CongestionNet: Routing Congestion Prediction Using Deep Graph Neural Networks

Robert Kirby
Nvidia
Santa Clara, CA, USA
rkirby@nvidia.com

Saad Godil
Nvidia
Santa Clara, CA, USA
sgodil@nvidia.com

Rajarshi Roy
Nvidia
Santa Clara, CA, USA
rajarshir@nvidia.com

Bryan Catanzaro
Nvidia
Santa Clara, CA, USA
bcatanzaro@nvidia.com

***Abstract*—As feature size shrinks, routing constraints become a more significant limiting factor to the manufacturability of VLSI designs. Routing congestion significantly impacts quality metrics such as area and timing performance, but congestion is not known accurately until late in the design cycle, after placement and routing. This can lead to unpleasant surprises during the design process. Accordingly, early prediction of routing requirements would enable design engineers to iterate faster, with more confidence that their designs were routable and high quality. Additionally, routability estimates can inform placement itself, preemptively eliminating routing problems. In this work, we present a graph-based deep learning method for quickly predicting logic-induced routing congestion hotspots from a gate-level netlist before placement. This model can provide early feedback to designers and EDA tools, indicating logic that may be difficult to route. Compared to using previous congestion prediction metrics to predict congestion hotspots without placement information, our solution provides a 29% increase in the Kendall ranking correlation score. Because our focus is on predicting congestion due to local logic structure, which manifests itself on lower metal layers, we also report accuracy for predicting lower metal layer congestion. When predicting congestion for the lower metal layers, the benefit of our solution over previous metrics increases to 75%. Additionally, our approach is fast. On a circuit with 1.3 million cells, our approach takes 19 seconds to predict congestion, compared with 10-60 minutes for other methods.**

***Index Terms*—machine learning, deep learning, neural networks, very large scale integration, routing congestion, physical design, graph convolutional neural networks**

I. Introduction

In newer semiconductor processes, wire area and delay have become dominant factors in determining chip area, timing and power. Routing congestion can overwhelm routing resources and lead to low cell utilization and routing detours. This translates to higher chip area, longer wires, and lower achievable operating frequency. When logic designers are aware of logic-induced routing congestion, they can restructure logic to alleviate the congestion, for example, by modifying design functionality or selecting alternative logic structures. Additionally, early feedback can increase the number of iterations possible when conducting design space exploration, and therefore improve quality of results.

978-1-7281-3915-9/19/$31.00 2019 IEEE

Many modern placement and synthesis tools leverage congestion estimation in their cost analysis in order to minimize the effects of congestion in the final physical design. Accurate and fast measurements which can be made *a priori*, without placement, would therefore be useful.

In this work, we present a fast early feedback system to provide logic designers and EDA tools with information about the likely routing congestion of synthesized designs, before any placement or routing step. This is achieved by training a deep neural network to predict the congestion properties of the final physical design on a per cell basis, given only the gate-level netlist graph as input. This approach is advantageous for a few reasons. First, it can be run early in the design cycle and therefore allows for more rapid prototyping. Additionally, because the model makes predictions based solely on the logical structure of the netlist, rather than on any specific cell placement, it removes artifacts of sub-optimal placement which are present in placement based approaches. Finally, this approach can be done without any physical information, and takes only seconds to perform inference, including on large designs.

Our approach is based on deep graph convolutional neural networks, which we train on real-world designs. We train the model using the netlist graph and cell features as input and the final local detail routed congestion as our target. We then validate our model using unseen partitions from the same process technology.

II. Related Work

Much work has been done to estimate routing congestion in VLSI circuits, with varying motivations and approaches. Global routing estimation is commonly done to provide feedback about the routability of a given placement. Most works in this space have invested in improving the correlation of this estimate with final detailed routing results to help drive intelligent placement decisions [13], [15], [17]. While we don't attempt to summarize all of this work here, it is important to note that our work differs from these because we don't use any prior placement and we make no attempt to do a rough placement to inform our estimate.

