DeepOTF: Learning Equations-constrained Prediction for Electromagnetic Behavior

PENG XU, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
SIYUAN XU, Huawei Technologies Noah's Ark Lab Hong Kong, Hong Kong, Hong Kong
TINGHUAN CHEN, The Chinese University of Hong Kong - Shenzhen, Shenzhen, China
GUOJIN CHEN, The Chinese University of Hong Kong, Hong Kong, Hong Kong
TSUNGYI HO, The Chinese University of Hong Kong, Hong Kong, Hong Kong
BEI YU, The Chinese University of Hong Kong, Hong Kong, Hong Kong

High-quality passive devices are becoming increasingly important for the development of mobile devices and telecommunications, but obtaining such devices through simulation and analysis of electromagnetic (EM) behavior is time-consuming. To address this challenge, artificial neural network (ANN) models have emerged as an effective tool for modeling EM behavior, with NeuroTF being a representative example. However, these models are limited by the specific form of the transfer function, leading to discontinuity issues and high sensitivities. Moreover, previous methods have overlooked the physical relationship between distributed parameters, resulting in unacceptable numeric errors in the conversion results. To overcome these limitations, we propose two different neural network architectures: DeepOTF and ComplexTF. DeepOTF is a data-driven deep operator network for automatically learning feasible transfer functions for different geometric parameters. ComplexTF utilizes complex-valued neural networks to fit feasible transfer functions for different geometric parameters in the complex domain while maintaining causality and passivity. Our approach also employs an Equations-constraint Learning scheme to ensure the strict consistency of predictions and a dynamic weighting strategy to balance optimization objectives. The experimental results demonstrate that our framework shows superior performance than baseline methods, achieving up to $1,700\times$ higher accuracy.

CCS Concepts: • **Hardware** → **Modeling and parameter extraction;**

Additional Key Words and Phrases: Passive Devices, Electromagnetic Behavior, Deep operator network

ACM Reference Format:
Peng Xu, Siyuan XU, Tinghuan Chen, Guojin Chen, Tsungyi Ho, and Bei Yu. 2024. DeepOTF: Learning Equations-constrained Prediction for Electromagnetic Behavior. *ACM Trans. Des. Autom. Electron. Syst.* 29, 5, Article 80 (September 2024), 22 pages. https://doi.org/10.1145/3663476

Authors' Contact Information: Peng Xu, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China; e-mail: 1155187705@link.cuhk.edu.hk; Siyuan XU, Huawei Technologies Noah's Ark Lab Hong Kong, Hong Kong, Hong Kong; e-mail: xusiyuan520@huawei.com; Tinghuan Chen, The Chinese University of Hong Kong - Shenzhen, Shenzhen, Guangdong, China; e-mail: thchen@cse.cuhk.edu.hk; Guojin Chen, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: gjchen21@cse.cuhk.edu.hk; Tsungyi Ho, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: tyho@cse.cuhk.edu.hk; Bei Yu (Corresponding author), The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: byu@cse.cuhk.edu.hk.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2024 Copyright held by the owner/author(s).
ACM 1084-4309/2024/09-ART80
https://doi.org/10.1145/3663476

1 Introduction

High-quality passive devices are becoming increasingly important for the development of mobile devices and telecommunications [1, 2]. Typically, passive devices include various antennas, circulators, isolators, and inductors (Figure 1) [3–6]. These devices offer functions such as electromagnetic radiation, directional wave propagation, and **radio frequency (RF)** signal processing. Thus, the fundamental functions of RF devices arise from the strong coupling of magnetization dynamics to electromagnetic fields. Therefore, understanding the role of electromagnetic behavior in these passive devices is crucial for enhancing device design and improving performance.

Simulation and analysis of **electromagnetic (EM)** behavior are crucial for obtaining high-quality passive devices, but those processes are also very time-consuming [7, 8]. In traditional design flow, a passive device is designed with its physical, geometrical, and electrical parameters. Then, simulation and analysis are used to obtain its distributed parameters (scattering parameter S, impedance parameter Z, and admittance parameter Y), impacting transmission and reflection of the electromagnetic waves. If the design specification of its distributed parameters is not satisfied, then the designer will return to the previous step to tune its geometrical and electrical parameters. The passive component design heavily relies on the engineer's experience and theoretical analysis. However, analyzing circuit behavior with distributed parameters is very sophisticated due to many mutual couplings, enclosures, and effects from other adjacent components on the same substrate. Such circuits are simulated and analyzed using specialized software packages and analyzers, i.e., EM simulation and analysis [9, 10]. However, both EM simulation and EM analysis are very expensive and time-consuming. Moreover, many design iterations are required to guarantee quality and performance, which seriously affects product development efficiency and the time to market.

An effective tool for parametric modeling of EM behaviors in the microwave area has been recognized as the **artificial neural network (ANN)** [11–13]. EM design optimization can be costly, because it usually requires repeated EM simulations due to changes in the geometrical parameter values. As it can learn the relationship between EM behaviors and geometrical parameters, the ANNs can quickly and accurately anticipate the EM behavior of microwave components after training. The developed parametric model can then be integrated into the high-level circuit and system design.

