Modeling Traffic Congestion using Graph Convolutional Networks

Madhwaraj Kango Gopal
Department of MCA
New Horizon College of Engineering(VTU)
Bangalore, India
dr.madhwaraj@newhorizonindia.edu

A. Asha
Department of MCA
New Horizon College of Engineering(VTU)
Bangalore, India
asha.gurudath@gmail.com

Arpana Prasad
Department of MCA
New Horizon College of Engineering(VTU)
Bangalore, India
arpanaprasad2013@gmail.com

Akahaury Nimitt Verma
Department of MCA
New Horizon College of Engineering(VTU)
Bangalore, India
nimittv6@gmail.com

Aditya Venkat Ganesh P
Department of MCA
New Horizon College of Engineering(VTU)
Bangalore, India
ganeshpadityaedu@gmail.com

Abstract— **Congestion in a network can cause data packet losses, delays, and reduced network performance. To prevent congestion, network engineers must be able to accurately predict and manage network traffic. In this research paper, we explore the use of Graph Convolutional Networks (GCN) for predicting congestion in a network. GCN is a type of deep learning algorithm that can analyze complex network structures to predict the behavior of nodes in a network. With GCN, predicting congestion in a network, identification of potential congested areas becomes a reality. Proactive measures to prevent congestion from occurring is also been attempted in this research work. The results of our experiments demonstrate that GCN outperforms other conventional machine learning techniques in predicting network congestion with high accuracy and precision using ReLU6, which was the most suitable activation function for implementing the model. This research also demonstrates the potential of using deep learning algorithms such as GCN to improve network management and optimize network performance.**

Keywords—congestion, networks, deep learning, graph convolutional networks

I. Introduction

Graph Convolutional Networks (GCN) have recently become popular for analysing network structures due to their ability to model complex relationships among network nodes. GCN has shown its potential to solve a wide range of tasks such as node classification, link prediction, and graph clustering. In this paper, we propose to apply GCN to predict congestion in a network. Congestion is a significant problem in computer networks that can result in severe degradation of network performance. Congestion happens when the available network resources become insufficient to meet the demand for data transmission. Congestion can lead to increased packet loss, delay, and throughput degradation, which can cause network applications to fail. Therefore, detecting and predicting congestion is a critical task for network operators to ensure the smooth functioning of the network. Traditionally, congestion detection in computer networks has been done using analytical or rule-based approaches, which can be inefficient and imprecise. These approaches rely on simple threshold-based methods, which do not consider the dynamic nature of the network. With the advancement of machine learning, researchers have explored different techniques to predict network congestion, such as supervised learning and unsupervised learning [1]. GCN uses graph convolution operations to aggregate information from neighbouring nodes and representations of nodes in the graph. This makes it particularly suitable for network analysis, as network nodes can be represented as graph nodes, and their relationships can be modelled as edges in the graph. By using GCN, the underlying structure of the network and the prediction of congestion is more accurate than traditional methods. Interpretability is essential for network operators to understand the causes of congestion and taking appropriate actions to mitigate it [2]. With the use visualization techniques to visualize the learned representations of nodes in the graph, identification of the most influential nodes in predicting congestion can also be accomplished. Section II discusses on the related literature work that has been done. The concepts of GCN are discussed in Section III. Section IV illustrates the implementation of GCN. Section V discusses the case study. Section VI depicts the results. Section VII summarizes the conclusions and future work.

