models, means they are often criticized for being neither transparent nor interpretable, and there is thus a need to build trust and fairness in these types of techniques. In this sense, explainable artificial intelligence [21] is a technique that has made significant progress in several areas. This field has two main issues, namely, interpretability and explainability. Interpretability refers to the design of models that can be understood by the very design of their architecture, resulting in white boxes, while explainability is based on the ability to explain different features of a black box model. The use of the term typically refers, especially in the field of deep learning, to functions or models with a behavior that is too complex for people to understand.

Interpretability and explainability have contrasting approaches. In the case of interpretability, an attempt is made to develop machine learning models from architectures that are based on the logic used by people, while, in the case of explainability, the intention is to develop methods that make it possible to bring the behavior of black boxes closer to human logic. Currently, explainability is one of the most important challenges facing AI on its journey to application in real tasks, with there being different objectives that need to be solved [22].

When interpretability is sought, fuzzy systems are an alternative to consider. Fuzzy systems are designed to solve problems by using human logic; they are able to represent the inaccuracy of imprecise concepts, which are typical of human language, like, for instance, expressing the degree to which granting a loan represents a low, medium or high risk. To do this, such systems make use of a series of rules that can be created from expert knowledge or from algorithms such as the Wang-Mendel (WM) algorithm [23]. Fuzzy systems belong to the category of white-boxes, but, like other methods in this category, they have the disadvantage that, despite having adequate explainability, the accuracy and adaptability of deep learning methods have been shown to be superior [24].

To overcome the limitations of fuzzy systems, different alternatives influenced by deep learning have been proposed. For instance, models based on fuzzy systems have been developed that imitate the architecture of deep learning models, where models are built with several layers of depth, as is the case of Random Locally Optimized Deep Fuzzy System (RLODFS) [25]. Furthermore, research has sought the integration of fuzzy systems with deep learning models [26], either through integrated models, which use fuzzy systems as part of the training of the model. For instance, in [27,28] the authors improve the training of the weights of the deep learning model. Others have implemented ensemble models, which use fuzzy systems sequentially [29] or in parallel [30] with deep learning models, combining both techniques.

The combination of the use of fuzzy systems and deep learning has received special attention in recent years, leading the field to evolve, which, together with their explainability properties, makes them an interesting alternative when explainability is sought. However, methods that integrate fuzzy systems and deep learning also have their limitations. In models where layers with different rules are concatenated, such as RLODFS, the increase in the number of rules, which enhances accuracy, reduces the interpretability of the model. Thus, a proper balance between the number of rules and the interpretability must be struck. In models that integrate deep learning and fuzzy system models, the structure becomes more complex and the computational cost increases. Nonetheless, in most models, interpretability does not greatly increase, and interpretability and accuracy cannot be combined, making a trade-off between both properties necessary [31].

These limitations have to be considered, along with the fact that although a number of works have focused on traffic prediction using fuzzy systems [32], these are focused on accuracy rather than explainability and are more limited than those performed with GNNs, which represent the state of the art for traffic prediction. Therefore, fuzzy systems are an alternative when it comes to obtaining models with explainability for traffic prediction, although the aim of this work is to provide explainability to the state of the art of traffic prediction. In so doing, we seek to make explainability applied to GNNs the line of research that is most aligned with the aim of this paper, since the development and scope of GNNs in the field of traffic prediction is greater in the current context.

Different methodologies exist to apply explainability to Supervised Machine Learning methods (SML) [33]. These methodologies have different characteristics that refer to different qualities. Some of these are applicable to white boxes, which are models that are interpretable, and others refer to methods applicable to black boxes, which require the use of explainability. These methods can be classified as pre-hoc or post-hoc, depending on the type of model. In the former, explainability is part of the design of the model itself, as is the case with white box models. In post-hoc models, an explainability technique is applied a posteriori, with this being the case of black box models. As we use a predictive model that combines the Graph Convolutional Network (GCN) and Recurrent Neural Network (RNN) architectures, it is necessary to use a post-hoc technique, since the model behaves as a black box.

In order to achieve transparency and interpretability in deep learning models and build trust and fairness, it is necessary to combine predictive deep learning techniques with explainability models. With this motivation in mind, the main aim of our proposal is to integrate two Artificial Intelligence (AI) models, one referring to the prediction, which allows the problem of traffic prediction in a city to be addressed, and the other referring to an explainability technique in order to better understand its predictive performance. To fulfil this objective, it is necessary to analyze the problem, propose a deep learning predictive model, and integrate it with a selected explainability technique that fits the general aim established. Furthermore, it is necessary to analyze the experimental results to determine whether the aforementioned general aim is achieved.

Our contribution in achieving this goal is thus summarized as follows:

- We conduct a comprehensive comparison of existing statistical regression models, classic machine learning models and deep learning models applied to different sets of traffic data.
- A systematic comparison of existing deep learning models is presented to learn the complex spatial and dynamic temporal dependencies in traffic datasets.
- We compare existing explanation techniques with the aim of selecting the one that best fits the chosen predictive architecture.
- We perform the necessary changes in the definition of the predictive model and the explainability technique to integrate both.
- We analyze the results of applying the explainability technique to traffic forecasting, showing how confidence in a predictive model can be increased due to a greater knowledge of its behavior.

To achieve this objective, the paper is organized as follows. Section 2 presents a comparison with the aim of selecting a predictive model and an explainability methodology. Section 3 provides a detailed description of the predictive and explainability models selected. In Section 4, we describe the steps involved in the explanation of the predictive model. The results of the proposed combination of models are discussed in Section 5. Finally, our conclusions are presented in Section 6.

2. Preliminaries

Achieving the aim of this paper, that is, integrating artificial intelligence (AI) predictive models with explainability techniques, entails selecting a predictive model and then one of explainability. Thus, within the wide range of existing Artificial Intelligence algorithms, we must select one. To do this, we first distinguish between statistical regression models, classic machine learning models and deep learning models. The choice of the explainability algorithm raises fewer doubts because it is a relatively new branch of AI and the number of existing algorithms is smaller. In addition, within the different existing explainability techniques, the one that best fits the selected predictive architecture was chosen.

To select the best predictive model, the comparison was conducted on different sets of traffic data. In addition, to make the comparison, use was made of the most widely employed precision metrics, namely, the mean absolute error (MAE), the root mean square error (RMSE) and the mean absolute percentage error (MAPE).

In this section, we therefore present the different datasets used and the metrics used to compare the models. We also discuss statistical regression, classic machine learning and deep learning algorithms, as well as a more refined comparison of the most advanced deep learning models used. Finally, we select the explainability model that best suits the chosen predictive algorithm.

2.1. Datasets

Different traffic datasets are used to compare models. Specifically, the datasets selected are PeMSD3, PeMSD4, PeMSD7 and PeMSD8. These are the most widely used datasets for the traffic prediction problem, all of which are taken from the Performance Measurement System (PeMS) [34] of the *California Department of Transportation*.

All these data are obtained from a set of sensors and consist of a single variable, which indicates the volume of traffic in each of the sensors in periods of 5 minutes. The different datasets vary in the number of sensors and in the span of time covered.

Table 1
Comparison between the metrics obtained by the regression statistical models, machine learning and deep learning models in four different traffic datasets.

Dataset	Metrics	ARIMA	VAR	SVR	LSTM-FC	DCRNN
PeMSD3	MAE	35.41	23.65	21.97	21.33	**18.18**
	RMSE	47.59	38.26	35.29	35.11	**30.31**
	MAPE	33.78%	24.51%	21.51%	23.33%	**18.91%**
PeMSD4	MAE	33.73	23.75	28.70	27.14	**24.70**
	RMSE	48.80	36.66	44.56	41.59	**38.12**
	MAPE	24.18%	18.09%	19.20%	18.20%	**17.12%**
PeMSD7	MAE	38.17	75.63	32.49	29.98	**25.30**
	RMSE	59.27	115.24	50.22	45.84	**38.58**
	MAPE	19.46%	32.22%	14.26%	13.20%	**11.66%**
PeMSD8	MAE	31.09	23.46	23.25	22.20	**17.86**
	RMSE	44.32	36.33	36.16	34.06	**27.83**
	MAPE	22.73%	15.42%	14.64%	14.20%	**11.45%**

- PeMSD3 consists of 358 sensors and spans from September 9, 2018 to November 30, 2018.
- PeMSD4 consists of 307 sensors and spans from January 1, 2018 to February 28, 2018.
- PeMSD7 consists of 883 sensors and spans from May 1, 2017 to August 31, 2018.
- PeMSD8 consists of 170 sensors and spans from July 1, 2018 to August 31, 2018.

In addition, the PeMS-Bay dataset was used since it includes the geolocation of the nodes (sensors), which facilitates the correct visualization, on a map, of the results obtained by applying the explainability technique on the chosen predictive model. This data set (PeMS-Bay) contains the average speed of vehicles measured in 325 sensors in periods of 5 minutes. This information was obtained from the data of the Performance Measurement System (PeMS) [34] in the period from January 1, 2017 to May 31, 2017.

2.2. *Precision metrics*

To perform the comparison, the following metrics were used: the mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE). These are the metrics most widely used to compare the results obtained when making predictions. The definitions of these metrics are:

$$\begin{aligned} \mathrm{MAE} &= \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \\ \mathrm{RMSE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \\ \mathrm{MAPE} &= \frac{100\%}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{y_i}, \end{aligned} \tag{1}$$

where N is the amount of data and y_i represents specific data.

2.3. *Comparison of statistical regression, machine learning and deep learning models*

The problem of traffic density has been extensively studied, leveraging methods from many different fields of Artificial Intelligence, such as statistical regression models, classic machine learning models and deep learning models, among others.

Statistical regression methods include models of the *Auto-Regressive Integrated Moving Average* (ARIMA) type with a Kalman filter and models of the *Vector Auto-regressive (VAR)* type, with these being one of the first to be applied to the traffic problem. In the field of classic machine learning, many of the studies conducted are based on *Linear Support Vector Regression* (SVR) [9]. Within deep learning models, a wide variety of methods is used, with the Long Short-Term Memory-Fully Connected (LSTM-FC) [10] and Deep Convolutional Recurrent Neural Network (DCRNN) [12] algorithms being the most noteworthy. These two types of models show the diversity of the deep learning architectures used to perform the comparison, since LSTM-FC is a recurrent neural network (RNN) that only makes use of temporary data, while DCRNN combines the use of RNNs and *Graph Convolutional Networks* (GCN) [11], using both temporal and spatial data.

Table 1 clearly shows that the DCRNN model obtains the best results in the three metrics and in the four datasets used. This evidences that the use of deep learning with both spatial and temporal data yields better results than statistical regression models, classic machine learning models or deep learning models that only use temporal information, such as LSTM-FC.

Table 2
Performance comparison between the metrics obtained by the deep learning models in PeMSD3, PeMSD4, PeMSD7 and PeMDS8.

