2024 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+SIPI) | 979-8-3503-6039-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/EMCSIPI49824.2024.10705639

# High-speed Channel Simulator using Neural Language Models

Hyunwook Park[#1], Yifan Ding[#2], Ling Zhang[$3], Natalia Bondarenko[*4], Hanqin Ye[*5], Brice Achkir[*6], and Chulsoon Hwang[#7]
[#]*EMC Laboratory, Missouri University of Science and Technology,* Rolla, MO, USA
[1]hpt3h, [7]hwangc@mst.edu
[$]*Zhejiang University*, Hangzhou, China
[*] *Cisco Inc., San Jose,* CA, USA

***Abstract*—In this paper, high-speed channel simulators using neural language models are proposed. Given the input sequence of geometry design parameters of differential channels, the proposed channel simulator predicts SI characteristic sequences such as insertion loss (IL) and far-end crosstalk (FEXT). Sequence-to-sequence (seq2seq) networks using a recurrent neural network (RNN) and a long short-term memory (LSTM) are utilized for the estimator. Moreover, a transformer network which is a recent neural engine of large language models (LLMs) is introduced for the first time. Compared to seq2seq networks, the transformer network-based simulator can achieve shorter computing time due to its parallel computation called an attention. The accuracy and training time of seq2seq and transformer networks are validated and compared. As a result, all the proposed simulators show ~1% error rates for both the IL and FEXT. However, for the training time, the transformer network achieves 75%–83% reduction compared to seq2seq networks.**

***Keywords— High-speed channel, Neural language model, Signal integrity, Transformer network***

## I. Introduction

Neural network (NN) models have been actively developed for signal integrity (SI), power integrity (PI), and electromagnetic compatibility (EMC) applications [1]–[3]. As shown in Fig. 1, deep NNs (DNNs) can characterize complex and non-linear relationships between inputs and outputs [1], [4]. In addition, DNN models are faster than the conventional electromagnetic (EM) and SPICE simulators and comparable in the accuracy [1], [5]. Therefore, those have kept proving their feasibility in practical usage.

Especially, various DNNs have been studied for the high-speed channel SI simulation [1], [5]–[9]. Basic multi-layer perceptron (MLP) models are developed to estimate eye diagram considering all the interconnect, transmitter, and receiver [1]. Also, MLP-based time domain reflectometry (TDR) impedance estimators for high-speed vias are investigated [5], [6]. To improve the accuracy compared to the MLP, advanced convolutional neural network (CNN) models are utilized for S-parameter prediction of high-speed interconnects [7], [8]. Torun *et al.* [8] validates that the proposed CNN model outperforms the MLP model. Graph neural network (GNN) and RNN are combined to estimate time-domain output waveforms of high-speed serial links [9]. By introducing GNN, Li *et al.* [9] enables a flexible and reusable solver that can used for different topologies and physical parameters.

In this paper, high-speed channel simulators using neural language models are proposed and compared for the first time. Given the input geometry design parameters of differential channels, the proposed simulator predicts SI characteristics. The seq2seq networks and transformer network are developed to estimate IL and crosstalk responses. Moreover, the developed neural language models including RNN-RNN, LSTM-LSTM, and transformer network are compared and analyzed in terms of the accuracy and the training time.

This work was supported in part by the National Science Foundation (NSF) under Grant IIP-1916535.

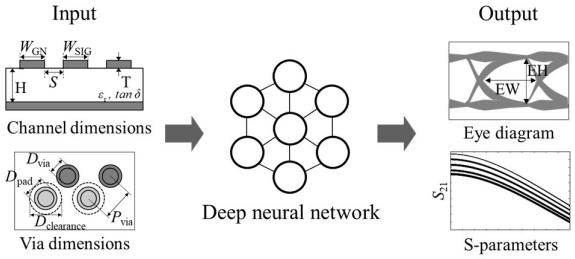


Fig. 1. Deep neural network model for fast and accurate SI simulation.

## II. Proposed Channel SI Simulator using Neural Language Models

When the NN models are developed for the prediction, the encoder-decoder structure is widely used as shown in Fig. 2 [10]–[13]. The encoder performs a feature embedding for given the input data $\boldsymbol{x}_e$ to output the embedded feature $h$: $h=f_{enc}(\boldsymbol{x}_e)$. Then, the decoder conducts the prediction to output $\boldsymbol{y}$ for given inputs as $h$ and the decoder's input $\boldsymbol{x}_d$: $\boldsymbol{y}=f_{\text{dec}}(\boldsymbol{x}_d, h)$. Therefore, depending on the task and the representations of the input and output, the proper NNs should be used for the encoder and decoder.

