End-to-end Learning for Distributed Circuit Design

Hao He[*] , **Guo Zhang**[*]
MIT CSAIL
{haohe, guozhang}@mit.edu

Jack Holloway
MIT EECS
holloway@mit.edu

Dina Katabi
MIT CSAIL
dina@csail.mit.edu

Abstract

We present a neural network model for designing distributed circuits. Today, designing such circuits is a slow process that can take months from an expert engineer. Our model both automates and speeds up the process. The model learns to simulate the electromagnetic (EM) properties of distributed circuits. Hence, it can be used to replace traditional EM simulators, which typically take tens of minutes for each design iteration. Further, since neural networks are differentiable, we can use our model to solve the inverse problem – i.e., given desirable EM specifications, we propagate the gradient descent to optimize the template parameters to satisfy the specifications. We compare our model with a commercial simulator showing that it reduces simulation time by five orders of magnitude. We also demonstrate the value of our model by using it to design a Terahertz channelizer, a difficult task that requires a specialized expert. The results show that our model produces a channelizer whose performance is as good as a manually optimized design, and can save the expert several weeks of iterative parameter optimization.

1 Introduction

Distributed circuit design refers to designing circuits at high frequencies, where the wavelength is comparable to or smaller than the circuit components. It is increasingly important since new 5G and 6G communication technologies keep moving to higher and higher frequencies. Unlike lower frequencies, where there are relatively fast tools and a large literature on design guidelines, designing distributed circuits is a slow and onerous process. The process goes as follows. An expert engineer comes up with an initial design based on desired specifications (e.g., design a bandpass filter at 300 GHz, with a bandwidth of 30 GHz). To do so, typically the engineer picks a suitable template and optimizes its parameters. For example, to design a high-frequency filter, the engineer may choose a template based on square-based or ring-based resonators [7, 8]. The engineer then spends extensive effort optimizing the parameters of the template so that the circuit satisfies the desired specifications. The optimization is done iteratively. In every iteration, the engineer sets the parameters in the template to some values, simulates the design, and compares the output of the simulation to the specifications. Each simulation takes tens of minutes during which the simulator runs mainly a brute-force numerical solution of the Maxwell EM equations. The process can take days, weeks, or months, during which the engineer keeps adjusting the parameters, and re-running the simulation.

In this paper, we introduce a learning model that speeds up and automates this process. As customary in circuit design [2, 13, 14, 15, 16], we focus on common templates and design a neural network model that helps the engineer optimize the template parameters. Our model addresses both the forward and inverse design problems. The forward task takes a particular parameter setting and produces the resulting s_{21} function, which relates the signal on the circuit's output port to the signal at its input port. Said differently, the forward task produces the output of the EM simulator but using a neural network. The inverse task on the other hand takes the specifications, i.e., a desired s_{21} function, and produces a parameter setting that obeys the desired specifications. To solve the inverse task, we leverage that neural networks are differentiable. Thus, given a desirable s_{21} and a circuit template, we back-propagate the gradient to optimize the template parameters to satisfy the desired function.

[*]Co-primary Authors.

Present at the workshop on ML for Systems at the 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

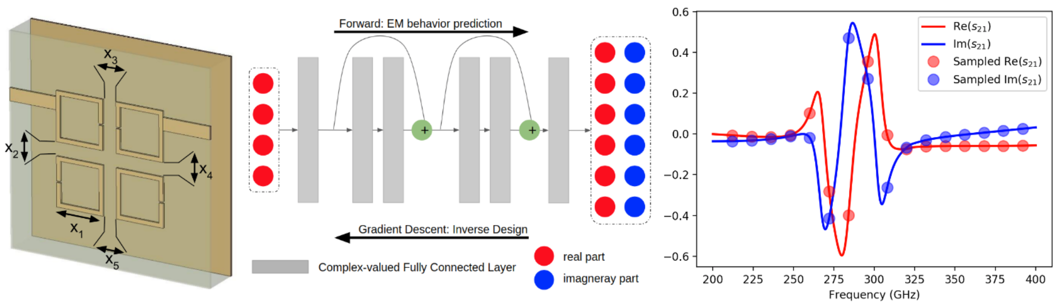


Figure 1: Left: An example of distributed circuit. Middle: our complex-valued neural network model that learns the relationship between circuit geometric parameters and its transfer function; Right: An example circuit transfer function.

