Graph Representation Learning for Gate Arrival Time Prediction

Pratik Shrestha
Drexel University
Philadelphia, Pennsylvania, USA
ps937@drexel.com

Saran Phatharodom
Drexel University
Philadelphia, Pennsylvania, USA
sp694@drexel.edu

Ioannis Savidis
Drexel University
Philadelphia, Pennsylvania, USA
isavidis@coe.drexel.edu

Abstract

An accurate estimate of the timing profile at different stages of the physical design flow allows for pre-emptive changes to the circuit, significantly reducing the design time and effort. In this work, a graph based deep regression model is utilized to predict the gate level arrival time of the timing paths of a circuit. Three scenarios for post routing prediction are considered: prediction after completing floorplanning, prediction after completing placement, and prediction after completing clock tree synthesis (CTS). A commercial static timing analysis (STA) tool is utilized to determine the mean absolute percentage error (MAPE) and the mean absolute error (MAE) for each scenario. Results obtained across all models trained on the complete dataset indicate that the proposed methodology outperforms the baseline errors produced by the commercial physical design tools with an average improvement of 61.58% in the MAPE score when predicting the post-routing arrival time after completing floorplanning and 13.53% improvement when predicting the post-routing arrival time after completing placement. Additional prediction scenarios are analyzed, where the complete dataset is further sub-divided based on the size of the circuits, which leads to an average improvement of 34.83% in the MAPE score as compared to the commercial tool for post-floorplanning prediction of the post-routing arrival time and 22.71% improvement for post-placement prediction of the post-routing arrival time.

CCS Concepts

• **Hardware** → **Timing analysis**; • **Computing methodologies** → *Machine learning*.

Keywords

timing prediction, physical design, machine learning, graph convolutional networks

ACM Reference Format:
Pratik Shrestha, Saran Phatharodom, and Ioannis Savidis. 2022. Graph Representation Learning for Gate Arrival Time Prediction. In *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '22), September 12–13, 2022, Snowbird, UT, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3551901.3556478

MLCAD '22, September 12–13, 2022, Snowbird, UT, USA

ACM ISBN 978-1-4503-9486-4/22/09…$15.00
https://doi.org/10.1145/3551901.3556478

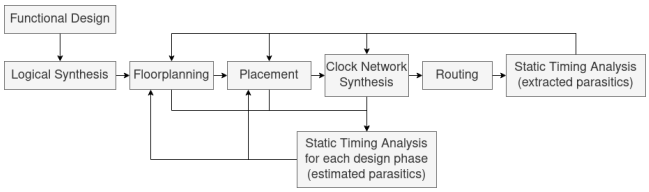


Figure 1: Physical design automation flow.

1 Introduction

A significant portion of the run time of modern electronic design automation (EDA) tools is spent on iterative execution of the design flow to meet the target circuit specifications and timing constraints. The different stages of the design flow, as illustrated in Fig. 1, include separate estimates of the parasitic impedance and physical characteristics of the circuit, which results in differences in the static timing analysis (STA) of the circuit. The estimates of the path delays during static timing analysis at each design stage require a comparatively shorter amount of time to complete, but significantly affect the quality of the downstream estimates of the path delay. Commercial tools such as Synopsys PrimeTime [1] provide an estimate of the timing profile after each stage, which allows for necessary modifications to the circuit to attain timing closure. With estimates of the timing profile of the circuit, provided at each design stage, preemptive changes to the circuit are possible before completing the given physical design step. Since each design step requires significant computational resources, a predictive tool that provides a good estimate of the delay of the timing paths significantly reduces the design time.

A wide variety of structures are represented as graphs and modeled into a low dimensional vector embedding using graph based neural networks (GNNs) [2], which are then utilized for various learning applications. Circuit netlists represented as graphs leverage structural information to allow for more holistic learning. Although graph based learning approaches have been utilized for various EDA tasks such as logic synthesis [3], placement [4], and clock tree synthesis [5], the use of graph based machine learning techniques has yet to be applied for arrival time prediction.

In this paper, a novel graph based regression framework is introduced to predict gate arrival time. The framework utilizes a timing path based graph representation that includes node features populated from the circuit netlist and the STA reports to predict path delay at different phases of physical design. Arrival time prediction is performed for three primary scenarios: a) post-completion of the floorplan to predict post-completion of routing, b) post-completion of placement to predict post-completion of routing, and c) post-completion of CTS to predict post-completion of routing. The contributions of the paper include

- Error characterization of a commercial STA tool to determine the mean absolute percentage error (MAPE) and mean absolute error (MAE) of the arrival time predictions as a baseline,
- A netlist to graph and timing path to graph representation with embedded node feature attributes,
- A graph-based deep neural regression model to predict the arrival time of a signal to a gate within a given timing path. A comparison is made against the baseline and a shallow linear model (linear regression), and
- Feature importance analysis on the neural models using GNNExplainer [6].

