TraceFormer: S-parameter Prediction Framework for PCB Traces based on Graph Transformer

Doyun Kim, Jaemin Park, Youngmin Oh, and Bosun Hwang
Samsung Advanced Institute of Technology (SAIT)
{doyun.d.kim,jm23.park,youngmin0.oh,bosun.hwang}@samsung.com

Abstract

Signal integrity becomes more critical to modern digital systems such as solid-state drives due to their high-speed operation. However, one of the challenges in signal integrity analysis is S-parameter modeling process for printed circuit boards (PCB). Due to increasing PCB design complexity, existing numerical methods take too long to solve governing equations for S-parameters. To overcome the issue, we present a novel deep learning framework, TraceFormer, to predict S-parameters of PCB traces. Our framework constructs a graph from PCB traces and tokenizes trace segments with geometric and topological information. A transformer encoder produces PCB representations from the tokens, followed by extraction networks which predict four different types of complex-valued S-parameters together. TraceFormer achieved above 0.99 R-squared score up to 15GHz for 4-port PCB designs, resulting in less than 3.1% and 4.2% errors in terms of the eye diagram's width and height, respectively.

Keywords: Signal Integrity, S-parameters, PCB, Trace, Eye Diagram, Channel Modeling

ACM Reference Format:
Doyun Kim, Jaemin Park, Youngmin Oh, and Bosun Hwang. 2024. TraceFormer: S-parameter Prediction Framework for PCB Traces based on Graph Transformer. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3658484

1 Introduction

As the demand for higher bandwidth and faster data rates increases, the issue of signal integrity (SI) becomes critical. In modern high-speed digital systems such as solid-state drives (SSD), even minor signal distortions can lead to significant data corruption and system failures. However, the trend towards miniaturization and high-density packaging amplifies the risk of signal interference and loss. Therefore, it is necessary to ensure robust signal integrity over a wide frequency range.

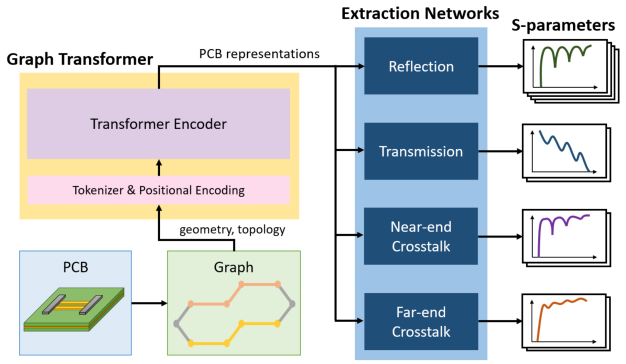


Fig. 1. TraceFormer overview. PCB trace data is converted into a graph by splitting trace into segments. Each segment's position and graph topology are fed into Graph Transformer model to produce PCB representations. Four extraction networks predicts each type of complex-valued S-parameters over a frequency range.

Channel simulation is one of the common ways of signal integrity analysis. We first model each part of a system: a driver, a receiver, and a channel, run transient simulations, and obtain an eye diagram as a measure of signal integrity. In modern digital systems, channels generally consist of signal paths on die, package, and printed circuit board (PCB) levels, which require separate modeling. PCB designs can be modeled by Scattering parameters (S-parameters), which are obtained by electromagnetic simulation. However, as PCB design complexity increases, for example, SSD boards with dozens of DRAM and NAND packages [1], S-parameter modeling for PCB designs takes a large amount of time. The prolonged modeling time slows down the channel simulation setup process in addition to its slow transient simulation time.

