Netlist Congestion Prediction with Graph Attention Network

Brian Chen brc004@ucsd.edu
Lisa Hwang lshwang@ucsd.edu
Lindsey Kostas lkostas@qti.qualcomm.com
Elahe Rezaei erezaei@qti.qualcomm.com
Masoud Sadeghian msadeghi@qti.qualcomm.com

Abstract

Abstract—In modern Very Large-Scale Integration (VLSI) chipdesign, congestion poses a significant challenge to the development process of semiconductor chip development. Currently, congestion is unknown until after placement and routing steps in the design process. This is inefficient and costly because if there is congestion, these long steps must be reiterated to improve the quality of the design. Predicting congestion early can expedite optimization efforts and enhance overall design efficiency. In this work, we explore the feasibility of leveraging deep learning techniques to predict local congestion prior to the routing stage. Using a digital integrated circuit (DigIC), we employ a Graph Attention Network (GAT) to predict the post-placement, pre-routing congestion levels in a netlist, in an attempt to preemptively address potential issues. Our analysis includes exploratory data analysis to understand key factors influencing congestion, pre-processing steps to prepare the data for implementation, and model training and testing using both GATs and a baseline extreme gradient boosting (XGBoost) model. Our results indicate that our GAT model, particularly when incorporating engineered features, outperforms the XGBoost model in learning patterns in graph data. While the XGBoost model exhibits better accuracy in assigning demand values to chip regions, the GAT model provides insights into the underlying patterns of congestion. Our findings highlight the potential of deep learning approaches for early congestion prediction in VLSI chip design.

Index Terms—very large scale integration, routing congestion, deep learning, graph attention network, post-placement, pre-routing

Code: https://github.com/lisasunhwang/Capstone-B07-3

1 Introduction

In a world powered by technology, chip manufacturers are constantly pushing the boundaries, trying to optimize resources to create the most efficient and least costly chips possible. In Very Large-Scale Integration (VLSI) chip design, an issue that continuously arises is routing congestion, where the demand of wires exceeds the amount that an area can handle. This can result in routing detours, under-utilization of regions in the chip, higher chip area, longer wires, and overall an inefficient use of resources. Currently, congestion is only able to be accurately calculated after the routing process, which is where the locations of wires between components on a chip are determined. This results in a re-iteration through the design flow if the congestion is problematic to the functionality of the chip (Figure 1). Becoming aware of congestion early in the design process allows designers to alter logic, logic structures, and overall allows more iterations of designs to be tested in a shorter period of time.

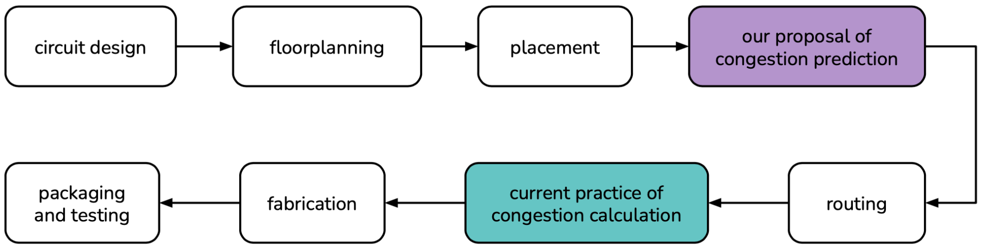
circuit design
floorplanning
placement
our proposal of congestion prediction
routing
current practice of congestion calculation
fabrication
packaging and testing

Figure 1: Flowchart of the chip design process. The current practice of congestion calculation is conducted after routing, and may or may not result in going back to the floorplanning stage if the chip is highly congested. Our proposed method attempts to prevent this back-tracking process.

In our project, we look to replicate and expand on a deep learning process used to predict local congestion. We take inspiration from Kirby et al. (2019) in using a Graph Attention Network to predict congestion prior to the routing stage of the design process, eliminating the need for re-iterating through the steps, displayed in Figure 1. There is a limited understanding of key factors that contribute to congestion in early design stages; many methods use hand-engineered features that don't necessarily capture complex characteristics of the design used, and models don't always generalize well to different datasets. By exploring a deep learning approach, we hope to assess generalizability of the architecture as well as generate insights as to what factors exactly are contributing to the success or failure of the approach.