The paper is organized as follows. Background on machine learning based arrival time prediction, graph representation learning, and explainable graph neural model analysis is provided in Section 2. An error analysis of timing estimates from a commercial STA tool and a baseline model of the errors are described in Section 3. The arrival time prediction framework is discussed in Section 4, and an analysis of the results produced by the proposed framework is provided in Section 5. Some concluding remarks are provided in Section 6.

2 Preliminaries and Related Works

Prior work on applying machine learning for gate level path delay estimation is discussed in Section 2.1. Both the use of graph neural network layers for graph representation learning and graph model explanation are described in Section 2.2.

2.1 Machine Learning for Gate Level Path Delay Estimation

One area of prior work explores applying machine learning (ML) to timing analysis to reduce the miscorrelation between two timing analysis tools that return different results for the same circuit netlist [7]. In [8], a learning-based approach is used to fit correlation based ML models of wire slew and delay to estimates from a signoff STA tool. A deep hierarchical model is utilized in [9] to predict the path delay of an unexecuted commercial EDA tool from the timing analysis of a commercial tool that was utilized. A support vector machine (SVM) was used during floorplanning for the prediction of post-layout timing failures of embedded memory in [10]. A random forest based approach is proposed in [11] to predict the pre-routing timing profile of a circuit designed in and analyzed by a commercial EDA tool. An artificial neural network is proposed in [12] to predict the statistical static timing analysis (sSTA) profile of a circuit characterized by a commercial STA tool. Previous approaches for timing prediction generally target a single stage of the overall design flow. Although structural features are utilized in prior work, the structural information provided by the connectivity relationship between two or more components of a netlist graph has not been explored in the context of timing profile prediction.

2.2 Graph Representation Learning and Neural Network Explanation

In this work, graph convolutional layers (GCN) [13] are utilized to model gate delay from timing path graphs. Convolution functions multiply the input neurons with a set of weights commonly known as filters or kernels. Given a directed graph $G = (V, E)$, where $v \in V$ is a node of the graph, $e \in E$ is an edge of the graph, and A denotes the adjacency matrix, the graph convolution layer is defined as

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \quad (1)$$

where σ is an activation function. The adjacency matrix of the graph $\tilde{G}$ is given by $\tilde{A} = A + I_N$, where $\tilde{G}$ is equivalent to the graph G with added self-connections on each node. I_N represents the identity matrix and $\tilde{D}$ the diagonal degree matrix for $\tilde{A}$. $H^{(l)}$ and $W^{(l)}$ are, respectively, the node embedding matrix and trainable weight matrix for the l^{th} layer. The input embedding layer of the network is $H^0 = X$, where X represents the node features of the graph.

One challenge with GCNs is the inability to comprehensively characterize the operation of the neural model after training. AI algorithms such as GNNExplainer [6] provide interpretable explanations of predictions by a GNN-based model by reducing redundant information in a graph that does not directly impact the prediction(s). GNNExplainer provides such explanations by maximizing mutual information (MI), a measure of the mutual dependence between the node feature set X and the predicted node target value Y of the corresponding minimal graph G_s of the computational graph G as defined by

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S). \quad (2)$$

3 Dataset Design and Baseline

The physical design of the benchmark circuits and the STA timing profile dataset generated for arrival time prediction are described in Section 3.1. A multi-scenario analysis of arrival time estimation error is provided in Section 3.2, which is utilized as the baseline for comparison with the proposed technique.

3.1 Dataset Generation

In this work, ISCAS'89 sequential benchmark circuits [14] are utilized to build a dataset of physical designs and timing paths. The dataset is generated on a TSMC 65 nm technology. The initial benchmark circuit is synthesized into a technology-specific gate-level netlist using Synopsys Design Compiler. Synopsys IC Compiler is utilized for floorplanning, placement, CTS, and routing. Multiple layouts are generated for each benchmark circuit by utilizing parameterized design constraints. PrimeTime is used to perform static timing analysis on the generated logical and physical design at each stage of the design flow. Timing paths, gate delays, and input/output delays are extracted from the timing report generated by PrimeTime for each stage of the design process.