As other silicon design processes, SI has also been adopting neural networks' capability of learning from data for design productivity. In [2, 3], neural networks design and optimize antenna designs rather than analyze them. Convolutional neural networks are utilized in [4] to predict S-parameters of bandpass filters but it only applies to limited design choices. A recent study [5] also predicts S-parameters, but it is limited to interpolation. Unlike other works, [6] directly predicts eye diagrams, but it only works with pre-determined PCB structures. Thus, the major drawback of previous studies is that they restrict the use of only predefined channel structures.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3658484

To address this issue, we present a novel deep learning framework, TraceFormer, in this paper. TraceFormer exploits neural networks' capability of learning patterns of PCB data to predict their S-parameters. Our framework first splits PCB traces into segments at the bending points and converts them into a graph where the bending points and segments become nodes and edges. Each segment's features such as 2D coordinates of its both endpoints and length alongside Laplacian summation positional encoding are used to generate the corresponding token. In this way, tokens are able to maintain geometric and topological information. A transformer encoder produces PCB representations using the tokens. Finally, extraction networks predict different types of S-parameters together (transmission, reflection, near-end crosstalk, and far-end crosstalk) over a wide frequency range.

Our contribution is followed below.

- To the best of our knowledge, TraceFormer is the first work to predict S-parameters based on PCB traces. In particular, we do not impose restrictions on PCB design, allowing traces to have arbitrary shapes.
- TraceFormer utilizes the graph transformer to utilize geometric and topological information of segments from PCB traces. It recognizes not only adjacent segments within a trace but also near segments in other traces so that it can comprehend the coupling effect.
- We validate performance with S-parameters and eye diagrams. TraceFormer achieved >0.99 R-squared (R2) score for S-parameter prediction up to 15GHz, resulting in <1% and <1% error in eye height and width, respectively.

The rest of the paper is organized as follows. We explain concepts of S-parameters, an eye diagram and graph Transformer in Section 2. In Section 3, we introduce TraceFormer which consists of graph tokenizer, transformer encoder, and S-parameter extraction networks. Then we show TraceFormer's performance in Section 4 and conclude the paper in Section 5.

2 Preliminaries

2.1 S-parameters

S-parameters are commonly used for design and analysis of high-speed circuits. S-parameters represent how signal from one port is scattered to other ports through a network. For PCB designs, the network consists of a set of traces and ports are both ends of each trace. For example, S-parameters of a PCB design with two traces (4 ports) are shown below.

$$S = \begin{pmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{pmatrix} \quad (1)$$

S_{ij} in eq. (1) refers to the amount of signal transmitted from jth port to ith port. Each S-parameter can be represented in the form of either a complex number or a magnitude-phase pair. We can categorize S-parameters: transmission, reflection, near-end crosstalk, and far-end crosstalk as shown in Figure 2. It is desirable to maximize transmission and minimize the others. Note that the S-parameter matrix is symmetry with the same termination on both sides, e.g., 50Ω termination. Therefore, we need to find out only a lower triangle part of the matrix (10 elements).

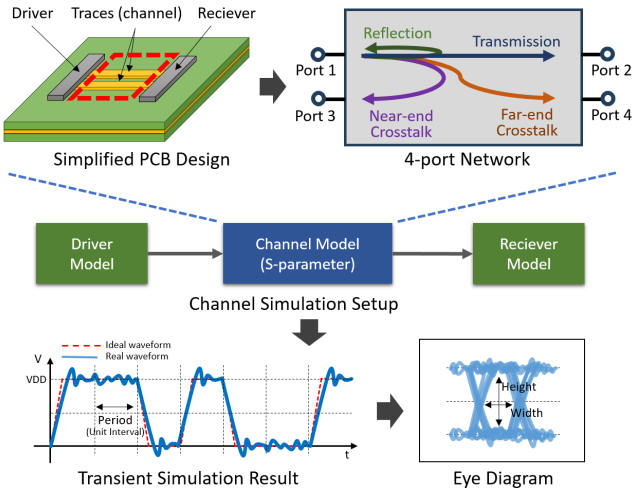


Fig. 2. S-parameter modeling and Eye Diagram.

Each S-parameter can be represented as a function of frequency. The frequency dependency of the traces is attributed to their inductive and capacitive components. It is important to analyze S-parameters over a wide frequency range since S-parameters of PCB traces change significantly in the high frequency range (>1GHz), degrading signal integrity.