For the neural language models, the task is to transform from sequence $\boldsymbol{x}_e$ to sequence $\boldsymbol{y}$, where usually $\boldsymbol{x}_e$ and $\boldsymbol{y}$ are the sequences of vectors [11]. Those previously adopted seq2seq networks using a RNN, LSTM, or gated recurrent unit (GRU) [11]. Recently, a transformer network has been adopted which is a main engine for recent large language models (LLMs) [12]. In this section, high-speed channel SI simulators using seq2seq networks and transformer network are proposed to estimate IL or crosstalk.

### A. Seq2seq networks

Fig. 3 (a) shows the proposed channel SI simulator using seq2seq network. It is configured with the encoder and the decoder, which consist of a series of encoding and decoding units. Either RNN or LSTM cell is used for the encoding and decoding units in this work. The task of the proposed channel SI simulator is to transform channel geometry and frequency sequences ($\boldsymbol{x}_e$, $\boldsymbol{x}_d$) to SI characteristic sequences ($\boldsymbol{y}_{\text{IL}}$, $\boldsymbol{y}_{\text{FEXT}}$) at $n$ number of frequency points. The encoder input $\boldsymbol{x}_e$ is represented as a set of vectors that are configured with 14 design parameters of each differential channel as shown in

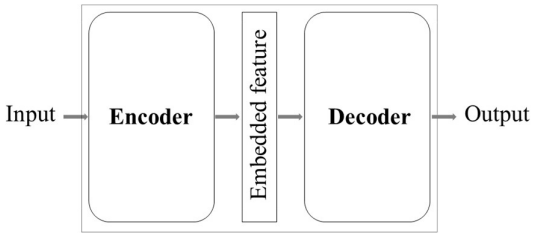


Fig. 2. Encoder-decoder NN architecture for the prediction.

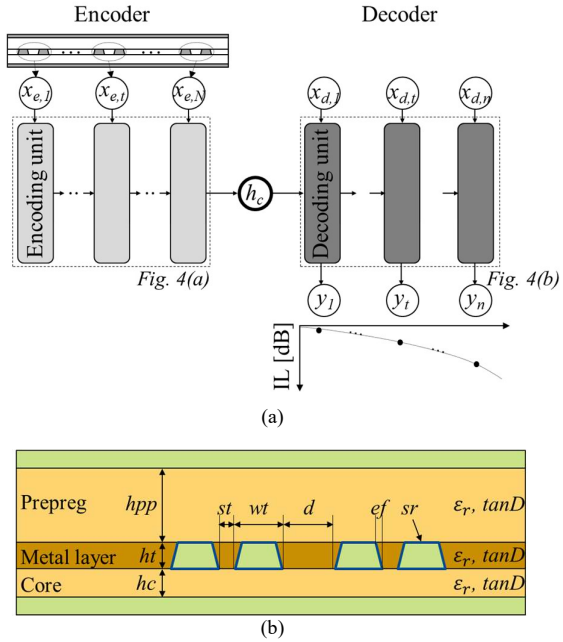


Fig. 3. (a) Proposed channel SI simulator using seq2seq network. (b) Target stack-up of 2 differential strip-line channels.

Fig. 3 (b): $\boldsymbol{x}_e=\{x_{e,1}, x_{e,2}, \ldots, x_{e,N}\}$ where $x_{e,t}=\{x_{1:14}\}$ and $N$ is the number of differential channels. Details on the design parameters $x_{1:14}$ are summarized in Table I. For output sequences $\boldsymbol{y}_{\text{IL}}$ and $\boldsymbol{y}_{\text{FEXT}}$, the decoder input $\boldsymbol{x}_d$ is a set of vectors consisting of a frequency point $f_t$ and the previous estimated value $y_{t-1}$: $x_{d,t}=\{f_t, y_{t-1}\}$.

