Batch Training of Gaussian Process for Up-sampling Problems in S-Parameter Predictions

Yiliang Guo*, Xingchen Li*, Yifan Wang*, Rahul Kumar†, Madhavan Swaminathan†
*Georgia Institute of Technology, Atlanta, GA 30332, USA, {yguo379, xingchen.li, ywang3184}@gatech.edu
† Pennsylvania State University, State College, PA 16801, USA, {fkumar7, mvs7249}@psu.edu

***Abstract*—In using Machine Learning (ML) methods to predict S-parameters, handling the dimensionality problem of mapping the low-dimension design parameters to high-dimension responses is important. We propose to use batch training of Gaussian Process (GP) to map the design parameters into latent Gaussian space instead of linear mappings to create the non-linearity property as well as avoiding the saturation of activation functions before applying transposed kernels. Results show that the proposed model achieves better performance with regard to loss and normalized mean-squared error.**

***Index Terms*—microvia interconnection, convolutional neural network, Gaussian process**

I. Introduction

Applying Machine Learning method to predict the target S-parameters has gained more and more attention, it not only achieves high accuracy but also reduces the computational resources as well as the time consumption. Previously, Spectral Transposed Convolution Neural Network (S-TCNN) [1] with 1D kernel and 2D kernel [2] have been proposed to solve the unbalanced dimensionality problem between low-dimension design parameters and high-dimension S-parameter matrix. It utilizes linear layers to map the design parameters into linear latent space for data preparation as the first step, after that the 1D or 2D transposed convolutional kernels are used to up-sample the data into desired dimension. However, during this training process two issues would happen, namely the 1)linearity problem and 2) saturation of activation functions. Linear layers apply linear transformations to the design parameters such that the output of linear layers could be reshaped as the input of transposed kernels. Due to the linear properties of linear layers, all the nonlinear patterns are assigned to the transposed kernel layers, which limits the performance of up-sampling. In transposed convolutional layers, non-linear activation functions faces the saturation problem, which is illustrated in Fig. 1. The inputs of x_1 and x_2 are distinguishable, however the difference between outputs of the activation function become negligible. It also results in the gradient vanishing problem, which increases the training iterations and shrinks the effective region of the neural network.

At its core, a Gaussian process defines a prior distribution over functions, where each function is a mapping from an input space to an output space. It assumes that any finite set of points from this function follows a multivariate Gaussian distribution. In other words, the Gaussian process defines a distribution over functions, rather than a distribution over parameters as in other models. When training a Gaussian process model, the prior distribution over functions is updated based on the observed data to obtain the posterior distribution. This posterior distribution is used to make predictions or infer the underlying function values at new, unseen points.

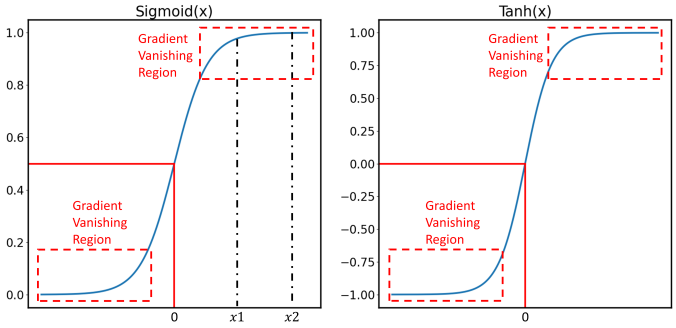

Fig. 1. The Saturation and gradient vanishing problems of Sigmoid and Tanh functions

In this paper, we propose to map each design parameter into its Gaussian latent space instead of using linear mappings to improve the performance of neural network.

