NETXPLAIN: REAL-TIME EXPLAINABILITY OF GRAPH NEURAL NETWORKS APPLIED TO COMPUTER NETWORKS

David Pujol-Perich [1] **José Suárez-Varela** [1] **Shihan Xiao** [2] **Bo Wu** [2] **Albert Cabellos-Aparicio** [1] **Pere Barlet-Ros** [1]

ABSTRACT

Recent advancements in Deep Learning (DL) have revolutionized the way we can efficiently tackle complex optimization problems. However, existing DL-based solutions are often considered as black boxes due to their high inner complexity. As a result, there is still certain skepticism among the computer network industry about their practical viability to operate data networks. In this context, explainability techniques have recently emerged to unveil *why DL models make each decision*. This paper focuses on Graph Neural Network (GNN) models applied to computer networks, which have already shown outstanding performance in different network optimization tasks. We thus present NetXplain, a novel real-time explainability solution that uses *a GNN to interpret the output produced by another GNN*. In the evaluation, we apply the proposed explainability method to RouteNet –a GNN model that predicts end-to-end performance metrics in computer networks. We show that NetXplain operates more than 3 orders of magnitude faster than state-of-the-art explainability solutions when applied to networks up to 24 nodes, which makes this solution compatible with real-time applications. Moreover, it demonstrated strong generalization capabilities over different network scenarios unseen during training.

1 INTRODUCTION

In recent years, Deep Learning (DL) has revolutionized the way we can to solve a vast number of problems – e.g., (Silver et al., 2016; Vinyals et al., 2019) – by finding meaningful patterns on large amounts of data. One main limitation of DL-based solutions, however, is that they offer probabilistic performance guarantees, which typically degrade as the data deviates from the distribution observed during training. Moreover, due to the high complexity of the internal architectures of Neural Networks (NN), they are often treated as black boxes (Meng et al., 2020). This limits the viability of applying these solutions to non-fault-tolerant systems such as computer networks, as these are critical infrastructures where it is essential to deploy fully reliable solutions.

In this context, explainability solutions (Samek et al., 2017; Bau et al., 2017) have recently emerged as practical tools to provide human-readable interpretations of complex DL models. This knowledge enables not only to produce more mature and reliable DL models, but also to enhance their performance by making ad-hoc adjustments. More specifically, these solutions analyze trained DL models from a black-box perspective – i.e., they only observe their inputs and outputs – and aim to identify which input elements mainly determine the model's output.

At the same time, the last few years have seen the explosion of Graph Neural Networks (GNN) (Scarselli et al., 2008), a new neural network family that has shown unprecedented generalization capabilities over graphs of different sizes and structures. These models have already attracted large interest given their numerous applications to different fields where the information is fundamentally represented as graphs –e.g., (Gilmer et al., 2017; Battaglia et al., 2016; Zitnik et al., 2018; Fan et al., 2019). In this context, GNNs have also proven to be specially suitable for applications in computer networks, particularly for network control and management, as most of the elements involved in these problems are fundamentally represented as graphs –e.g., topology, routing (Rusek et al., 2019; Geyer and Carle, 2018; Mao et al., 2019; Almasan et al., 2019). However, the black-box nature of GNN-based solutions represents nowadays a major barrier to achieve their adoption in real-world networks, as potential malfunctions can lead to temporal service disruptions with serious economic damages for network operators.

Explainability of GNNs has been recently explored in two main works. A first work emerging from the ML community (Ying et al., 2018) analyzes a well-known GNN model applied to several problems –e.g. chemistry (Debnath et al.,

[1]Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain [2]Network Technology Lab., Huawei Technologies Co.,Ltd.. Correspondence to: David Pujol-Perich <david.pujol.perich@upc.edu>.

Proceedings of the First MLSys Workshop on Graph Neural Networks and Systems (GNNSys'21), San Jose, CA, USA, 2021.

1991). Likewise, the computer networks community has made a first attempt to apply a similar approach to GNN-based network optimization solutions (Meng et al., 2020). However, both solutions are based on costly iterative optimization algorithms that are executed individually on each input sample to obtain interpretations. Hence, they do not meet the requirements to make comprehensive analysis over large datasets and, more importantly, to be used in real-time applications.