While we are not the first to leverage machine learning for circuit design, most prior work focuses on lumped design [2, 13, 15, 16, 17, 19], where the circuit is represented as connections between lumped components: resistance, capacitance, inductance, transistor, etc. Such an approach however does not apply to high-frequency circuits, which require a distributed design.

Prior attempts at learning distributed circuit design have key limitations [1, 4, 5]. Like our model, given a particular template and parameter setting, they aim to predict the circuit transfer function. However, unlike our model which directly predicts the values of the transfer function for different frequencies, they predict a parameterized version of the transfer function. Specifically, they leverage that the transfer function can be approximated as $s_{21}(\omega) = \sum_{i=1}^{N} \frac{a_i}{j\omega - b_i}$, where ω is the frequency and a_i and b_i are complex parameters. Thus, instead of predicting the function $s_{21}(\omega)$, they train a neural network to predict the parameters a_i and b_i. This approach has three limitations. First, given a particular template and parameter setting, it is not clear how many parameters, a_i and b_i, one needs to have for a good representation of the transfer function. Thus, past work, for each template, trains multiple neural networks. Second, during inference, it is not clear how to pick the best network for a particular template. Thus, past work trains an additional model that selects which network to use from the set of networks associated with that template. Third, because the model predicts the parameters in the transfer function as opposed to the function itself, one cannot use the model to solve the inverse problem effectively because the gradient is not smooth, as shown in Sec. 3.5.

The paper makes the following contributions:

- The paper demonstrates the feasibility of designing an end-to-end model for distributed circuit design. Prior models for this problem are significantly more complex and require training multiple neural networks for each template [1, 4, 5]
- The paper shows that one model can solve both the forward and inverse problems. To our knowledge, there is only one past proposal for solving the inverse problem [20]. This solution however works only for templates that have one parameter, and hence is not practical. Alternatively, they can map the specifications to an intermediate analytical representation called the "coupling matrix", but there is no general solution that maps a coupling matrix to an actual circuit design.
- The paper also highlights the benefits of using a complex-valued neural network model for learning complex valued signals and transfer functions.
- Finally, the paper presents an empirical evaluation of the proposed model. The results show that our design can solve the forward task $100,000$ faster than a state-of-the-art commercial simulator while maintaining accuracy. We also show the benefit of solving the inverse problem by using the model to design a Terahertz channelizer that operates at around 300 GHz and has three 30GHz-wide channels. This is a difficult task that requires a Terahertz circuit expert. The results show that our model produces a channelizer whose performance is as good as the channelizer produced by a circuit expert who spent several weeks optimizing the parameters of the template.

2 Learning for Distributed Circuit Design

Fig. 1 shows the pipeline in our model. In the forward direction, the model maps a given circuit to the corresponding transfer function (i.e. s_{21}). In the inverse direction, our model uses gradient descent to optimize the circuit parameters to produce a desired transfer function.

As common in the literature on circuit design (both lumped and distributed) [5, 13, 15, 17], our neural network is trained for a particular template. In the context of distributed circuit design, circuits have a geometric representation as illustrated in the left panel of Fig. 1 (see Fig. 2 for more examples). The template in Fig. 1 has four square resonators. The transfer function of this template depends on its geometric parameters, which describe the length, width, and placement of the mental in each resonator. In this paper, we focus on templates based on square resonators because they are widely used in distributed circuit [7, 8]. Our approach, however, is general and applies to other templates.