II. Technical Approaches

A. Spectral Transposed Convolution Neural Network

In order to map data from lower dimensional feature space to higher dimensional input space, transposed convolution operation could be applied. In Toeplitz form, the output y given input x and kernel function h can be written as:

$$y = f(\boldsymbol{h} *^T \boldsymbol{x}) = f(\boldsymbol{H^T x}) \tag{1}$$

with

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{H} = \begin{bmatrix} w_1\, w_2 \cdots w_k & 0 & \cdots & 0 \\ 0 & w_1\, w_2 \cdots w_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_1\, w_2 \cdots w_k \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{2}$$

where $*^T$ is the transposed convolution operation and $\boldsymbol{y}$ is the upsampled output of size $m = n + k - 1$. In this paper, we select 2D transposed convolutional kernel for the NN model. Mathematically, the 2D kernel is written as:

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j]h[m-i, n-j] \tag{3}$$

where x and y are input and output functions and h is the kernel function. The general structure of S-TCNN with 2D kernel is illustrated in Fig. 2. Fully connected linear layers

979-8-3503-1798-5/23/$31.00 ©2023 IEEE

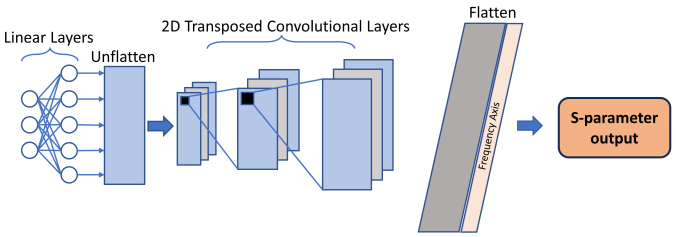

Fig. 2. General S-TCNN Structure

(FCNN) are used to up-sample the input data as the initial step. After that, the outputs from previous layer are unflattened to add extra dimensions in order to fit batches of 2D transposed convolutional filters. The matrix will be flattened and expanded along frequency axis to serve as the output of S-parameter predictions.

B. Gaussian Process

Gaussian Process (GP) is regarded as the extension of standard multivariate Gaussian distribution to infinitely many variables, where any finite number of samples form a joint Gaussian distribution. A standard GP can be written as:

$$y = f(x) \sim \mathcal{N}(\mu(X), K_X) \tag{4}$$

In most cases, a constant mean is used as $\mu(x) = m$. The kernel function $K(\boldsymbol{x})$ that describes the relation between points in the function is written as:

$$K(\boldsymbol{x}) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_t) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_t, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_t, \boldsymbol{x}_t) \end{bmatrix} \tag{5}$$

Kernel functions are designed to capture specific patterns, which is the key of GP. The details of various kernels are described in [3]. The hyperparameters in kernel functions are updated during the training process by minimizing the negative log marginal likelihood of the GP. Once the GP model is trained using the dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, it can be used to predict the unknown response y^* for a new set of input data $x^* \in \mathbb{R}^{M \times D}$ using the relationship:

$$p(y^*, Y | x^*, X, \theta) = \mathcal{N}\left(\begin{bmatrix} \mu X \\ \mu x^* \end{bmatrix}, \begin{bmatrix} K_X & K_{X,x^*} \\ K_{X,x^*}^T & K_{x^*,x^*} \end{bmatrix}\right) \tag{6}$$

where θ is the set of hyperparameters used as part of the training process [4].

III. Model Setup

A. Staggered Via Setup

Consider the staggered via model described in [2], [5]. The via is modeled using Ansys HFSS as shown in Fig.3. The model incorporates an embedded co-planar waveguide (CPW) chip inside a glass cavity and two copper RDL layers (M1 and M2) plated on ABF, a polymer dielectric, laminated above the glass core. To tune the performance of the staggered via structure in terms of s-parameters, there are ten trainable parameters that can be optimized as shown in the second subplot of Fig.3 (b). Ranges of these parameters are listed in

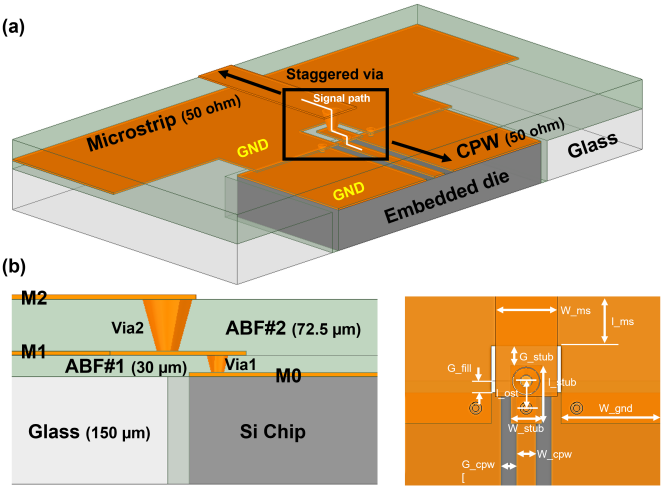

Fig. 3. Modeling and size of a staggered via.(a) Overview of the model. (b) Sizes of the structure.

