Modeling Graph Neural Networks for Gate Level Arrival Time Prediction of Long Tailed Datasets

Pratik Shrestha, Saran Phatharodom and Ioannis Savidis
Department of Electrical and Computer Engineering, Drexel University
Philadelphia, Pennslyvania 19104
ps937, sp694, is338@drexel.edu

***Abstract*—Iterative re-calibration of the design parameters to meet timing closure is a critical and computationally costly component of the overall integrated circuit (IC) design flow. By predicting timing profiles at the initial stages of the design flow, early modifications to the circuit reduce overall design time. In this work, a graph-based deep regression model is utilized for the prediction of the gate-level arrival time of the timing paths of a circuit. Three scenarios for the post-routing prediction of timing path arrival time are considered: prediction after floorplanning, prediction after placement, and prediction after clock tree synthesis (CTS). A commercial static timing analysis tool is used to determine the mean absolute percentage error (MAPE) and 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 commercial physical design tools, with an average improvement of 61.58% in the MAPE score when predicting the post-routing arrival time after floorplanning and a 13.53% improvement when predicting the post-routing arrival time after placement. Given the long-tailed nature of the data, further prediction scenarios are analyzed, where the complete dataset is subdivided based on both circuit size (small, medium, and large) and timing path size (timing path logical depths of less than 5 and timing path logical depths greater than or equal to 5). Models trained for small and medium datasets result in an average MAPE of 2.61% and 4.03%, respectively, while the average MAPE of the complete dataset is approximately 24.47%. Similarly, models trained on datasets with timing paths of logical depths < 5 and logical depths $>= 5$ result in an average MAPE of 24.01% and 15.62%, respectively.**

***Index Terms*—timing path prediction, physical design, machine learning, graph convolutional networks**

I. Introduction

Modern electronic design automation (EDA) tools spend a considerable amount of time iterating through the design process to ensure that the circuit specifications and timing constraints are met. 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 when executing static timing analysis (STA) on a 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] and Cadence Tempus [2] 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.

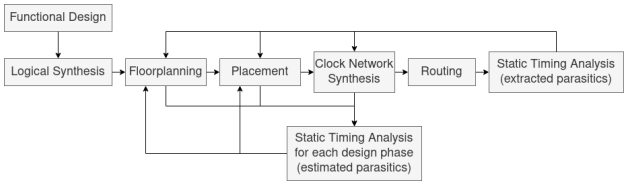

Fig. 1: Physical design automation flow.

One of the main challenges of timing prediction is the long-tailed nature of the data. Larger circuits are particularly affected by long-tailed data as such circuits include a large number of timing paths of various sizes, which results in a distribution of the timing data that is highly variable (long tail of extreme values). The accurate modeling and prediction of the timing of the circuit is, therefore, challenging and can lead to timing errors when finalizing design.

In this paper, a novel graph based regression framework is introduced to predict gate arrival time. The framework uses a graph representation based on timing paths and includes node features from the circuit netlist and STA reports to predict path delay in different phases of physical design. A dataset of timing paths is divided into sub-datasets based on the size of the circuit from which the timing path originated and the size of the timing path itself. Three primary scenarios are considered when predicting the post-routing arrival time: post-floorplan, post-placement, and post-CTS prediction of the post-routing arrival time. The contributions of this 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 linear regression model, and
- Feature importance analysis on the neural models using GNNExplainer [3].

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 II. An error analysis of timing estimates from a commercial STA tool and a baseline model of the errors are described in Section III. The framework for arrival time prediction is discussed in Section IV, and an analysis of the results produced by the proposed framework is provided

in Section V. Some concluding remarks are provided in Section VI.

II. Background and Related Work

Prior work on applying machine learning for gate level path delay estimation is discussed in Section II-A. The use of graph neural network layers for graph representation learning and graph model explanation is described in Section II-B.