Other work has instead focused on finding a graph based metric to serve as a proxy for wire length and congestion during synthesis. Some methods predict at a cell level, and

others predict average or maximum congestion, based on entire graph properties. In [7] the authors show that for many circuits the peak congestion is strongly correlated to the cell count, number of literals and the sum of all min-cut edge sets for all node pairs in the graph. In [4] the authors show an approach which can generate and rank clusters of logic in the netlist graph which have the potential for high congestion. Also in [5] and [10] the authors present methods for accurately measuring net length. It is known that the total net length can be a good proxy for congestion and therefore these methods can act as a basic congestion estimation technique. In this paper we outline an alternative approach, based on graph convolutional neural networks, and show improved results compared to these metric based techniques.

Unlike our work which focuses on congestion estimation, many other works have shown improvement in final congestion by leveraging these congestion estimates (both physical and graph based) in various ways. This has been done by fast global routing and cell inflation during placement, as well as by informing earlier synthesis steps. [2] used the GTL metric from [4] to identify potential clusters of tangled logic and inflated those cells to ease routing congestion problems. [1] uses global routing based estimates to inflate problematic cells. In the past, some works have also investigated using congestion estimates from technology independent placements during the technology mapping step such as in [8]. Many industry standard synthesis tools already perform low effort placement and congestion estimation along with synthesis in order to improve the final quality of the synthesis netlist, as well as improve correlation to final placement and routing. In this way improving congestion prediction can be applied to physically aware synthesis flows to potentially improve run time and quality of results.

Although we focus on predicting congestion from netlist structure, other efforts have attempted to solve related congestion estimation problems using machine learning and deep learning techniques. In [18] and [12], the authors trained a machine learning model using multivariate adaptive regression splines to predict detail route DRC violations from global routing and placement level features. This idea was extended by the authors of [16] who added deep convolutional neural networks to include more spatial information.

Recently, deep learning techniques for operating on graphs have improved significantly. We based our model on Graph Attention Networks (GAT), introduced in [14]. These models also draw from Graph Convolutional Networks, introduced in [6]. These models apply one or more graph convolution layers, which aggregate information from a graph neighborhood, and then perform a transformation on the aggregated features to produce an output for every node. Graph Attention Network models demonstrate further improvement in performance on inductive graph structure problems such as Protein-Protein Interaction datasets. These techniques have not been widely applied in VLSI problems, although we are aware of one project that used graph convolutions to predict node observability during test logic insertion [11].

III. Problem Statement

In this work, we propose a predictive model that can predict the detail routed lower metal layer congestion values from a technology specific gate-level netlist for every cell in a design. We build our training dataset by separating the final detail routed congestion maps into discrete grid cells, and assign the congestion value in each grid as the ground truth (or target) value for each cell that was placed in that grid cell.

We focus on the congestion in the bottom half of metal layers of the design because the congestion found on lower metal layers is driven primarily by local logic structures, and not by longer interconnects between unrelated clusters of logic, which tend to run on higher metal layers. While we present results both for all metal layers and lower metal layers, we believe predicting lower metal layer congestion is not only more important for the underlying task of identifying congested logic structures, but also simplifies the task for our graph based network, which has no information about the relative placement of distinct high-level logic structures.

TABLE I
Validation Partition Size

Partition	A	B	C	D	E	F	G
Cell Count (1e6)	1.94	2.42	2.52	2.67	0.74	1.71	0.73

IV. Our Approach

We employ Graph Attention Networks, a form of deep neural network. Deep learning models work by constructing a differentiable, highly parameterized model which produces predictions and then training that model via gradient descent. The training procedure optimizes the model such that its predictions match the ground truth. Of course, a model that can merely duplicate the training dataset is of little use, so model accuracy is then evaluated by applying the trained model to a new, unseen dataset (validation dataset) and evaluating the model predictions against the ground truth.

A. Graph Attention Network

Our neural network is an 8 layer Graph Attention Network (GAT) with size 16 intermediate (or hidden) state. We have seen that the flexible attention based aggregation step in GAT models allow the network to make higher frequency decisions about the graph structure. Additionally, when compared to the original Graph Convolution Networks, this attention based aggregation performs better on inductive inference problems where no labels are given for the graph at inference time [14].