TABLE I. We generated 600 groups of datasets by sampling each of the parameters independently though Latin hypercube sampling (LHS) method, among which 480 groups are used for training and the rest 120 groups are reserved for testing. The frequency range of s-parameter is DC to 170 GHz with 0.1 GHz steps, which corresponds to 1690 points.

TABLE I
Characterization Parameters (μm)

Parameters	Min	Max	Parameters	Min	Max
G_{stub}	20	60	l_{ost}	30	80
l_{ms}	500	700	W_{cpw}	45	65
W_{gnd}	300	700	G_{cpw}	20	80
W_{stub}	60	120	G_{fill}	30	60
l_{stub}	80	200	W_{ms}	70	200

B. Network Structure

Prior to input the data into neural network, we normalize the training set by subtracting the mean and then divided by the standard deviation. Instead of utilizing sequential fully-connected linear layers, we create N batches of Gaussian process layers where each layer takes the input design parameters and map them into latent Gaussian space, as depicted in Fig. 4. After that, 2D transposed convolutional layers with $tanh$ activation function are applied to map the data from latent Gaussian space to frequency space in order to capture the response patterns. The data is reshaped into the forms of [batches, channels, rows, columns], where channels determined by the number of Gaussian process layers in the precious stage. The data is then flattened and a $CoordConv$ layer [6] is utilized to maintain the frequency axis information.