To address these limitations, this paper proposes NetXplain, a novel real-time *explainability solution for GNNs*. This solution uses a GNN that learns –from Tabula Rasa– how to interpret the outputs produced by another GNN –trained for a specific purpose. NetXplain produces human-readable interpretations of GNNs comparable to state-of-the-art solutions (Ying et al., 2018; Meng et al., 2020). However, it achieves this at a much more limited cost. In our evaluation, we apply NetXplain to RouteNet (Rusek et al., 2019) – a GNN model used to predict the delay of traffic flows in computer networks. Our evaluation results reveal the possibility to produce interpretations over a wide variety of network scenarios after training NetXplain over a reduced dataset generated by costly explainability solutions –e.g., (Ying et al., 2018; Meng et al., 2020). Particularly, we first train NetXplain on a small dataset with samples produced by Metis (Meng et al., 2020). Then, we test the generalization power of our GNN-based method when applied to network scenarios fundamentally different to those seen during training. Likewise, we show that NetXplain far outperforms state-of-the-art algorithms in terms of computational cost, running more than 3 orders of magnitude faster on average than Metis (Meng et al., 2020) when applied to samples of three real-world network topologies (up to 24 nodes). This eventually enables to make comprehensive analysis of GNN solutions at limited cost and, more importantly, to integrate NetXplain with real-time network optimization solutions to improve their performance – as discussed in Section 6.

2 Graph Neural Networks applied to computer networks

The strong generalization capabilities of GNN over graphs make these models interesting for applications in the computer networks field, since the most natural way to formalize many network control and management problems involves the use of graphs – e.g., topology, routing, inter-flow dependencies (Meng et al., 2020). In this regard, several GNN-based solutions have been proposed to tackle different use cases – e.g., network modeling (Rusek et al., 2019; Badia-Sampera et al., 2019), automatic routing protocols (Geyer and Carle, 2018). In this section, for illustrative purposes, we focus on RouteNet (Rusek et al., 2019), as it is quite representative of how GNN-based solutions process network-related data to solve complex problems.

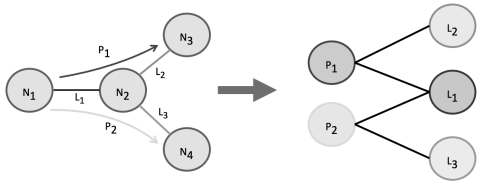

Figure 1: Transformation from the physical network scenario to the graph representation of RouteNet.

RouteNet targets the problem of modeling per-flow Quality-of-Service metrics in networks (e.g., delay, jitter). For this purpose, a network snapshot is provided as input –i.e., a network topology, a routing configuration, and a traffic matrix. To this end, this model makes a transformation of the physical network scenario into a refined graph representation in which physical and logical elements are explicitly represented –*paths* and *links* in this case. More specifically, every *link* of the physical network topology and every source-destination path is transformed into a node in the input graph of the GNN. Finally, edges connect *links* with *paths* according to the routing configuration. Thus, each path is connected to those links that it traverses given the input routing scheme. This process is illustrated in Fig. 1, where we can observe how a physical network scenario with two paths and three links is transformed into the input graph of RouteNet. This graph representation enables to model the circular dependencies between the state of paths and links, and how they relate to the output network performance metrics (e.g., delay).

In this context, applying explainability over this model would enable to identify the most critical edges of its input graphs (i.e., path-link relations). These critical edges thus represent the set of path-link pairs that mostly affect the per-flow QoS metrics produced by RouteNet. As a result, explainability solutions would enable to extract relevant knowledge of the processing made by the GNN given a network scenario, which can eventually enable many interesting applications for networks – as discussed later in Section 6.

3 Related work

Recent years have attracted increasing interest in producing explainability solutions for neural network models – e.g., *Convolutional Neural Networks* (Bau et al., 2017). Despite this, explainability techniques for GNN have been scarcely explored so far. In this context, GNNExplainer (Ying et al., 2019) is, to the best of our knowledge, the first proposal approaching this problem.