2 Problem Statement

Given a post-placement and pre-routing netlist, our objective is to predict local congestion within the semiconductor chip. Post-placement refers to the stage in VLSI design after gate

Table 1: Correlation Coefficients of Variables to Demand

	demand
xloc	-0.121
yloc	0.115
individual pins	-0.252
width	-0.556
GRC pin count	0.169
GRC cell count	0.314
connections	0.282

placement on the chip, which sets the stage for subsequent design steps (Figure 1). Pre-routing refers to the stage prior to physically placing wires onto the chip. It is important that we aim to predict congestion pre-routing to minimize the time and resources spent on re-routing of wires later on in the design process. Local congestion refers to the demand of a specific region's nodes in the chip, in contrast to the congestion of the entirety of the semiconductor chip. Predicting the local congestion in a chip allows us to identify which features play a pivotal role in congestion on a smaller, digestible scale.

This research aims to address the paramount challenge of predicting local congestion in VLSI design, targeting a specific region's demand within the chip. Improved accuracy in congestion prediction at early design stages holds the potential to alleviate challenges during logic synthesis, enhancing routability, and facilitating optimal chip placement. Recognizing a gap in understanding key factors influencing congestion, our research endeavors to contribute valuable insights to advance the field of VLSI design.

For this work, we define congestion as total demand. We measure congestion as a level of demand rather than also tying in the capacity, primarily because capacity is fixed in a given area, no matter how many nodes and wires are within it. Since our model is not focusing on predicting the capacity, predicting an area's demand is sufficient in understanding whether the area would be congested.

3 Methods

1. *Exploratory Data Analysis*

   Understanding our given data is one of the greatest challenges in building this model. Rather than a linear process of exploratory data analysis (EDA) then building the model and testing, we go back to EDA multiple times throughout implementation. Fully understanding the data is crucial to building a successful model.

   We work with a digital integrated circuit (DigIC) dataset provided by our mentors at Qualcomm, Inc. This dataset contains 13 different netlist configurations, shown in Table 2, which we conduct our training and testing on during this project. Each instance is an individual component in the chip, with attributes like height, width,

Table 2: Netlist Data and Demand Statistics

	count		demand statistics			
netlist	instances	nets	min	max	mean	std
1	3952	4482	1	38	25.646	4.522
2	6872	7404	0	62	35.412	9.399
3	6913	7445	0	46	27.675	7.562
4	7323	7855	0	47	27.720	7.535
5	7258	7790	0	47	27.158	7.340
6	7120	7652	0	48	28.408	7.680
7	7879	8411	3	59	29.668	6.682
8	7626	8158	1	65	27.435	6.233
9	7620	8153	3	50	27.816	7.883
10	7772	8304	2	43	25.305	6.896
11	7814	8346	0	62	28.686	6.180
12	6529	7061	0	70	31.536	6.465
13	6548	7082	7	60	34.163	6.615

and pin count that we use as features, in addition to engineering our own.

During various stages of EDA, we uncover several intriguing findings. For example, initially, when examining correlation coefficients, we observe that the variables 'xloc' and 'yloc,' representing the x- and y-coordinates of instances, displayed relatively weak correlations with the demand values, with coefficients of -0.110 and 0.125 respectively (Table 1). These coefficients do not reveal much about the relationship between spacial location and the demand values at face value, and could easily be overlooked. However, visually representing the data by plotting xloc against demand and yloc against demand (as shown in Figure 2) provides a clearer understanding of the relationship between instance locations and demand. Scatterplots illustrating the correlation reveal that instances situated closer to the center of the x-axis tend to have higher demand values, and a similar trend was observed for the y-axis, though with a weaker correlation.

Additionally, we use heatmaps to display the demand in different netlists. Figure 3 shows Netlist 1's demand values, plotted in global routing cell (GRC) coordinates. Our chip is divided into GRCs, represented in Figure 4. To represent a chip in a 2-D space, it is divided into blocks or regions, each defined as a GRC. Each GRC contains a number of instances and wires, which helps us to understand the netlist in a 2-dimensional perspective. The instanecs are the components of the chip, and the wires contribute to demand and congestion. Each GRC has a capacity value, which represents the maximum demand that it can have. Although demand cannot be a negative value, we plot the heatmap by normalizing demand values from -10 to 10 for visualization purposes, to emphasize the high demand values on the chip netlist.