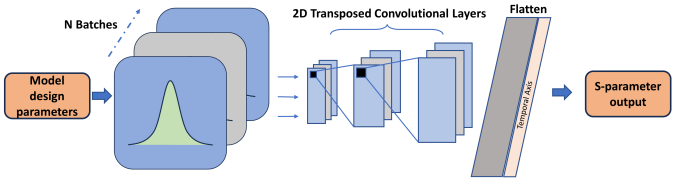

Fig. 4. NN structure

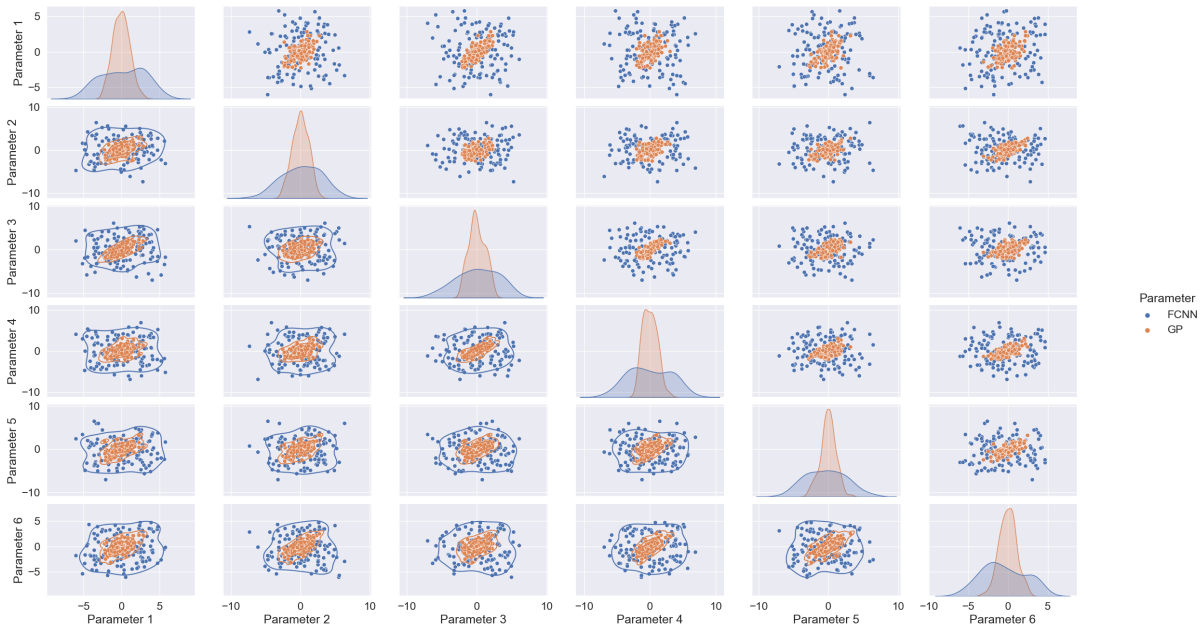

Fig. 5. Distributions of mapping input design parameters to linear space and Gaussian space

The machine learning model is built on PyTorch using CUDA to accelerate the training process. The Gaussian process is built upon Gpytorch.

IV. Results

Once the model is trained, we applied both the proposed S-TCNN with 6 GP layers of RBF kernels and the S-TCNN with 6 traditional FCNN layers to the testing set. Results of transformations of the input design parameters after being centered to $(0, 0)$ are shown in Fig. 5, where Parameter 1-6 stand for the normalized outputs of FCNN or GP layers. As we can see, the outputs of GP layers are more centered to 0 compared with the FCNN layers, which means that the model has a higher probability of working on the effective regions of $sigmoid$ and $tanh$ activation functions. The outputs of FCNN layers, instead, have more dispersive distributions, where a large amount of the points will locate in the saturation region. The final losses for S-TCNN with FCNN layers and the proposed model are 0.311 and 0.276. The validation normalized mean-squared error (NMSE) for the two models are 0.022 and 0.019, respectively. An example of predictions of $S11$ and $S21$ in testing set are shown in Fig. 6, where the predictions of the proposed model are more smooth and achieves better accuracy compared with the original model.

V. Conclusion

In this paper, we propose to utilize GP layers to map the input design parameters of the model to latent Gaussian space to overcome the common gradient vanishing problem in machine learning. Compared with linear transformations, the outputs of GP layers will be more concentrated around the region where the activation functions have larger derivatives. In the staggered via application, the final loss reduces from 0.311 to 0.276 and the NMSE reduces from 0.022 to 0.019 compared with S-TCNN with FCNN layers.

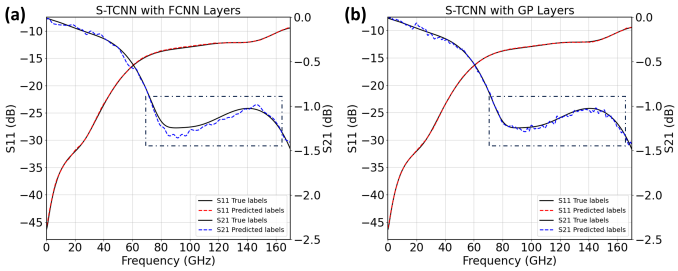

Fig. 6. Prediction results. (a) S-TCNN with linear layers. (b) S-TCNN with GP layers

Acknowledgment

This work was supported by DARPA under the Warden program (Project Number GR00013386).

References

[1] H. M. Torun, H. Yu, N. Dasari, V. C. K. Chekuri, A. Singh, J. Kim, S. K. Lim, S. Mukhopadhyay, and M. Swaminathan, "A spectral convolutional net for co-optimization of integrated voltage regulators and embedded inductors," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2019.

[2] Y. Guo, X. Li, and M. Swaminathan, "2d spectral transposed convolutional neural network for s-parameter predictions," in *2022 IEEE 31st Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pp. 1–3, 2022.

[3] D. Duvenaud, *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[4] M. Swaminathan, O. W. Bhatti, Y. Guo, E. Huang, and O. Akinwande, "Bayesian learning for uncertainty quantification, optimization, and inverse design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 11, pp. 4620–4634, 2022.

[5] X. Jia, X. Li, S. Erdogan, K.-S. Moon, J. W. Kim, K.-Q. Huang, M. B. Jordan, and M. Swaminathan, "Antenna with embedded die in glass interposer for 6g wireless applications," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 13, no. 2, pp. 219–229, 2023.

[6] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.