The main purpose of the encoder is the channel geometry embedding to compute relationships between differential channels. Details on the encoder using RNN is shown in Fig. 4(a). For every time step $t$, a hidden state $h_{e,t}$ is recursively computed as a linear combination of $x_{e,t}$ and the previous hidden state $h_{e,t-1}$:

$$h_{e,t} = tanh(W^e_{xh}x_{e,t} + b^e_x + W^e_{hh}h_{e,t-1} + b^e_h). \quad (1)$$

$W^e_{xh} \in R^{d_h \times d_x}$, $W^e_{xh} \in R^{d_h \times d_x}$, $b^e_x \in R^{d_h}$, and $b^e_h \in R^{d_h}$ are the learnable parameters. $tanh$ is a hyperbolic tangent function as a non-linear activation function, which is typically used for the RNN [4]. Through the recursive computation of $h_{e,t}$, all the geometry information of $x_{e1:eN}$ is embedded in the final hidden state $h_{e,N}$ at time step $N$. $h_{e,N}$ becomes the encoder's output embedded feature node $h_c$, which is also called a context node. Then, $h_c$ is given to the decoder as an important clue for the prediction.

TABLE I. DESIGN PARAMETERS OF DIFFERENTIAL CHANNELS

| Design parameter | Description | Range | |
|---|---|---|---|
| | | Min | Max |
| $wt$ ($x_1$) | Width of trace | 2 mil | 16 mil |
| $st$ ($x_2$) | Space of trace | 2 mil | 16 mil |
| $hc$ ($x_3$) | Height of core | 2 mil | 10 mil |
| $\varepsilon_{r,core}$ ($x_4$) | Dielectric constant of core | 2.5 | 4.5 |
| $tanD_{\text{core}}$ ($x_5$) | Loss tangent of core | 0.001 | 0.03 |
| $ht$ ($x_6$) | Height of trace | 0.4 mil | 2.6 mil |
| $\varepsilon_{r,m}$ ($x_7$) | Dielectric constant of metal layer dielectric fill | 2.5 | 4.5 |
| $tanD_m$ ($x_8$) | Loss tangent of metal layer dielectric fill | 0.001 | 0.03 |
| $hpp$ ($x_9$) | Height of prepreg | 2 mil | 30 mil |
| $\varepsilon_{r,pp}$ ($x_{10}$) | Dielectric constant of prepreg | 2.5 | 4.5 |
| $\text{tanD}_{pp}$ ($x_{11}$) | Loss tangent of prepreg | 0.001 | 0.03 |
| $sr$ ($x_{12}$) | Surface roughness | -10 | 10 |
| $d$ ($x_{13}$) | Pair 2 pair distance | 6 mil | 40 mil |
| $ef$ ($x_{14}$) | Etch factor | 0 | 1 |

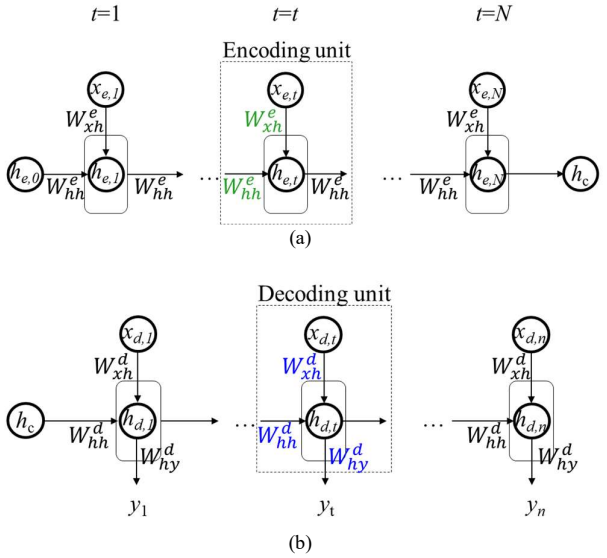


Fig. 4. (a) Encoder RNN. (b) Decoder RNN.