2. *Pre-Processing Data*

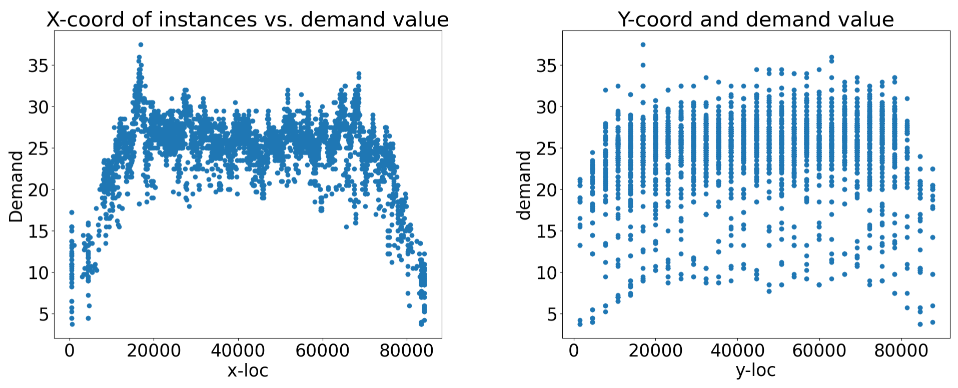

Figure 2: Scatterplot displaying relationship between instance location coordinates and demand values.

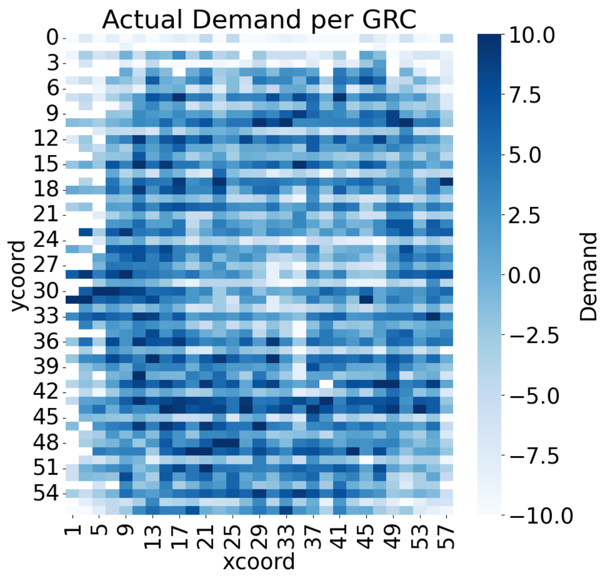

Figure 3: Heatmap displaying the demand values spread across Netlist 1 by GRC. Demand values were normalized from -10 to 10.

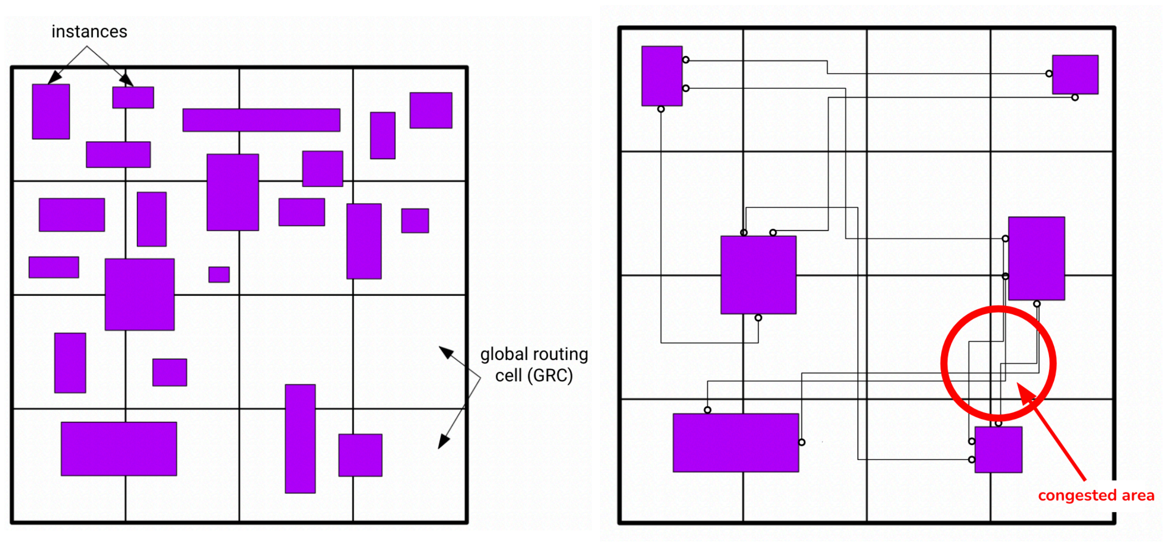

Figure 4: Global routing cell instances (left) and congestion visualization (right). A GRC can have multiple, one, or no instances, wires, and pins within it.