GNNExplainer is given as input a GNN model and a sample

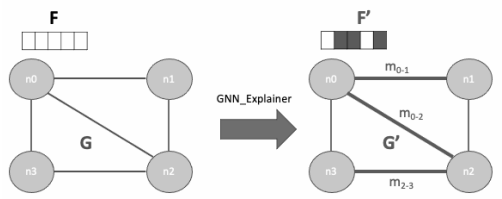

Figure 2: Schematic description of explainability solutions for GNN (e.g., GNNExplainer).

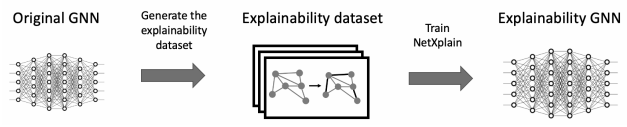

Figure 3: High-level workflow of NetXplain.

graph G that feeds this model, and it produces as output a subset with the connections $G' \subset G$ and the node features $F' \subset F$ that affect most critically the output of the target GNN (see Fig. 2). This is done by computing a set of weights $w_{i,j}$ that represent how critical are the pair-wise connections of input graphs to the prediction accuracy of the target GNN. Particularly, most relevant connections are those that have more impact on the loss function used to train the model (e.g., Mean Squared Error for regression tasks). The amount of relevant connections produced by the algorithm can be tuned by setting some parameters.

Overall, GNNExplainer is an explainability solution that targets the explainability of any GNN applied to classification tasks (e.g., graph-level, node-level, edge-level classification). However, this solution does not support GNN-based models used for regression. In this context, a posterior solution proposed from the computer networks community presents Metis (Meng et al., 2020), a similar approach adapted to GNN models trained for regression problems. Particularly, this work showcases the use of this solution over several network applications.

Although GNNExplainer (Ying et al., 2019) and Metis (Meng et al., 2020) are able to produce quality explainability solutions for a vast range of problems, both have an important limitation. To compute G' and F' for each input sample, these solutions use a time-consuming iterative convex optimization method. For instance, producing a single explainability solution can take up to hundreds of seconds in scenarios with topologies between 14 and 24 nodes –as shown later in Section 5. This arguably prevents these methods to be used for real-time applications. Moreover, their high cost makes them impractical to perform a comprehensive test analysis of GNN-based solutions – covering a wide range of network scenarios – before such solutions are released to the market.

4 NetXplain: Proposed Explainability Method

In this section, we introduce NetXplain, a novel explainability method for GNN that is compatible with real-time applications. NetXplain is able to produce the same output as state-of-the-art solutions based on costly iterative optimization algorithms (Ying et al., 2019; Meng et al., 2020), while operating at much limited cost – at the scale of a few milliseconds in our experiments in Section 5. This not only enables to perform real-time troubleshooting of GNN-based solutions, but also opens the possibility of combining these solutions with automatic optimization algorithms (e.g., Local Search, Reinforcement Learning) to solve more efficiently online optimization problems –as discussed later in Section 6. To this end, *NetXplain uses a GNN that learns how to interpret a target GNN model* that was trained for a particular task. As shown in Fig. 3, the proposed GNN-based solution is trained with a dataset generated by an existing explainability solution (Meng et al., 2020) and, once trained, the resulting model is able to make one-step explainability predictions for each input sample of the target GNN. Note that thanks to the generalization capabilities of GNN over graph-structured information, once NetXplain is trained over a particular target GNN solution, it can be applied to different input graphs not included in the training dataset. In practice, when applied to GNN-based network solutions, NetXplain is able to generalize to network scenarios with topologies of variable size and structure not seen in advance –as shown later in the experiments of Section 5. The following subsections describe in more detail the main components of this solution.

4.1 Explainability mask

We refer to the *explainability mask* as an $n \times n$ matrix that defines the relevance of each edge of an input graph $G = (V, E)$ on the output produced by the target GNN, where $n = |V|$. This mask enables to interpret which are the graph connections that mostly affect the predicting power of the GNN in each case.