The GAT neural network comprises multiple layers which are applied sequentially to the node features. Each layer, l, consists of two stages which are applied to each node, i, in the graph independently. All model parameters are shared for every node in the graph for a given layer. The first stage takes the input feature vector (or previous layer output), $\mathbf{h}_{l-1}^{i}$, for a node and all nodes in the 1-hop neighborhood, N, and performs a weighted sum to obtain a new aggregated feature

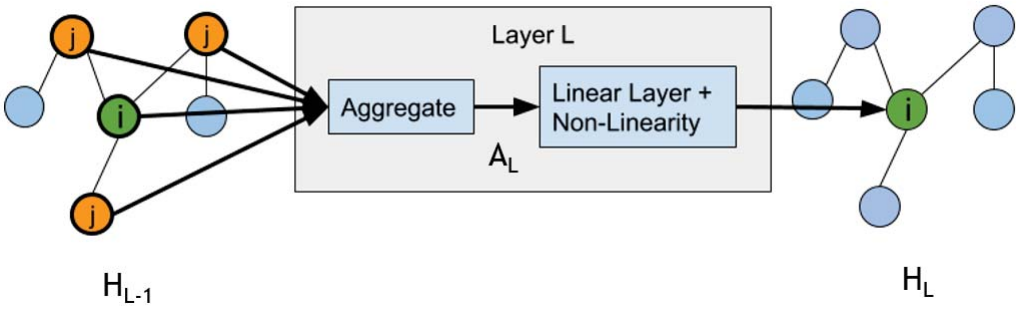

Fig. 1. Graph Convolutional Network

vector, $\mathbf{a}_l^i$ (Equation 1). The weighting for each neighboring node in this sum, $\mathbf{u}_l^{i,j}$, is obtained by concatenating the primary node features with the neighboring node features and taking the dot product with a trainable parameter vector, $\mathbf{w_l^{attn}}$. Then, a softmax operation is performed over the results from all neighbors to ensure the weightings add to one (Equation 2). The second stage performs a parameterized linear transformation followed by a non-linearity to produce a new vector for the node (Equation 3). In this way, every layer consumes a feature vector for every node and produces a new feature vector based on the previous features of that node and its neighbors. The final layer's output feature is a single scalar for each node which predicts the local routing congestion.

$$\mathbf{a}_l^i = \sum_{j \in N} \mathbf{u}_l^{i,j} \mathbf{h}_{l-1}^j \tag{1}$$

$$\mathbf{u}_l^{i,j} = SoftMax_{j \in N}([\mathbf{h}_{l-1}^j || \mathbf{h}_{l-1}^i] \cdot \mathbf{w_l^{attn}}) \tag{2}$$

$$\mathbf{h}_l^i = ReLu(\mathbf{a}_l^i \mathbf{W_l^{trans}}) \tag{3}$$

Within this model the parameters $\mathbf{W_l^{trans}}$ and $\mathbf{w_l^{attn}}$ are trained via gradient back-propagation.

B. Model Training Setup

Deep learning requires large amounts of data, and so we collected a large real-world dataset to train our model. Our dataset consists of roughly 50 million cells spread over tens of different physical design partitions. This dataset contains roughly 5000 distinct cell types from real designs. We split the collected data into train and validation sets by choosing a subset of the partitions that we know to be distinct in function from the remainder of the dataset. We do this to prevent cross contamination between train and validation partitions, which would lead to overfitting and poor generalization. All images and results we report in this paper are from the validation set, which is not seen during training. The process technology (and therefore the enumeration of cell types) and tool flows are shared across both training and validation partitions. The size of each named validation partition is provided in Table IV.

We chose as inputs to the model a simple undirected connectivity graph of the netlist, as well as selected features for every cell in the design. The connectivity graph is a graph where each node corresponds to a cell, and if two cells are connected by a net then there exists an edge between those nodes in the graph. The cell features consist of a trainable embedding of length 50 for each cell type and each cell's logic description as well as the pin count and cell size of that cell.

We train the network using the PyTorch auto-grad framework on a single NVIDIA Tesla V100 GPU. We use mini-batches of 800k cells (spread across multiple partitions) and the Adam optimizer to control the parameter update step for both the GAT model as well as the embeddings. We minimize the mean squared error of the congestion prediction for each cell, comparing against the final congestion for that cell from the final detail routed design.

Training required roughly 60 hours on a single GPU. Our model needs to be retrained for every new process technology, since the embeddings are over cell types specific to a process technology.