During the pre-processing phase, we generate the connectivity graph with connectivity information, which our dataset provides to us in the form of an incidence matrix. However, to pass the connectivity into our model, we turn this incidence matrix into an edge index, which is simply two arrays of instance indices whose positions in the array correspond to the connection. As an example, in an edge index of arrays [0, 1, 0] and [1, 2, 3], node 0 goes to nodes 1 and 3, and node 1 goes to node 2.

Next, we split the data into train and test sets. Kirby et al. (2019) partition the graph by function for generalization purposes; however, we pivot from this method, as our data is not provided to us by function. Instead, we utilize certain netlists as train data and test on others. In this way, we eliminate the need for splitting edge indexes to pass through the GAT, and pass the entirety of graphs into the algorithm.

In addition to the edge index, we hand-select as well as engineer features to pass through the model. From the dataset provided, we identify several features that intuitively contribute to our model's logic. Specifically, the x- and y-coordinates, width, and the number of pins (points at which wires connect to instances) were valuable to our model's logic. The physical location of the instances were shown to be significant in Figure 2, so we make sure to incorporate them in our feature selection. While the widths of the instances exhibit the largest (negative) correlation coefficient (Figure 2), the heights display minimal variance in the netlist data, leading us to exclude them from consideration. We normalize the x- and y-coordinates and width data due to their wide range of values, ranging from thousands to hundreds of thousands. The sizes of netlists vary from chip to chip and manufacturer to manufacturer, thus necessitating normalization to ensure generalizability of our model. Furthermore, the inclusion of the number of pins makes intuitive sense, as more pins results in more wires, represented on the right of Figure 4.

We engineer features from GRC data (cell density, pin density, connections) that we were provided alongside the DigIC netlists. Cell density is the total amount of cells in the GRC, pin density is the total amount of pins in the GRC, and connections are simply the number of cells each cell is connected to. The rationale behind these engineered features is straightforward: higher numbers of cells, pins, and connections suggest increased wire density in the GRCs of the corresponding nodes, resulting in higher levels of demand. As shown in Figure 4, a highly congested area is likely to contain more cells, pins, and/or wires.

Our labels are extracted from congestion data provided to us. We iterate through the instances, look at their x-location and y-location, map those coordinates to a GRC, and assign the GRC demand value to the coordinate as the label. This method runs the risk of having a lot of repeat values in the labels if there are a lot of instances in the same GRC. To combat this problem, we try to average the values of all the GRCs that each instance occupies using information about their sizes. However, this does not lead to any significant changes in model performance. We also consider the use of supernodes: making a node for each GRC and connecting them based on the connections between the instances between GRCs. While this method would help the distribution of labels, we lose a lot of vital information about the connections between instances. These connections are why we use a GAT in the first place, so we opted not to use this method and instead focused on instance-based demand prediction.

It is important to note that, while we normalize the values of large, location-deterministic features, we deliberately choose to not normalize the demand values. Normalizing demand values results in minimal variance, posing challenges for the model in differentiating between demand values. By maintaining the original scale of the demand values, we aim to preserve their inherent variance and ensure that the model could effectively distinguish between different levels of demand.

3. *Our Model*

We use a Graph Neural Network (GNN), a neural network that performs convolutions on graph data, aggregating information from each node's neighbors, shown in Figure 5 on the left. The basic intuition behind using GNNs for this problem is that the logic of the netlist is encoded in its topology. By representing our data in graph form, we are able to capture complex relationships between nodes and their connectivities that we may not see through tabular data. Leveraging this topological structure within the neural network is expected to yield favorable outcomes.