II. Literature Review

Several machine learning algorithms are being used today due to the benefits they offer like prediction, analysis and model building etc… Madhwaraj & Amirthavalli [3] used machine learning techniques to predict the maintainability of open source software. Viswanath et al. [4] performed a case study to predict the number of deaths due to dengue disease. Madhwaraj et al. [5] identified a novel machine learning idea to predict the sale of washing machines in an organization. Machine learning algorithms are used for a variety of purposes. While considering network analysis, congestion issues and other areas, GCN have been used in a variety of fields, such as transportation network analysis, bioinformatics, and social network analysis. Due to their capacity to discover intricate relationships within graph-structured data, GCN have attracted a lot of attention lately in the transportation industry to forecast network congestion, a significant issue for urban transportation planning and management etc. This literature review provides an overview of the application of GCN using deep learning to predict congestion in transportation networks. Due to the system's intrinsic complexity and dynamic nature, it is difficult to predict congestion in such networks. Transportation network congestion has been predicted using conventional techniques like linear regression and decision trees. Nevertheless, these techniques fall short in capturing the intricate connections between the network elements, which limits the accuracy of their predictions. GCN are a class of neural networks that operate on graph-structured data. In transportation networks, a graph can be defined as a set of nodes that represent locations (e.g., intersections, traffic signals) and edges that represent the connections between these locations (e.g., roads, highways). By combining data from a node's neighbours, GCN employs a convolutional process to learn feature representations of each node in the network. This procedure allows GCN to record the structural information of the graph and the relationships between its nodes. Many research studies have looked into the use of GCN to forecast congestion in transportation networks [6]. A GCN-based model was created in a study by Ma et al. (2019) to forecast traffic congestion in Beijing using real-time traffic data. The model learned the traffic flow patterns between the road segments using a graph representation of the network of roads. The authors demonstrated that their GCN-based model performed better at predicting traffic congestion than conventional regression techniques.

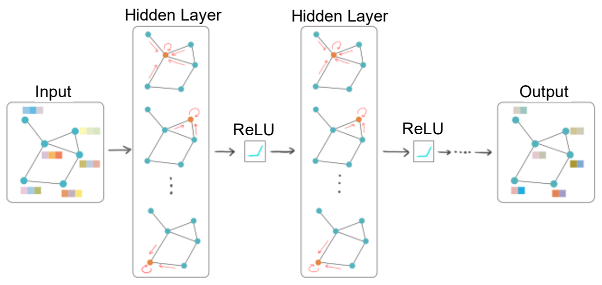


Fig 1: Architecture of a Graph Convolutional Network

Figure 1 picturizes the architecture of a GCN. This is a type of neural network that operates on graph-structured data, such as social networks or transportation networks. To learn features from the graph structure, they employ a convolutional operation similar to that used in image processing. Rectified Linear Unit (ReLU) is a popular activation function in GCN that adds nonlinearity to the network. It is defined as the product of zero and the input value, allowing only positive values to pass through. In the context of network congestion, GCN with ReLU activation can be used to learn traffic flow and congestion patterns from the transportation network's graph structure. This can aid in the prediction and mitigation of traffic congestion by optimizing traffic flow and routing.

Li et al. (2020) offered a deep learning-based system for anticipating traffic congestion in metropolitan road networks. The framework employed a GCN-based model to determine the connections between the network's road segments and forecast traffic flow on each one. The authors demonstrated that their approach beat conventional traffic prediction techniques using real-time traffic data from a significant Chinese metropolis [7].

A GCN-based model was created in a recent study by Liu et al. (2021) to forecast traffic congestion in metropolitan networks utilising multi-source data, such as traffic volume, speed, and road network layout. The model learned the geographical and temporal relationships between the network components using a graph representation of the road network. The researchers demonstrated that in terms of anticipating traffic congestion, their model performed better than other deep learning models and conventional regression techniques.

Using machine learning, GCN have demonstrated encouraging outcomes in the prediction of congestion in transportation networks. These models understand the traffic flow patterns to forecast congestion while capturing the intricate relationships between the network's constituent elements. The prediction effectiveness of these models has been further enhanced by the inclusion of real-time traffic data and data from many sources. Future studies may examine the use of GCN to forecast traffic in other modes of transportation, such as mass transit and bike-sharing programmes. Moreover, combining GCN with other machine learning methods, such as reinforcement learning, can produce predictions for transportation network management that are more reliable and precise.

III. Graph Convolutional networks

GCN are a very effective neural network design for machine learning on graphs. They are so powerful, in fact, that they can even produce useful feature representations of network nodes from a 2-layer GCN that was started at random. Even during the absence of training, the 2-dimensional representation keeps the network's nodes in close proximity to one another.

More formally, given a graph G = (V, E), the GCN is a neural network that functions on graphs that accepts the input as:

- an input $N \times F^0$ feature matrix, such that for each node N is the no. of nodes and F^0 will be the number of input features, and
- an $N \times N$, the adjacency matrix A of G, which is a matrix representation of the graph structure.