C. Measuring Correlation

In order to evaluate our predictions, we project our per cell predictions back onto their respective 2D grid (using the final ground truth physical placement) and average all cells within each grid cell to come up with a predicted value that can be compared to the original ground truth grid value. We then report the correlation metrics on a per grid cell basis across the entire validation set. This ensures we give equal significance to physical areas with low cell density which might otherwise have a small contribution to the cell based correlation metrics.

We report correlation values using the Kendall ranking coefficient. This metric is calculated by evaluating if a sample pair's ordering is the same in both the real and predicted values for all possible pairs. The Kendall coefficient is the difference between the percentage of correct pairs and incorrect pairs. Therefore, a score of 0.5 means that 75% of all possible pairs are correctly ordered. Looking at correlation to evaluate accuracy allows us to study the accuracy of metrics without requiring the distribution of the predictions to match the distribution of the metrics. Other works, such as [5], also use similar pairwise metrics to evaluate their predictions.

什么东西?

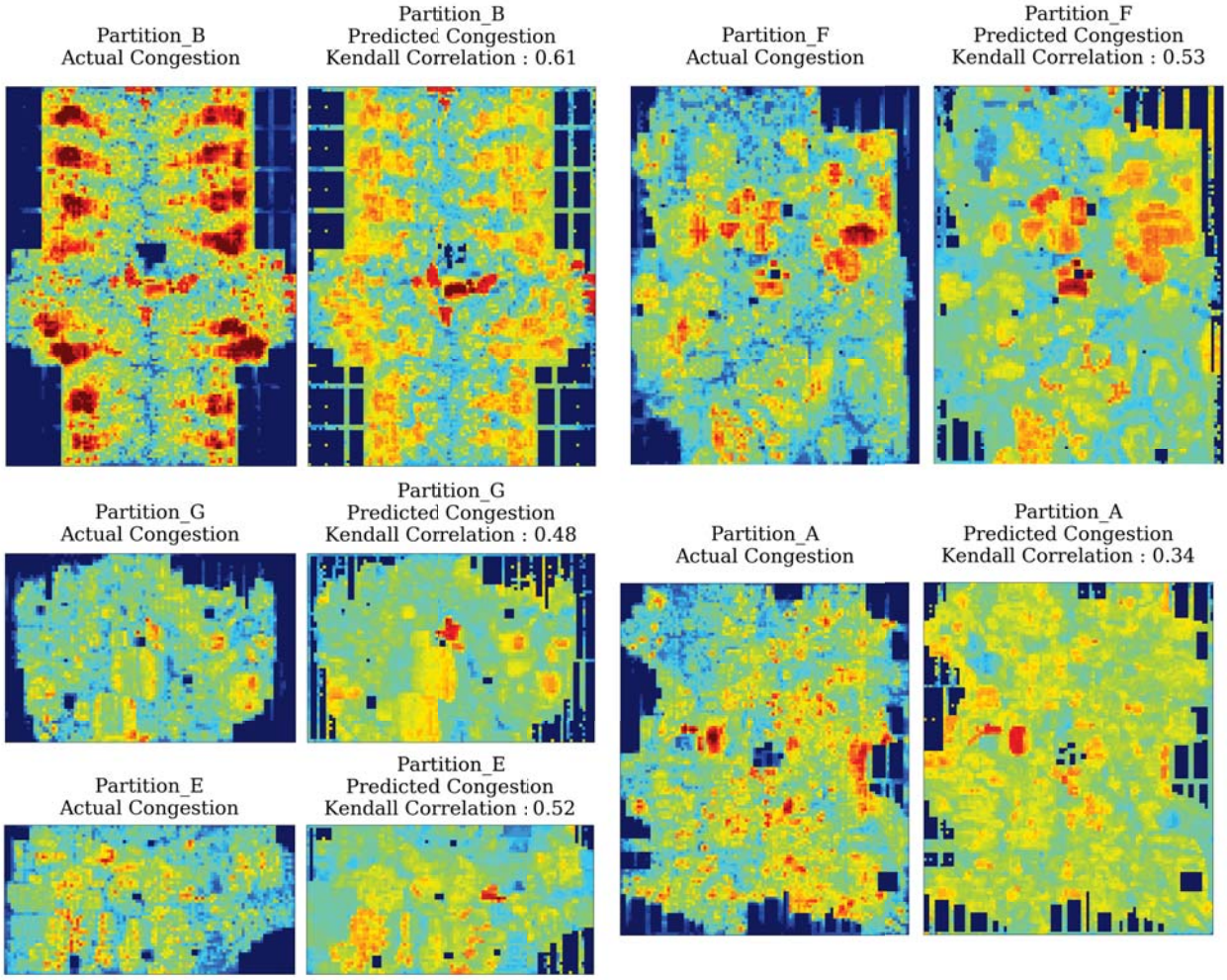

Fig. 2. Left: Detail Routed Congestion, Right: GAT Congestion Prediction (Lower Half of Metal Stack; Validation Holdout Set)

TABLE II
DETAILED ROUTING CORRELATION (KENDALL CORRELATION; LOWER METAL LAYERS)

Partition	A	B	C	D	E	F	G	Avg
Adhesion	0.06	0.00	0.00	0.14	0.00	0.03	0.18	0.06
Neighborhood Size	0.11	0.48	0.43	0.12	0.23	0.30	0.29	0.28
GTL	0.19	0.35	0.35	0.27	0.21	0.29	0.33	0.28
GAT Prediction	0.34	0.61	0.58	0.37	0.52	0.53	0.48	0.49

领居节点的相关性

TABLE III
DETAILED ROUTING CORRELATION (KENDALL CORRELATION; ALL METAL LAYERS)

Partition	A	B	C	D	E	F	G	Avg
Adhesion	0.00	-0.04	0.00	0.10	-0.09	-0.01	0.13	0.01
Neighborhood Size	0.05	0.48	0.42	0.12	0.27	0.32	0.26	0.27
GTL	0.11	0.28	0.27	0.23	0.20	0.27	0.26	0.23
GAT Prediction	0.26	0.54	0.49	0.22	0.32	0.38	0.28	0.35

V. RESULTS

Our results are shown in Figure 2 and Table II and III. As can be seen, the network properly identifies many congested areas in the final detail routed designs. In Partition_B (the best performing of all partitions) almost all areas of high congestion are properly detected. The performance across different partitions varies significantly though. The model seems to fail mostly in two ways. First, it occasionally over predicts congestion in areas of low to moderate congestion, such as in most failing parts of Partition_A. Second, due to the