2.2 Eye diagram

An eye diagram is an important measure in evaluating signal quality. To obtain the eye diagram, we perform a channel simulation where a driver sends a random digital signal to a receiver through a channel, which is modeled by S-parameters. Then the eye diagram is drawn by overlaying segments of the waveform sampled across numerous periods on top of each other as shown in Figure 2 bottom. The resulting pattern helps to analyze signal quality like noise, interference, and distortion in the signal path. A more open eye vertically and horizontally suggests fewer errors on the receiver side, meaning better signal integrity. Thus, we can indicate SI with an eye diagram.

2.3 Graph Transformer

In this study, we employ the Graph Transformer to predict S-parameters with PCB trace shape data. The original transformer in [7] utilizes an attention mechanism, which determines the amount of influence from one token to the other tokens based on their relationship, for natural language tasks. The transformer architecture has been extended to other tasks with different data formats such as image [8] and graph [9] due to its capability of capturing the relationship between tokens. We refer to the Transformer model for graph data as Graph Transformer (GT). GT can grasp the influence from not only adjacent nodes but also distant

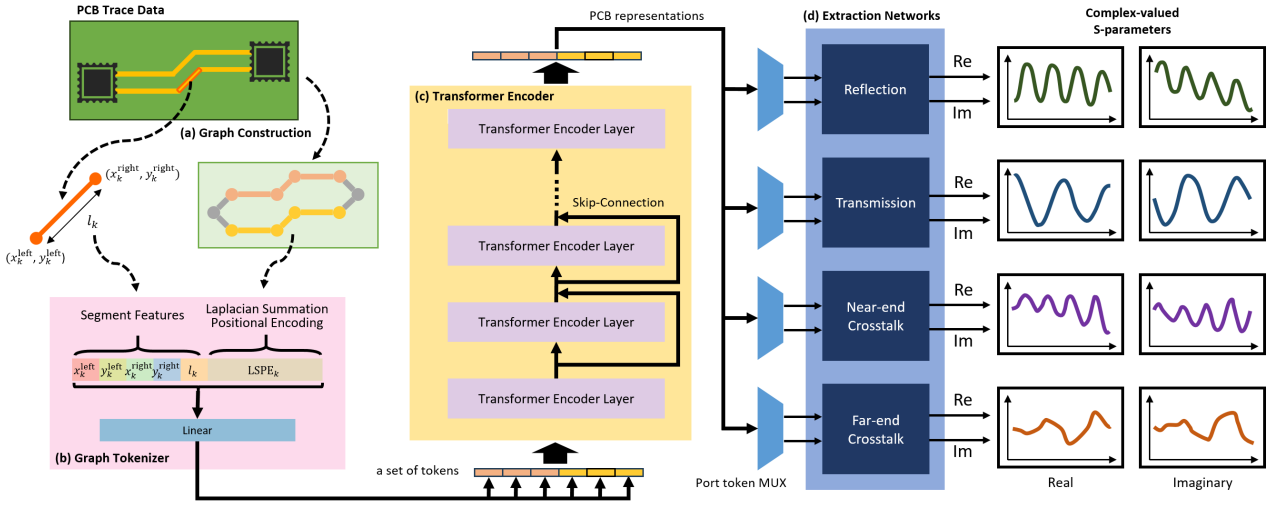


Fig. 3. TraceFormer architecture: (a) PCB traces are segmented to form the edges of the graph, where the geometric features of each segment serve as the segment features. (b) Each segment is tokenized through a linear network with its features and Laplacian Summation Positional Encoding to preserve the segment's geometric and topological information. (c) A set of tokens is fed to the transformer encoder to produce PCB representation. The skip-connections between layers mitigate the issue of the graph bottleneck. (d) Extraction networks (linear) for different S-parameter types use the corresponding port tokens to predict complex-valued S-parameters. These processes are implemented within the multi-task learning framework.

nodes in a graph. From PCB trace data, the GT is able to learn geometric (2D coordinate) and topological (connection) features so that the model understands the shape of traces from segments and coupling among traces. As a result, the GT can generate powerful representations of the PCB design. Based on the representations, TraceFormer can predict S-parameters with additional neural networks.