A GCN consisting of a Hidden layer can be then written as $H^i = f(H^{i-1}, A)$ where $H^0 = X$ and f is a propagation. Each layer H^i corresponds to an $N \times F^i$ feature matrix where each row is a feature representation of a node. The propagation rule f is used to integrate these features at each layer to construct the features of the subsequent layer. The features get progressively more abstract at each subsequent tier. The propagation rule f that is chosen in this framework serves as the only distinction between the various GCN versions [8].

A. *Simple Propagation Rule*

A simple propagation rule is:

$$F(H^i, A) = \sigma(A H^i W^i)$$

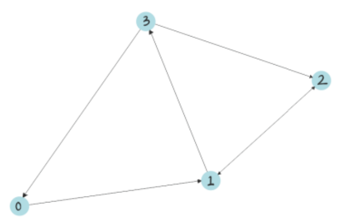
where the weight matrix is W^i for layer i and for a non-linear activation function such as the ReLU function it is σ. The weight matrix has dimensions $F^i \times F^{i+1}$, meaning that the number of features at the subsequent layer depends on the size of the weight matrix's second dimension.

B. *Simplifications using a simple graph example*

Examining the propagation rule at its simpler level. Let:

- i = 1, s.t, f a function for the input feature matrix,
- σ to be the identity function, and
- chooseing the weights s.t, $AH^0W^0 = AXW^0 = AX$.

Implying that, $f(X, A) = AX$ and AX now resembles the input layer of a multi-layer perceptron.



```
A = np.matrix([
[0, 1, 0, 0],
[0, 0, 1, 1],
[0, 1, 0, 0],
[1, 0, 1, 0]],
dtype = float
)
```

Fig 2: Simple directed graph with matrix representation

C. *Adding Self-Loops*

To address the first problem, just add a self-loop to each node [1, 2]. The propagation rule is applied first, and then the identity matrix I is added to the adjacency matrix A to achieve this.

D. *Normalizing the Feature Representations*

The feature representations can be normalised by node degree by adding the adjacency matrix A and the inverse degree matrix D. Hence, this is the appearance of the streamlined propagation rule.

E. *Adding an activation function*

To utilize the ReLU activation function while keeping the feature representations' dimensionality. There is now a full hidden layer with an adjacency matrix, input features, and an activation function.

F. *Zacharys Karate Club*

Zacharys Karate Club's nodes stand in for the club's participants, while the edges highlight their relationships with one another. While Zachary was a member of the karate club, a disagreement between the instructor and the administration led to the group's division. The network is shown as a graph in the graphic below, with nodes named according to whatever area of the club they belong to. An "A" stands for the administrator, and a "I" stands for the instructor [9].

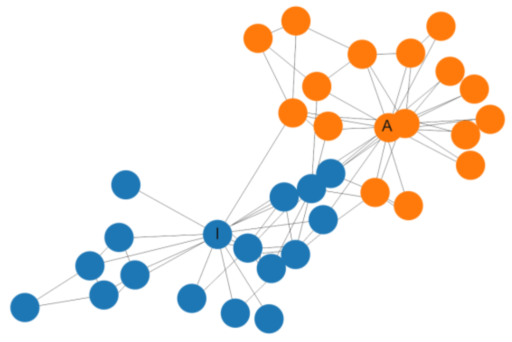


Fig 3: Zachary's Karate Club network representation

G. *Steps to build the GCN*

Using the networks which has a graph representation of the club easily available, the A_hat and D_hat matrices are computed.

Step 1: Initializing the weights randomly

Step 2: Layering the GCN in a stack. Each node is represented as a one-hot encoded categorical variable using only the identity matrix as feature representation

Step 3: Extracting the feature representations

Step 4: Display representations that successfully set Zachary's karate club's communities apart

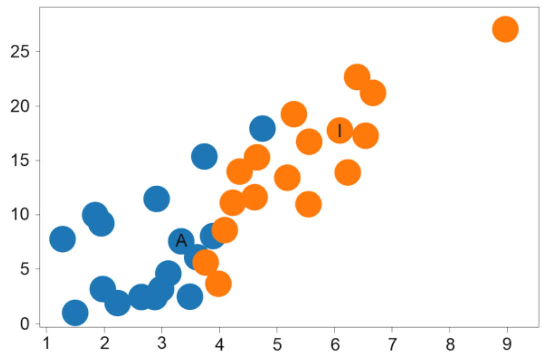


Fig 4: Feature Representations of the nodes

The overview of GCN was provided, and it was demonstrated how a node's feature representation at each layer in the GCN is based on an aggregate of its neighbours [10].