Dataset	Metrics	DCRNN	GraphWaveNet	STGCN	ASTGCN(r)	STSGCN	STFGNN	STGODE	AGCRN
PeMSD3	MAE	18.18	19.85	17.49	17.69	17.48	16.77	16.50	**15.86**
	RMSE	30.31	32.94	30.12	29.66	29.21	28.34	**27.84**	27.98
	MAPE	18.91%	19.31%	17.15%	19.40%	16.78%	16.30%	16.69%	**15.04%**
PeMSD4	MAE	24.70	25.45	22.70	22.93	21.19	19.83	20.84	**19.45**
	RMSE	38.12	39.70	35.55	35.22	33.65	31.88	32.82	**31.82**
	MAPE	17.12%	17.29%	14.59%	16.56%	13.02%	13.02%	13.77%	**12.82%**
PeMSD7	MAE	25.30	26.85	25.38	28.05	24.26	22.07	22.99	**21.20**
	RMSE	38.58	42.78	38.78	42.57	39.03	35.80	37.54	**35.14**
	MAPE	11.66%	12.12%	11.08%	13.92%	9.21%	9.21%	10.14%	**8.99%**
PeMSD8	MAE	17.86	19.13	18.02	18.61	17.13	16.64	16.81	**15.75**
	RMSE	27.83	31.05	27.83	28.16	26.80	26.22	25.97	**24.94**
	MAPE	11.45%	12.68%	11.40%	13.08%	10.60%	10.60%	10.62%	**10.17%**

2.4. Comparison of deep learning models

The results obtained in Table 1 clearly suggest that the models based on deep learning present the best results in traffic-related data. Thus, the next step is to carry out another performance comparison between different modern deep learning models that make use of both temporal and spatial data from the PeMSD3, PeMSD4, PeMSD7 and PeMSD8 datasets.

All the models used to carry out the second comparison are recent models with different characteristics. The Diffusion Convolutional Recurrent Neural Network (DCRNN) uses a combination of graph recurrent neural networks to capture temporal information, and diffusion convolutions to capture spatial information, using an encoder-decoder structure. Graph WaveNet [35] combines the use of stacked dilated 1D convolutions to capture temporal information, with the use of graph convolutional network layers to process spatial information. Spatio-Temporal Graph Convolutional Networks (STGCN) [36] are composed of different blocks comprising by a combination of one-dimensional convolutions followed by gated linear units and graph convolutional networks layers to capture both temporal and spatial information. Attention Based Spatial-Temporal Graph Convolutional Networks (ASTGCN) [37] capture spatio-temporal information using both attention mechanisms and convolutions. Spatial-Temporal Synchronous Graph Convolutional Networks (STSGCN) [38] seek to obtain localized complex correlations that exist between temporal and spatial information through the use of different modules composed of a group of graph convolutional operations. Spatial-Temporal Fusion Graph Neural Networks (STFGNN) [39] learn hidden spatio-temporal dependencies, using a novel operation that is applied, in parallel, to different spatial and temporal graphs. To this end, a series of temporal graphs are generated from a data-driven method. Spatial-Temporal Graph Ordinary Differential Equation Networks (STGODE) [40] capture dynamic spatio-temporal relationships through the use of tensor-based ordinary differential equations, which allow networks to be synchronously deepened to use spatio-temporal features. Finally, the Adaptive Graph Convolutional Recurrent Network (AGCRN) [41] model uses two modules to learn unique characteristics of each node and a non-predefined matrix that allows correlations to be detected at the spatial and temporal level in the time series.

As can be seen in Table 2, no model presents the best performance in all metrics and datasets. However, the model that generally presents a better set of results is AGCRN, obtaining the best metrics in 11 out of 12 cases. Consequently, this was the model chosen for the study.

The selected AGCRN model presents an architecture focused on learning specific characteristics of the different nodes of the graph, as well as the relationships between them at spatial and temporal level. Thus, since the aim of the article is to integrate prediction models with explainability techniques, the choice of the deep learning predictive technique is a key issue. This is because the explainability model chosen must take into account the characteristics of the architecture based on graph neural networks.

2.5. Comparison of explainability models

Explainability in graph neural networks (GNNs) is an emerging field, where the number of techniques is still very limited, and where the metrics for evaluating and comparing methods are not standardized [21,42]. A number of articles [43] address the problem of comparing the performance of explainability methods, but without a standard metric. However, precision and fidelity, in their different variants, are the two most widely used metrics.

Taking into account that an explainability technique needs a model on which to be evaluated, a number of predictive models of varying efficiency, based on the architecture, are used to evaluate such techniques. Graph Convolutional Networks (GCN) [11], Graph Attention Networks (GAT) [44], Graph Sample and Aggregate (GraphSAGE) [45] or Graph Convolutional Network via Initial residual and Identity mapping (GCNII) [46] are the most frequently used.

Due to the specificity and complexity of determining the optimal explainability method for a GNN model, we focus on finding the method that provides the best performance on a graph convolutional network (GCN) architecture. This is because the chosen predictive model (AGCRN) is based on GCN. In addition, datasets with real data are used, since those employed in the experimentation (PeMS) belong to this category.

Table 3
Comparison between the mean value of fidelity for sparsities in a range of 0.5 to 0.95 in the GCN architecture.

	Cora	Pubmed
GNNExplainer	0.3912	0.1475
PGExplainer	0.2164	0.0960
GraphMask	**0.5777**	**0.3080**

It is worth noting that the AGCRN predictive model behaves like a black box, which makes it necessary to use a post-hoc explainability technique. The existing literature shows that of the various explainability methods that meet the aforementioned requirements, two main groups can be distinguished:

- **Non GNSs-specific techniques**: These are techniques that were not initially designed for GNNs but have been adapted to be used in them.
- **GNNs-specific techniques**: These techniques have been designed from the outset to be specifically used in the GNN architecture, and are thus designed and adapted to their needs.

Of the two groups of explainability techniques, we evidently chose to use those that are specific to GNNs, since they are better adjusted to the needs of this architecture, and therefore present better performance.

Before selecting the explainability method to use, it is necessary to establish a series of criteria in order to compare effectiveness. Of the different existing metrics, it was decided to use sparsity and fidelity since the combination of these two metrics allows us to indicate the influence of the changes produced by the explainability method in a predictive model.

Fidelity indicates the fidelity of the prediction made by the model to which the explainability technique is applied, compared to the prediction of the original model, and is defined as:

$$\begin{gathered} \mathit{Fidelity} = P(Y = c|G) - P(Y = c|G \setminus G_S) \\ c = \underset{c \in C}{\mathit{arg\,max}}\ P(Y = c|G), \end{gathered} \tag{2}$$

where:

Y	$\rightarrow$ Model prediction.
G	$\rightarrow$ Original graph.
G_S	$\rightarrow$ Graph after applying the explainability method.
$\setminus$	$\rightarrow$ Set difference operator.
C	$\rightarrow$ Set containing all classes.

On the other hand, sparsity measures the number of edges of the graph that are eliminated. A lower number of edges is normally considered better when explainability techniques are applied, since a smaller subgraph is considered to have been achieved where the information the model needs to make its prediction is found. Sparsity is defined as:

$$\mathit{Sparsity} = 1 - \frac{m}{M}, \tag{3}$$

where:

m	$\rightarrow$ Number of edges of the subgraph.
M	$\rightarrow$ Number of edges of the original graph.

When we perform the different comparisons, sparsity is used as a parameter that serves to define the number of edges to be preserved. In this way, the evolution of fidelity is compared by varying the sparsity parameter.

Table 3 shows a comparison of the mean fidelity for sparsities between 0.5 and 0.95 for the explainability methods of GNNExplainer [47], PGExplainer [48] and GraphMask [49]. These three methods are post-hoc and specific for GNNs, while the comparison is made on the Cora and Pubmed datasets, both of which are made up of real data [21]. The architecture used to obtain the different metrics for the three explainability methods was GCN. In so doing, we sought to obtain a comparison that is as close as possible to the operation of the explainability method on the AGCRN model.

Table 3 shows that, for both datasets, the fidelity value for the GraphMask method is notably higher than for the other two methods. Consequently, GraphMask was selected as the explainability method to be used, since the average fidelity obtained suggests it should provide better results.

3. Definition of the problem

This section defines the problem to be addressed through a detailed explanation of the selected predictive model, as well as the explainability method.

3.1. Predictive model

To understand traffic dynamics and make future predictions, it is essential to model complex spatial and temporal correlations with time series data. In this sense, many works have focused on the design of complex graph convolutional network (GCN) models, which capture shared patterns in the nodes and use predefined graphs. However, recent studies (see [41]) show that, for predictive traffic models, the definition of the predefined graph can be avoided and the learning of node-specific patterns is essential. In addition, the graphs generated in this way are typically incomplete and not specific to prediction tasks, and may present biases. To avoid all these problems, in [41], the authors propose a mechanism of two adaptive modules, which are:

1. Node Adaptive Parameter Learning (NAPL) module: this module learns node-specific patterns by factoring traditional GCN network parameters and generating node-specific parameters from weights and biases from a shared pool.
2. Data Adaptive Graph Generation (DAGG) module: it infers inter-dependencies between nodes from the data and generates the graph during training.

These two independent modules can be adapted to GCN-based prediction models, both separately and together, and have parameters that can be easily learned. The combination of NAPL and DAGG with recurrent networks gives rise to a unified predictive model called the Adaptive Graph Convolutional Recurrent Network (AGCRN), which is capable of capturing spatial and temporal correlations of specific nodes in time series of traffic data. Additionally, the training of the AGCRN model generates a significant node representation vector for each time series that contains important information about the dataset, traffic, in our case, that can be applied to other tasks. Thus, the goal of the AGCRN model is to predict the future value of a time series.

Next, we explain in detail the two modules that make up the AGCRN predictive method, since understanding these parts is essential for integration with the explainability method.

3.1.1. Node Adaptive Parameter Learning (NAPL)

Let $G(V,E)$ be an undirected graph, where V represents the set of N nodes, E is the set of edges and A the adjacency matrix of the graph G. As the model is based on a graph convolutional network (GCN) [11] whose graph convolution operation approximates a first-order Chebyshev polynomial expansion, it can be generalized to a multidimensional network as follows:

$$Z = (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X\Theta + b, \tag{4}$$

with:

$Z \in R^{n\times f}$	$\rightarrow$ Output of the GCN.
$I_n \in R^{n\times n}$	$\rightarrow$ Identity matrix of size $n \times n$.
$D \in R^{n\times n}$	$\rightarrow$ Degree matrix.
$A \in R^{n\times n}$	$\rightarrow$ Adjacency matrix of G
$X \in R^{n\times c}$	$\rightarrow$ Input of the GCN layer.
$\Theta \in R^{c\times f}$	$\rightarrow$ Weights of the GCN Layer.
$b \in R^{f}$	$\rightarrow$ Bias of the GCN layer.
n	$\rightarrow$ Number of nodes.
c	$\rightarrow$ Number of input variables.
f	$\rightarrow$ Number of output variables.

From an individual node viewpoint, this convolution operation can be seen as a transformation of an input $X^i \in R^{1\times c}$ into $Z^i \in R^{1\times f}$ with Θ and b equal for all nodes.

Note that the matrix $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, is the Laplacian matrix $\tilde{\mathcal{L}}$ of graph. In addition, the previous equation allows us to know all the components of the GCNs that must be taken into account to understand the changes proposed by the NAPL and DAGG modules into which the AGCR model is divided.