To improve the ANNs' capacity for learning and generalization, the **Neuro-Transfer function (Neuro-TF)** approaches [14–16] combine neural networks and transfer functions. In this method, transfer functions describe the passive devices' EM responses as a function of frequency. However, limited by the specific forms of the transfer functions, those methods are faced with dramatic degradation of performance caused by the discontinuity issues [14] and the high sensitivities problems [15, 16] when geometrical parameters vary.

A more serious issue of existing methods is that the predicted results are inconsistent with the known conversion equations between distributed parameters. ANN models are developed in these works to predict one of the distributed parameters. In practice, the simulation results of distributed parameters must be verified whether they satisfy the design specifications. However, a significant error in conversion results is usually brought when the predicted parameter locates in a sensitive region to the calculated parameter. Although the predictor can achieve reasonably good accuracy for the EM behavior, the transferred prediction has significant errors, which is unacceptable. This is because the conversion equations among distributed parameters are not considered. DeepOTF further extends these physical relationships between predictions, allowing consistent predictions of different distributed parameters.

This work proposes a novel EM behavior learning framework, *a deep operator network for learning the transfer function with equation constraints*, to achieve accurate and consistent modeling

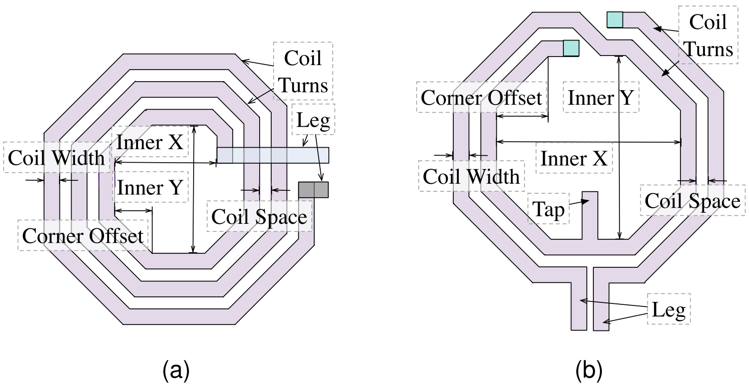

Fig. 1. The structures of different types of inductors with definitions of their geometric parameters: (a) single-ended inductors; (b) differential inductors.

of EM behavior. Specifically, we first design a customized deep operator network for EM prediction, which can identify the suitable transfer function for different data. Then, a novel equations-constraint learning scheme is developed to enforce a strict restriction on distributed parameter predictions. Finally, we introduce a dynamical weighting strategy via uncertainty to tackle the potential tradeoffs.

Our major contributions are summarized as follows:

— To overcome these limitations of existing NeuroTF methods, we propose two different neural network architectures: DeepOTF and ComplexTF. DeepOTF is a customized deep operator network for automatically learning feasible transfer functions from data. ComplexTF utilizes complex-valued neural networks to fit feasible transfer functions for different geometric parameters in the complex domain while maintaining causality and passivity.
— We have developed an equations-constraint learning scheme for learning predictions for EM behavior. The proposed scheme can guarantee consistency between predictions of different distribution parameters by incorporating the conversion equations among them into the learning process.
— We further propose a dynamic weighting method to solve the potential conflict problem of different objectives in the equations-constraint learning process.
— Experimental results show our framework's effectiveness compared with baseline methods.

2 Preliminaries

This section provides the formulation of the equations-constrained prediction for EM behavior problem (Section 2.1), followed by an overview of NeuroTF methods for modeling EM behavior (Section 2.2) and deep operators learning (Section 2.3).

2.1 Problem Formulation

The objective is to accurately estimate distributed parameters by satisfying the physics laws among them. In electromagnetics, typically distributed parameters contain scattering parameter S, impedance parameter Z, and admittance parameter Y. They impact the transmission and reflection of EM waves in the RFIC. Without loss of generality, we use the typical RF circuit component, inductors, in the experiments of this article.

Table 1. Comparison between Different NeuroTF Methods

NeuralTF Methods	Function Format	Characteristics
Rational-based [14]	$\sum_{i=1}^{N} \frac{r_i(\boldsymbol{x}, \boldsymbol{w}_r)}{s-p_i(\boldsymbol{x}, \boldsymbol{w}_p)}$	High sensitivities of the coefficients due to its high-order format.
Pole-residue-based [15]	$\frac{\sum_{i=1}^{N} a_i(\boldsymbol{x}, \boldsymbol{w}_a)s^{i-1}}{1+\sum_{i=1}^{N} b_i(\boldsymbol{x}, \boldsymbol{w}_b)s^i}$	Discontinuity and non-smoothness issues due to its pole-residue format.
Hybrid-based [16]	$\sum_{i=1}^{N_1} \frac{r_i(\boldsymbol{x}, \boldsymbol{w}_r)}{s-p_i(\boldsymbol{x}, \boldsymbol{w}_p)} + \frac{\sum_{i=1}^{N_2} a_i(\boldsymbol{x}, \boldsymbol{w}_a)s^{i-1}}{1+\sum_{i=1}^{N_2} b_i(\boldsymbol{x}, \boldsymbol{w}_b)s^i}$	Restriction of the specific function format

Formally, the three distributed parameters are defined as follows:

Definition 1 (Scattering Parameter (S-parameter)). The response of a network to signal(s) incident to any or all of the ports.