3 TraceFormer

The aim of TraceFormer is to predict S-parameters based on PCB trace data. To effectively train neural networks that reflect the physical characteristics of S-parameters, it is crucial to represent the PCB data accurately and choose appropriate architectures. In this section, we explain TraceFormer in detail.

3.1 Graph Construction

We build a graph with PCB data to be used for the GT model as illustrated in Figure 3 (a). Banding points along each trace become nodes of a graph and segments that are divided by those points become edges. Each trace is not supposed to be connected to each other so they become separate sub-graphs. To handle all traces as a whole, we interconnect these sub-graphs using two dummy nodes. One dummy node is attached to the end (port) nodes on the driver side whereas another dummy node is attached to the end (port) nodes on the receiver side, constructing a merged graph.

3.2 Graph Tokenizer

A primary challenge in our study lies in effectively converting the graph into tokens. A token is the smallest unit which can convey useful information for transformer architectures. Transformer architectures utilize a set of tokens as their input. However, a set of traces is required as a whole to estimate S-parameters because traces mutually influence themselves. Thus, we need to keep critical information while splitting them into tokens. Therefore, we propose innovative approaches to tokenizer with positional encoding, illustrated in Figure 3 (b).

We first include each segment's position and length as features in the corresponding token. Segments are created by splitting traces during graph construction as explained above. A segment's geometric information helps to estimate its length and relative position to other segments, which contribute to inductive and capacitive components, thereby S-parameters. Therefore, we include the 2D coordinate of each segment's endpoints in each token. In addition, we found that a trace's length significantly affects S-parameters so we also include each segment's length for robust training in a token.

We also introduce a novel positional encoding method designed to encapsulate the topological information inherent in PCB traces. Positional encoding indicates a token's relative position in a set of tokens so transformer models can recognize the relationship between tokens by features as well as positions. Typical GT models often employ Laplacian eigenvectors for positional encoding [9]. However, PCB traces to construct a graph share a similar shape, resulting in multiple path graphs. Therefore, the traditional Laplacian eigenvector positional encoding fails to distinguish between PCB traces and convey meaningful topological content. In response to this challenge, we propose a novel positional encoding method, named Laplacian Summation Positional Encoding (LSPE). During graph construction, where each trace segment corresponds to an edge, LSPE at a token is

defined as the sum of Laplaican eigenvectors at nodes connected to the corresponding edge. Our LSPE enhances the attention mechanism based on the topological aspect of the graph, even though it cannot distinguish the topology of the graph.

After concatenating token features and LSPE, a linear layer converts them into input tokens for the GT. Accordingly, we formulate the k-th token X_k as follows:

$$\begin{aligned} F_k &= [x_k^{\text{left}}, y_k^{\text{left}}, x_k^{\text{right}}, y_k^{\text{right}}, l_k]; \\ X_k &= \text{Linear}\left([F_k \mid \text{LSPE}_k]\right). \end{aligned} \tag{2}$$

Here, $(x_k^{\text{left}}, y_k^{\text{left}})$ and $(x_k^{\text{right}}, y_k^{\text{right}})$ are coordinates of the corresponding segment's endpoints and l_k is the legnth of the segment.

3.3 Transformer Encoder

The transformer encoder aggregates the entire PCB trace data in the form of tokens to generate PCB representations. The transformer encoder is built by multiple transformer encoder layers. Each layer creates a new set of tokens by extracting more abstract features and their positions, which keeps their relational quantities, for the next layer. However, stacking multiple layers causes a potential issue in that the model produces constant representations regardless of the input data. This problem is associated with a challenge called the graph bottleneck commonly faced by graph neural networks since aggregation methods such as summation or mean average out features from the input [10]. To address this issue, we employ the skip-connection technique after each transformer encoder layer (Figure 3 (c)), as expressed by the following equations:

$$H^{(l)} = H^{(l-1)} + \text{Enc}_l\left(H^{(l-1)}\right), \tag{3}$$

where $H^{(l)}$ is the hidden representation after l-th layer and Enc_l is the l-th transformer encoder layer [11]. Here, we note that $H^{(0)} = X$ and $Z = H^{(N_{LAYER})}$.