The ISCAS'89 benchmark circuits used for the data generation are listed in Table 1, and the parameters utilized as design constraints are listed in Table 2. A total of 1500 physical designs are generated for the 25 benchmark circuits (60 per circuit).

3.2 Baseline Path Delay

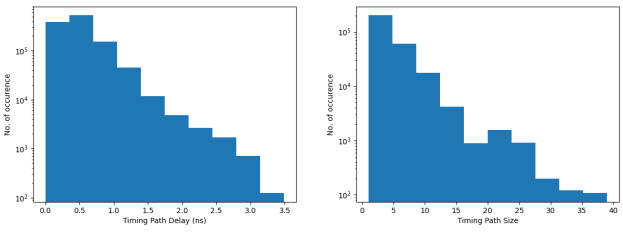
Specifically, for a post-floorplan to post-routing prediction, the estimates of the arrival time of logical paths of the post-floorplanned

Table 1: ISCAS'89 benchmark circuits and dataset characterization. Circuits selected as test data are highlighted in red.

Dataset Group	Circuit	Circuit size	# of timing paths		
			Floorplan to routing	Placement to routing	CTS to routing
Small	s27	16	540	540	540
	s298	98	1027	1565	1566
	s344	136	1905	2310	2310
	s349	139	1885	2262	2262
	s382	129	1601	2571	2571
	s386	138	1195	1380	1380
	s400	136	1561	2644	2640
	s420	175	1175	1681	1681
	s444	149	1465	2687	2688
	s510	211	293	830	830
Medium	s641	185	3276	4283	4281
	s713	216	3138	4167	4167
	s820	298	1269	2186	2186
	s832	304	1439	2221	2222
	s838	355	2123	3284	3286
	s953	379	1796	3892	3892
	s1196	434	438	2678	2684
	s1238	474	538	2795	2797
	s1423	586	2753	6268	6309
Large	s9234	2313	8088	12525	12517
	s13207	3425	33174	47845	49092
	s15850	4209	21432	54959	55758
	s35932	14287	112861	221529	223327
	s38417	10479	33786	88409	92555
	s38584	13216	48423	142829	147508

Table 2: Dataset constraints and parameters.

Parameters	Values or ranges	# of Samples
Clock periods (ns)	{0.5, 1, 2, 5}	4
Aspect ratio	{0.5, 1, 1.5}	3
Max utilization	{0.3, 0.4, 0.5, 0.6, 0.7}	5
Max skew (ns)	[0.01 - 0.2]	1 random sample
Max fanout	[50 - 250]	1 random sample
Max clock network capacitance (pF)	[0.05 - 0.3]	1 random sample
Max latency (ns)	[0 - 1]	1 random sample
Total circuits per design		**60**
Overall circuits in dataset		**60 * 25 = 1500**



(a) (b)

Figure 2: Histograms (in log scale) of (a) gate arrival time, and (b) timing path depth.

circuit are treated as the baseline for the same timing paths after routing.

Histograms of both the post-routing gate arrival time and post-routing timing path depth (number of gates in a timing path) for the entire dataset are shown in Fig. 2. There is a large variation in the size of the benchmark circuits, and a significantly left-skewed logarithmic distribution of the timing path depth is observed. To balance the dataset, all data points are further divided into three categories, small, medium, and large, based on both the total number of gates and the total number of inputs and outputs in the circuit. From the overall dataset, six circuits and all parameterized designs and timing paths of the selected circuits are separated as testing sets. The distribution of the timing paths across the small, medium, and large circuits are listed in Table 1. The circuits selected for the test set are highlighted and in red font in Table 1.

Scatter plots of the initial and final stage estimates of the path delay for the small, medium, large, and entire dataset and for each of the three scenarios described in Section 3.1 are shown in Fig. 3. There is a significant error between the timing estimates provided from two different design stages. The error in the estimated timing is the largest when predicting the post-routing path delay after floorplanning, smallest for predicting the post routing path delay after CTS, and greater for circuits with larger size. The metrics evaluating the error, specifically MAPE and MAE, are selected as a baseline and are calculated as, respectively,

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}|1-\frac{y_i}{\tilde{y}_i}|, \text{ and} \tag{3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i-\tilde{y}_i|, \tag{4}$$

where n is the total number of gates in the overall dataset, y is the STA delay estimate for the initial design stage and $\tilde{y}$ is the STA delay estimate for the final stage. The baseline errors for all 12 models are listed in Table 3.

Table 3: Baseline evaluation of MAPE and MAE for each dataset scenario.