A. Machine Learning for Gate Level Path Delay Estimation

Prior work has applied machine learning (ML) to timing analysis. In [4], the discrepancy between results from different timing analysis tools for the same circuit netlist is reduced. In [5], a ML-based approach is employed to fit correlation-based models of wire slew and delay to estimates from a sign-off STA tool. A deep hierarchical model is proposed in [6] to predict the path delay of an unexecuted commercial EDA tool from the timing analysis of another commercial tool. Support vector machines (SVMs) were utilized for floorplanning to predict post-layout timing failures of embedded memory in [7]. In [8], a random forest based approach is proposed to predict the pre-routing timing profile of a circuit. An artificial neural network is proposed in [9] to predict the statistical static timing analysis (sSTA) profile of a circuit characterized by a commercial STA tool. In [10], GCNs are utilized to predict the post routing arrival time post-floorplan, post-placement, and post-CTS. However, partitioning of the dataset based on the timing path depth is not considered in [10].

B. Graph Learning and Neural Network Explanation

A wide variety of structures are represented as graphs and modeled into a low dimensional vector embedding using graph neural networks (GNNs) [11], which are then utilized for various learning applications. Circuit netlists represented as graphs leverage structural information to allow for more holistic learning. In this work, graph convolutional layers (GCN) [12] are used to model the delay of gates from timing path graphs. GCNs utilize convolution functions, which multiply input neurons by a set of weights 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.

A limitation of GCNs is the lack of interpretability of the neural model after training. GNNExplainer [3] provides interpretable explanations of graph predictions by maximizing mutual information (MI), which is 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

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

III. Dataset Design and Baseline

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

A. Dataset Generation

In this work, ISCAS'89 sequential benchmark circuits [13], which are listed in Table I, are used to create 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 used for physical design. Multiple layouts are generated for each benchmark circuit by utilizing parameterized design constraints, which are listed in Table II. A total of 1500 physical designs are generated for the 25 benchmark circuits, 60 designs of each benchmark circuit. PrimeTime is used to perform static timing analysis on the generated logical and physical designs at each stage of the design flow. Timing paths, gate delays, and input/output delays are extracted from the timing reports provided by PrimeTime.

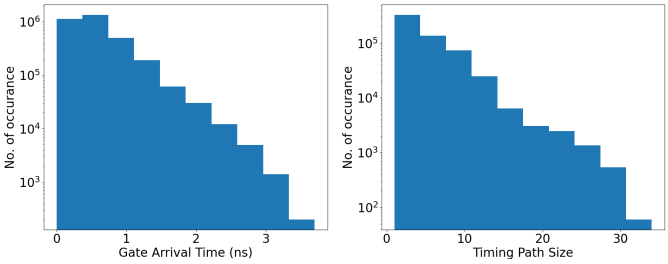

Fig. 2: Histogram (in log scale) of occurrence of (a) gate arrival time, and (b) timing path logical depth.

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. To balance the dataset, all data points are further divided into two sub-dataset classes. The first sub-dataset class is divided into small, medium, and large circuit sizes, based on the total number of gates and the total number of inputs and outputs of the circuit. The second sub-dataset class is divided into two categories based on the logical depth of the timing path, where logical depths of less than 5 and logical depths of greater than or equal to 5 are partitioned. From the overall dataset, six circuits and all parameterized designs and timing paths of the six circuits are separated as the testing sets. The distribution of the timing paths across the small, medium, and large circuits are listed in Table I, of which the circuits selected for the test set are highlighted and in red font.

B. Baseline Path Delay

Scatter plots of the initial and final stage estimates of the path delay for all six groups of data including the entire dataset and for each of the three scenarios described in Section III-A are shown in Fig. 3. There is significant variation in the error between the timing estimates provided from different design

TABLE I: ISCAS'89 benchmark circuits and dataset statistics. Circuits selected as test data are highlighted and in red font.