2.1 Forward Model

The forward model allows the designer to quickly obtain the transfer function of his/her design. It takes as input a vector $\mathbf{x} \in \mathbb{R}^n$ whose elements specify the values of all geometric parameters in the template. At its output, it generates a complex-valued vector $\mathbf{y}$ that provides a discrete representation of the circuit transfer function, i.e., $\mathbf{y} \triangleq [S_{21}(\omega_1), \cdots, S_{21}(\omega_m)]^T$ where $\{\omega_i\}_{i=1}^m$ indicates the frequency samples from the circuit working frequency $\Omega \triangleq [\omega_{min}, \omega_{max}]$. As Fig. 1 shows, our model captures the relationship between the geometric circuit parameters $\mathbf{x} \in \mathbb{R}^n$ and the (discretized) transfer function $\mathbf{y} \in \mathbb{C}^m$ via a neural network $\mathbf{f}$, i.e. $\mathbf{f}(\mathbf{x}; \theta) \simeq \mathbf{y}(\mathbf{x})$ where θ are the weights.

We use the supervised learning paradigm to train our model. The loss function contains two l_1-norm terms. Each of them penalizes the prediction errors in the real and the imaginary part of the transfer function respectively. Mathematical, the loss function has the following form, $\mathcal{L}(\theta) \triangleq \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \|\Re(\mathbf{f}(\mathbf{x};\theta)) - \Re(\mathbf{y})\|_1 + \|\Im(\mathbf{f}(\mathbf{x};\theta)) - \Im(\mathbf{y})\|_1$ where $\mathcal{D}$ is the dataset containing all samples $(\mathbf{x}, \mathbf{y})$.

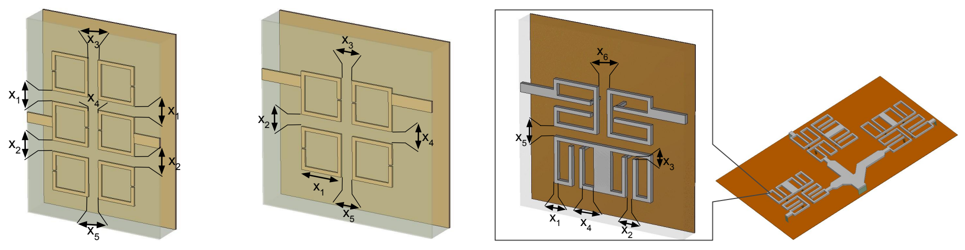
In contrast to past work, our model is based on a complex-valued neural network. This choice is more adequate given that signals and transfer functions have complex values. Such model allows for capturing the interaction between real and imaginary values that occurs as the signal traverses the circuit. However, the choice of the activation function in complex-valued neural networks is more involved. The author of [18] describes three complex versions of ReLU activation, ModReLU, $\mathbb{C}$ReLU and zReLU. In our experiments, we use $\mathbb{C}$ReLU which is also the recommended choice in [18]. As shown in Fig. 1, our model also leverages the technique of residual links. In all experiments, we use a six layers neural network with a hidden size of 512.[2]

2.2 Inverse Optimization

Our model's ability to solve inverse design problem is automatically gained by the differentiable nature of the neural network. For any differentiable objective function $\mathcal{J}(\mathbf{y})$ over the transfer function $\mathbf{y}$, we can apply gradient descent methods to optimize the input parameters of the neural network. In particular, consider a designer who wants to obtain a circuit geometry that satisfies a desired transfer function $\mathbf{y}^*$. We can define the objective function as the l_2-norm of the difference between the desired transfer function and that delivered by our design, i.e. $\mathcal{J}(\mathbf{y}) \triangleq \|\mathbf{y}(\mathbf{x}) - \mathbf{y}^*\|^2$. Since the transfer function is decided by the geometric parameters $\mathbf{x}$, the objective will be imposed on $\mathbf{x}$. Using the gradient method to optimize $\mathcal{J}(\mathbf{y}(\mathbf{x})) \simeq \mathcal{J}(\mathbf{f}(\mathbf{x};\theta))$, ideally will output a geometry parameters $\mathbf{x}^*$ with the transfer function $\mathbf{y}(\mathbf{x}^*)$ very close to the goal $\mathbf{y}^*$. The choice of the objective function can also change depending on the design. For example, when designing a band-pass filter, the goal is to allow signals in specific frequency bands to pass through, and block signals outside the desired bands. In this case, we can set the objective function to capture the ratio of the output energy between the signal in the pass-bands and the signal outside the pass-bands. For example, we may use the logarithm of that ratio as our objective as shown in Eqn. 1.