Dataset Group	Scenario	MAPE	MAE
Complete	Floorplan to Routing	82.3	16.44
	Placement to Routing	45.25	11.33
	CTS to Routing	2.96	1.64
Small	Floorplan to Routing	6.26	1.73
	Placement to Routing	2.41	0.72
	CTS to Routing	2.39	0.71
Medium	Floorplan to Routing	8.59	4.18
	Placement to Routing	3.57	1.71
	CTS to Routing	2.63	1.21
Large	Floorplan to Routing	99.6	19.44
	Placement to Routing	51.49	12.79
	CTS to Routing	3.02	1.71

4 Graph based Arrival Time Prediction

To predict the arrival time of the gates of a timing path, the timing paths extracted from the physical implemetation of the circuit (DEF file) are mapped into a directed graph and populated with node features extracted from the netlist and STA timing reports. Additional details on the generation of the graph are provided in Section 4.1. The generated graph representation is then used as input for a graph neural network architecture, which is described in Section 4.2. Models based on linear regression are also utilized for comparison.

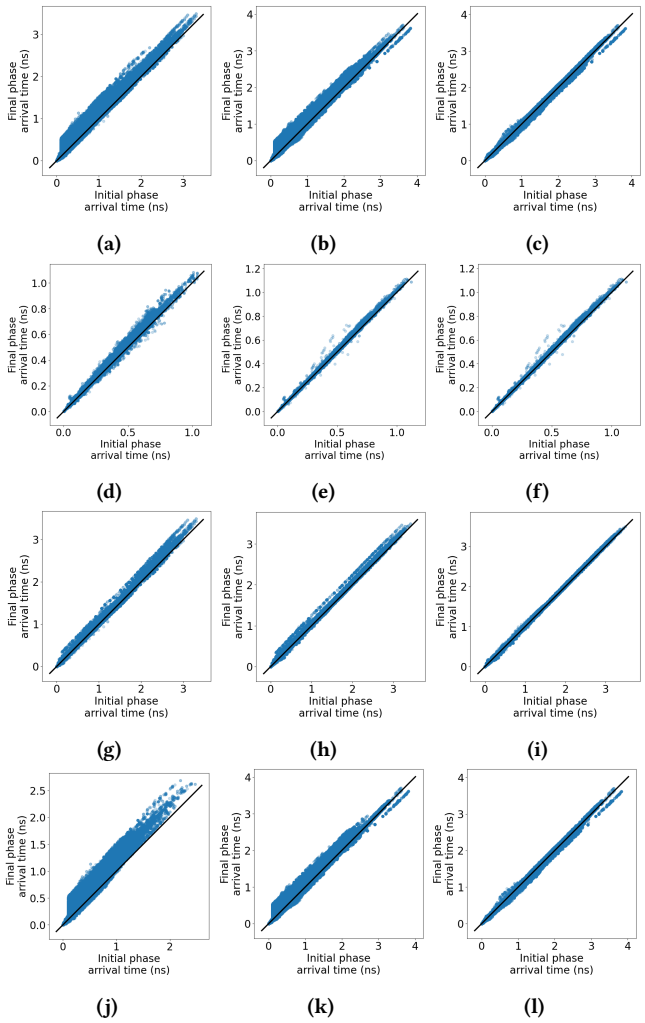


Figure 3: PrimeTime based initial and final phase arrival time estimates for (a) all circuits from floorplan to routing, (b) all circuits from placement to routing, (c) all circuits from CTS to routing, (d) small circuits from floorplan to routing, (e) small circuits from placement to routing, (f) small circuits from CTS to routing, (g) medium circuits from floorplan to routing, (h) medium circuits from placement to routing, (i) medium circuits from CTS to routing. (j) large circuits from floorplan to routing, (k) large circuits from placement to routing, (l) large circuits from CTS to routing,

4.1 Graph Structure and Feature Engineering

The overall flow of the framework and training process is shown in Fig. 4. The physical characteristics of the circuit, described in the Verilog and DEF files, are parsed and represented as a netlist graph $G = (V, E)$, where node $v \in V$ represents inputs, outputs, and gates of a netlist and edge $e \in E$ represents a wire connecting two node components. Subgraphs of the timing paths are extracted from each netlist graph utilizing timing reports generated from the STA tool. The subgraphs of the timing paths represent primary inputs to

Table 4: Node features of the timing graph.