Assigning a set of specific $\Theta \in R^{c\times f}$ parameters to each node generates a set of $n \times c \times f$ parameters, a number so large that it would make the model prone to overfitting. To avoid this, it is proposed that the model learns two smaller matrices, to then generate the matrix $\Theta \in R^{n\times c\times f}$ using the multiplication of both, that is,

$$\Theta = E_G \cdot W_G, \tag{5}$$

where $E_G \in R^{n\times d}$ is the matrix of embeddings of the nodes, $W_G \in R^{d\times c\times f}$ is the set of shared weights and d is the size of the embedding, with d being less than n and independent of the number of nodes. The use of both matrices makes it possible to reduce the number of parameters to be learned by the network.

For the bias generating b the same principle is followed, that is

$$b = E_G \cdot b_G, \tag{6}$$

where b_G is a set shared bias.

Substituting in (4) the convolutional equation is obtained

$$Z = (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X E_G W_G + E_G b_G. \tag{7}$$

3.1.2. *Data Adaptive Graph Generation (DAGG)*

Most of the graph convolutional networks (GCN), applied to predictive models of traffic data, require a predefined adjacency matrix A for the calculation of the convolution operation. There are two basic approaches to precomputing the graph: 1) using a distance function, which defines the graph according to the geographical distance between different nodes; 2) by means of a similarity function, which defines the proximity of the node by measuring the similarity of the node's attributes. However, with these approaches, the predefined graph cannot contain complete information about spatial dependency, and is also not directly defined for its use in prediction tasks, which can generate considerable biases.

To solve the problem, this module generates graphs that adapt to the data in such a way that the hidden dependencies of the data can be inferred, with the model itself calculating an interdependency matrix between the nodes. This interdependence matrix will be the Laplacian matrix of the graph $\tilde{\mathcal{L}}$, from which the adjacency and degree matrices can be calculated. Thus, the Laplacian $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is redefined as

$$\tilde{\mathcal{L}} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = \mathit{softmax}(\mathit{ReLU}(E_A \cdot E_A^T)), \tag{8}$$

where the function *ReLU* is used to avoid negative values and *softmax* to normalize the values.

During training, E_A is automatically updated to learn the hidden dependencies between the different time series and obtain the adaptive matrix for the graph convolutions.

3.1.3. *Adaptive Graph Convolutional Recurrent Network (AGCRN)*

A link is established between the two modules defined to establish a predictive model of traffic time series without biases. Thus, substituting (8) in (7) we obtain

$$Z = (I_N + \mathit{softmax}(\mathit{ReLU}(E_A \cdot E_A^T))) X E_G W_G + E_G b_G = (I_N + \tilde{\mathcal{L}}) X E_G W_G + E_G b_G. \tag{9}$$

This equation represents the version of a graph convolutional network modified with the NAPL and DAGG modules. This allows specific patterns to be captured in each node using NAPL, and inferring interdependencies between them with the DAGG module.

Formula (9) makes use of spatial information, but the traffic prediction also has a temporal component. Thus, to add this temporal component, another module called Gated Recurrent Units (GRU) [50] is used. To integrate the three modules (GRU, NAPL and DAGG), the fully connected layers, which GRU uses, are replaced by the NAPL and DAGG modules.

Finally, the implementation of the GRU module with the addition of bias is

$$\begin{aligned}
\tilde{A} &= (I_n + \mathit{softmax}(\mathit{ReLU}(E E^T))), \\
z_t &= \sigma(\tilde{A}[X_{:,t}, h_{t-1}] E W_z + E b_z), \\
r_t &= \sigma(\tilde{A}[X_{:,t}, h_{t-1}] E W_r + E b_r), \\
h_t &= z_t \odot \hat{h}_t + (1 - z_t) \odot \hat{h}_{t-1}, \\
\hat{h}_t &= \tanh(\tilde{A}[X_{:,t}, r \odot h_{t-1}] E W_{\hat{h}} + E b_{\hat{h}}),
\end{aligned} \tag{10}$$

where E is the array of embeddings shared by NAPL and DAGG, z_t is the update gate, r_t is the reset gate, h_t is the output in step t, x_t is the input at step t, $\hat{h}_t$ is the candidate activation in step t, W, U, b are parameters to be learned by the model, $\odot$ is the Hadamart product, σ is the sigmoid function, and $[\cdot]$ is the concatenation operator.

To obtain the final model, which will perform the prediction, several AGCRN layers are chained, followed by a final layer that will perform a linear transformation, obtaining as a result, a matrix of dimension $R^{n \times \tau}$, with τ the size of the window, indicating the number of steps to predict.

3.2. *GraphMask explainability model*

The GraphMask [49] explainability technique provides adequate performance in models based on Graph Convolutional Network (GCN) architectures. It is a specific graph neural network explainability technique designed and adapted to the needs of this type of network. Furthermore, it is a post-hoc method, that attempts to understand how graph neural networks (GNNs) use the graph edges when making a prediction. This is intended to detect the connections that are important for the model, and thus identify superfluous edges in order to mask them and avoid their use. In short, this method makes decisions about whether to maintain or remove edges of the graph, to ensure that discarded edges have no relevance to the model's predictions. Given a graph G, for each layer of the convolutional neural network, the method returns a subgraph that reduces the number of edges used and, therefore, delimits the relevant connections so that the model can make predictions.

Given an input graph of a GNN, in each layer k, a representation $h_u^{(k)}$ is calculated for each node u, which is a function of the representations of the nodes of the previous layer. All the nodes are assigned an initial representative $h_u^{(0)}$ such that, for the different layers $k > 0$, a GNN can be defined through a message function M and an aggregation function A:

$$m^k_{u,v} = M^k(h^{k-1}_u, h^{k-1}_v, r_{u,v}), \tag{11}$$

$$h^k_u = A^k(\{m^k_{u,v} : u \in \mathcal{N}(v)\}), \tag{12}$$

where:

u, v	$\rightarrow$ Both nodes of an edge.
$r_{u,v}$	$\rightarrow$ Relationship between nodes u y v.
$\mathcal{N}(v)$	$\rightarrow$ The set of neighboring nodes of v.
k	$\rightarrow$ Number of the layer.

The messages can be understood as the information that each node transmits to its neighbors, while the aggregation function is responsible for combining the messages that arrive at a node from its neighbors. In the case of GCNs, the messages correspond to the result of applying the convolutions on each of the nodes.

The goal of GraphMask is to detect the set of edges (u, v) that can be ignored without affecting the model's prediction or accuracy below a set threshold. When an edge (u, v) can be ignored, both it and its corresponding messages $m^k_{u,v}$ are considered superfluous. Ignoring a message means that when calculating the status of a node using equation (12), the edge (u, v) will not be taken into account.

In masking the edges, two key elements must be considered. The first is that GNNs have the problem of being highly sensitive to changes in the graph architecture, while the second is that many of the edges may be superfluous except for the normalization task. Taking both elements into account, GraphMask ignores the edges, based on a binary option $z^k_{u,v} \in \{0, 1\}$, but replacing its message with a learned base value (learned baseline) b^k, following the formula:

$$\tilde{m}^k_{u,v} = z^k_{u,v} \cdot m^k_{u,v} + b^k \cdot (1 - z^k_{u,v}), \tag{13}$$

with $z^k_{u,v} = g_\pi(h^k_u, h^k_v, m^k_{u,v})$, determining $z^k_{u,v}$ the significance of an edge for the prediction of the model, indicating whether it should be used or ignored.

Function g_π is the same for all cases, and is defined from a set of parameters, π, learned during the training of the model –one layer neural network– that implements the function.

When training for the calculation of the messages and with the aim of learning the function g_π –needed to calculate $z^k_{u,v}$– only the information available in the original model is used, and thus only the state of the nodes and the messages of the layer k are used as input. In this way, g_π only handles information known by the original model, avoiding having information beyond that available to the layer k. This technique is called amortization and is intended to prevent a hindsight bias. Furthermore, g_π is not trained on a case-by-case basis, but is trained on several cases, using examples that were not seen by the original model in its training.

Matrix $\tilde{M}$ of values $z^k_{u,v}$ is called a mask

$$\tilde{M} = \begin{pmatrix} z^k_{1,1} & z^k_{1,2} & \dots & z^k_{1,v} \\ z^k_{2,1} & z^k_{2,2} & \dots & z^k_{2,v} \\ \dots & \dots & \dots & \dots \\ z^k_{u,1} & z^k_{u,2} & \dots & z^k_{u,v} \end{pmatrix}. \tag{14}$$

To obtain this, the original model must first be executed, to calculate the states and messages, obtaining the values of h^k_u, h^k_v and $m^k_{u,v}$. The mask is then obtained, which indicates the edges and, therefore, the messages that should be ignored. Finally, if the mask is applied to the original model, a new prediction can be obtained that ignores the masked messages. In the present work, this is called alternative prediction, in order to differentiate it from the prediction of the original model before applying any masks.

Note that the parameters that are trained to apply GraphMask are those of the network that produces g_π and the set of bias vectors b_k. However, the GNN does not change its parameters, but only has to apply the mask obtained.

4. Methodology

This section, describes each of the steps involved in explaining the predictive model presented in the paper, as shown in Fig. 1, where data preprocessing, partition generation, model training and metric calculation, among others, can be observed.

4.1. Combining GraphMask and AGCRN

As mentioned in previous sections, the GraphMask explainability model detects superfluous connections in the model in order to mask them and prevent their use. However, to implement this explainability model in a predictive model, such as AGRCN, it is necessary to modify the initial implementation of the model so that it supports the use of one mask, or several in the case of using a batch, without affecting the prediction. This section will explain how to modify the AGRCN predictive algorithm so that it can accept masks that eliminate superfluous connections from the graph.

The GraphMask explainability model is based on message masking, defined by the following equations:

$$m^k_{u,v} = M^k(h^{k-1}_u, h^{k-1}_v, r_{u,v}), \quad h^k_u = A^k(\{m^k_{u,v} : u \in \mathcal{N}(v)\}), \tag{15}$$

where:

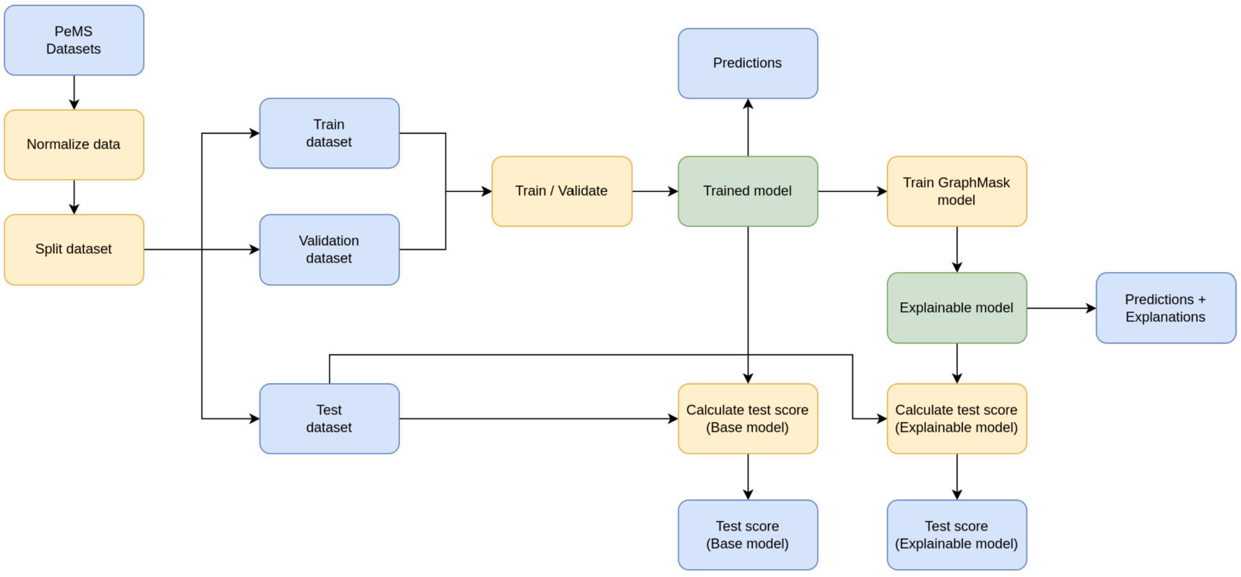


Fig. 1. Block diagram of the approach. In blue the data, in yellow the operations and in green the models.