Specifically, we use a Graph Attention Network (GAT). The GAT originates from Veličković et al. (2018), and is a type of GNN which assigns attention coefficients to each of a node's neighbors during the aggregation process, weighing more important

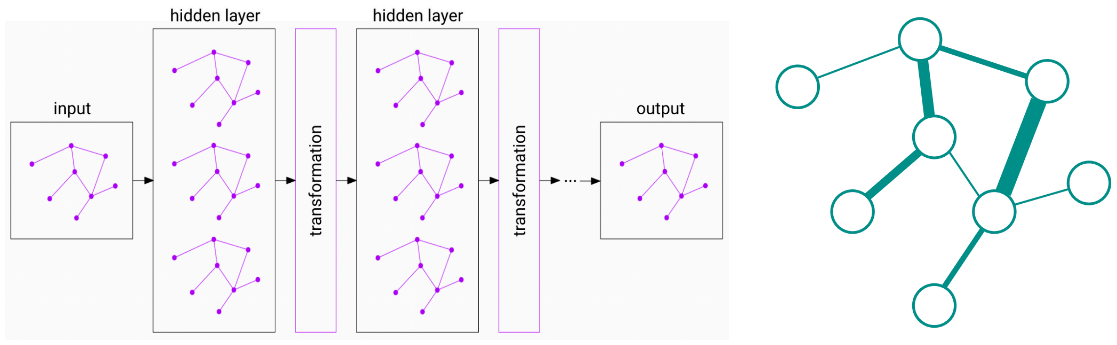

Figure 5: Left: a GNN takes a graph structure as an input, aggregates the data and updates the weights of nodes in iterations.
Right: GAT attention mechanism assigns different weights to nodes and their neighbors depending on their importance in the graph structure.

neighbors more heavily. The attention mechanism is displayed in Figure 5 on the right.

During the training phase, we experiment with adjusting the architecture of the model in several ways, including varying the number of layers, hidden layers, or attention heads. We find that increasing these parameters is helpful for the model's performance; however, after a certain point, we encounter issues with oversmoothing. Exceeding a certain number of layers overloads the model with information, ultimately impeding its learning capabilities.

4. *Testing*

We use mean-squared error (MSE) as our metric for error, as it is a commonly used measure and is relatively easy to interpret. Additionally, MSE treats prediction errors equally, which is beneficial for our purposes of minimizing error of congestion prediction across all regions of the chip.

To test our GAT model, we split our node features into two groups: given features and engineered features. By performing this split, we are able to analyze the effectiveness of the engineered features for congestion prediction.

We also create a baseline extreme gradient boosting (XGBoost) model, a machine learning algorithm that is used for predictive modeling. XGBoost utilizes gradient boosting and decision trees to predict values, all while controlling overfitting. While XGBoost is considered an extremely powerful tool in machine learning, GNNs are more suited to the context of chip design, as they are able to capture the data of instances and wires in a graph structure.

4 Results

We run experiments using many different architectures and hyperparameter configurations. In our results, we compare the GAT including our engineered features, the GAT with only given features, and the XGBoost model, looking at their results on a separate test netlist dataset.

Test distribution results are shown in Figure 6. The GAT with engineered features predicts a distribution of values that is closest to the actual distribution. Both the GAT with only given features and the XGBoost predict values that overfit to our training distribution. Despite having higher test MSE, the distributions indicate that the GAT with engineered features learns more about the underlying patterns of the data in comparison to the XGBoost and the GAT model trained with only given features.

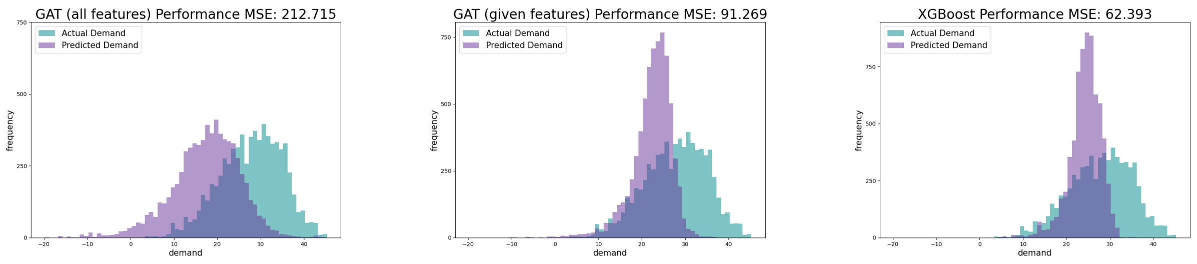

Figure 6: Distributions of various tested models with the actual demand label values.