In the decoder, SI characteristic $\boldsymbol{y}_{\text{IL}}$ or $\boldsymbol{y}_{\text{FEXT}}$ is sequentially predicted by decoding units. Details on the decoder using RNN is shown in Fig. 4(b). Using $h_c$ as a first hidden state for the decoder $h_{d,0}$, decoder hidden states $h_{d,t}$ are computed similarly to the encoder, but with the learnable parameters $W^d_{xh}$, $W^d_{hh}$, $b^d_x$, and $b^d_h$. The decoder has an additional shared parameter $W^d_{hy} \in R^{d_y \times d_h}$ and $b^d_y \in R^y$ to convert from $h_{d,t}$ to $y_t$ at $f_t$:

$$y_t = W^d_{hy}h_{d,t} + b^d_y. \quad (2)$$

The main limitation of the RNN is a long-term dependency problem. RNN uses $tanh$ function as the activation function which always maps to the value between 0 and 1 [2].

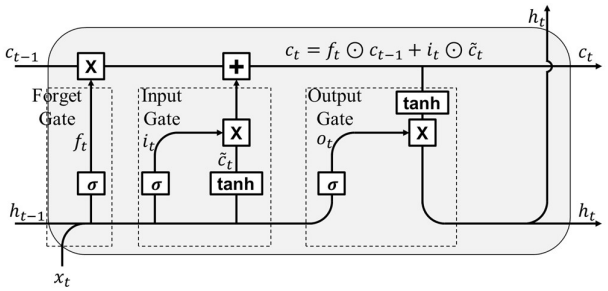


Fig. 5. LSTM cell.

Therefore, the previous hidden states are not well preserved along the time steps since the $tanh$ is being taken recursively. Likewise, the geometry-embedded feature node $h_c$ also easily vanishes along the time-steps in the decoder. Eventually, this problem causes the accuracy degradation of the NN estimator.

To mitigate this problem, the LSTM cell can be used instead of the RNN cell. As shown in Fig. 5, the LSTM introduces a cell state $c_t$ that is responsible for long-term memory and three gates including the forget gate $f_t$, input gate $i_t$, and output gate $o_t$ that control the degree of the memory, input, and output respectively. $c_t$ can propagate down the LSTM cell chain, with only some minor linear interactions by the forget gate and input gate, which enables long-term dependency [4], [11].

The number of layers of the RNN shown in Fig. 4 is 1. Both the RNN and LSTM cells can be vertically stacked up to make a deeper model to further describe complex and non-linear relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$ [4].

*B. Transformer network*

Fig. 6 shows the proposed channel SI simulator using transformer network. The main roles for both the encoder and decoder transformer networks are the same as those of the seq2seq network. The encoder performs geometry embedding and the decoder estimates $\boldsymbol{y}_{\text{IL}}$ or $\boldsymbol{y}_{\text{FEXT}}$. However, the main difference compared to the seq2seq network is that the parallel computation is enabled to reduce the computing time.

The main parallel computation of the proposed transformer network is called an attention. By a self-attention, the encoder embeds the input $\boldsymbol{x}_e$ into high-dimensional embedded node $\boldsymbol{h}$. By an encoder-decoder attention, the decoder estimates the sequence $\boldsymbol{y}_{\text{IL}}$ or $\boldsymbol{y}_{\text{FEXT}}$ using $h_c$ from the encoder and $\boldsymbol{x}_d$ from the decoder. Since the output sequences are computed in a parallel way at once, $x_d$ only contains frequency point $f$. It can't include the previously estimated result $y_{t-1}$, which is also different compared to the seq2seq networks.

The attention computation is all about computing relationship between node as shown in Fig. 7 [13]. The attention function always starts from query $q$, key $k$ and value $v$. $q$ is the object questioning the neighbor nodes $h$'s including itself. $k$ and $v$ are the descriptions or characteristics of each node [12]. Every node $h_i$ can have query $q$, key $k$ and value $v$, which are linearly transformed by learnable parameters $W_{\text{q}}$, $W_{\text{k}}$, and $W_{\text{v}}$:

$$q_i = W_q h_i\,, \qquad k_i = W_k h_i, \qquad v_i = W_v h_i. \tag{4}$$

$i$ is a node index. The linear weights $W_{\text{q}} \in R^{d_k \times d_h}$, $W_{\text{k}} \in R^{d_k \times d_h}$, and $W_{\text{v}} \in R^{d_v \times d_h}$ are shared between all the nodes $h_i$'s. $d_h$, $d_q$, $d_k$, and $d_v$ are the dimension of the $h$, $k$, $q$, and $v$ respectively. $d_k$ is always same as $d_q$. Then, each node $h_i$ can compute its compatibility $u$ by scaled-dot product of its $q$ and $k$ from neighbor nodes including itself:

$$u_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}}. \tag{5}$$

$u_{ij}$ indicates the relevance between two nodes $h_{\text{i}}$ and $h_{\text{j}}$. To normalize, a *softmax* function is taken to $u$ to get an attention weight $a$:

$$a_{ij} = softmax(u_{ij}) = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}}. \tag{6}$$