graph based nature of the model, it sometimes makes overly soft decision boundaries. This can be seen in over prediction in areas such as the lower center portion of Partition_G and much of the high congestion areas of Partition_F.

A. Comparison to Other Methods

In order to determine the effectiveness of our model, we then attempt to compare the Kendall rank correlation coefficient of our results to that of other popular metrics for *a priori* (no placement) congestion estimation. We choose to evaluate neighborhood size, adhesion [7] and the GTL-Score metric from [4] which the authors adapted from the ratio cut and rent metrics. These metrics have previously been evaluated on a circuit graph, rather than per node in a graph. Therefore, in order to predict at a per cell level, we evaluate the metric on a sub-graph of the partition centered on each node and use that as a prediction for the center node. This should give us a localized congestion estimation.

B. Neighborhood Size

A simple approximation for congestion prediction is to use the size of the local neighborhood [7]. We calculate this size for every cell in the design (calculated at a 5 hop neighborhood) and report the correlation between this value and the final reported congestion as one of our baselines.

C. Adhesion

Adhesion is a metric that attempts to measure the interconnectedness of a graph and has been suggested in the past as a good proxy for helping predict peak congestion. In [7], it is defined as the sum of all min-cuts between all pairs in a graph. Because our sub-graphs have varying sizes we adapted this metric slightly to fit our purposes. To investigate the correlation between adhesion and congestion for every cell in the design, we looked at a given cell's degree-5 neighborhood and calculated the min-cut value between the given cell and every cell in the neighborhood. We then took the maximum min-cut value as a representation of how interconnected the sub-graph is around the given cell. We found the maximum min-cut value had a higher correlation with final routing congestion than the average min-cut value. As has been noted in previous work, this method is quite computationally intensive, since it involves solving min-cut problems.