M	$\rightarrow$ Message function
A	$\rightarrow$ Aggregation function
u, v	$\rightarrow$ Both nodes of an edge.
$r_{u,v}$	$\rightarrow$ Relationship between nodes u y v.
$\mathcal{N}(v)$	$\rightarrow$ The set of neighboring nodes of v.
k	$\rightarrow$ Number of the layer.

These two equations are described for graph neural networks (GNNs), but in the case of the chosen predictive model, AGCRN, it is a convolutional recurrent neural network of adaptive graphs. This makes it necessary to adapt the architecture.

For this adaptation, the way to mask the messages is first determined. Thus, the messages are composed of a function M, dependent on the states h_u^{k-1} and h_v^{k-1} and on the relationship between nodes $r_{u,v}$. This gives rise to two ways of masking the messages.

1. The first one consists of generating the message and then applying the mask, that is, multiplying by 0 or 1 the base to the value of $z_{u,v}^k$, which determines the mask. This option implies that when calculating the messages in (9), the mask must be applied to each message.
2. The second possibility is to indicate that the edge $r_{u,v}$ does not exist before calculating the message itself. This involves applying the mask to the matrix where the values of $r_{u,v}$ are defined, with this being the adjacency matrix.

In this work, we use the second option given its theoretical simplicity. As can be seen in (9), this option requires the use of the adjacency matrix through the Laplacian matrix of the graph

$$Z = (I_N + \tilde{\mathcal{L}})XE_GW_G + E_Gb_G. \tag{16}$$

A significant difficulty arises here, since the predictive model does not have an adjacency matrix. Rather, this is replaced by a matrix calculated in the training process to which it assigns the name of $\tilde{\mathcal{L}} = softmax(ReLU(E_A \cdot E_A^T))$. That is, there is no graph adjacency matrix on which to apply the mask. By way of a solution, the mask defined by $\tilde{M}$ will be applied directly to the matrix $\tilde{\mathcal{L}}$, such that the equation of Z is redefined as

$$Z = (I_N + \tilde{\mathcal{L}}\tilde{M})XE_GW_G + E_Gb_G. \tag{17}$$

Based on Equation (17), two important issues must be highlighted:

1. Firstly, the nodes will always have the information of their state in the previous layer, since, when multiplying by the identity matrix I_n, these values are preserved, regardless of the mask $\tilde{M}$.
2. Secondly, the matrix $\tilde{\mathcal{L}}$ tends to be a matrix without null values, and so it can be considered that, for AGCRN, the graph is totally connected, since all the nodes are seen to be influenced by the states of the other nodes.

Table 4
Results of GraphMask training on AGCRN.

	PeMSD3	PeMSD4	PeMSD7	PeMSD8	PeMS-Bay
MAE (Baseline)	15.86	19.45	21.20	15.74	1.67
MAE (GraphMask)	16.90	21.37	22.44	16.93	1.80
MAE variation	6.55%	9.87%	5.84%	7.56%	7.78%
Retained messages	31.99%	31.72%	44.21%	39.99%	5.18%

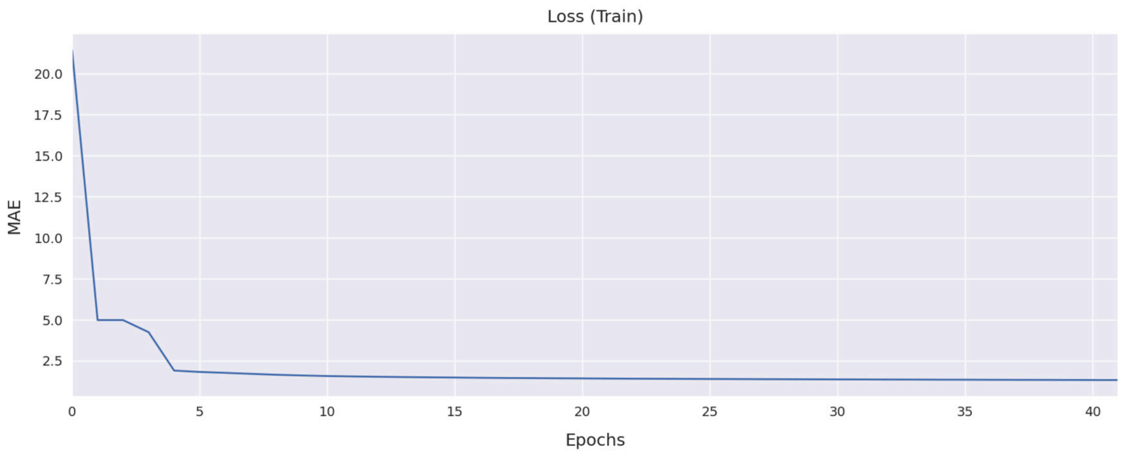


Fig. 2. Progression of the loss function over the training set during training on PeMS-Bay.

Additionally, when the mask is applied, it is necessary to add a bias (see the equation (13)) by substituting zeros for ones and ones for zeros. This new matrix is represented by $\tilde{M}_I$, and, since the size of the bias vector is equal to the length of a message, it is necessary to repeat the biases once per message, forming matrix B^k.

By applying these changes to (17), the following equation is obtained

$$Z = ((I_N + \tilde{\mathcal{L}}\tilde{M})XE_GW_G + E_Gb_G) + (\tilde{M}_I B^k). \tag{18}$$

However, when implementing equation (18) to perform the training, the function that calculates mask g_π, defined in (13), does not converge. This means that the integrated model is incapable of obtaining a mask and a set of biases that give a prediction close to the original. This drawback is due to matrix $\tilde{\mathcal{L}}$, in the AGCRN predictive model, having a regularizing effect, since it is not restricted to representing the connection between the nodes. The values of $\tilde{\mathcal{L}}$ conform to the relationship between the nodes, and these values can use different scales. Therefore, applying the bias, without taking into account the scale of $\tilde{\mathcal{L}}$, leads to these not being able to adapt to the values expected by the model.

In order to solve the non-convergence of the function g_π, the $\tilde{\mathcal{L}}$ matrix is introduced in the second part of equation (18), through an element-by-element Hadamart product $\tilde{\mathcal{L}} \odot (\tilde{M}_I B^k)$. Multiplying by matrix $\tilde{\mathcal{L}}$ makes g_π converge, since the biases are not multiplied only by ones or zeros, but are scaled by the values of the masked positions of $\tilde{\mathcal{L}}$. By applying this step, the behavior of the model is intended, making predictions similar to the original predictive model.

Finally, the equation of Z is

$$Z = ((I_N + \tilde{\mathcal{L}}\tilde{M})XE_GW_G + E_Gb_G) + (\tilde{\mathcal{L}} \odot \tilde{M}_I B^k). \tag{19}$$

It is worth noting that equation (19) is only applied when a non-null mask has been defined, since, if the mask is null, the behavior of the model will be the same as the original, working in the same way as without the implementation of GraphMask. It should be noted that to calculate both z_t and r_t the same mask $\tilde{M}$ is used, as they also share the same matrix $\tilde{\mathcal{L}}$.

In the case of the Gated Recurrent Unit (GRU) layer, which provides the temporal component to AGCRN, its implementation is not changed by the GraphMask integration, since upon receiving the value of Z, the messages are already discarded.

5. Experimental results

In this section, we validate the effectiveness of the combined deep learning and explainability model. First, the application of the model to the five types of datasets selected (PeMSD3, PeMSD4, PeMSD7, PeMSD8 and PeMS-Bay) is compared. This is followed by a detailed explanation of the model training. Finally, we discuss the results of the model on the PeMS-Bay dataset of the city of San Jose in California, USA.

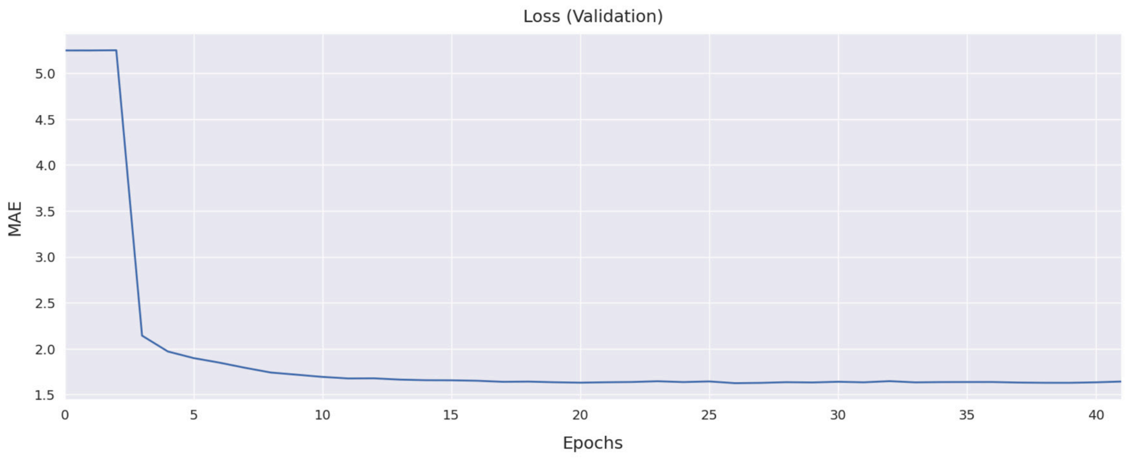


Fig. 3. Progression of the loss function over the validation set during training on PeMS-Bay.

Table 5
GraphMask training hyperparameters values on AGCRN.