Formally, given an input graph $G = (V, E)$, the proposed explainability method aims to produce an explainability mask $M \in \{0, 1\}^{|V| \times |V|}$, where cell (u, v) defines a weight $w_{u,v}$ indicating the importance of the connection between node u and node v on the overall accuracy of the target GNN. Particularly, M contains a weight for each pair $(u, v) \in E$. Fig. 4 further illustrates how the explainability mask is built from a given input (undirected) graph.

4.2 Explainability dataset

To train NetXplain, we first need to generate a training dataset D with a set of input samples $s \in S$ and their

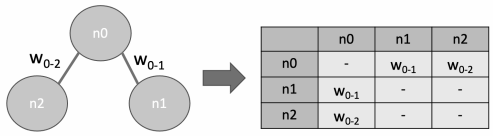

Figure 4: Explainability mask of an input graph.

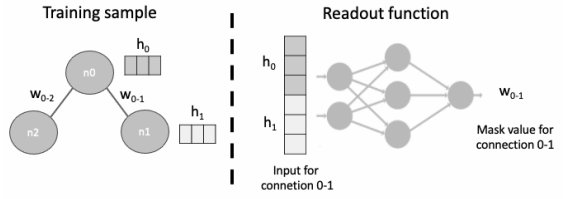

Figure 5: Adaptation of the readout function in NetXplain to produce the explainability mask.

associated explainability masks M when applied to the target GNN. Note that this process is made from a black-box perspective –i.e., the explainability mask interprets the relevance of the input graph connections by analyzing the input-output correlations in the target GNN. To this end, we can use specific state-of-the-art iterative optimization algorithms as those described in Section 3, depending on the particularities and the purpose of the target GNN (e.g., regression, classification). This kind of solutions work as follows: given a target GNN, which has been trained with a dataset D, the explainability algorithm applies an iterative gradient descent method to find the optimal explainability mask M^* that better explains the accuracy of the model – i.e., it defines weights $w_{u,v}$ that represent the impact of each graph edge (u, v) on the loss function of the target GNN. More specifically, the calculation of the explainability mask is driven by the loss function $D(M)$ of Equation 1, which depends on three factors: (i) predictive loss, (ii) entropy of the values in the mask, and (iii) L1 regularization computed over the mask. The predictive loss quantifies how the accuracy of the target GNN ($V_{original}$) degrades when weighting the connections according to M (V_{mask}). The two remaining factors, entropy (Equation 2) and L1 regularization, regulate the homogeneity of the mask's values and the portion of critical connections respectively. These latter factors can be weighted according to two hyper-parameters (i.e., α, β) that can be fine-tuned according to the problem needs and the target GNN, to eventually achieve masks that can be easily interpretable by humans.

Through a gradient descent method, these algorithms gradually converge to the optimal mask M^* that minimizes the loss function (Eq. 1).

$$D(M) = l(V_{original}, V_{mask}) + \alpha H(M) + \beta ||M||_{L1} \quad (1)$$

$$H(M) = -\sum_{(u,v)\in E} w_{u,v} \log(w_{u,v}) + (1 - w_{u,v}) \log(1 - w_{u,v}) \quad (2)$$

We then repeat this process for a small subset of samples $A \subset D$, formally defined in Equation 3.

$$D' = \{(s, M^*) | s \in A\} \quad (3)$$

Note that D' contains a small fraction of the original dataset D, making the cost of generating the explainability dataset affordable.

4.3 Training the explainability GNN

Finally, we propose the use of an independent GNN (NetXplain) to learn how to predict explainability masks M over the target GNN.

First, let us define the underlying architecture of this GNN. For this purpose, we keep the same architecture of the target GNN and make some small modifications. The intuition behind this decision is that the complexity for the target GNN to learn how to make its output predictions should be similar to solving the explainability problem over that GNN –i.e., explaining which connections affected most such predictions. However, we make a minor change on the readout function $r(\cdot)$ to adapt it to produce the explainability mask M. As illustrated in Fig. 5, for every edge $(u, v) \in E$, we concatenate the hidden-state vectors of these nodes after the message passing phase is finished (i.e., $h_u||h_v$) and this is passed as input to $r(\cdot)$, which predicts the mask weight for that edge $w_{u,v}$. Note that this operation can be computed in parallel for each edge $(u, v) \in E$ of the input graph G.