Finally, output head $h_i'$ can be embedded in the weighted average of $v$'s by the attention weights $a_{ij}$'s:

$$h_i' = \sum_j a_{ij} v_j. \tag{7}$$

Therefore, relationships between node $h_i$ and all the neighbor node s $h_j$'s, where $j \in \{1{:}N\}$ and $N$ is the number of nodes, can be captured in $h_i'$. Since it outputs a single output head, it is also called as single-head attention (SHA). A multi-head

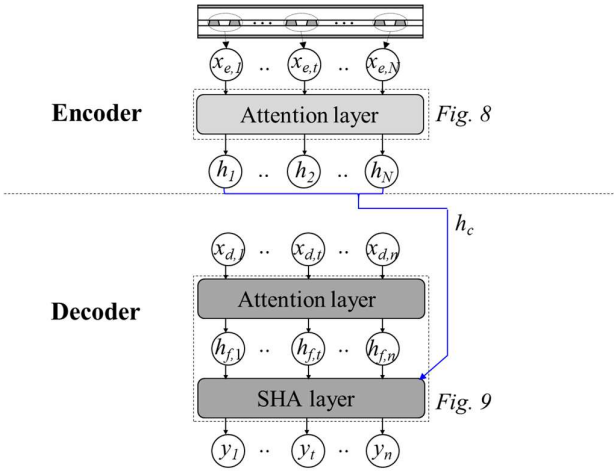


Fig. 6. Proposed channel SI simulator using transformer network.

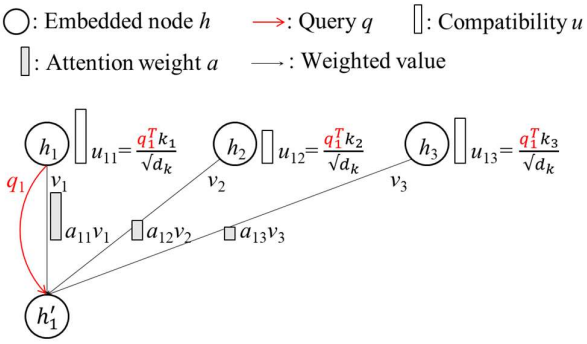


Fig. 7. Single-head attention (SHA) [13].

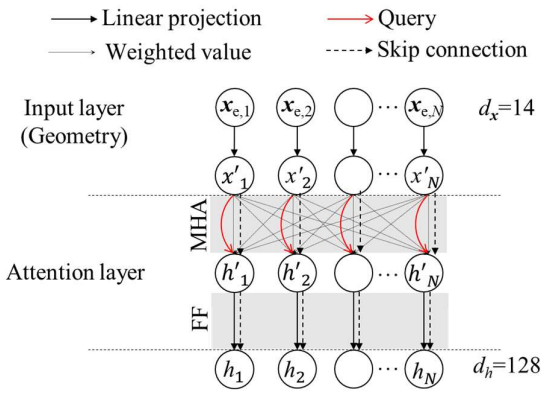


Fig. 8. Encoder transformer network.

attention (MHA) is a parallel computation of SHA in order to learn diverse representation [12], [13].

Fig. 8 shows details of the proposed encoder transformer network. Initial node embed $x_i'$ is embedded by the linear projection of input node $x_{e,i}$:

$$x_i' = W_x x_{e,i} + b_x. \tag{8}$$

where $i \in \{1{:}N\}$ indicates the node index. $W_x \in \mathrm{R}^{d_h \times d_x}$ and $b_x \in \mathrm{R}^{d_h}$ are learnable parameters. Then, $x_i'$ is updated to $h_i$ by an attention layer. The attention layer consists of one MHA sublayer and one feed-forward (FF) sublayer with skip connections:

$$h_i' = x_i' + MHA(x_1', x_2', \ldots, x_N') \tag{9}$$

$$h_i = h_i' + FF(h_i'). \tag{10}$$

$h_i'$ is the output of the MHA sublayer using learnable parameters $W_q$, $W_k$, $W_v$, and $W_o$. $h_i$ is the output of the FF sublayer. The skip connection $x+f(x)$ prevents the gradient vanishing [13]. Especially in the MHA sublayer, all the nodes have their own queries which are denoted in the red lines in Fig. 8. This means each geometry node compute its relationship to all the nodes through the self-attention.