3.4 Extraction Networks

Adaptive deep learning networks referred to as the *extraction networks*, are employed to perform complex-valued S-parameter prediction tasks. As previously mentioned, the S-parameters are categorized into four types: reflection, transmission, near-end crosstalk, and far-end crosstalk. TraceFormer utilizes four extraction networks, each specialized in generating a specific type of S-parameters as illustrated in Figure 3 (d). Certain tokens in the PCB representation from the transformer encoder are developed to represent each port (port tokens). Formally, for each element S_{ij} of the S-parameter, the sum of port i and port j's tokens is fed into the extraction networks according to the type of S_{ij}, as follows:

$$S_{ij} = \text{Extraction}_{[\text{type}]}\left(Z_{k_i} + Z_{k_j}\right), \tag{4}$$

where the type of S_{ij} is [type]. Here, the indices k_i and k_j are token indices in the PCB representation which corresponding edges are connected with port i and port j, respectively. Each extraction network comprises two linear networks which output dimension is the number of frequency points for S-parameters. The networks separately predict real and imaginary components of S-parameters. We note that our framework can be understood as a form of hard-sharing multi-task learning. In this configuration, the transformer encoder acts as a shared network, and the extraction networks serve as task-specific networks. The training loss designed from this viewpoint is introduced in the following subsection.

3.5 Training Loss

We employ the mean squared error (MSE) as a loss function to train the graph transformer and extraction networks together. The MSE loss is calculated separately for real and imaginary components. The true (target) S-parameters obtained from the simulation tool (SIWave) have an imbalanced distribution with respect to the frequency range: 1601 points and 800 points in the range of 0GHz to 1GHz and 1GHz to 15GHz, respectively. This may cause inaccurate predictions for the high frequency range. To mitigate the imbalance, we assign weights to each frequency point that are inversely proportional to their density. We formulate the loss function for predicting S_{ij} as follows:

$$\begin{aligned} \mathcal{L}_{ij} &= \|S_{ij} - \text{Extraction}_{[\text{type}]}(Z)\|_\gamma^2 \\ &= \sum_{f=1}^{2401} \left\{S_{ij}(\omega_f) - \text{Extraction}_{[\text{type}]}(Z)(\omega_f)\right\}^2 \cdot \gamma_f. \end{aligned} \tag{5}$$

Here, the type of S_{ij} is [type], $\gamma = (\gamma_1, \ldots, \gamma_{2401})$ are weights and $\omega_1, \ldots, \omega_{2401}$ are frequency points.

In addition, we adopt a multi-task learning framework to predict multiple S-parameters simultaneously. Following [12], we construct a multi-task likelihood loss by aggregating weighted MSE losses for each S-parameter component. For each S-parameter type, we assign an observation noise scalar, denoted as $\sigma_{[\text{type}]}$, indicating its relative importance. Formally, the loss function is described as follows:

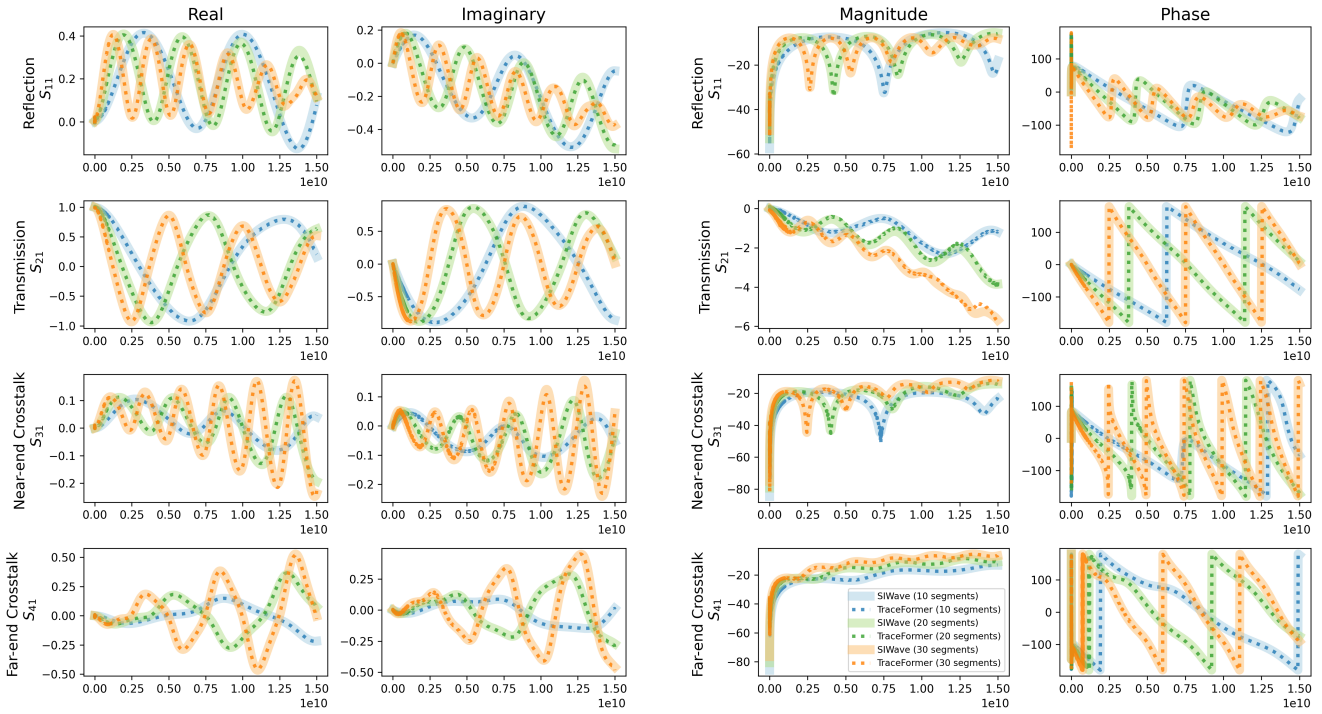
$$\begin{aligned} \mathcal{L} &= -\log p\left(\{S_{ij} : 1 \le i \le j \le 4\} \mid Z\right) \\ &= \sum_{[\text{type}]} \sum_{\substack{i \le j \\ (i,j) \in [\text{type}]}} \frac{1}{2\sigma_{[\text{type}]}^2} \mathcal{L}_{ij} + \log \sigma_{[\text{type}]}, \end{aligned} \tag{6}$$

where $(i, j) \in [\text{type}]$ means the type of S_{ij} is [type]. Here, the summation runs over $i \le j$ since S-parameter elements are symmetric, i.e., $S_{ij} = S_{ji}$.

4 Experiments

4.1 Experiment Setup

We generate PCB datasets for TraceFormer's training, validation, and testing. Each PCB design has two layers and two traces are drawn with uniform spacing (0.2mm) and width (0.055mm) on the top layer (microstrip structure). Each trace



(a) Real / Imaginary **(b)** Magnitude / Phase

Fig. 4. Prediction results of each type of S-parameters of PCB traces with 10, 20, 30 segments.

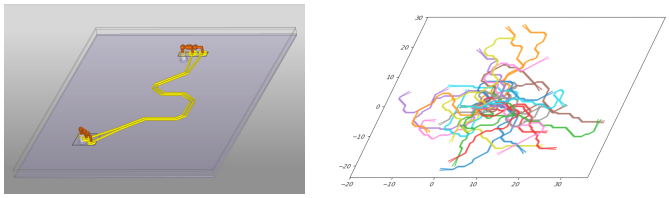


Fig. 5. An examaple PCB (*left*) and Distribution of the 30 PCB traces (*right*) .

in a PCB design consists of multiple segments which length and direction are randomly chosen between 0.3mm and 3mm and among +45°, 0°, -45°, respectively, generating diverse PCB designs. Figure 5 shows an example PCB design and distribution of PCB traces. Our dataset consists of 1000 PCB designs each, with 10, 20, and 30 segments respectively. We run SYZ simulations using Ansys SIWave to obtain S-parameters of each PCB design, which takes about 90 seconds for a 30-segment PCB design. S-parameters are calculated with 1601 points and 801 points in the range of 0Hz to 1GHz and 1GHz to 15GHz, respectively. Note that we sweep the frequency linearly to prevent convergence issues in the channel simulation. Experiments are performed on a Linux host with AMD EPYC 7543 32-Core Processor and one NVIDIA A6000 40GB GPU.