As introduced in Section 4.2, one key aspect of our proposal is to reduce as much as possible the size of the training dataset (D') to make this approach feasible. To achieve this, we adopt a Transfer Learning approach by reusing the weights of the original target GNN, except for the readout function, whose architectures differ slightly (Fig. 5). This enables to effectively initialize the explainability GNN model, as the message-passing functions of this GNN is expected to be close to those of the target GNN (e.g., similar feature distributions on the input). Finally, we train the explainability model with a reduced dataset D' generated by a reference explainability algorithm, and this enables to learn how to produce accurately explainability masks with low cost (both for training and inference).

4.4 Generalization power of NetXplain

By analysing the training process of NetXplain, we identify the generation of the dataset D' as the most computationally expensive task, even considering that the size of D' is only a small portion of the original dataset D.

Note that our proposal aims to learn how to explain potentially any sample that our target GNN could face during

operation. This motivates our choice of using a GNN to explain a target GNN, as they are specially applicable given its high generalization power over graph-structured data. As a result, once trained, the GNN explainability model generalizes to network scenarios not present in its training dataset D'. This means that NetXplain's GNN can be trained over a small dataset to make predictions of the critical connections from the perspective of the target GNN and, once trained, it can predict these critical connections over arbitrary network scenarios (e.g., topologies of variable size and structure). All this offering an accuracy comparable to state-of-the-art costly solutions.

5 EVALUATION

In this section, we first evaluate the accuracy of the predictions made by NetXplain with respect to the state-of-the-art solution Metis (Meng et al., 2020). Then, we quantify the speed-up when using NetXplain compared to Metis. In our experiments, we train an explainability model that makes interpretations over RouteNet (Rusek et al., 2019) – previously introduced in Section 2.

5.1 Generation of the explainability model

This section first defines the process to generate the explainability dataset and then the architecture of the GNN explainability model used by NetXplain.

We generated the explainability dataset using Metis (Meng et al., 2020). We iteratively applied this algorithm to a subset of samples from the NSFNet dataset (BNN Center, 2021) used in RouteNet (Rusek et al., 2019). Particularly, we limited Metis to run 2,000 gradient-descent iterations per sample, after observing this was enough to ensure convergence. After some experimentation, we observed that using only 5% of samples randomly selected from the original dataset was enough to train NetXplain (i.e., 15k samples). Moreover, the explainability dataset was divided into training, validation and test datasets by randomly picking 80%, 10% and 10% of the samples, respectively.

As previously mentioned in Section 4.3, we use for the explainability GNN a similar architecture to the target GNN, RouteNet (Rusek et al., 2019) in this case. However, we introduce a change in the readout function. In this case, outputs are the weights $w_{i,j}$ of a link-path edge in the input graph of RouteNet (see Fig. 1). To this end, we concatenate the corresponding hidden states of $link_i$ (h_{li}) and $path_j$ (h_{pj}), and introduce this as input of the readout function. The output of the readout is thus the weight w_{ij}, which can be interpreted as quantifying the importance for RouteNet of a particular src-dst path as it passes through a certain link in the network.

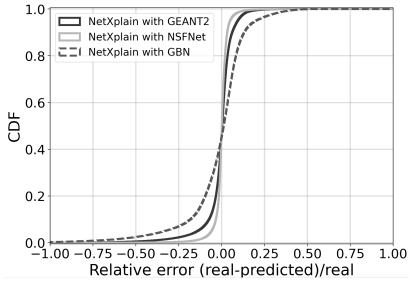

Figure 6: CDF of the relative error of NetXplain evaluated on three real-world network topologies.