Dataset group	Circuit	Circuit logical depth	Floorplan to Routing			Placement to Routing			CTS to Routing		
			# of timing paths	timing paths logical depth < 5	timing paths logical depth >= 5	# of timing paths	timing paths logical depth < 5	timing paths logical depth >= 5	# of timing paths	timing paths logical depth < 5	timing paths logical depth >= 5
Small	s27	16	540	300	240	540	300	240	540	300	240
	s298	98	1027	966	61	1565	1038	527	1566	1038	528
	s344	136	1905	1329	576	2310	1425	885	2310	1425	885
	s349	139	1885	1399	486	2262	1518	744	2262	1518	744
	s382	129	1601	911	690	2571	1585	986	2571	1585	986
	s386	138	1195	224	971	1380	232	1148	1380	232	1148
	s400	136	1561	1020	541	2644	1687	957	2640	1687	953
	s420	175	1175	358	817	1681	388	1293	1681	388	1293
	s444	149	1465	804	661	2687	1422	1265	2688	1422	1266
	s510	211	293	42	251	830	44	786	830	44	786
Medium	s641	185	3276	870	2406	4283	886	3397	4281	886	3395
	s713	216	3138	871	2267	4167	884	3283	4167	884	3283
	s820	298	1269	349	920	2186	564	1622	2186	564	1622
	s832	304	1439	516	923	2221	708	1513	2222	708	1514
	s838	355	2123	251	1872	3284	335	2949	3286	335	2951
	s953	379	1796	1545	251	3892	1557	2335	3892	1557	2335
	s1196	434	438	167	271	2678	420	2258	2684	420	2264
	s1238	474	538	174	364	2795	433	2362	2797	433	2364
	s1423	586	2753	994	1759	6268	1428	4840	6309	1427	4882
Large	s9234	2313	8088	5302	2786	12525	6146	6379	12517	6146	6371
	s13207	3425	33174	27333	5841	47845	31641	16204	49092	32057	17035
	s15850	4209	21432	15338	6094	54959	24240	30719	55758	24264	31494
	s35932	14287	112861	85387	27474	221529	174568	46961	223327	174869	48458
	s38417	10479	33786	27900	5886	88409	38357	50052	92555	38691	53864
	s38584	13216	48423	27242	21181	142829	54245	88584	147508	54444	93064

TABLE II: 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**

TABLE III: 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

stages. The error is larger when predicting the post-routing path delay after floorplanning, smaller when predicting the post-routing path delay after CTS, and generally greater for circuits with larger size. In addition, the error is larger for timing paths with smaller logical depths. Given the initial and final stages of the IC design flow, the timing profile of the initial stage (post-floorplan) provided by the STA tool is considered as the baseline prediction for the timing profile of the final stage (post-routing). MAPE and MAE are selected as the baseline metrics of evaluation and are defined as, respectively,

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

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

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 18 models are listed in Table IV.

IV. Graph Learning based Arrival Time Prediction

To predict the arrival time of gates of a timing path, the timing paths extracted from the physical designs 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 IV-A. The generated graph representation is then used as input for a graph neural network architecture, which is described in Section IV-B. Note that, models based on linear regression are used for comparison.

A. 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 the model. Each node of a timing path subgraph is populated with a carefully selected feature set, which is listed in Table III. Setup features are design constraints utilized by DC compiler during logical synthesis 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.