Feature Type	Feature	Size	Source
Setup features	Clock period	1	Design parameters /constraints
	Aspect ratio	1	
	Max utilization	1	
	Max skew	1	
	Max fan-out	1	
	Max clock network capacitance	1	
	Max latency	1	
Standard cell features	Standard cell one hot encoding	859	LEF/DEF file
Structural features	Logic level	1	Calculated from netlist graph representation
	Number of fan-out gates	1	
Timing report features	Initial phase gate delay	1	STA timing report
	Initial phase arrival time	1	
Parasitic features	Total interconnect capacitance	1	IC Compiler generated SPEF file

the model. Each node of a timing path subgraph is populated with a carefully selected feature set, which is listed in Table 4. Setup features are design constraints utilized by DC compiler and IC compiler during physical design. Standard cell features represent the gate functionality of each node of the timing path using a one-hot encoding vector of all cells of a specific technology node (TSMC 65 nm for this work). The overall cell information is extracted from a technology-specific LEF file. Netlist graphs are traversed to compute structural features, whereas timing features are extracted from the timing reports generated by the STA tool and mapped into the circuit netlist. Parasitic features are net capacitance values extracted from the SPEF files generated by IC Compiler. Although edges represent wires in the graph, the total capacitance of a net is set as an attribute of a node, which results in the net being represented by the edge following a given node. The allowed range of capacitances at the output nodes is provided as a constraint during circuit synthesis.

4.2 Model Architecture and Training

After completing feature extraction and graph representation of the circuit, a regression model is developed to predict the gate arrival time. The regression model, as shown in Fig. 5a, is a deep neural network consisting of two GCN layers connected to four linear layers. Each hidden layer utilizes a rectified linear unit (ReLU) [15] as an activation function. The primary objective function is to minimize the mean squared error (MSE) loss between the estimate of the STA delay for the initial design stage, y, and the estimate of the STA delay for the final design stage, $\tilde{y}$. The MSE loss is defined as

$$MSELoss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2, \tag{5}$$

where n is the total number of gates in the overall dataset.

As listed in Table 3, the estimates of the arrival time for the initial stage (floorplanning) are an important feature for the estimates of the final stage arrival time (post-routing). Although the objective of the model is to learn complex non-linear features by leveraging the structural attributes of the graph, *wide and deep* models that include simultaneous interactions between low and high order features provide better model results as compared to models that consider either low or high order features alone [16]. Therefore, a second regression model is developed where the estimates of the arrival time of the initial stage are added as an additional input

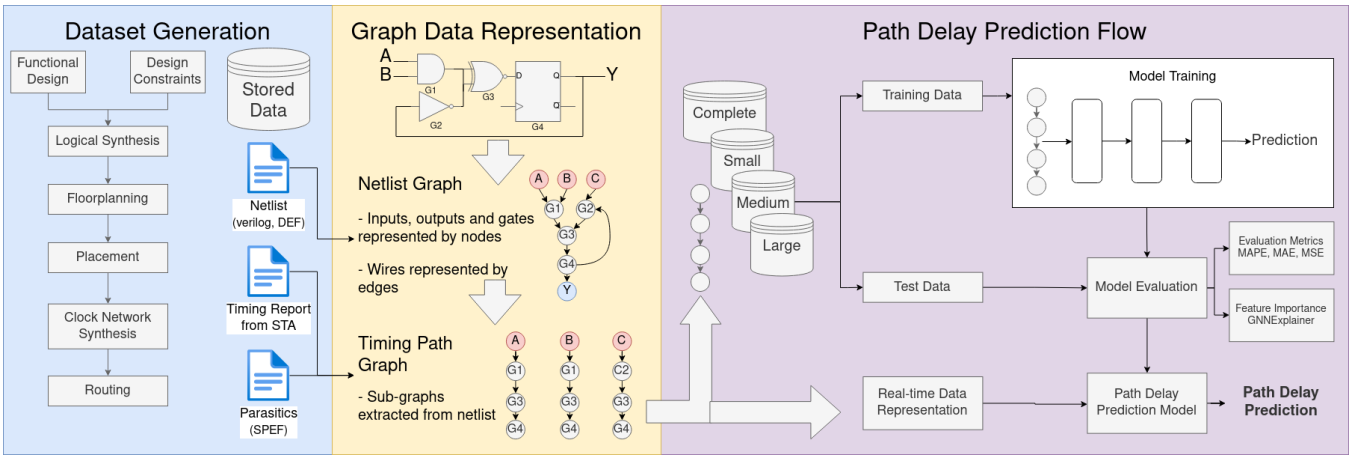


Figure 4: Path delay dataset generation and prediction framework.