4.2 S-parameter Evaluation

We evaluate the prediction performance of TraceFormer using a separate test set comprising 300 samples that were not included in the training set. For PCB with two traces, there exists 10 distinct complex-valued S-parameters: 4 of reflection, 2 of transmission, 2 of near-end crosstalk, and 2 of far-end crosstalk. By decomposing complex numbers into real and imaginary components, TraceFormer produces 20 real-valued functions defined for the frequency range of interest. Figure 4a demonstrates the real and imaginary components of each S-parameter type computed by SIWave (shaded lines), alongside predicted values by TraceFormer (dotted lines). In addition, we convert the complex numbers into magnitude and phase, as shown in Figure 4b. Note that the magnitude plot's dB scale tends to visually amplify errors at smaller magnitude values in the plot by nature but actual MSE losses for the errors are negligible. However, the small scale may lead to an abrupt change in phase. Therefore, post-processing such as filtering may be required.

Our evaluation metrics indicate excellent prediction performance. In Table 1, we present the R-squared score (R2) and Mean Absolute Error (MAE) for each type of S-parameters. Across all cases, the R2 scores surpass 0.979, with an average of 0.9911. For follow-up studies predicting S-parameters, we leave MAE as benchmark scores.

4.3 Eye Diagram Validation

We validated the predicted S-parameters with channel simulations. To minimize the influence from models of a driver and a receiver, we use an ideal driver and an S-parameter channel model to run channel simulations using Keysight Advanced Design System. The driver sends 1.1V, 6.4GHz random bit signal with 18ps of rising and falling time. Each port of the PCB channel is terminated with 50 ohm. We compared the eye diagrams for PCB designs with 10, 20,

		10 segments		20 segments		30 segments		Average	
		R2 ↑	MAE ↓	R2 ↑	MAE ↓	R2 ↑	MAE ↓	R2 ↑	MAE ↓
Reflection	s_{11}	0.9932	0.0069	0.9912	0.0068	0.9968	0.0048	0.9937	0.0062
	s_{22}	0.9931	0.0067	0.9915	0.0068	0.9967	0.0049	0.9937	0.0061
	s_{33}	0.9914	0.0079	0.9925	0.0063	0.9973	0.0044	0.9937	0.0062
	s_{44}	0.9921	0.0075	0.9925	0.0066	0.9972	0.0045	0.9939	0.0062
Transmission	s_{21}	0.9974	0.0156	0.9957	0.0201	0.9976	0.0157	0.9969	0.0171
	s_{43}	0.9965	0.0180	0.9958	0.0207	0.9978	0.0151	0.9967	0.0179
Near-end Crosstalk	s_{31}	0.9842	0.0032	0.9704	0.0057	0.9845	0.0048	0.9797	0.0046
	s_{42}	0.9838	0.0033	0.9700	0.0057	0.9855	0.0046	0.9798	0.0046
Far-end Crosstalk	s_{41}	0.9908	0.0028	0.9889	0.0052	0.9938	0.0060	0.9912	0.0046
	s_{32}	0.9925	0.0025	0.9885	0.0051	0.9937	0.0061	0.9916	0.0045
Average		0.9915	0.0074	0.9877	0.0089	0.9941	0.0071	0.9911	0.0078

Table 1. Evaluation metrics. R2, R-squared score; MAE, Mean Absolute Error.