$$\mathcal{J}(\mathbf{y}) = \log(\sum_{i:\omega_i \in \Omega^*} \|y_i\|^2) - \log(\sum_{i:\omega_i \notin \Omega^*} \|y_i\|^2) \tag{1}$$

[2]To choose the network architecture, we performed a grid search on neural network hyper-parameters and tried depth values from four to ten, and hidden layer size from 64 to 512.



(a) 6-resonator filter (b) 4-resonator filter (c) THz channelizer

Figure 2: Distributed circuits templates: (a) is a dual-band on-PCB filter composed of 6 open-loop square resonators; (b) is a single-band on-chip filter composed of 4 open-loop square resonators; (c) is an on-chip channelizer with three sub-filters.

Here, Ω^* indicates the union of all pass-bands.

3 Experimental Evaluations

3.1 Distributed Circuits Templates

We experiment with three distributed circuits templates: a dual-band on-PCB open-loop coupling square resonator filter (6-resonator filter in Fig. 2(a)), a single-band on-chip open-loop coupling square resonator filter (4-resonator filter in Fig. 2(b)) and an on-chip THz channelizer (Fig. 2(c)).

Six-resonator filter. The filter operates between 2 GHz and 3 GHz. The circuit substrate is Al_2O_3 with a thickness of 1.27mm, which is a common choice in microwave component. The filter has 5 parameters: $x_1, \dots x_5$, which refer to the spacing between resonators. All parameters take values in the range $[5\text{mm}, 50\text{mm}]$.

Four-resonator filter. The filter operates between 200 GHz and 400 GHz. It is designed on the metal layers of an integrated chip based on IHP SG13G2 process [9]. The filter has 5 parameters: $x_1, \dots x_5$, which refer to the size of and the spacing between resonators. In our experiments, x_1 varies in $[60\mu\text{m}, 100\mu\text{m}]$ while $x_2, \cdots, x_5$ vary in $[5\mu\text{m}, 50\mu\text{m}]$.

THz channelizer. Designing a Terahertz circuit is very challenging. We consider a real-world THz channelizer, which is used for a 100-Gbps-level chip-to-chip communication IC. The channelizer is designed by a senior Ph.D. student in the Terahertz research group in our department. This design took him weeks and involved simulations on a supercomputer. The channelizer operates from 200 GHz to 400 GHz and has three channels centered at 235, 275 and 315 GHz, each having a bandwidth of 30 GHz. As shown in Fig. 2(c), the channelizer has 3 sub-modules that correspond to the three channels. The geometric parameters of each sub-module are as follows: $x_1 \in [1.75\mu\text{m}, 10\mu\text{m}]$ denotes the width of metal; $x_2 \in [2\mu\text{m}, 6.5\mu\text{m}]$, $x_3 \in [1.9\mu\text{m}, 10\mu\text{m}]$, $x_4 \in [4\mu\text{m}, 16\mu\text{m}]$ define the shape of folded structure; $x_5, x_6 \in [4\mu\text{m}, 10\mu\text{m}]$ are the horizontal and vertical distances between different folded structures.

3.2 Dataset and Training Protocol

To train our network, we generate many labeled examples using the CST STUDIO SUIT [3], a commercial EM simulator. Generally, it takes about 10 to 50 CPU minutes to simulate one circuit (i.e., one configuration of the parameters in the template). For every template in Fig. 2, we generate about 30,000 to 70,000 samples by using a distributed computing cluster with 800 virtual CPU cores. For each template, we use 80% of the data for training and the rest for testing. Training uses the Adam optimizer [11] and a batch-size of 64. In total, the model is trained 1000 epochs. The learning rate is initialized as 10^{-2} and decayed every 100 epochs by a factor of 0.5.