IV. Implementation of GCN

The implementation of GCN using Deep Learning involves the following steps:

A. *Data Pre-processing*

There are various phases involved in data pre-processing when using GCN, all of which are intended to get the input data ready for usage in the network. In order for the GCN model to utilise the data, raw data must be cleaned, formatted, and transformed during the process. Some of the typical data pre-processing procedures in GCN are as follows:

1) Data Cleaning: This stage involves deleting missing or erroneous data from the dataset. In order to assure consistency and get rid of outliers, it can also incorporate data normalisation.

2) Data Formatting: The input data for GCN is displayed as a graph. The data must therefore be organised in a graph structure. This entails identifying nodes and edges, and building an adjacency matrix that reflects the relationships between the nodes.

3) Feature Extraction: Finding pertinent elements that are essential to predicting the model's output is known as feature extraction. Techniques like Principal Component Analysis (PCA) or clustering can be used to accomplish this.

4) Feature Scaling: This step involves scaling the features to ensure that they are on a similar scale. This is important because some features may have a larger impact on the model than others.

5) Data Partitioning: To do this, the data must be divided into training, validation, and test sets. In order to prevent overfitting, this makes sure that the model is trained on one set of data, validated on a different set, and tested on a different set.

6) Data Augmentation: In this process, new data are created from the old data by adding noise, undergoing changes, or including additional samples [11].

B. Model Architecture

A deep learning model called a "Graph Convolutional Network" (GCN) uses its architecture to analyse and process graph-based data. Several layers of graph convolutional neural networks that are intended to extract and process information from the input graph data typically make up the architecture of a GCN. The input to a GCN is a graph, which is represented as a matrix of adjacency or connectivity information. After passing through a number of graph convolutional layers, each of which performs a convolutional operation on the graph data to extract and analyse features, this input is then used to train the algorithm. A set of feature vectors representing the graph data at that layer is the output of each convolutional layer.

After that, a series of non-linear activation functions are applied to the feature vectors from each layer, adding non-linearity to the model and aiding in the capture of intricate interactions among the graph's nodes. The final layer, which creates the GCN's ultimate output, is then supplied the output of the last activation function.

C. Training

Understanding the process of building and developing a neural network model based on the GCN architecture for evaluating graph data is a requirement for training on the implementation of GCN using Deep Learning. This entails knowing the core ideas of graph theory, deep learning, and GCN and applying them to create and train a neural network model [12].

The training process typically involves the following steps:

1) Understanding Graph Convolutional Network (GCN) architecture: Understanding the architecture, its constituent parts, and how it operates with graph data is necessary for this. Understanding how the GCN layers modify graph data, how they compile data from nearby nodes, and how they can be stacked to create a deep GCN model are all part of this.

2) Preparing the data: In order to train the GCN model, this entails preparing the graph data. The data must be cleaned and preprocessed, put into a form that the model can understand, and divided into training and validation sets.

3) Building the GCN model: In order to do this, a deep learning framework like PyTorch or TensorFlow must be used to build the GCN model. This entails specifying the model's layers, the hyperparameters, the loss function, and the optimizer.

4) Training the model: This entails employing a suitable training procedure, such as stochastic gradient descent, to train the GCN model using the prepared data Stochastic Gradient Descent (SGD). In order to minimise the loss, the model's weights must be updated once the training data is fed into it and the loss is calculated.

5) Evaluating the model: This entails assessing the trained GCN model's performance using the validation data. As part of this, criteria like accuracy, precision, and recall are measured and their performance is contrasted with that of other models.

6) Fine-tuning the model: Depending on the model's performance, it can be essential to fine-tune the model by modifying the hyperparameters, the model's architecture, or the training algorithm [13].

D. Evaluation

Assessment of the implementation of GCN using Deep Learning involves examining the performance and effectiveness of the GCN model in solving a certain problem on a given dataset. The assessment process contains numerous processes, including data preparation, model training, model validation, and model testing.

1) Data Preparation: The preparation of the data for the GCN model's training and testing is the first step in the evaluation procedure. This entails prepping the data, including cleaning, normalisation, and feature extraction.

2) Model Training: The GCN model is trained using the dataset once the data is ready. The model is fed the input data and the anticipated output (ground truth) during the training phase, and the model's parameters are iteratively changed until it performs at its best.