Hyperparameters	Value
Epochs (layer 1)	2
Epochs (layer 0)	10
Learning Rate	0.003
Allowance	0.125

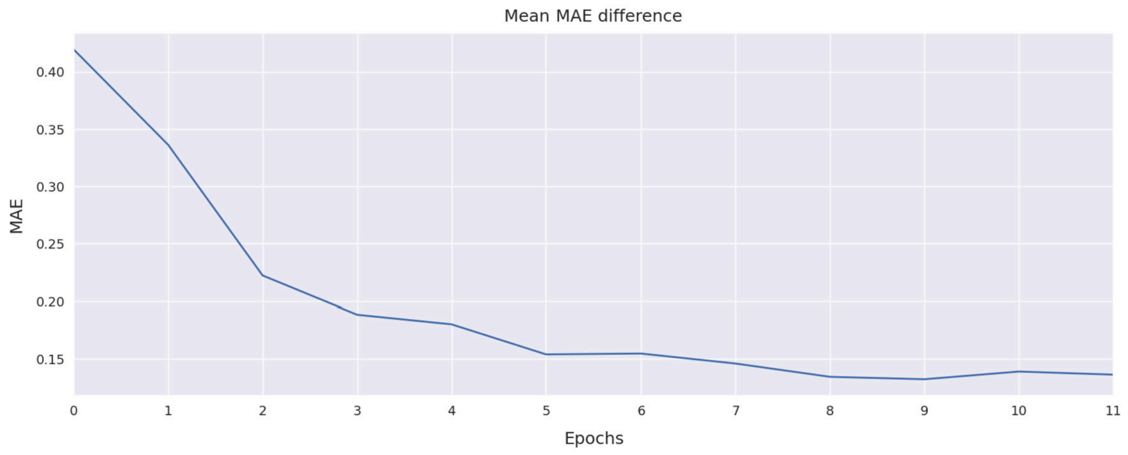


Fig. 4. Progression of the mean MAE difference during GraphMask training on PeMS-Bay.

5.1. Comparison of PeMS results

To observe the effect of applying GraphMask on the model, a series of tests was performed on the datasets used in Table 2, as well as on the PeMS-Bay dataset, which contains the geographic coordinates of different nodes, in order to subsequently visualize the results on a map. This dataset (PeMS-Bay) contains the average speed of vehicles, measured on 325 sensors in 5-minute periods.

The training of the model, applying GraphMask, was carried out seeking to conserve the minimum number of edges without exceeding a 10% increase in the MAE. In this way, we sought to apply explainability to the model, minimizing the effect on the prediction accuracy, eliminating only the edges with less weight on the prediction.

Table 4 shows the results obtained, varying the number of messages retained from 44.21% to 5.18%. The best result was obtained on PeMS-Bay, in which with only a 7.78% increase in the MAE, only 5.18% of the messages were retained, discarding almost 95% of them. This result drastically reduces the number of variables in the model predictions, simplifying the exploration and analysis of the inferences made.

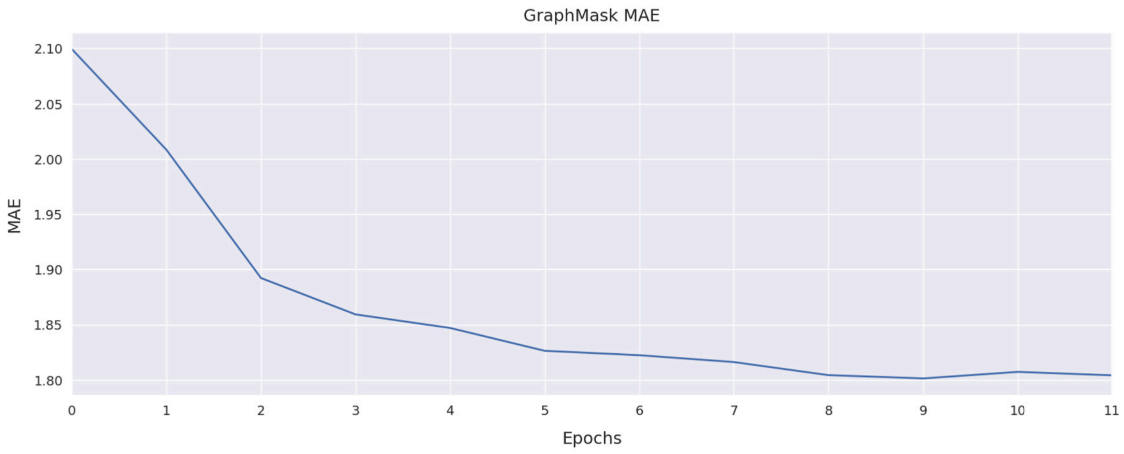


Fig. 5. MAE progression during GraphMask training on PeMS-Bay.

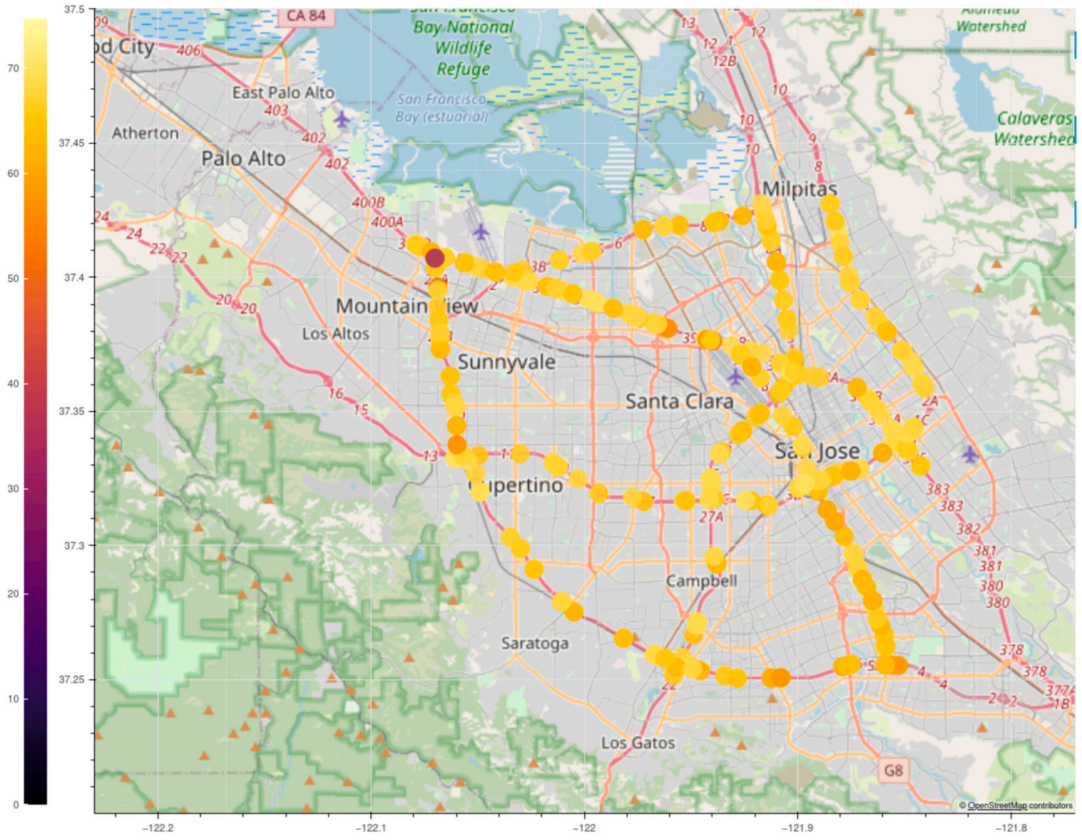


Fig. 6. Node set prediction.

5.2. Model training

The aim of the paper is not only to explore the different alternatives for traffic prediction, but also to explore the explainability options. Therefore, the model was trained on the PeMS-Bay dataset, which contains the geographic coordinates of the nodes, facilitating a visualization on a map and enhancing the exploration possibilities of the explainability results.

Figs. 2 and 3 show the evolution of the loss function over the training and evaluation set, showing a behavior that seems to indicate the ability of the model to generalize over both the training and validation sets.

The model was trained, implementing GraphMask using the hyperparameters listed in Table 5. Of all the hyperparameters, it is important to highlight allowance, which refers to the threshold of allowed error. In this case, it was set to 0.125, and so

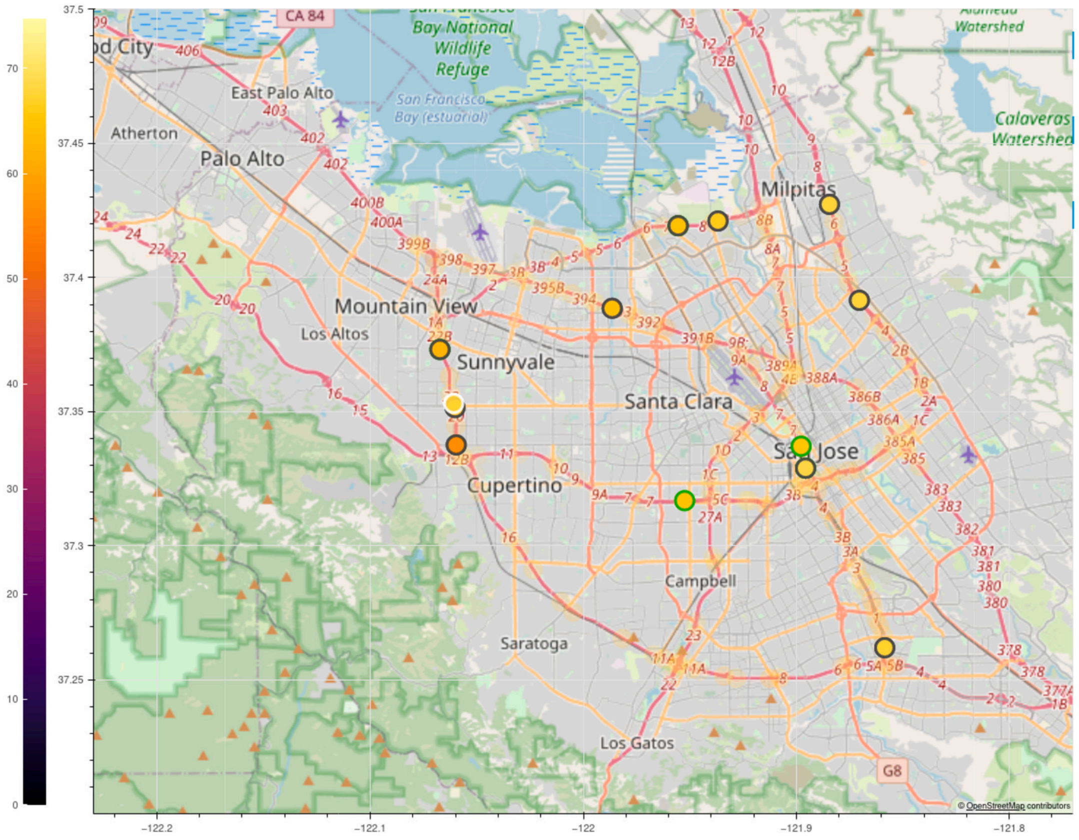


Fig. 7. Selected node together with the significant nodes in the first and second AGCRN layers.

GraphMask attempts to mask as many messages as possible, provided that the average MAE of the predictions does not exceed this threshold.

Regarding the GraphMask training, Figs. 4 and 5 show a good behavior, since the difference between the MAE of the original model and the model masked by GraphMask is reduced. Both values decrease until the last iterations, where the MAE starts to oscillate.