5.2 Evaluation of the accuracy

We evaluate the accuracy achieved by the NetXplain model on samples simulated in three real-world topologies (BNN Center, 2021): *NSFNet* (14 nodes), *GEANT2* (24 nodes), and *GBN* (17 nodes). Figure 6 depicts the Cumulative Distribution Function (CDF) of the relative error when predicting the *explainability mask* over 1,000 samples not seen during training, for each topology respectively. We observe that our explainability model achieves a Mean Relative Error (MRE) of 2.4% when it is trained and evaluated over samples of the NSFnet topology (14 nodes). We then repeat the same experiment training and evaluating the model with samples of Geant2 (24 nodes), and observe a MRE of 4.5%. Note that despite NetXplain was trained and evaluated over samples of the same topology, the network scenarios (i.e., routing and traffic matrices) are different across the training and evaluation samples, which means that the input graphs seen by the GNN in the evaluation phase are different from those observed during training. Finally, we further test the generalization capabilities of NetXplain by training the explainability GNN with samples from *NSFNet* and *GEANT2*, but in this case we evaluate the model on samples of a different network with 17 nodes (*GBN*). As a result, NetXplain achieves a MRE of 11% over this network topology unseen in advance – dashed line in Figure 6. All these results are in line with the generalization numbers already observed in the target GNN model (Rusek et al., 2019).

These results show that with NetXplain, we can achieve a similar output to state-of-the-art solutions based on iterative optimization – even when we tested it over network scenarios not seen during training.

5.3 Evaluation of the execution cost

In this section, we evaluate the inference time to compute an explainability mask with NetXplain with respect to state-of-the-art solutions – Metis (Meng et al., 2020) in this case. This was done by randomly selecting 500 samples from the three datasets previously used: *NSFNet*, *GEANT2* and *GBN* (BNN Center, 2021).

Table 1: Execution time of NetXplain with respect to Metis, evaluated on three real-world network topologies.

Topology	Method	Mean (s)	Std deviation (s)
NSFNet	Benchmark (Metis)	98.139	2.455
	NetXplain	0.012	0.001
GBN	Benchmark (Metis)	150.83	1.79
	NetXplain	0.0214	0.005
GEANT2	Benchmark (Metis)	191.46	2.76
	NetXplain	0.029	0.002

Table 1 summarizes the execution times (in seconds) differentiated over samples of the three network topologies. Note that both solutions were executed in CPU and in equal conditions. As we can observe, NetXplain achieves an average speed-up of ≈7,200x across all the network topologies (i.e., it is more than 3 orders of magnitude faster).

This shows the benefits of NetXplain with respect to state-of-the-art solutions, as it can be used to make extensive explainability tests at limited cost. More importantly, its operation at the scale of milliseconds makes it compatible with real-time applications for computer networks.

6 Discussion on possible applications

This section discusses two main use-case categories where the application of GNN-based explainability solutions can be especially beneficial for computer networks: (i) Test & troubleshooting, and (ii) Improving network optimization tasks. Particularly, we put the focus on the advantages of leveraging the fast and low-cost interpretations of NetXplain with respect to state-of-the-art explainability methods.

6.1 Test & Troubleshooting

To create GNN-based solutions for computer networks, we need guarantees that they will work optimally when deployed in real-world networks. In this context, manufacturers would typically need to make extensive tests to their GNN solutions to check how they generalize to different network conditions. Using NetXplain would enable to collect human-readable interpretations of the internal data processing made by GNNs. For instance, we can identify the network elements that mainly drive the decisions made by the model –which are given by the explainability mask of NetXplain– and then observe if the properties of the selected elements are consistent across similar network scenarios. This would be a good indicator that the model generalizes well and, consequently, it is reliable for deployment. In this context, making such a comprehensive analysis using state-of-the-art explainability solutions would result in large costs for manufacturers; while the limited cost of NetXplain would enable to reduce dramatically both the cost and the time needed before releasing the product to the market.

6.2 Improving network optimization solutions

Network optimization often requires to deal with very high-dimensional action spaces –e.g., all the valid src-dst routing combinations in networks up to thousands of nodes. As a result, network optimization tools only evaluate a small portion of configurations before they make a final decision. Thus, the exploration strategy used by these tools has a critical impact on the performance they can eventually achieve.