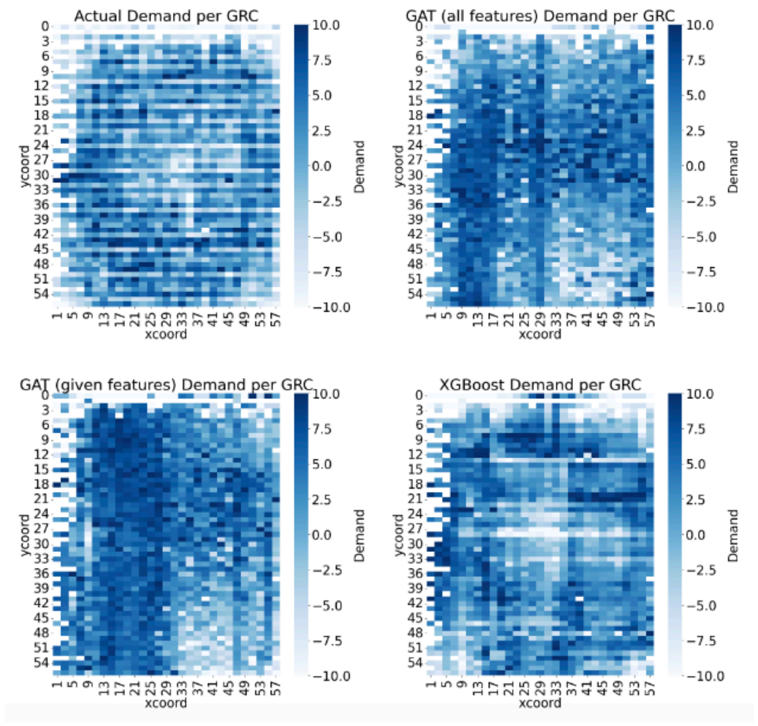

Figure 7: Heatmaps of various tested models' demand values.

The XGBoost model, however, more accurately assigns demand values to their corresponding areas in the netlist, resulting in a more visually accurate heatmap. This can be seen in Figure 7; the XGBoost heatmap is the most similar to the actual demand heatmap.

5 Conclusion

Our study delves into the challenging domain of Very Large-Scale Integration (VLSI) chip design, particularly focusing on addressing routing congestion, a persistent issue in the field. Through our exploration, we have developed and analyzed a Graph Attention Network (GAT) model augmented with engineered features, aiming to predict local congestion within semiconductor chips. Our investigation uncovers several key findings and insights:

1. Leveraging graph-based representations and attention mechanisms, our GAT model demonstrates promising capabilities in predicting local congestion.
2. Through meticulous exploratory data analysis (EDA), we gain valuable insights into the relationships between instance features and demand values, informing our feature selection and engineering processes.
3. The inclusion of engineered features derived from Global Routing Cell (GRC) data significantly enhances the predictive performance of our model, as evidenced by closer alignment of predicted demand distributions.
4. Despite higher Mean Squared Error (MSE), our GAT model with engineered features showcases a deeper understanding of underlying data patterns, indicating superior learning capabilities compared to baseline models.

The clearest next step in our work is to train the GAT on multiple and larger netlists, as opposed to the single netlist we currently train on. We run into issues combining connectivity data to be passed into the model, as we can concatenate instance data from multiple netlists but not the edge indices. However, if we are able to train on more representative output distributions, we expect the model to be able to generalize better.

Additionally, we currently use the GAT for relatively small netlists consisting of thousands of nodes. There are commonly millions of nodes in VLSI designs, so scaling our architecture would be a solid next step for our experiment. Bigger datasets should be able to better leverage the complexities of a graph neural network.

Upon successfully completing these steps, we can address the complicated problem of predicting demand in GRCs without instances. Currently, our pipeline predicts demand for instances, and we can then predict demand in GRCs by taking the average of the demands of the instances in each GRC. However, if there are no instances in a GRC, we have no way to predict the demand, or even to create features for the GRC. Congestion can still occur in these GRCs, and so perfecting our model would have to involve addressing this issue.

References

Kirby, Robert, Saad Godil, Rajarshi Roy, and Bryan Catanzaro. 2019. “CongestionNet: Routing Congestion Prediction Using Deep Graph Neural Networks.” In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*. [Link]

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. “Graph Attention Networks.” [Link]