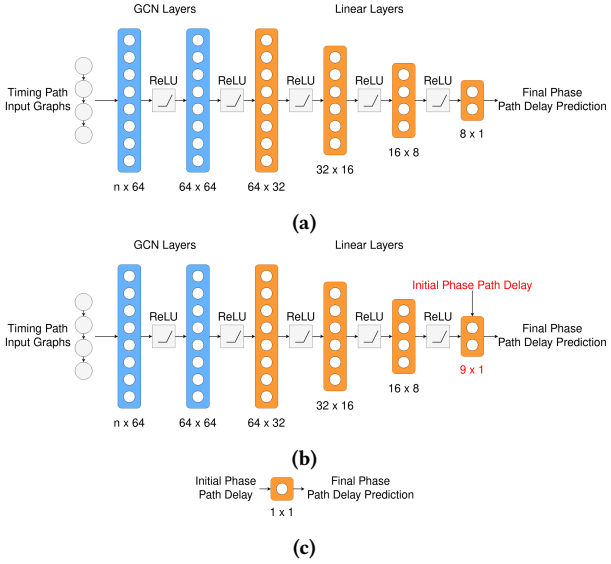


Figure 5: Machine learning network architectures, which include (a) deep GCN network, (b) wide and deep GCN network, and (c) linear regression.

to the final layer of the neural network. The perceptron added to the output layer acts as a generalized linear *wide* component, whereas the remaining layers act as the *deep* component of the overall network. The network architecture of the *wide and deep regression model* is shown in Fig. 5b. Including the estimated path delay from the initial stage as an input of the final layer heavily biases the prediction. The model is expected to learn from both the co-linearity between the arrival time of the initial and final stage and the non-linear characteristics of the overall graph features. To analyze whether the performance of the model is solely based on the linear relationship between the estimated arrival time of the initial and final stage, a linear regression model, shown in Fig. 5c is utilized, where only the initial stage estimate is used as a feature to predict the final stage path delays.

5 Results

In this section, an analysis of the results from execution of the prediction framework described in Section 4 is provided for the three scenarios described in Section 3.2. The proposed neural regression models are trained with a stochastic gradient descent (SGD) optimizer [17] on the training dataset listed in Table 1. Since the complete dataset of timing paths is too large to train all at once, the dataset is divided into batches of 256 timing paths each for a single iteration of training and validation. An initial learning rate of 0.1 is applied that decays at a rate of 95% every 10 training steps. Early stopping is triggered when there is no improvement in the training loss for over 10 epochs, which both terminates the training process as the model stops generalizing and prevents the model from overfitting by not learning the statistical noise in the training dataset. The framework is implemented in Python 3 using the PyTorch-geometric deep learning library[18], and the training is performed on a system with 96 GB memory, twenty four 4 GHz CPU cores, and an NVIDIA GeForce GTX 1080 graphics card.

The trained regression models are validated against the test dataset. MAPE and MAE are calculated for comparison of the models, with the results as listed in Table 5. From the MAPE scores listed, the baselines in all scenarios are consistently outperformed by the wide and deep GCN model except for the post-CTS to post-routing path delay prediction for the complete and large datasets. The dataset groups that comprise larger circuits result in more variations in the path delay, as shown in Fig. 3, and are more challenging to model accurately. The wide and deep GCN models, therefore, perform better for small and medium circuit datasets as compared to large circuit datasets. The average MAPE for the wide and deep GCN models for the small, medium, and large circuits is 2.61%, 4.03%, and 25.34%, respectively. In addition, the difference between the post-CTS and post-routing estimated path delay is minimal as compared to the difference between the post floorplanning to post routing prediction and post-placement to post routing prediction. The average baseline MAPE for post-CTS to post routing prediction is 3.05% as compared to an average baseline MAPE of 44.45% and 25.57% for, respectively, post floorplanning to post routing prediction and post-placement to post routing prediction. The baseline

Table 5: Mean absolute percentage error (MAPE) and mean absolute error (MAE) comparisons across the different datasets, scenarios, and models.