3.3 Evaluation of the Forward Task

We evaluate our model's prediction ability on all three templates introduced in Sec. 3.1, and report the results Table 1. As common in circuit literature, we express the error in dB. The error is computed

Table 1: Performance of forward learning for all three templates

Circuit	Number of Examples	Avg. Training Error	Avg. Test Error
Four-resonator filter	41087	0.22 dB	0.25 dB
Six-resonator filter	32742	0.27 dB	0.30 dB
THz channelizer	65850	0.57 dB	0.58 dB

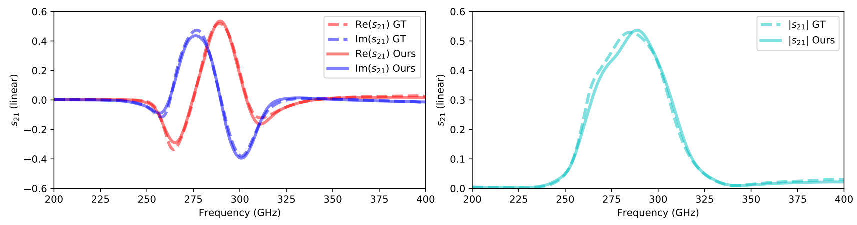


Figure 3: A random example of our model predicted transfer function for a given THz channelizer. In the plots, dash lines and solid lines indicate the ground truth and our predictions. We display the real and imaginary parts of the transfer function in the left panel and visualize the magnitude in the right.

as the mean value of the absolute difference error between the prediction value and the ground truth in dB, i.e. $\epsilon_{db}(\mathbf{y}, \hat{\mathbf{y}}) \triangleq \frac{1}{m}\sum_{i=1}^{m} |20\log_{10}(|y_i|) - 20\log_{10}(|\hat{y}_i|)|$.

Table 1 shows that our model achieves a mean test error of 0.25dB, 0.30dB, 0.58dB on the three templates, respectively. Such an error is very small, indicating that our model is highly accurate. For qualitative evaluation, we visualize a random test sample in Fig. 3. The figure shows an example from the Terahertz channelizer dataset. The graphs on the left show both the real and imaginary values of the S_{21}, whereas the graphs on the right show its magnitude. Although the Terahertz channelizer is the hardest design problem in our experiments, as we can see in Fig. 3, our model's prediction error is negligible. Visualization on other random samples performs no difference as the example we showed. In terms of the run-time for prediction, our model conducts one prediction within two milliseconds on a single NVIDIA 1080Ti GPU which is five orders faster than running one simulation using CST Studio on a modern desktop.

3.4 Evaluation of the Inverse Optimization

We evaluate our model's ability to solve the inverse design problem by using it to design the Terahertz channelizer described in Sec. 3.1. We optimize each sub-modules of the channelizer individually as the human expert did. For every sub-module, we use the gradient back-propagation method as introduced in Sec. 2.2. Since the design goal here is the same as designing a uni-band filter, we employ the filter design objective function in Eqn. 1. The optimization procedure goes as follows. We first uniformly sample 32 random initial values for the geometric parameters in the design space. We then iteratively apply gradient descent 100 times on those geometric parameters in parallel using Adam [11]. We pick the geometric parameters that produces the largest objective value as our inverse design result. The whole process is fast and completes in less than two seconds on a single GPU.

We compare the transfer function produced by our model with the transfer function for the channelizer designed by the human expert using the CST simulator. Fig. 4 shows the transfer functions for both designs. The figure shows that both designs have a good inter-channel isolation and meet the pass-band requirement. Finally, we note that the human expert spent several weeks optimizing the parameters in the channelizer's template to obtain his design, This indicates that the model could save the expert weeks of work.

3.5 Comparison with Past Work

As explained in Sec. 1, past neural network models for distributed circuit design learn the parameters of the transfer function as opposed to the function itself [1, 4, 5]. As a result, their approach does not lend itself to solving the inverse problem since the gradient is not smooth. In this section, we support this intuition with empirical results.