D. Groups of Tangled Logic (GTL-Score) Metric

The GTL metric, proposed by [4], identifies wire-dominated groups of logic. It looks at a cluster of cells and gives that cluster a ranking based on cut-size, average cluster pin count, cluster size, the Rent coefficient and the Rent exponent. We adapt this metric in order to rank clusters rather than merely identify them. We evaluate a fast estimate for the peak GTL-Score for all given clusters we wish to rank. We calculate this metric for a growing neighborhood (from 1 to 5 hop) and report the highest GTL-Score out of all these neighborhoods. In this way, the GTL-Score becomes a rough measure of how interconnected the local area around the cell is to the rest of the circuit. Because Rent exponent is a top down estimated metric sensitive to clustering decisions we chose a Rent exponent for each partition such that the accuracy of the GTL-Score when predicting congestion was maximized. This should provide an advantage to the GTL-Score metric which would otherwise estimate the Rent exponent without access to ground truth congestion information.

TABLE IV
ABLATION STUDY ACCURACY

Node Feature Inputs	Kendall Correlation (Lower Layers)
Cell Type, Function, Geometry	0.49
Cell Function, Geometry	0.49
Cell Geometry	0.40

E. Accuracy and Run Time

Our model provides a ranking correlation that outperforms the best baseline above by 29% when predicting all metal layer congestion and by 75% on lower metal layer congestion prediction. Our approach also takes 19 seconds to calculate for a 1.3 million cell design on a V100 GPU compared to calculating neighborhood and ratio-cut for every cell in the same design, which takes roughly 10 minutes and 60 minutes respectively using 20 threads of an E5-2698 Intel(R) Xeon(R) CPU. While our attempt at calculating neighborhood size and GTL are using off the shelf graph libraries, and we have not made significant effort to improve the run time of these methods, it is not likely that improvement in the algorithms or parallel implementation would lead to run times significantly lower than our GAT run time. It is also worth noting that while our method requires little compute time during inference, as previously mentioned, we do require a large amount of compute for training. While the accuracy of our models vary significantly across partitions, we can see that there are correlations in the difficulty of predicting congestion across different methods. This suggests that there are some partitions that are inherently harder to predict than others, but our method outperforms other methods to varying degrees across all partitions. The poor performance of the adhesion metric could have been anticipated given the previous efforts results which rank it as less correlated to peak congestion than cell count and literal count [7]. We include them for completeness.

VI. ABLATION

In this section we remove different input features from our model to identify which features contribute the most to the accuracy of our predictions. We present results from model training leaving out explicit cell type (but keeping the logical description of the cell) and additionally leaving out the logical description of the cell but retaining the cell geometry (physical size and pin count). We observe that the cell type or function is an essential part of our predictions. This is notable because the only features our baselines take into account is the cell pin

count and graph structure. Models that were trained with only access to cell geometry dramatically under-performed models which leveraged the logic function and specific cell type information. Non-learning baselines do not have the benefit of being tuned to specific technologies (cell types) or tool flows other than by tuning the scalar value of the Rent exponent. We believe this advantage is where much of the power of our model comes from.

VII. Congestion Estimation Philosophy

While this work focused on congestion estimation from graph structure, we are aware that many other methods of congestion estimation exist which rely on varying levels of placement and global routing estimation. Some of these methods have even been successfully incorporated into existing synthesis level tools. It is not our intention to present this work as a replacement for these physically based estimations, but rather as a supplement to these methods when these estimates either prove too costly to compute, or when estimation based solely on graph structure might be more desirable. Examples might include high frequency design iteration (such as in physically aware synthesis tools) or pre-placement design analysis. Frequently, these methods include some form of Rent's rule or ratio cut estimation to estimate wire length. We believe our method has the ability to improve these estimations in the future.

VIII. Conclusion and Future Work

We demonstrated a novel graph neural network based approach to congestion estimation that does not require any placement information. This approach outperforms historical methods for congestion estimation that use only netlist graph structure. It does this by using deep learning to identify frequent patterns in standard cell connectivity which cause congestion and has the ability to learn which graph edges are most relevant for the task. Our model is also able to make predictions about congestion on a per cell level as opposed to other approaches which attempt to predict peak congestion values for sub-circuits or identify specifically congested sub-circuits. It is also faster than previous methods for identifying clusters of congested logic, often by an order of magnitude. We also believe this approach can benefit from additional exploration in neural network architectures, input features (such as pin/edge types), and extending the graph representation to a directed or hyper graph.