Dataset Group	Scenario	Baseline		Deep GCN		Wide and Deep GCN		Improvement (Wide and Deep GCN vs Baseline)		LR		Improvement (Wide and Deep GCN vs LR)	
		MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE
Complete	Floorplan to Routing	77.86	15.74	161.61	8.56	29.91	7.22	61.58	54.12	32	9.17	6.55	21.29
	Placement to Routing	46.19	13.83	47.34	15.53	39.94	10.51	13.53	24	40.81	13.88	2.13	24.28
	CTS to Routing	2.84	1.81	15.87	7.2	3.56	1.79	-25.35	1.1	4.32	1.99	17.55	10.03
Small	Floorplan to Routing	7.25	1.94	13.35	3.07	4.28	1.06	41.01	45.15	4.37	1.12	2.11	4.88
	Placement to Routing	2.71	0.67	19.08	5	1.77	0.38	34.5	42.92	1.98	0.47	10.31	18.9
	CTS to Routing	2.69	0.66	23	6.55	1.8	0.39	33.06	41.59	2	0.48	9.9	19.1
Medium	Floorplan to Routing	6.46	2.68	35.35	9.81	6.42	1.6	0.67	40.17	6.79	1.69	5.54	5.13
	Placement to Routing	3.97	1.82	27.48	13.67	3.34	1.15	15.85	36.91	3.97	1.27	15.86	9.43
	CTS to Routing	3.9	1.8	22.2	7.86	2.35	1.12	39.69	37.62	2.79	1.22	15.74	7.74
Large	Floorplan to Routing	86.23	17.34	26.6	8.16	32.06	7.42	62.82	57.19	31.55	9.11	-1.6	18.5
	Placement to Routing	49.41	14.77	33.31	11.43	40.61	10.36	17.8	29.83	41.74	14.01	2.69	26.04
	CTS to Routing	2.8	1.84	16.83	7.97	3.6	1.83	-28.6	0.39	3.75	1.95	3.87	5.89

Table 6: Mutual importance of features across trained network architectures. Listed features have mutual importance scores greater than 0.4 on a 1.0 scale.

Dataset Group	Scenario	Deep GCN Network	Wide and Deep GCN Network
Complete	Floorplan to Routing	Interconnect capacitance (0.74), DFQD1 (0.72), Initial stage arrival time (0.67)	DFQD1 (0.72), Initial stage arrival time (0.72), Interconnect capacitance (0.62)
	Placement to Routing	Initial stage arrival time (0.74), DFQD1 (0.72), Node delay (0.69), Interconnect capacitance (0.67), Logic depth (0.45)	Initial stage arrival time (0.74), DFQD1 (0.7), Logic depth (0.69), Interconnect capacitance (0.51)
	CTS to Routing	Initial stage arrival time (0.73), Logic depth (0.49), Interconnect capacitance (0.41), DFQD1 (0.4)	Initial stage arrival time (0.72)
Small	Floorplan to Routing	Initial stage arrival time (0.76), No. of fan-out (0.76), NR3D0 (0.72)	Interconnect capacitance (0.74), Initial stage arrival time (0.73), No. of fan-out (0.72)
	Placement to Routing	NR3D0 (0.74), Interconnect capacitance (0.73), Initial stage arrival time (0.73)	Initial stage arrival time (0.74), No. of fan-out (0.72), Interconnect capacitance (0.44)
	CTS to Routing	ND2D1 (0.77), Initial stage arrival time (0.76), DFQD1 (0.72), Logic depth (0.53)	Initial stage arrival time (0.72)
Medium	Floorplan to Routing	Initial stage arrival time (0.74)	Initial stage arrival time (0.71)
	Placement to Routing	Initial stage arrival time (0.76), Logic depth (0.74), No. of fan-out (0.73), OAI211D1 (0.7), CKND3 (0.69)	Initial stage arrival time (0.74)
	CTS to Routing	Initial stage arrival time (0.7)	Initial stage arrival time (0.74)
Large	Floorplan to Routing	Initial stage arrival time (0.74), Interconnect capacitance (0.71), Node delay (0.61)	Initial stage arrival time (0.72) DFQD1 (0.72), Interconnect capacitance (0.44)
	Placement to Routing	Initial stage arrival time (0.71), Node delay (0.7), DFQD1 (0.65), Logic depth (0.51)	Initial stage arrival time (0.72), DFQD1 (0.69), Logic depth (0.66), Interconnect capacitance (0.59)
	CTS to Routing	Initial stage arrival time (0.73)	Initial stage arrival time (0.74)

model is, therefore, performing very well, which leaves less room for gain when predicting post-routing path delay after completing CTS.

The linear model, similar to the wide and deep GCN, also outperforms the baseline in the same scenarios; however, underperforms the wide and deep GCN for all scenarios except for the post-floorplan to post-routing arrival time prediction of large circuits, where a 1.6% improvement is observed for the given single case. When comparing the MAPE between the arrival time predicted by the wide and deep GCN model and the linear model, an average improvement of 2.9%, 12.18%, and 15.7% is observed for post-floorplanning to post-routing prediction, post-placement to post-routing prediction, and post-CTS to post-routing prediction, respectively. Although the deep GCN model converges, indicating some learning, the resulting predictions are consistently worse than the baseline. The MAE calculation of the deep GCN models behaves similar to the MAPE, which re-enforces the insights observed from the analysis of the MAPE of the models.