We compare our model with the most recent prior work [5], on the 4-resonator template. We train both models on the forward task, and use the trained models to solve the inverse problem. To test

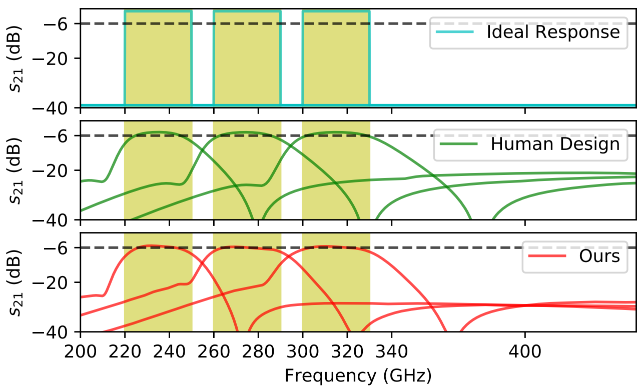
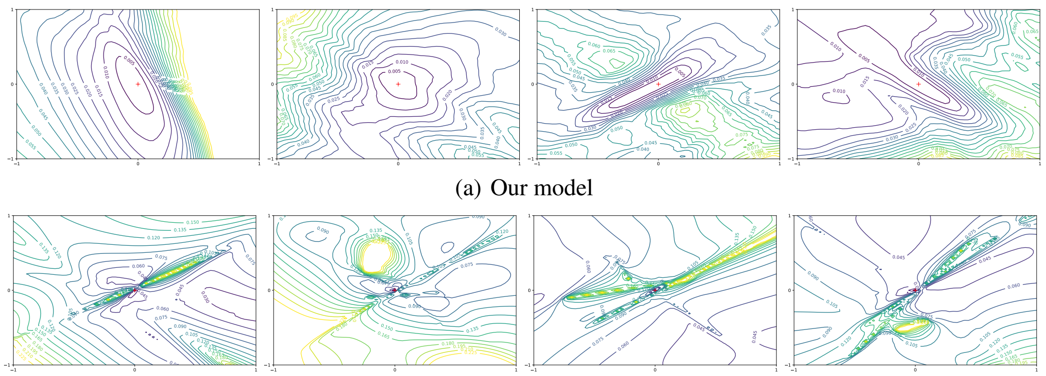


Figure 4: The transfer function of our optimized channelizer vs. the channelizer designed by the human expert. The yellow area indicate the desired pass-bands. The Y-axis plots range from $0dB$ to $-60dB$. Generally, the frequency region where the transfer function is over $-6dB$ is considered as the pass-band of the circuit.



(a) Our model

(b) Model in [12]

Figure 5: Comparison of the optimization landscapes of our model and past work. Each row contains four randomly generated 2D visualization of the gradient. In every plot, the red cross refers to the global optimum which has a zero objective value. The plots show that the gradient of our model is smooth allowing for inverse optimization, which is not the case for past work.

their performance on the inverse problem, we randomly pick a parameter setting for the template, $\mathbf{x}^*$, and use the commercial simulator to generate the circuit's actual transfer function, denoted as $\mathbf{y}^*$. We then consider the inverse problem, where given $\mathbf{y}^*$ the objective is to use the neural network to generate a parameter setting that satisfies this transfer function.

To evaluate the hardness of finding the optimal parameters via first-order methods, we visualize the value of the objective function near $\mathbf{x}^*$, which is a valid solution for the inverse problem. Since this is a high dimensional space, we leverage a 2D visualization technique, commonly used in the literature [6, 10, 12]. Specifically, we randomly sample two directions $\mathbf{x}_j, \mathbf{x}_k$ from the parameter space $\mathcal{X}$, and project the objective function on the plane created by them.