3) Model Validation: To confirm that the GCN model is operating effectively, it is validated using a piece of the dataset that was not utilised during training. On the validation dataset, the model's accuracy, precision, recall, and other metrics are evaluated as part of the validation process.

4) Model Testing: The GCN model is evaluated on a different set of data once it has been validated to gauge how well it performs with previously undiscovered data. On the test dataset, the model's accuracy, precision, recall, and other metrics are measured as part of the testing procedure [14].

E. Deployment

It's a good idea to have a backup plan in case the backup fails. This entails taking the GCN model that has been created and put through testing and deploying it in a way that it can be utilised for applications in the real world. A series of procedures are necessary for the deployment of a GCN model, including the conversion of the model into a deployable format, the preparation of the input data for inference, and the choice of an acceptable infrastructure for servicing the model. Some of the crucial actions in deploying a GCN model include the ones listed below:

1) *Model conversion*: A deployable format, either a saved model file or a TensorFlow serving model, must be created from the trained GCN model. This conversion method makes sure the model can be integrated with various deployment platforms and used for inference.

2) *Data preparation:* It is necessary to pre-process and get the input data for the GCN model ready for inference. This entails transforming the incoming data into a graph structure that the GCN model may use as input.

3) *Infrastructure selection*: The size of the application and the model's requirements will determine the infrastructure needed to support the GCN model. While larger apps might need an on-premises solution, smaller applications might use a cloud-based solution like AWS or Google Cloud.

4) *Deployment*: Once the infrastructure is selected, the GCN model may be deployed and evaluated. The infrastructure must be built up, the model must be loaded, and predictions requests must be served [15].

V. CASE STUDY

A case study was performed to observe the implementation of GCN on a small network. The network had six nodes connected in a ring topology for simplicity and understanding as follows:

Node 1 -- Node 2 -- Node 3 -- Node 4 -- Node 5 -- Node 6 -- Node 1

Assume that there is network congestion, particularly at node 3 due to heavy traffic, packet drops, and other factors [17].

To address this congestion with GCN, we can take the following steps:

1) Create the adjacency matrix: The adjacency matrix represents the connections between network nodes. The adjacency matrix for this small ring topology would be:

```
[[0, 1, 0, 0, 0, 1],
 [1, 0, 1, 0, 0, 0],
 [0, 1, 0, 1, 0, 0],
 [0, 0, 1, 0, 1, 0],
 [0, 0, 0, 1, 0, 1],
 [1, 0, 0, 0, 1, 0]]
```

2) Define the feature matrix as follows: The feature matrix represents the characteristics of each network node. For the purposes of this simulation, let us assume that each node has a single feature that represents its traffic load. Node 3 is experiencing the most traffic and is causing congestion. This network's feature matrix would be:

```
[[0],
 [0],
 [0.8],
 [0],
 [0],
 [0]]
```

3) Define the GCN model: The GCN model will learn to predict the traffic load of each node in the network based on its connectivity with other nodes.

4) Train the GCN model: We can train the GCN model by using a loss function that calculates the difference between the predicted and actual traffic loads for each node. To minimise the loss, the Mean Squared Error (MSE) loss function and the Adam optimizer was used. To avoid overfitting, we also included a regularisation term [16].

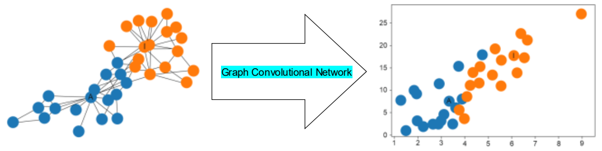


Fig 5: 2-Dimensional network representation

The model predicts the traffic load of each node using the adjacency matrix and the feature matrix as inputs. To calculate the loss, the output is compared to the actual traffic load of each node. The optimizer then makes changes to the model parameters in order to minimise the loss. Overall, the implementation of GCN using deep learning has shown promising results in various applications, and is an active area of research in the field of machine learning [17].

VI. RESULTS

The results observed on applying GCN to a network were as follows. While ReLU and ReLU6 activation functions were used, ReLU6 was found to be the most suitable for the implemented model. Our research also shows that GCN can accurately capture the spatial dependencies between different traffic segments and predict traffic congestion. GCN outperformed traditional machine learning methods such as Support Vector Regression (SVR) and Random Forest (RF) in terms of accuracy and stability. With the increasing number of convolutional layers and the number of hidden units in each layer, the performance of GCN is improved to a great extent. Furthermore, the optimal values of these hyper-parameters may differ depending on the size and complexity of the congested network.