An additional advantage of utilizing graph neural networks is the ability to apply a diverse set of inputs to the model and obtain insights by analyzing the important features of the model. GNNExplainer is utilized to measure the mutual information of each feature of the graph neural network and characterize the feature importance. Critical features for each GNN model across all scenarios are listed in Table 6. Structural features such as logic depth and the number of fan-outs, and parasitic features consistently rank as the greatest contributing features of the model. The estimated arrival time determined after completion of the initial stage of the design flow is a significant feature of both the deep GCN neural network architecture and the wide and deep GCN neural network architecture and results in a mutual importance score greater than 0.67 across all models. The deep GCN model includes greater variation in the important gate features, which consist of the DFDQ1, NR3D0, ND2D1, and CKND3 standard cells. The wide and deep GCN only lists DFDQ1 as the sole important gate feature. There are fewer features with mutual importance greater than 0.4 in the wide and deep GCN models, which suggests the models leverage fewer features for prediction, especially for the estimation of the arrival time in the initial design stage.

6 Conclusion

In this paper, a methodology that utilizes a wide and deep graph regression model is proposed to predict the post routing gate arrival time from a timing path generated post-floorplanning, post-placement, and post-CTS of the IC design flow. Graph representations of netlists and timing paths for ISCAS'89 benchmark circuits generated from commercial EDA and STA tools are used as datasets, and an error analysis is performed to obtain baseline errors from the commercial tools. When validating the trained model, an average improvement of 65.58% in MAPE is observed when predicting the post-routing arrival time after completing floorplanning and 13.53% when predicting the post-routing arrival time post-placement. By subdividing the dataset based on overall netlist size, an average improvement of 34.83% for post floorplan prediction and 22.17% for post-placement prediction is observed as compared to the baseline. The proposed machine learning methodology largely reduces errors and outperforms the baseline across all modeling scenarios except for the post-CTS to post routing predictions, for which the commercial tool already provides a low baseline error.

7 Acknowledgements

This research is supported in part by the National Science Foundation under Grant CNS-1751032.

References

[1] "Gold standard in static timing analysis - Primetime," https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html.

[2] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, Vol. 1, pp. 57–81, Jan. 2020.

[3] W. Haaswijk, E. Collins, B. Seguin, M. Soeken, F. Kaplan, S. Süsstrunk, and G. De Micheli, "Deep learning for logic optimization algorithms," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, May 2018.

[4] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, *et al.*, "Chip placement with deep reinforcement learning," *arXiv preprint arXiv:2004.10746*, pp. 1–15, Apr. 2020.

[5] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, Nov. 2019.

[6] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," *Advances in Neural Information Processing Systems*, Vol. 32, pp. 9240–9251, Dec. 2019.

[7] A. B. Kahng, "Machine learning applications in physical design: Recent results and directions," *Proceedings of the International Symposium on Physical Design (ISPD)*, pp. 68–73, Mar. 2018.

[8] A. B. Kahng, S. Kang, H. Lee, S. Nath, and J. Wadhwani, "Learning-based approximation of interconnect delay and slew in signoff timing tools," *Proceedings of the ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1–8, Dec. 2013.

[9] S.-S. Han, A. B. Kahng, S. Nath, and A. S. Vydyanathan, "A deep learning methodology to proliferate golden signoff timing," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, Mar. 2014.

[10] W.-T. J. Chan, K. Y. Chung, A. B. Kahng, N. D. MacDonald, and S. Nath, "Learning-based prediction of embedded memory timing failures during initial floorplan design," *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 178–185, Jan. 2016.

[11] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism," *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, Jun. 2019.

[12] S. R. Ramesh and R. Jayaparvathy, "Artificial neural network model for arrival time computation in gate level circuits," *Automatika*, Vol. 60, No. 3, pp. 360–367, Oct. 2019.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, pp. 1–14, Sept. 2016.

[14] "The ISCAS-89 Benchmark Circuits.," http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/verilog.

[15] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," *arXiv preprint arXiv:1803.08375*, pp. 1–7, Mar. 2018.

[16] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide and deep learning for recommender systems," *Proceedings of the Workshop on Deep Learning for Recommender Systems*, pp. 7–10, Sept. 2016.

[17] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, pp. 1–14, Sept. 2016.

[18] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," *arXiv preprint arXiv:1903.02428*, pp. 1–9, Mar. 2019.