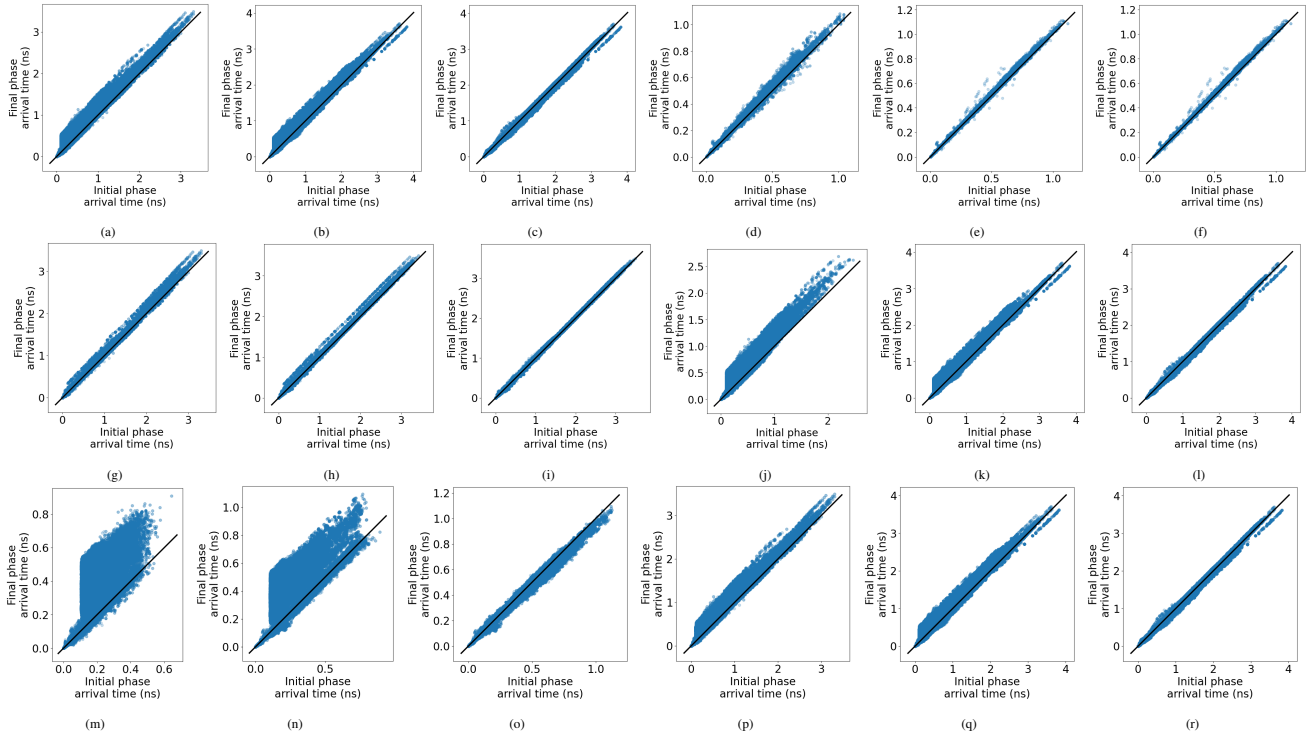

Fig. 3: PrimeTime based initial and final design stage arrival time estimates of (a) all circuits from floorplan to routing, (b) all circuits from placement to routing, (c) all circuits from CTS to routing, (d) small circuits from floorplan to routing, (e) small circuits from placement to routing, (f) 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, (m) timing paths with logical depth < 5 from floorplan to routing, (n) timing paths with logical depth < 5 from placement to routing, (o) timing paths with logical depth < 5 from CTS to routing, (p) timing paths with logical depth $>= 5$ from floorplan to routing, (q) timing paths with logical depth $>= 5$ from placement to routing, (r) timing paths with logical depth $>= 5$ from CTS to routing.