Fig. 9 shows details of the proposed decoder transformer network for $\boldsymbol{y}_{\mathrm{IL}}$ or $\boldsymbol{y}_{\mathrm{FEXT}}$ estimation. The decoder is configured of an attention layer and a SHA layer. First, input frequency $f$'s are embedded into high-dimensional $h_f$ by initial linear projection and the attention layer. Those computations are the same with (8)–(10) of the encoder.

Since the $h_c$ becomes a key factor for the estimation, it is defined as follows:

$$h_c = Concat\left(h_{vic}, h_{aggr}\right). \tag{11}$$

$h_{vic}$ and $h_{aggr}$ are the victim and aggressor channels' geometry-embedded nodes from the encoder. For the IL, $h_{aggr}$ and $h_{vic}$ are set to be the same. $Concat$ is the concatenation operation. With the defined $h_c$, the compatibility $u_i$ is computed by the SHA between $q_c$ and $k_i$ where $q_c$ is projected from $h_c$, and $k_i$ is from $h_{f,i}$:

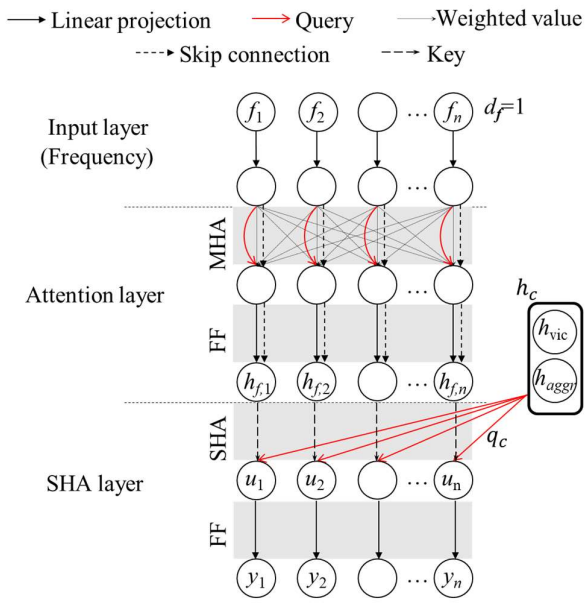


Fig. 9. Decoder transformer network.

$$u_i = \frac{q_c^T k_i}{\sqrt{d_k}}. \tag{12}$$

Finally, in the FF layer, $\boldsymbol{u}$ is linearly projected to output final sequence $\boldsymbol{y}$.

## III. Verification

The hyper-parameters of seq2seq networks are set as follows. Dimension of hidden states for both the RNN and LSTM is set to 128. Dimension of the cell state of the LSTM is also 128. The number of layers of RNN and LSTM are set to 2 for both the encoder and decoder. For the transformer network, the dimensions of the embed (hidden) and feedforward layers in the attention layers are set to 128, 512 respectively. The number of heads for the MHA is 8. The dimensions of the key and value ($d_k$, $d_v$) are set to 16 and 16 for the MHA layer respectively. The key dimension is 128 for the decoder SHA layer.

The data sets for the validation are generated by Intel Interconnect Modeler Loss Calculator (IMLC) tool. Total 99991 of two pair of differential channels in Fig. 3(b) are simulated at 40 frequency points from 1 GHz to 40 GHz, with 1 GHz linear spacing. The generated data sets are split into 84992, 7499, and 7500 for the training, validation, and test sets respectively. The total number of 500 epochs are trained. The size of one epoch is 84992. The batch size is 256 and the validation size is 7499. The initial learning rate is 5×10-4 and is linearly decayed by 0.99 times every epoch. The root mean squared error (RMSE) in dB scale is used for the loss function. All the inputs and outputs $\boldsymbol{x}_e$, $\boldsymbol{x}_d$, and $\boldsymbol{y}$ are normalized by using min-max normalization.