The objective function is set to $\mathcal{J}(\mathbf{y}) = \|\mathbf{y} - \mathbf{y}^*\|_2^2$, where $\mathbf{y}^*$ is the desired transfer function and $\mathbf{y}$ is the function returned by the model. For our model, the objective function can be computed directly as $\|\mathbf{f}(\mathbf{x}; \theta) - \mathbf{y}^*\|_2^2$. In contrast, the model in [5] approximates the transfer function as $\mathbf{y}(\omega) = \sum_{i=1}^{N} \frac{a_i}{j\omega - b_i}$ and returns the parameters a_i's and b_i's. Thus, to compute the value of the objective function, we substitute the parameters returned by the model in the above equation, then compute the objective function.

Fig. 5 shows random examples of the optimization landscape of our model and the model in [5]. The red cross marks the projection of $\mathbf{x}^*$. The figure shows that our landscape has a shape close to a convex function. In contrast, the landscape of the model in [5] is highly non-convex. In our landscape, there are always large basins around $\mathbf{x}^*$ which attract intermediate solutions to the global optimum during the gradient descent process. On the contrary, the landscape of past work is far from smooth, which can prevent the gradient method from converging to the optimal solution.

References

[1] Yazi Cao, Gaofeng Wang, and Qi-Jun Zhang. A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions. *IEEE Transactions on Microwave Theory and Techniques*, 57(11):2727–2742, 2009.

[2] David M Colleran, Clemenz Portmann, Arash Hassibi, César Crusius, Sunderarajan S Mohan, Stephen Boyd, Thomas H Lee, and Maria del Mar Hershenson. Optimization of phase-locked loop circuits via geometric programming. In *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pages 377–380. IEEE, 2003.

[3] CST official website. Cst - computer simulation technology. `https://www.cst.com/`, 2018. [Online].

[4] Feng Feng, Chao Zhang, Jianguo Ma, and Qi-Jun Zhang. Parametric modeling of em behavior of microwave components using combined neural networks and pole-residue-based transfer functions. *IEEE Transactions on Microwave Theory and Techniques*, 64(1):60–77, 2016.

[5] Feng Feng, Chao Zhang, Jianguo Ma, Qi-Jun Zhang, et al. Parametric modeling of microwave components using adjoint neural networks and pole-residue transfer functions with em sensitivity analysis. *IEEE Transactions on Microwave Theory and Techniques*, 65(6):1955–1975, 2017.

[6] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[7] Jia-Sheng Hong. *Microstrip Filters for RF/Microwave Applications*, volume 235. John Wiley & Sons, 2011.

[8] Jia-Sheng Hong and Michael J Lancaster. Couplings of microstrip square open-loop resonators for cross-coupled planar microwave filters. *IEEE Transactions on Microwave theory and Techniques*, 44(11):2099–2109, 1996.

[9] IHP. Ihp low-vol & multi-project service. `https://www.ihp-microelectronics.com/en/services/mpw-prototyping/sigec-bicmos-technologies.html`, 2018. [Online].

[10] Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An empirical analysis of deep network loss surfaces. 2016.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[13] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V Fernández. Analog circuit optimization system based on hybrid evolutionary algorithms. *INTEGRATION, the VLSI journal*, 42(2):137–148, 2009.

[14] Bo Liu, Dixian Zhao, Patrick Reynaert, and Georges GE Gielen. Gaspad: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(2):169–182, 2014.

[15] Wenlong Lyu, Pan Xue, Fan Yang, Changhao Yan, Zhiliang Hong, Xuan Zeng, and Dian Zhou. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(6):1954–1967, 2018.

[16] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International Conference on Machine Learning*, pages 3312–3320, 2018.

[17] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Multi-objective bayesian optimization for analog/rf circuit synthesis. In *Proceedings of the 55th Annual Design Automation Conference*, page 11. ACM, 2018.

[18] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. *arXiv preprint arXiv:1705.09792*, 2017.

[19] Ye Wang, Michael Orshansky, and Constantine Caramanis. Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6. IEEE, 2014.

[20] Chao Zhang, Jing Jin, Weicong Na, Qi-Jun Zhang, and Ming Yu. Multivalued neural network inverse modeling and applications to microwave filters. *IEEE Transactions on Microwave Theory and Techniques*, 66(8):3781–3797, 2018.