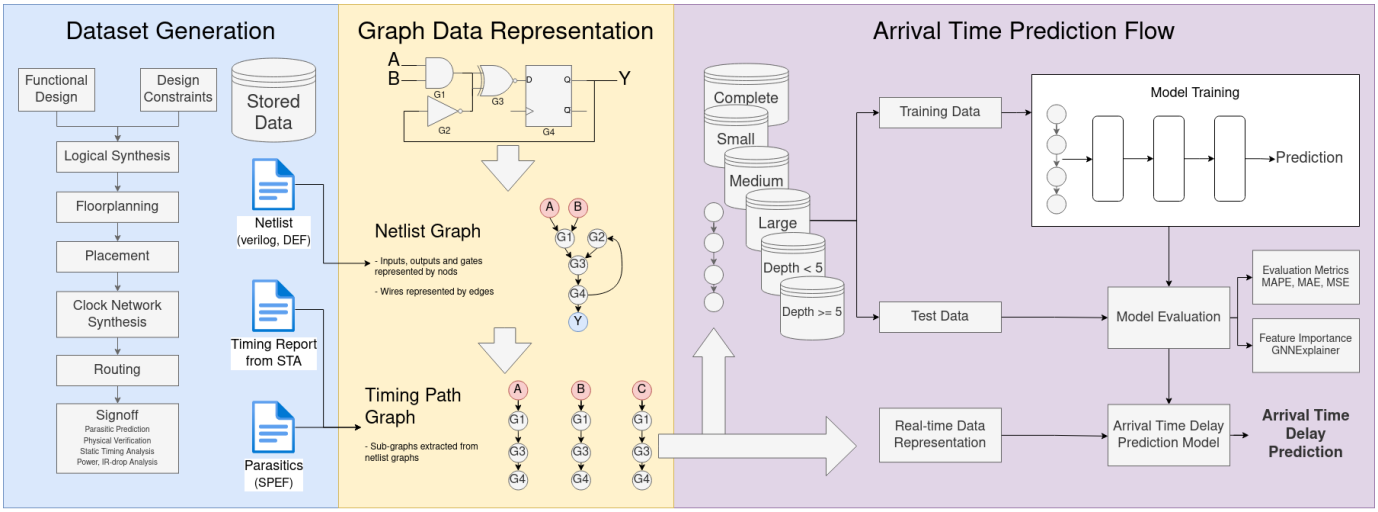

Fig. 4: Path delay dataset generation and prediction framework.

B. Model Architecture and Training

After feature extraction and graph representation of a circuit, a regression model is developed to predict the gate arrival time. The model, as shown in Fig. 5a, implements **wide and deep regression** and consists of two GCN layers connected to four linear layers. The hidden layer utilizes a rectified linear unit (ReLU) [14] as an activation function. As listed in Table IV, the estimates of the path delay of the initial stage (floorplanning) are an important feature for the estimates of the path delay of the final stage (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 [15]. Therefore, the estimates of the arrival time of the initial stage are added as an additional input 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 primary objective function is to minimize the mean squared error (MSE) loss between the estimate of the

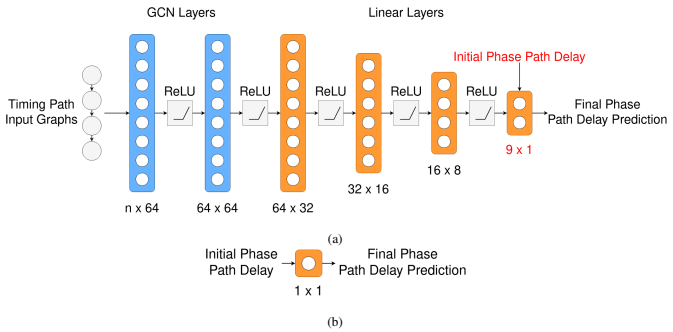

Fig. 5: Architectures of the learning network, which includes (a) a wide and deep GCN network and (b) a linear regression network.

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.

Including the path delay of 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 arrival time estimates of the initial and final stage, a linear regression model, shown in Fig. 5b, is utilized, where only the initial stage estimate is used as a feature to predict the delay of the final design stage.

V. Results

In this section, analysis of the results from the execution of the prediction framework described in Section IV for the three scenarios described in Section III-B is provided. The proposed neural regression models are trained on the training dataset listed in Table I with a stochastic gradient descent (SGD) optimizer [16]. 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 used that decays at a rate of 95% every 10 training steps. Early stopping is triggered when there is no improvement in training loss for over 10 epochs, which terminates the training process as the model stops generalizing. The framework is implemented in Python 3 using the PyTorch-geometric deep learning library [17] and trained on a machine with 94 GB memory, 4 GHz CPUs, and an NVIDIA GeForce GTX 1080.