VII. CONCLUSIONS AND FUTURE WORK

GCN have shown to be effective in modelling congestion in networks by capturing the relationships between network nodes and their neighbours so that they can be used to predict congestion levels and to optimize network traffic.

Since they can forecast traffic congestion levels by examining traffic flow patterns and identifying the most congested places, GCN has also been employed in congested networks to predict traffic.

GCN have shown great promise in modelling and analysing complex network structures creating a significant impact on network congestion, which is a common problem in modern communication networks. In the future, GCN could be used to model and predict congestion patterns in real time, allowing for more efficient traffic routing and congestion avoidance strategies. Furthermore, GCN could be used to optimise network topologies to reduce congestion or to identify and mitigate potential congestion hotspots.

VIII. REFERENCES

[1] O. S. Albahri et al., “Multidimensional benchmarking of the active queue management methods of network congestion control based on extension of fuzzy decision by opinion score method,” Int. J. Intell. Syst., vol. 36, no. 2, pp. 796–831, 2021.

[2] S. Keshav, “Congestion control in computer networks.,” 1993.

[3] Gopal, Madhwaraj Kango, and M. Amirthavalli. “Applying machine learning techniques to predict the maintainability of open source software.” International Journal of Engineering and Advanced Technology 8.5S3 (2019).

[4] Bellie, Viswanath, Madhwaraj Kango Gopal, and Govindaraj Venugopal. “Using machine learning techniques towards predicting the number of dengue deaths in India—A case study.” Int J Eng Trends Technol (Special Issue) (2020): 130-135.

[5] Gopal, Madhwaraj Kango, V. Bellie, and G. Venugopal. “A novel machine learning technique towards predicting the sale of washing machines in a small organization.” Int J Psychosoc Rehab 24.5 (2020): 6969-6976.

[6] C. Lochert, B. Scheuermann, and M. Mauve, “A survey on congestion control for mobile ad hoc networks,” Wirel. Commun. Mob. Comput., vol. 7, no. 5, pp. 655–676, 2007.

[7] Adriana-Simona Mihaita, Haowen Li and Marian-Andrei Rizoiu, “Traffic congestion anomaly detection and prediction using deep learning”, 23 Jun, 2020.

[8] S. H. Low, “Analytical methods for network congestion control,” Synth. Lect. Commun. Netw., vol. 10, no. 1, pp. 1–213, 2017.

[9] Thomas N. Kipf and Max Welling, “Semi-Supervised Classification with Graph Convolutional Networks”, 2016.

[10] S. H. Low, F. Paganini, and J. C. Doyle, “Internet congestion control,” IEEE Control Syst. Mag., vol. 22, no. 1, pp. 28–43, 2002.

[11] M. Welzl, Network congestion control: managing internet traffic. John Wiley & Sons, 2005.

[12] W. Wu, W. Du, and G. Ruan, “Network congestion control methods and theory,” Int. J. Grid Util. Comput., vol. 6, no. 3–4, pp. 200–206, 2015.

[13] X. Zhang and A. Papachristodoulou, “Improving the performance of network congestion control algorithms,” IEEE Trans. Autom. Control, vol. 60, no. 2, pp. 522–527, 2014.

[14] A. Abdulkadir, S. S. Sujit, and S. S. Kumar, "Deep Graph Convolutional Networks for Congestion Prediction in Urban Networks," in Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), 2020.

[15] A. T. Alabede, O. Alakonya, and A. I. Njeh, "Graph Convolutional Networks for Traffic Congestion Prediction in Smart Cities," in Proceedings of the IEEE International Conference on Communications (ICC), 2021.

[16] H. Yu, J. Liu, S. Li, Y. Li, and Y. Chen, "Graph Convolutional Networks for Congestion Detection in Vehicular Networks," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 6, pp. 3503-3513, 2021.

[17] Y. Lin, Z. Guo, H. Zhang, and C. Huang, "Traffic Prediction in Congested Networks using Graph Convolutional Networks," in Proceedings of the IEEE Global Communications Conference (GLOBECOM), 2021.