5.3. Discussion

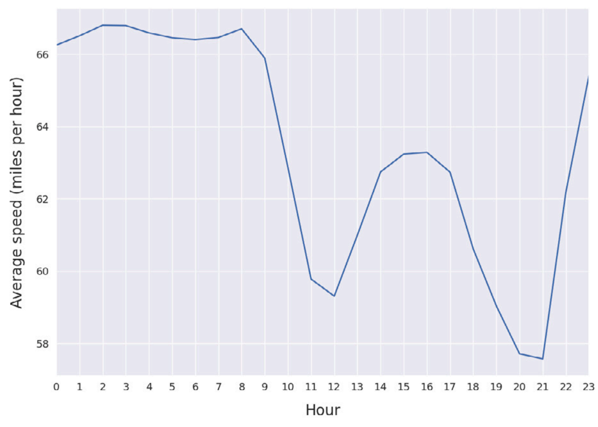
First, the model performance was tested, using one week's data from the PeMS-Bay dataset of the city of San Jose in California (USA). In order to understand the model, the masks and predictions obtained by the ACGRN model with GraphMask implemented and trained were visualized. It should be noted that each prediction is composed of a window of 12 values with 5 minutes difference between them, with the mask being shared by the 12 values of the prediction.

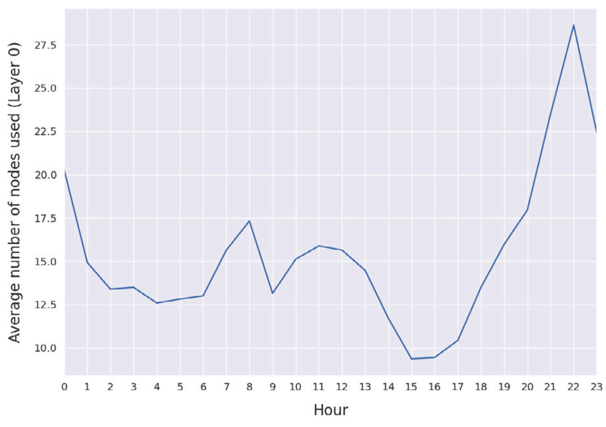
Fig. 6 shows a color map with the traffic prediction of the set of nodes in the network. Each node represents a sensor, with the colors showing the average speed of vehicles in the different zones. Thus, darker colors represent a lower average speed, and, therefore, higher traffic congestion, and lighter colors represent a higher average speed and, consequently, lower traffic congestion.

Fig. 7 shows the mask applied by GraphMask. The selected node is highlighted with a white border and only the nodes used in the prediction of that node are shown. These nodes are indicated with a green or gray border. This differentiation only distinguishes the nodes used in the prediction in layer 0 of AGCRN (green color) and those used in layer 1 of AGCRN (black color). In this way, it is easy to observe not only the nodes used by the model to make the prediction of a specific node, but also the depth at which they are used (indicated by the layer). This differentiation between layers is important, since, due to the functioning of the GCN architecture on which AGCRN is based, the information reaching layer 1 is more complex, given that it originates from the convolutions of layer 0.

In order to carry out an exhaustive analysis of the behavior of the model, we began by examining the relationship between the number of nodes used and the amount of existing traffic. To do this, the average hourly speed prediction was first compared with the number of nodes that had not been masked (hidden), and had therefore been taken into account by the model to make the prediction. In addition, for a more complete analysis, the number of masked nodes in layer 0 was separated from those in layer 1.

In Figs. 8a, 8b and 8c, the average velocity in each hour is compared with the number of nodes used in each of the two layers of the model. In Fig. 8c, a robust correlation between velocity and number of nodes can be observed, indicating that the lower the


(a) Average speed per hour.


(b) Average number of nodes used in layer 0 per hour.

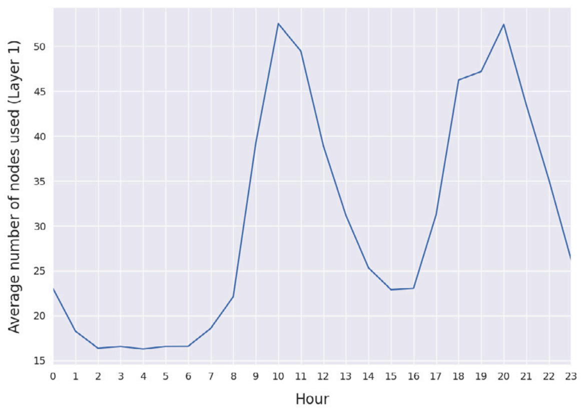

(c) Average number of nodes used in layer 1 per hour.

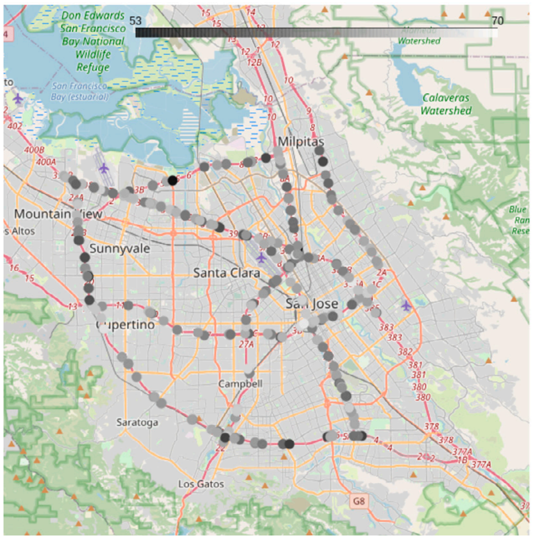
Fig. 8. Comparison between the average speed and the number of nodes used per hour in layers 0 and 1.

velocity, the larger is the number of nodes used in that layer. This behavior shows that the model uses a greater number of nodes in layer 1 to make the prediction at times of higher traffic, due to the larger number of variables to be taken into account. It is worth noting that, due to the architecture of graph convolutional networks (GCN), in layer 1, the messages are propagated between the different nodes, such that their information also contains information related to other nodes, providing more data about the traffic context. Therefore, it can be deduced that the model makes use of a greater amount of information at times of higher traffic, with it being able to predict the traffic at times of lower traffic only with the information from the nodes of layer 0.

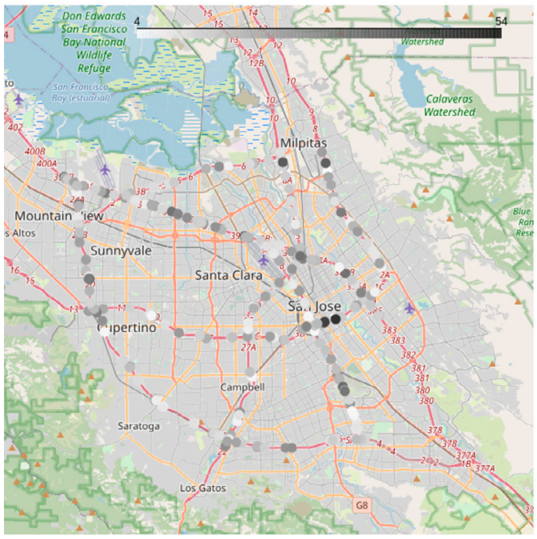
Once the correlation between the average speed and the number of nodes used is explored, the next step is to analyze the relationships at the specific node level and at spatial level. For this purpose, a series of maps were generated where the different nodes, the average velocity at these points and the number of times they are used by the model were plotted. Thus, Fig. 9 shows a comparison made between the average speed per hour of each of the nodes of the network (Fig. 9a) and the average number of nodes used in layers 0 and 1, respectively (Fig. 9b, Fig. 9c). In neither layer 0 nor layer 1 is a clear correlation observed between the average velocity at each node and the number of times it is used by the model. However, it is possible to observe a series of nodes consulted more than the average number of times, mainly in the central area of the map. Likewise, there is another series of nodes, scattered across the map, which are also used more than the average.

Examining the behavior of node use by hours, different patterns related to traffic can be observed. To perform an analysis of such patterns, the data was explored at three different times: at 3am, 12pm and 9pm. These three times were selected based on the behavior of the average speed per hour observed in Fig. 8a, which clearly shows the times of maximum and minimum speed. At 3am, the traffic is at one of its lowest levels (which is why the average speed is higher). At 12am, traffic is at a local maximum, and, at 9pm, it is at a global maximum. In this way, traffic patterns can be observed more clearly when comparing the results.

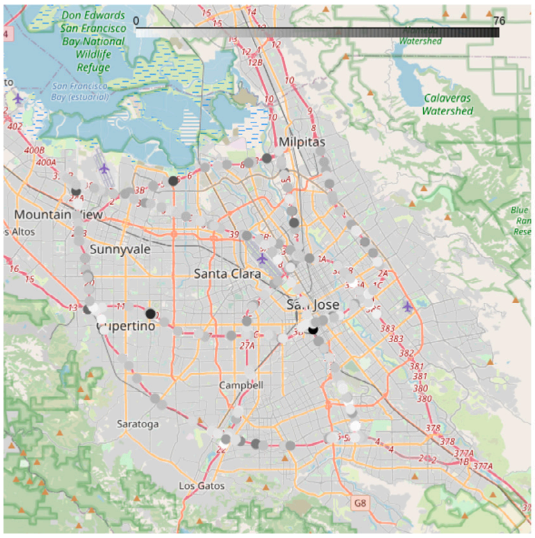
The first comparison, corresponding to 3am, relates the average speed per node (Fig. 10a) to the number of nodes used in layer 0 (Fig. 10b), and layer 1 (Fig. 10c). In this case, analyzing the nodes in isolation, no correlation is observed between the traffic and the number of nodes used. However, if we look at the whole, we can see the patterns detected above. At low speeds, the use of nodes in layer 1 is minimal and the use of nodes in the layer 0 is much higher, showing a tendency to consider the nodes in the central area of the map as more significant.



(a) Average speed per hour at each node.



(b) Average number of nodes used in layer 0 per hour.



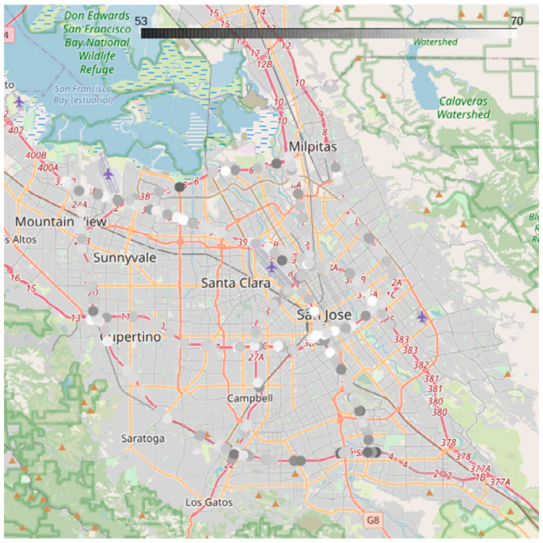
(c) Average number of nodes used in layer 1 per hour.

Fig. 9. Comparison between the average speed and nodes used per hour in layers 0 and 1.