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 IV. From the MAPE listed in Table IV, 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 of the datasets that contain all circuits, large circuits, timing paths with logical depth less than 5, and timing paths with logical depth greater than or equal to 5. When considering the complete dataset, an average improvement of 61.58% and 13.53% in MAPE is observed when predicting the post-routing arrival time after completing floorplanning and post-placement, respectively. However, when predicting the post-routing arrival time after completing CTS, a degradation of 25.35% in MAPE is observed. The difference between the post-CTS and post-routing estimated path delay prediction of the baseline is minimal as compared to the difference between the post floorplanning to post routing prediction and the post-placement to post routing prediction. The average baseline MAPE for post-CTS to post routing prediction is 3.03% as compared to an average baseline MAPE of 59.53% and 39.34% for, respectively, post floorplanning to post routing prediction and post-placement to post routing prediction. Therefore, the baseline model is performing well, which leaves small room for gain with learning based models when predicting the post-routing path delay after completing CTS.

The datasets 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 datasets as compared to the large dataset. The average MAPE for the wide and deep GCN models of the small, medium, and large datasets is 2.61%, 4.03%, and 25.34%, respectively, as compared to the 24.47% average MAPE for the complete dataset. For small and medium sized circuits, models trained on the small and medium datasets perform better than models trained on the complete dataset. In addition, GCN models for datasets with timing path logical depths of < 5 result in an average MAPE of 24.01%, while GCN models for datasets with timing path logical depths of $>= 5$ result in an average MAPE of 15.62%. Therefore, for timing path logical depths of $>= 5$, GCN models trained on timing path logical depths of the same range provide better performance than models trained on the complete dataset.

The linear model, similar to the wide and deep GCN, also outperforms the baseline for the same scenarios. However, the linear model 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 1.62% improvement over the wide and deep GCN is achieved for this single case. When comparing the MAPE between the arrival time prediction of 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.

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 V. Structural features such as logic depth and the number of fan-outs, and parasitic features consistently rank as the greatest contributing components to the model. The estimated arrival time determined after completion of the initial stage of the design flow is a significant feature of the wide and deep GCN neural network architecture and results in a mutual importance score greater than 0.7 across all models. DFDQ1 is the sole important gate feature. There are fewer features with mutual importance scores greater than 0.4 in the wide and deep GCN models, which suggests that the models leverage fewer features for prediction, especially for the estimation of the arrival time in the initial design stage (post floorplan).

TABLE IV: Mean absolute percentage error (MAPE) and mean absolute error (MAE) comparison across the different datasets, scenarios, and models.