Finally, to our knowledge, this is the first work exploring the use of graph based deep learning for physical design problems and we believe that the power of this approach can be used for other netlist based EDA tasks in the future, such as predicting wire length, power or other physical design metrics that have a dependency on the netlist graph structure.

IX. Acknowledgments

We would like to thank David Brown and Sreedhar Pratty for their dependable support in parsing netlists and building our datasets.

References

[1] U. Brenner and A. Rohe, "An effective congestion-driven placement framework," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 4, pp. 387-394, April 2003. doi: 10.1109/TCAD.2003.809662

[2] J. Cong, Guojie Luo, K. Tsota and Bingjun Xiao, "Optimizing routability in large-scale mixed-size placement," 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, 2013, pp. 441-446. doi: 10.1109/ASPDAC.2013.6509636

[3] Bo Hu and M. Marek-Sadowska, "Congestion minimization during placement without estimation," IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002., San Jose, CA, USA, 2002, pp. 739-745. doi: 10.1109/ICCAD.2002.1167614

[4] T. Jindal, C. J. Alpert, Jiang Hu, Zhuo Li, Gi Joon Nam and C. B. Winn, "Detecting tangled logic structures in VLSI netlists," Design Automation Conference, Anaheim, CA, 2010, pp. 603-608. doi: 10.1145/1837274.1837422

[5] A. B. Kahng and S. Reda, "Intrinsic shortest path length: a new, accurate a priori wirelength estimator," ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005., San Jose, CA, USA, 2005, pp. 173-180. doi: 10.1109/ICCAD.2005.1560059

[6] Kipf, T. N., Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv e-prints arXiv:1609.02907.

[7] P. Kudva, A. Sullivan and W. Dougherty, "Metrics for structural logic synthesis," IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002., San Jose, CA, USA, 2002, pp. 551-556. doi: 10.1109/ICCAD.2002.1167586

[8] T. Kutzschebauch and L. Stok, "Congestion aware layout driven logic synthesis," IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281), San Jose, CA, USA, 2001, pp. 216-223. doi: 10.1109/ICCAD.2001.968621

[9] T. Lin and C. Chu, "POLAR 2.0: An effective routability-driven placer," 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, 2014, pp. 1-6.

[10] Qinghua Liu and Malgorzata Marek-Sadowska. 2004. Pre-layout wire length and congestion estimation. In Proceedings of the 41st annual Design Automation Conference (DAC '04). ACM, New York, NY, USA, 582-587. DOI: https://doi.org/10.1145/996566.996726

[11] Yuzhe Ma, Haoxing Ren, Brucek Khailany, Harbinder Sikka, Lijuan Luo, Karthikeyan Natarajan, Bei Yu, High Performance Graph Convolutional Networks with Applications in Testability Analysis, ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, June 2-6, 2019. (preprint)

[12] Z. Qi, Y. Cai and Q. Zhou, "Accurate prediction of detailed routing congestion using supervised data learning," 2014 IEEE 32nd International Conference on Computer Design (ICCD), Seoul, 2014, pp. 97-103. doi: 10.1109/ICCD.2014.6974668

[13] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement," 2007 Design, Automation & Test in Europe Conference Exhibition, Nice, 2007, pp. 1-6. doi: 10.1109/DATE.2007.364463

[14] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. 2017. Graph Attention Networks. arXiv e-prints arXiv:1710.10903.

[15] Jurjen Westra, Chris Bartels, and Patrick Groeneveld. 2004. Probabilistic congestion prediction. In Proceedings of the 2004 international symposium on Physical design (ISPD '04). ACM, New York, NY, USA, 204-209. DOI: 10.1145/981066.981110

[16] Z. Xie et al., "RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network," 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, 2018, pp. 1-8. doi: 10.1145/3240765.3240843

[17] Xiaojian Yang, R. Kastner and M. Sarrafzadeh, "Congestion estimation during top-down placement," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 1, pp. 72-80, Jan. 2002. doi: 10.1109/43.974139

[18] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou and Y. Cai, "An accurate detailed routing routability prediction model in placement," 2015 6th Asia Symposium on Quality Electronic Design (ASQED), Kula Lumpur, 2015, pp. 119-122. doi: 10.1109/ACQED.2015.7274019