Fig. 10 shows the loss convergence results when training the NNs for IL estimation. All the RNN-RNN, LSTM-LSTM, and transformer network converge both in the training and validation loss at the end of the training epoch. The training losses at the 500th training epoch are 0.0041, 0.0022, and 0.0009 for RNN, LSTM, and transformer respectively. The

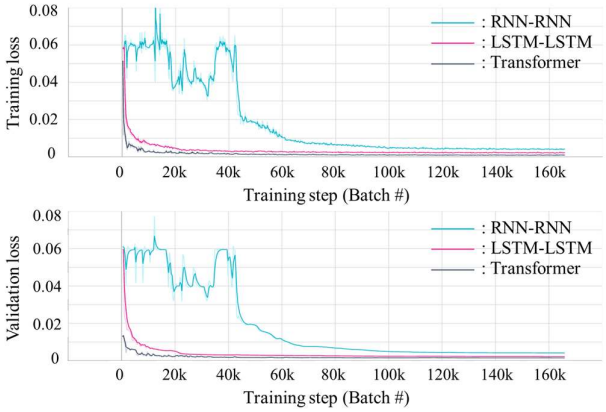


Fig. 10. Training and validation loss convergence results for IL estimator.

TABLE II. ACCURACY TEST RESULTS

| | Error rate | | |
|---|---|---|---|
| | **IL** | **FEXT** | **Average** |
| **RNN-RNN** | 1.7% | 0.8% | 1.25% |
| **LSTM-LSTM** | 0.7% | 0.44% | 0.57% |
| **Transformer** | 0.48% | 0.86% | 0.67% |

validation losses are 0.0041, 0.0022, and 0.0014 respectively. All the training and validation losses when training the NNs for FEXT estimation are also converged well.

Table II shows the comparison of accuracy test results. For the IL estimation, the error rates of the RNN, LSTM, and transformer are 1.7 %, 0.7 %, and 0.48 % respectively. For the FEXT, 0.8 %, 0.44 %, and 0.86 % respectively. All the seq2seq and transformer networks show good correlations with the labels, achieving ~1 % of the average error rate: 1.25 % for the RNN, 0.57 % for the LSTM, and 0.67 % for the transformer network. Compared to the RNN, the LSTM shows better accuracy thanks to its long-term dependency. Since the transformer network compute attention in a parallel way both in the encoder and decoder, there is less memory distortion compared to the RNN. Therefore, it shows a lower error rate than the RNN. However, compared to the LSTM, it shows a slightly larger error rate because it can't take the previous estimated value $y_{t-1}$ as a decoder input. Detailed comparison plots of 10 random test sets between the prediction by NNs and labels are depicted in Fig. 11. Fig. 11(a), 11(b), and 11(c) illustrate the RNN-RNN, LSTM-LSTM, and transformer network respectively.

Table III shows the comparison of the training time between the proposed NNs. The training times are 2h 13m, 3h 20m, and 33m for RNN, LSTM, and transformer network respectively. Transformer network conducts parallel computations for the geometry embedding in the encoder and for the IL or FEXT estimation in the decoder. However, sequential computations should be performed for both the RNN and LSTM. Therefore, the training time of the transformer network can be reduced by 75% compared to RNN; 83% compared to LSTM. Compared to RNN, LSTM shows the longer training time since it has more learnable parameters to enhance long term dependency.