Dataset Group	Scenario	Baseline		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
Complete	Floorplan to Routing	77.86	15.74	29.91	7.22	61.58	54.12	32	9.17	6.55	21.29
	Placement to Routing	46.19	13.83	39.94	10.51	13.53	24	40.81	13.88	2.13	24.28
	CTS to Routing	2.84	1.81	3.56	1.79	-25.35	1.1	4.32	1.99	17.55	10.03
Small	Floorplan to Routing	7.25	1.94	4.28	1.06	41.01	45.15	4.37	1.12	2.11	4.88
	Placement to Routing	2.71	0.67	1.77	0.38	34.5	42.92	1.98	0.47	10.31	18.9
	CTS to Routing	2.69	0.66	1.8	0.39	33.06	41.59	2	0.48	9.9	19.1
Medium	Floorplan to Routing	6.46	2.68	6.42	1.6	0.67	40.17	6.79	1.69	5.54	5.13
	Placement to Routing	3.97	1.82	3.34	1.15	15.85	36.91	3.97	1.27	15.86	9.43
	CTS to Routing	3.9	1.8	2.35	1.12	39.69	37.62	2.79	1.22	15.74	7.74
Large	Floorplan to Routing	86.23	17.34	32.06	7.42	62.82	57.19	31.55	9.11	-1.6	18.5
	Placement to Routing	49.41	14.77	40.61	10.36	17.8	29.83	41.74	14.01	2.69	26.04
	CTS to Routing	2.8	1.84	3.6	1.83	-28.6	0.39	3.75	1.95	3.87	5.89
Timing path logical depth < 5	Floorplan to Routing	120.37	16.96	33.57	7.33	72.11	56.78	35.12	8.46	4.42	13.4
	Placement to Routing	90.31	14.44	33.14	6.94	63.31	51.95	43.26	11.54	23.4	39.89
	CTS to Routing	3.33	1.27	5.34	0.86	-60.23	32.52	8.58	1.44	37.78	40.47
Timing path logical depth $>= 5$	Floorplan to Routing	50.66	17.73	18.09	7.98	64.29	55.01	18.17	8.01	0.43	0.38
	Placement to Routing	40.27	14.84	25.30	10.54	37.18	28.96	31.13	13.39	18.73	21.25
	CTS to Routing	2.69	1.96	3.47	2.01	-28.97	-2.39	3.83	2.16	9.40	7.13

TABLE V: Mutual importance of features across trained network architectures. Features are listed when mutual importance scores are greater than 0.4 on a 1.0 scale.

Dataset Group	Floorplan to Routing	Placement to Routing	Placement to Routing
Complete	DFQD1 (0.72), Initial stage arrival time (0.72), Interconnect capacitance (0.62)	Initial stage arrival time (0.74), DFQD1 (0.7), Logic depth (0.69), Interconnect capacitance (0.51)	Initial stage arrival time (0.72)
Small	Interconnect capacitance (0.74), Initial stage arrival time (0.73), No. of fan-out (0.72)	Initial stage arrival time (0.74), No. of fan-out (0.72), Interconnect capacitance (0.44)	Initial stage arrival time (0.72)
Medium	Initial stage arrival time (0.71)	Initial stage arrival time (0.74)	Initial stage arrival time (0.74)
Large	Initial stage arrival time (0.72) DFQD1 (0.72), Interconnect capacitance (0.44)	Initial stage arrival time (0.72), DFQD1 (0.69), Logic depth (0.66), Interconnect capacitance (0.59)	Initial stage arrival time (0.74)
Timing path logical depth < 5	Initial stage arrival time (0.72), DFQD1 (0.68)	Initial stage arrival time (0.74), DFQD1 (0.72), Logic depth (0.68)	Initial stage arrival time (0.71)
Timing path logical depth $>= 5$	Initial stage arrival time (0.72)	Initial stage arrival time (0.73), Interconnect capacitance (0.55), No. of fan-out (0.43)	Initial stage arrival time (0.75), Logic depth (0.72)

VI. Conclusions

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 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 models, an average improvement of 61.58% 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 the overall size of the circuit, an average MAPE of 2.61% and 4.03% is observed for models trained on the small and medium datasets, respectively, as compared to the 24.47% average MAPE for the complete dataset. Similarly, models trained on datasets of timing paths with logic depth of < 5 and logic depth of $>= 5$ result in an average MAPE of 24.01% and 15.62%, respectively. The proposed methodology largely reduces errors and outperforms the baseline across all modeling scenarios except for the post-CTS to post routing arrival time prediction, for which the commercial tool already provides a low baseline error.

References

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

[2] Inc. Cadence Design Systems, "Tempus Timing Solution," https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] P. Shrestha, S. Phatharodom, and I. Savidis, "Graph representation learning for gate arrival time prediction," *Proceedings of the ACM/IEEE Workshop on Machine Learning for CAD*, pp. 127–133, Sept. 2022.

[11] 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.

[12] T.-N Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proceedings of the International Conference on Learning Representations*, pp. 1–14, Feb. 2016.

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

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

[15] 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.

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

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