In this context, explainability methods can provide meaningful interpretations of the current network state that can be useful to guide more efficiently optimization algorithms – e.g., Reinforcement Learning (Almasan et al., 2019), Local Search (Gay et al., 2017). For instance, using a NetXplain model trained over RouteNet –as the one of Section 5– would enable to point to critical paths and links that are mostly affecting the network performance. This could be highly beneficial for optimization algorithms to explore alternative configurations targeting specifically these critical points. In this context, computational efficiency is a must for optimization tools, as it directly affects to the amount of configurations that can be evaluated before producing the final decision. Thus, counting on real-time solutions, like NetXplain, offers an important competitive advantage with respect to state-of-the-art explainability methods.

7 Conclusions

In this paper, we proposed NetXplain, an efficient explainability solution for GNN. NetXplain uses a GNN that *learns* how to produce accurate interpretations over the outputs produced by another GNN model. In contrast to state-of-the-art explainability solutions based on costly optimization algorithms, the proposed solution can be integrated into network control and troubleshooting systems operating in real time. We tested NetXplain over RouteNet –a GNN model that predicts per-flow delays in computer networks– and showed that our solution can produce an output equivalent to state-of-the-art solutions with an execution time more than 3 orders of magnitude shorter (in networks up to 24 nodes). Lastly, we discussed the potential applications that can have this real-time GNN-based explainability solution particularly for computer networks.

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme within the framework of the NGI-POINTER Project funded under grant agreement No. 871528. This paper reflects only the author's view; the EC is not responsible for any use that may be made of the information it contains. This work was also supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE) and the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

Almasan, P., Suárez-Varela, J., Badia-Sampera, A., Rusek, K., Barlet-Ros, P., Cabellos-Aparicio, A., 2019. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. arXiv preprint arXiv:1910.07421 .

Badia-Sampera, A., et al., 2019. Towards more realistic network models based on graph neural networks, in: Proceedings of the ACM CoNEXT student workshop, pp. 14–16.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al., 2016. Interaction networks for learning about objects, relations and physics, in: Advances in neural information processing systems (NIPS), pp. 4502–4510.

Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A., 2017. Network dissection: Quantifying interpretability of deep visual representations, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6541–6549.

BNN Center, 2021. Network modeling datasets. URL: https://github.com/knowledgedefinednetworking/NetworkModelingDatasets.

Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C., 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry 34, 786–797.

Fan, W., et al., 2019. Graph neural networks for social recommendation, in: The ACM World Wide Web Conference (WWW), pp. 417–426.

Gay, S., Hartert, R., Vissicchio, S., 2017. Expect the unexpected: Sub-second optimization for segment routing, in: IEEE INFOCOM 2017, pp. 1–9.

Geyer, F., Carle, G., 2018. Learning and generating distributed routing protocols using graph-based deep learning, in: Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, pp. 40–45.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 .

Mao, H., Schwarzkopf, M., Venkatakrishnan, S.B., Meng, Z., Alizadeh, M., 2019. Learning scheduling algorithms for data processing clusters, in: Proceedings of ACM SIGCOMM, pp. 270–288.

Meng, Z., Wang, M., Bai, J., Xu, M., Mao, H., Hu, H., 2020. Interpreting deep learning-based networking systems, in: Proceedings of ACM SIGCOMM, pp. 154–171.

Rusek, K., Suárez-Varela, J., Mestres, A., Barlet-Ros, P., Cabellos-Aparicio, A., 2019. Unveiling the potential of graph neural networks for network modeling and optimization in sdn, in: Proceedings of the 2019 ACM Symposium on SDN Research, pp. 140–151.

Samek, W., Wiegand, T., Müller, K.R., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 .

Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G., 2008. The graph neural network model. IEEE Transactions on Neural Networks 20, 61–80.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., 2016. Mastering the game of go with deep neural networks and tree search. Nature 529, 484–489.

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al., 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 575, 350–354.

Ying, R., et al., 2018. Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 974–983.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J., 2019. Gnnexplainer: Generating explanations for graph neural networks, in: Advances in neural information processing systems, pp. 9244–9255.

Zitnik, M., Agrawal, M., Leskovec, J., 2018. Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics 34, i457–i466.