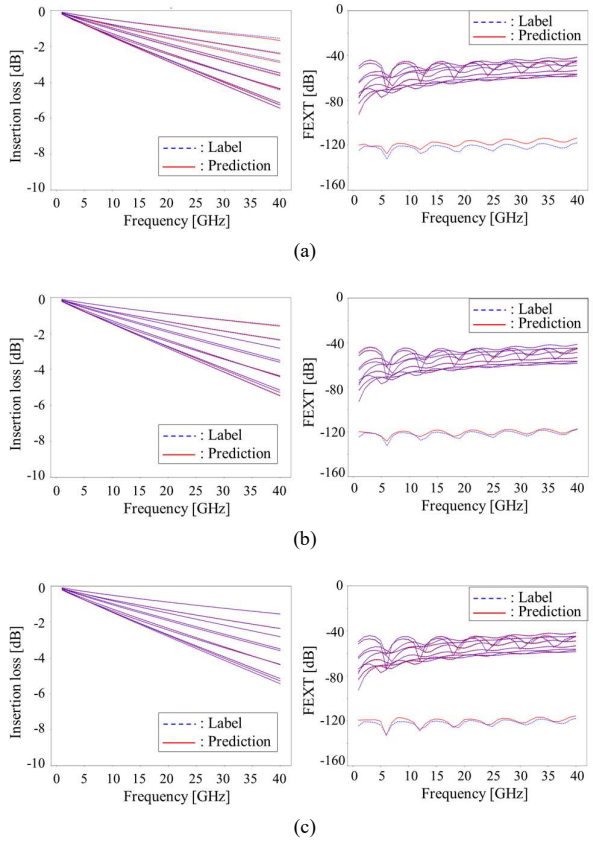


Fig. 11. Comparison of IL and FEXT estimation results with labels. (a) RNN-RNN. (b) LSTM-LSTM. (c) Transformer.

TABLE III. COMPARISON OF THE TRAINING TIME

| | **RNN-RNN** | **LSTM-LSTM** | **Transformer** |
|---|---|---|---|
| **Training time (500 epochs)** | 2h 13m | 3h 20m | 33m |

*CPU: Intel 12th Gen i5, GPU: NVIDIA RTX 3060

## IV. CONCLUSION

In this paper, neural language model-based high-speed channel SI simulators are proposed, validated, and compared. The seq2seq networks using RNN and LSTM, and the transformer network are investigated to predict IL and FEXT responses. As a result, all the proposed networks achieve ~1 % error rates. Especially, thanks to its powerful parallel attention computation, the proposed transformer network reduces the training time more than by 75% compared to the seq2seq networks.

## REFERENCES

[1] T. Lu, J. Sun, K. Wu and Z. Yang, "High-Speed Channel Modeling With Machine Learning Methods for Signal Integrity Analysis," *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 6, pp. 1957-1964, Dec. 2018.

[2] H. Manoharan *et al.*, "Augmented Genetic Algorithm v2 with Reinforcement Learning for PDN Decap Optimization," *2023 IEEE Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMC+SIPI)*, Grand Rapids, MI, USA, 2023, pp. 255-258.

[3] L. Zhang, W. Huang, Z. Sun, N. Erickson, R. From and J. Fan, "Deep Learning Based Poisson Solver in Particle Simulation of PN Junction with Transient ESD Excitation," *2020 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*, Reno, NV, USA, 2020, pp. 241-244.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[5] H. Park *et al*., "Fast and Accurate Deep Neural Network (DNN) Model Extension Method for Signal Integrity (SI) Applications," *2019 Electrical Design of Advanced Packaging and Systems (EDAPS)*, Kaohsiung, Taiwan, 2019, pp. 1-3

[6] J. Xu, L. Zhang, M. Sapozhnikov and J. Fan, "Application of Deep Learning for High-speed Differential Via TDR Impedance Fast Prediction," *2018 IEEE Symposium on Electromagnetic Compatibility, Signal Integrity and Power Integrity (EMC, SI & PI)*, Long Beach, CA, 2018, pp. 645-649.

[7] Y. Guo, X. Li and M. Swaminathan, "2D Spectral Transposed Convolutional Neural Network for S-Parameter Predictions," *2022 IEEE 31st Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, San Jose, CA, USA, 2022, pp. 1-3.

[8] H. M. Torun, A. C. Durgun, K. Aygün and M. Swaminathan, "Causal and Passive Parameterization of S-Parameters Using Neural Networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 10, pp. 4290-4304, Oct. 2020.

[9] Z. Li, X. -C. Li, Z. -M. Wu, Y. Zhu and J. -F. Mao, "Surrogate Modeling of High-Speed Links Based on GNN and RNN for Signal Integrity Applications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 71, no. 9, pp. 3784-3796, Sept. 2023.

[10] H. Park *et al*., "Transformer Network-Based Reinforcement Learning Method for Power Distribution Network (PDN) Optimization of High Bandwidth Memory (HBM)," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 11, pp. 4772-4786, Nov. 2022.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Advances Neural Information Processing Systems*, 2014, pp. 3104–3112.

[12] A. Vaswani et al., "Attention is all you need," in *Proc. Advances Neural Information Processing Systems*, 2017, pp. 5998–6008.

[13] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–25.