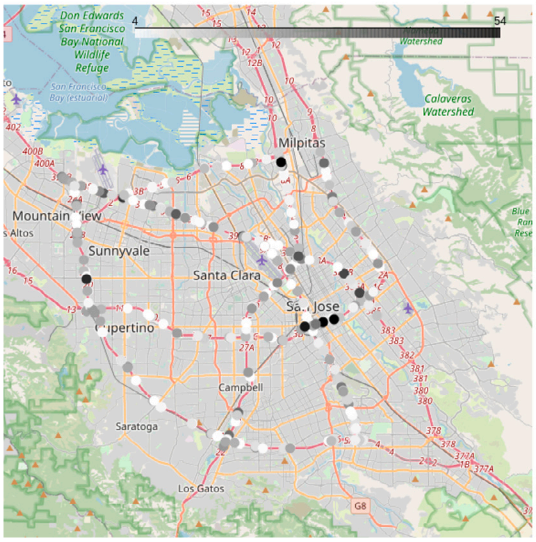
In the 12pm case, Fig. 11b shows that traffic increases, but the number of nodes used in layer 0 increases only slightly. However, in layer 1 in Fig. 11c, there is a noticeable increase in the number of nodes used. In this case, there is a correlation between the traffic detected at a node and the number of times this node is used, with the nodes with the greatest amount of traffic being those most used.

The time with highest level of traffic is 9pm and the behavior of the model can be seen in Figs. 12b and 12c. While at 12 o'clock noon, the model shows a correlation between traffic and the number of times the nodes in layer 1 are used, in this case, this correlation is extended to layer 0 where, the higher the level of traffic, the greater is the number of times it is queried by the model. This phenomenon can be visualized in Fig. 12b and is consistent with the situation in Figs. 8b and 8c, which show that, at 9pm, the average number of nodes is close to their maximum. From this behavior, it can be deduced that the model, at the moments of maximum traffic, continues to increase the use of nodes to be used, but, at a certain point, instead of increasing the number of nodes in layer 1, it starts to increase the number of nodes in layer 0. This is because the model needs more information, and, being unable to obtain it from the combination of messages from a few nodes in layer 0 and its propagation through layer 1, the number of nodes to be consulted also increases in the layer 0, with the information being composed of more nodes and with greater complexity.

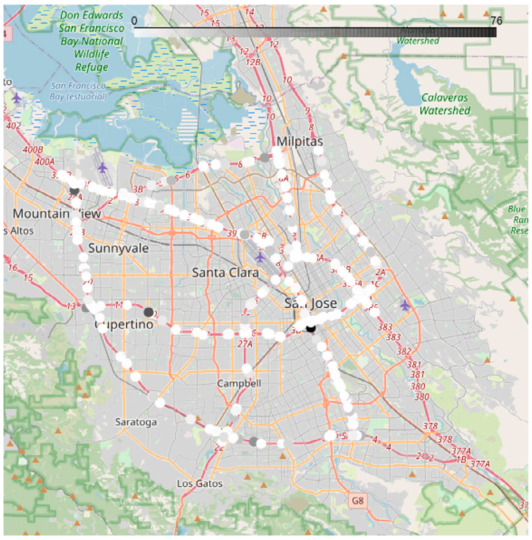
To summarize, the analysis conducted facilitates a better understanding of how the model works. Thus, regardless of the traffic, the model makes use of a few nodes scattered around the map, which it consults to obtain the prediction. Of these base nodes, those located in the central area of the map are the most significant. Likewise, the model queries a smaller number of nodes when traffic is



(a) Average speed per node at 3am.



(b) Average number of nodes used in layer 0 at 3am.



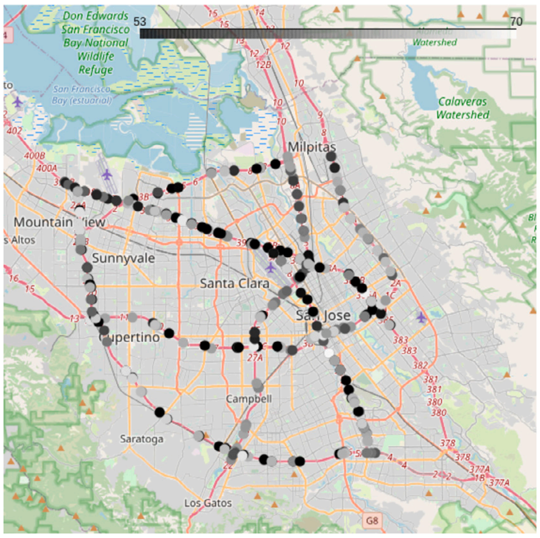
(c) Average number of nodes used in layer 1 at 3am.

Fig. 10. Comparison between the average speed and the nodes used for layers 0 and layer 1 at 3am.

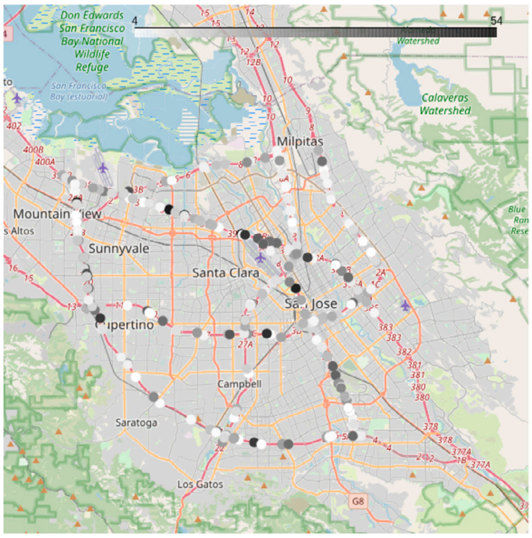
low, and the number of nodes increases as traffic increases and vice versa. In addition, it uses more complex information and more of it, the higher the level of traffic to be predicted. This can be seen by observing how, for a low amount of traffic, it mostly uses nodes in layer 0. However, as traffic increases, it increases the number of nodes in layer 1, making use of more complex information propagated through the network. Finally, when reaching the peak traffic, it makes use of more nodes in both layers 0 and 1, in order to obtain the greatest amount of information possible.

6. Conclusions

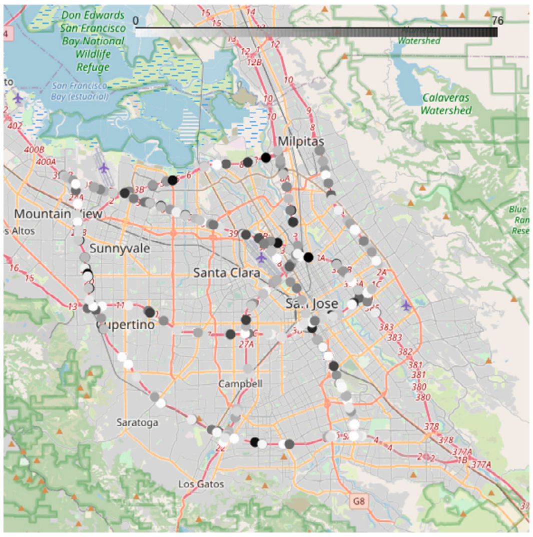
This work carried out an analysis of different fields of artificial intelligence, with the aim being to provide a model that performs predictions with an explainability component. Firstly, to approach data related to traffic flows, a comparison was made between the three most commonly used techniques: statistical regression, classic machine learning and deep learning. The results show that the techniques based on deep learning present the best results. However, as a consequence of the great variety of models based on deep learning, it was necessary to carry out another performance comparison between different architectures applied to spatio-temporal traffic flow data (PeMSD3, PeMSD4, PeMSD7 and PeMSD8). We compared the following different models: Diffusion Convolutional Recurrent Neural Network, Graph WaveNet, Spatio-Temporal Graph Convolutional Networks, Attention Based Spatial-Temporal Graph Convolutional Networks, Spatial-Temporal Synchronous Graph Convolutional Networks, Spatial-Temporal



(a) Average speed per node at 12pm.



(b) Average number of nodes used in layer 0 at 12pm.



(c) Average number of nodes used in layer 1 at 12pm.

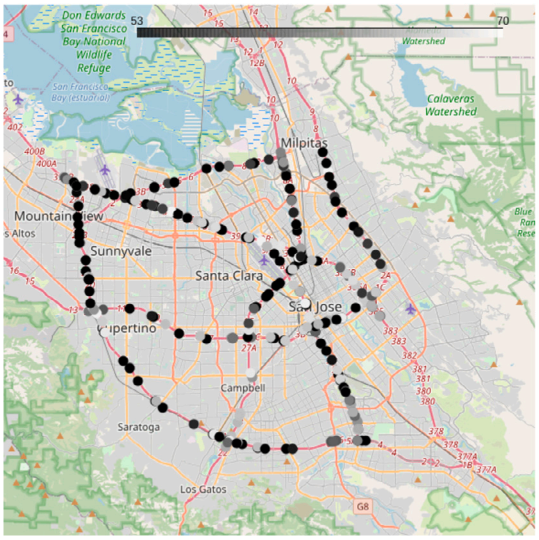
Fig. 11. Comparison between the average speed and the nodes used for layers 0 and layer 1 at 12pm.

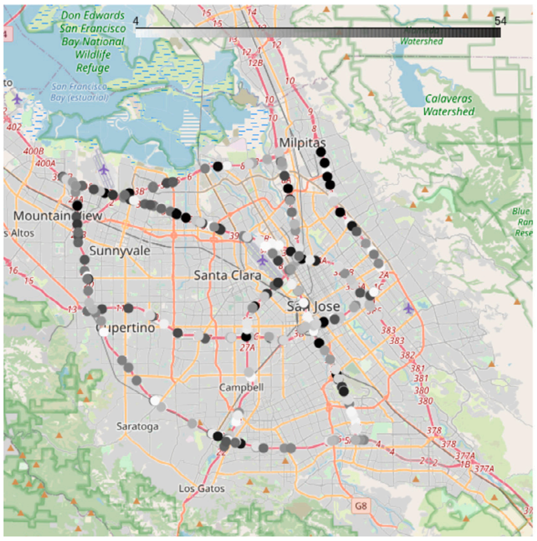
Fusion Graph Neural Networks, Spatial-Temporal Graph Ordinary Differential Equation Networks and Adaptive Graph Convolutional Recurrent Networks. The final conclusion is that the predictive model that best adapts to spatio-temporal traffic data is the Adaptive Graph Convolutional Recurrent Network (AGCRN).

However, the AGCRN predictive model behaves as a black box, requiring the use of a post-hoc explainability technique to better understand its behavior. At this point, due to the specificity and complexity of determining the optimal explainability method for Graph Convolutional Recurrent Network models, we focus on finding the method that provides the best performance with respect to sparsity and fidelity. We conclude that the sparsity and fidelity values for the GraphMask method are notably higher than with other explainability methods used.

The main aim of this article can be considered to be the integration of technologies based on Graph Convolutional Networks and on explainability (Graphmask) applied to traffic prediction. In order to test the combination capacity of predictive techniques based on deep learning and explainability methods, experiments were conducted on traffic density data in the city of San Jose (California).

The analysis and discussion of the results obtained reveal a logical behavior. The model uses only a few nodes, preferably in the central part, to make predictions in low traffic contexts, where prediction should be easier to perform. Moreover, in these low traffic contexts, the nodes used contain simpler information as they belong to layer 0. In turn, the model uses a greater amount of more complex information as traffic increases, which is to be expected, as more elements influence the prediction and more information is required. As an example, with the application of explainability through GraphMask, it was possible to mask about 95% of the nodes,


(a) Average speed per node at 9pm.


(b) Average number of nodes used in layer 0 at 9pm.