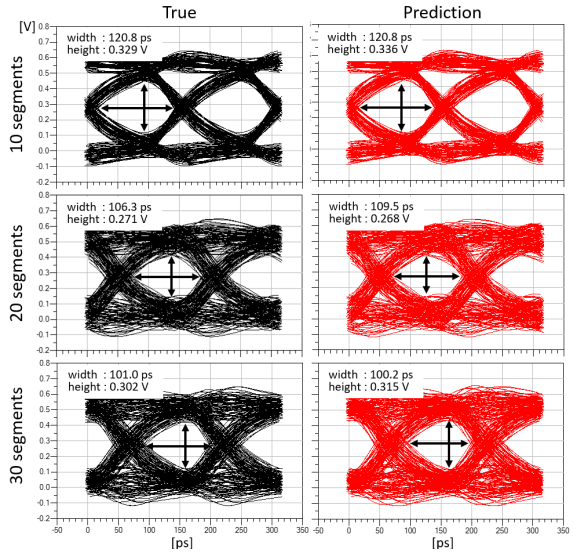


Fig. 6. Eye diagram comparison between true *(left)* and predicted *(right)* S-parameters.

and 30 segments, each having their worst prediction cases ($R2 = 0.9836, 0.9648, 0.9388$). As shown in Figure 6, the errors in the eye diagram's width (timing) and height (voltage) are less than 3.1% and 4.2%, respectively. Although TraceFormer's prediction performance in the low frequency range is sub-optimal, its impact on the eye diagram is minimal since errors in the S-parameter mainly occurred in the frequency range where the signal magnitude is lower than -40 dB. Thus, TraceFormer can speed up SI analysis process without degrading simulation accuracy.

5 Conclusions

In this paper, we present a novel S-parameter prediction framework, TraceFormer. TraceFormer converts PCB traces into a graph and generates tokens based on topological and geometric information, which is used for generating PCB representations. The following S-parameter extraction networks with the representations predict S-parameters according to their type. For PCB designs with two traces, TraceFormer achieved 0.9911 average R-squared score and 0.0078 mean absolute error, resulting in only 3.1% and 4.2% of errors in the eye diagram's width and height, respectively. We can extend this work to more complex PCB designs such as a larger number of traces and multiple layers with vias for potential future works.

Acknowledgments

We would like to thank Taehee Kim and Hyunsuk Jung from Samsung Electronics for insightful discussions and tool support.

References

[1] Jinwook Song et al. "Novel Target-Impedance Extraction Method-Based Optimal PDN Design for High-Performance SSD Using Deep Reinforcement Learning". *IEEE Transactions on Signal and Power Integrity*, 2023.
[2] Shutong Qi et al. "Deep neural networks for rapid simulation of planar microwave circuits based on their layouts". *IEEE Transactions on Microwave Theory and Techniques*, 2022.
[3] Aggraj Gupta et al. "Tandem Neural Network based Design of Multiband Antennas". *IEEE Transactions on Antennas and Propagation*, 2023.
[4] Ren Shibata et al. "A Novel Convolutional-Autoencoder Based Surrogate Model for Fast S-parameter Calculation of Planar BPFs". In *IEEE International Microwave Symposium*, 2022.
[5] Sriram Ravula et al. "One-Dimensional Deep Image Prior for Curve Fitting of S-Parameters from Electromagnetic Solvers". *ICCAD*, 2023.
[6] Daehwan Lho et al. "Channel characteristic-based deep neural network models for accurate eye diagram estimation in high bandwidth memory (HBM) silicon interposer". *IEEE Trans. on Electromagnetic Compatibility*, 2021.
[7] Ashish Vaswani et al. "Attention is all you need". *NeurIPS*, 2017.
[8] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". *ICLR*, 2021.
[9] Jinwoo Kim et al. "Pure transformers are powerful graph learners". *NeurIPS*, 2022.
[10] Uri Alon et al. "On the bottleneck of graph neural networks and its practical implications". *arXiv preprint arXiv:2006.05205*, 2020.
[11] Jiawei Zhang et al. "Gresnet: Graph residual network for reviving deep gnns from suspended animation". *arXiv preprint arXiv:1909.05729*, 2019.
[12] Alex Kendall et al. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In *CVPR*, 2018.