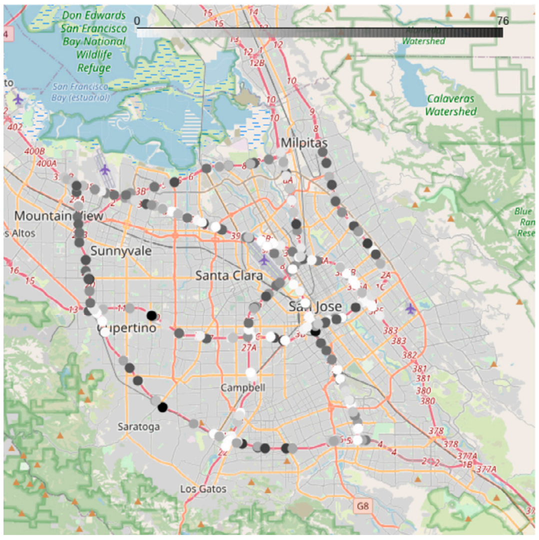

(c) Average number of nodes used in layer 1 at 9pm.

Fig. 12. Comparison between the average speed and the nodes used for layers 0 and 1 at 9pm.

with only a 7.78% increase in the *Mean Absolute Error* (MAE), as can be seen in the Table 4. This reduction of nodes allowed us to examine the masks obtained and so perform an analysis, which helped detect patterns in the performance of the model, allowing, in turn, for a better understanding of said model.

All the above provides confidence in the model, since it went from being a black box of whose working we had no knowledge, to a model in which we could intuit the elements that influence its behavior. Summarizing, the introduction of explainability in a predictive deep learning model allows for a better knowledge of its behavior and why the prediction has been made, thus increasing confidence in the model.

Based on these results, several avenues open up for future work. From the usability approach, it would be possible to study the use cases of explainability in GCNNs, for instance, to apply explainability in models before their use in real problems to verify that the model follows a logical behavior before making predictions that affect people. In this way, a model with explainability would be used for model validation and the original model would be employed for user predictions. Additionally, in the case of wishing to provide an explanation for each prediction, the model with explainability could be used to make a prediction together with an explanation for the user, for instance, indicating not only the best route to use, but also why that route is the best option. Another possible use would be to identify the most important nodes when traffic congestion occurs and thus assist in the process of improving the transportation network.

Regarding the improvement of the presented model, the reduction of the loss of accuracy in the explainability method could be explored, as well as combining the deep learning model with fuzzy systems to produce outputs closer to human logic, labeling the output as of low, medium or high traffic.

CRediT authorship contribution statement

The contribution of all the authors has been the same. We have all participated in the design and implementation of the research, the methodology, the formal analysis of the results and the writing of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the data at the manuscript.

References

[1] Mascha Van Der Voort, Mark Dougherty, Susan Watson, Combining Kohonen maps with arima time series models to forecast traffic flow, Transp. Res., Part C, Emerg. Technol. 4 (5) (1996) 307–318.

[2] Marco Lippi, Matteo Bertini, Paolo Frasconi, Short-term traffic flow forecasting: an experimental comparison of time-series analysis and supervised learning, IEEE Trans. Intell. Transp. Syst. 14 (2) (2013) 871–882.

[3] Wen Zhang, Shaoshan Yan, Jian Li, Tcp-bast: a novel approach to traffic congestion prediction with bilateral alternation on spatiality and temporality, Inf. Sci. 608 (2022) 718–733.

[4] B.L. Smith, M.J. Demetsky, Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 2, 1994, pp. 1706–1709.

[5] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.

[6] Ruhi Sarikaya, Geoffrey E. Hinton, Anoop Deoras, Application of deep belief networks for natural language understanding, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (4) (2014) 778–784.

[7] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.

[8] Ian Goodfellow, Mehdi Mirza, Aaron Courville, Yoshua Bengio, Multi-prediction deep Boltzmann machines, in: C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 26, Curran Associates, Inc., 2013.

[9] Guancen Lin, Aijing Lin, Danlei Gu, Using support vector regression and k-nearest neighbors for short-term traffic flow prediction based on maximal information coefficient, Inf. Sci. 608 (2022) 517–531.

[10] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to sequence learning with neural networks, Adv. Neural Inf. Process. Syst. 27 (2014).

[11] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[12] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu, Graph convolutional recurrent neural network: data-driven traffic forecasting, CoRR, arXiv:1707.01926, 2017.

[13] Xiaohui Huang, Yuming Ye, Weihua Ding, Xiaofei Yang, Liyan Xiong, Multi-mode dynamic residual graph convolution network for traffic flow prediction, Inf. Sci. 609 (2022) 548–564.

[14] Zhijun Chen, Zhe Lu, Qiushi Chen, Hongliang Zhong, Yishi Zhang, Jie Xue, Chaozhong Wu, Spatial–temporal short-term traffic flow prediction model based on dynamical-learning graph convolution mechanism, Inf. Sci. 611 (2022) 522–539.

[15] Zachary C. Lipton, The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery, ACM Queue 16 (3) (2018) 31–57.

[16] Haifeng Li, Jun Cao, Jiawei Zhu, Yu Liu, Qing Zhu, Guohua Wuo, Curvature graph neural network, Inf. Sci. 592 (2022) 50–66.

[17] Alain Manzo-Martinez, Fernando Gaxiola, Graciela Ramirez-Alonso, Fernando Martinez-Reyes, A comparative study in machine learning and audio features for kitchen sounds recognition, Comput. Sist. 26 (2) (2022) 603–621.

[18] David Silva, Alain Manzo-Martinez, Fernando Gaxiola, Luis Gonzalez-Gurrrola, Graciela Ramirez-Alonso, Analysis of CNN architectures for human action recognition in video, Comput. Sist. 26 (2) (2022) 623–641.

[19] Minhao Zou, Zhongxue Gan, Ruizhi Cao, Chun Guan, Siyang Leng, Similarity-navigated graph neural networks for node classification, Inf. Sci. 633 (2023) 41–69.

[20] Yongliang Wu, Yue Fu, Jiwei Xu, Hu Yin, Qianqian Zhou, Dongbo Liu, Heterogeneous question answering community detection based on graph neural network, Inf. Sci. 621 (2023) 652–671.

[21] Peibo Li, Yixing Yang, Maurice Pagnucco, Yang Song, Explainability in graph neural networks: an experimental survey, arXiv preprint, arXiv:2203.09258, 2022.

[22] Weiping Ding, Mohamed Abdel-Basset, Hossam Hawash, Ahmed M. Ali, Explainability of artificial intelligence methods, applications and challenges: a comprehensive survey, Inf. Sci. (2022).

[23] L-X. Wang, Jerry M. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Syst. Man Cybern. 22 (6) (1992) 1414–1427.

[24] Saptarshi Sengupta, Sanchita Basak, Pallabi Saikia, Sayak Paul, Vasilios Tsalavoutis, Frederick Atiah, Vadlamani Ravi, Alan Peters, A review of deep learning with special emphasis on architectures, applications and recent trends, Knowl.-Based Syst. 194 (2020) 105596.

[25] Yunhu Huang, Dewang Chen, Wendi Zhao, Hong Mo, Deep fuzzy system algorithms based on deep learning and input sharing for regression application, Int. J. Fuzzy Syst. 23 (2021) 727–742.

[26] Rangan Das, Sagnik Sen, Ujjwal Maulik, A survey on fuzzy deep neural networks, ACM Comput. Surv. 53 (3) (2020) 1–25.

[27] Fernando Gaxiola, Patricia Melin, Fevrier Valdez, Oscar Castillo, Interval type-2 fuzzy weight adjustment for backpropagation neural networks with application in time series prediction, Inf. Sci. 260 (2014) 1–14.

[28] Fernando Gaxiola, Patricia Melin, Fevrier Valdez, Oscar Castillo, Generalized type-2 fuzzy weight adjustment for backpropagation neural networks in time series prediction, Inf. Sci. 325 (2015) 159–174.

[29] Leyao Wang, Yijie Ding, Prayag Tiwari, Junhai Xu, Wenhuan Lu, Khan Muhammad, Victor Hugo C. de Albuquerquee, Fei Guo, A deep multiple kernel learning-based higher-order fuzzy inference system for identifying DNA N4-methylcytosine sites, Inf. Sci. 630 (2023) 40–52.

[30] Phu Pham, Loan T.T. Nguyen, Ngoc Thanh Nguyen, Robert Kozma, Bay Vo, A hierarchical fused fuzzy deep neural network with heterogeneous network embedding for recommendation, Inf. Sci. 620 (2023) 105–124.
[31] Yuanhang Zheng, Zeshui Xu, Xinxin Wang, The fusion of deep learning and fuzzy systems: a state-of-the-art survey, IEEE Trans. Fuzzy Syst. 30 (8) (2022) 2783–2799.
[32] Dongfang Ma, Bowen Sheng, Xiaolong Ma, Sheng Jin, Fuzzy hybrid framework with dynamic weights for short-term traffic flow prediction by mining spatio-temporal correlations, IET Intell. Transp. Syst. 14 (2) (2020) 73–81.
[33] Ángel Delgado-Panadero, Beatriz Hernández-Lorca, María Teresa García-Ordás, José Alberto Benítez-Andrades, Implementing local-explainability in gradient boosting trees: feature contribution, Inf. Sci. 589 (2022) 199–212.
[34] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, Zhanfeng Jia, Freeway performance measurement system: mining loop detector data, Transp. Res. Rec. 1748 (1) (2001) 96–102.
[35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Chengqi Zhang, Graph wavenet for deep spatial-temporal graph modeling, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.
[36] Bing Yu, Haoteng Yin, Zhanxing Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018.
[37] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, Huaiyu Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, Proc. AAAI Conf. Artif. Intell. 33 (01) (2019) 922–929.
[38] Chao Song, Youfang Lin, Shengnan Guo, Huaiyu Wan, Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting, Proc. AAAI Conf. Artif. Intell. 34 (01) (2020) 914–921.
[39] Mengzhang Li, Zhanxing Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, Proc. AAAI Conf. Artif. Intell. 35 (5) (2021) 4189–4196.
[40] Zheng Fang, Qingqing Long, Guojie Song, Kunqing Xie, Spatial-temporal graph ode networks for traffic flow forecasting, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021.
[41] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, Can Wang, Adaptive graph convolutional recurrent network for traffic forecasting, CoRR, arXiv:2007.02842, 2020.
[42] Chirag Agarwal, Marinka Zitnik, Himabindu Lakkaraju, Probing GNN explainers: a rigorous theoretical and empirical analysis of GNN explanation methods, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 8969–8996.
[43] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, Ce Zhang, Graphframex: towards systematic evaluation of explainability methods for graph neural networks, arXiv preprint, arXiv:2206.09677, 2022.
[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, Graph attention networks, arXiv preprint, arXiv:1710.10903, 2017.
[45] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[46] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, Yaliang Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[47] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, Jure Leskovec, Gnnexplainer: generating explanations for graph neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).
[48] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, Xiang Zhang, Parameterized explainer for graph neural network, Adv. Neural Inf. Process. Syst. 33 (2020) 19620–19631.
[49] Michael Sejr Schlichtkrull, Nicola De Cao, Ivan Titov, Interpreting graph neural networks for NLP with differentiable edge masking, in: International Conference on Learning Representations, 2021.
[50] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR, arXiv:1412.3555, 2014.