Definition 2 (Admittance Parameter (Y-parameter)). The incoming and outgoing voltages and currents of a network.

Definition 3 (Impedance Parameter (Z-parameter)). The linear characteristics of RF electronic circuits and components.

The S-parameters depict the characteristics of incidence and reflection, which are typically the primary focus. In the case of inductor devices, we are also interested in the Z-parameters.

Problem 1 (Equations-constrained Prediction for EM Behavior). *Given a passive component's physical, geometrical, and electrical parameters, predict its S-parameter, Y-parameter, and Z-parameter with satisfying explicit equations.*

Specifically, we can consider using an ANN $f_{\boldsymbol{\theta}}$ to predict the corresponding distributed parameters S-parameter, Y-parameter, and Z-parameter:

$$\hat{\boldsymbol{y}}_S, \hat{\boldsymbol{y}}_Y, \hat{\boldsymbol{y}}_Z = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{\theta}$ is the parameters of the ANN.

Based on the predicted parameters ($\hat{\boldsymbol{y}}_S$, $\hat{\boldsymbol{y}}_Y$, $\hat{\boldsymbol{y}}_Z$) and equations between the distributed parameters, we can calculate the corresponding equation conversion values using the conversion formulas between them for each predicted item. While the predictor can achieve reasonable accuracy for each distributed parameter, the conversion results derived by the formulations conversion function have significant numerical errors. We refer to this issue as *consistency error*.

Definition 4 (Consistency Error). The error is derived from the conversion results of the predicted distributed parameters to each other.

2.2 NeuroTF Methods for Modeling Electromagnetic Behavior

To improve the ANNs' capacity for learning and generalization, the NeuroTF approaches (Table 1) [14–16] combine neural networks and transfer functions, as shown in Figure 2. In this method, transfer functions describe the passive components' EM responses as a function of frequency. However, the coefficients of transfer functions experience discontinuity issues when faced with large geometric variations.

The rational-based NeuroTF [14] develops a training method for the rational transfer function to create NeuroTF models in the frequency domain. The training method in Reference [14] provides a solution to the discontinuity of coefficients in transfer functions over geometrical variables for the rational transfer function. However, the model's precision and robustness are constrained by the high sensitivities of the coefficients of the rational transfer function w.r.t. geometrical parameters due to its high-order format.

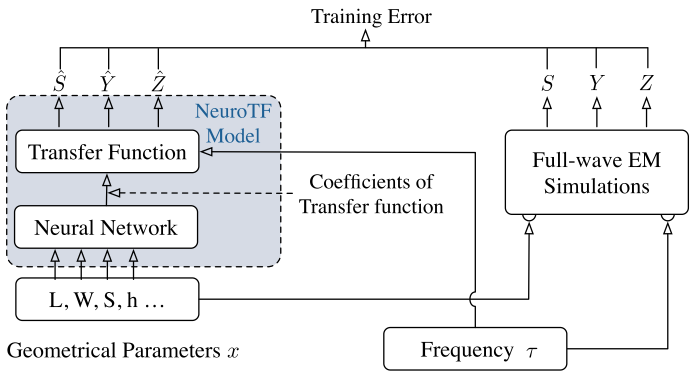

Fig. 2. The overview of previous NeuroTF methods.

With the help of the order-changing technique, the pole-residue-based NeuroTF [15] develops a combined neural network and pole-residue-based transfer function model. An alternative pole-residue method to the rational transfer function was provided in Reference [17]. Nevertheless, the training method to address the discontinuity issues in Reference [14] cannot directly apply to the pole-residue-based transfer functions. Thus, to address this issue, the pole-residue-based NeuroTF [15] proposes an order-changing technique, which allows the pole-residue-based method to take different orders in regions with different variations of geometric parameters.

To combine the advantages of the rational-based transfer function and pole-residue-based transfer function, Reference [16] proposes a hybrid-based NeuroTF with a hybrid-based transfer function. It starts with the pole-residue-based transfer function and automatically separates the smooth and continuous poles/residues from the poles/residues with discontinuity and non-smoothness issues. Then, the poles/residues that have those issues are converted into the coefficients of the rational-based transfer function to resolve the discontinuity and non-smoothness issues.

Even though hybrid-based NeuroTF combines rational-based and pole-residue-based methods, it is not exceptionally limited by the specific forms of transfer functions. In addition, the computation of transfer functions is time-consuming, which is also the bottleneck when training the NeuroTF models. Therefore, to address the existing issues of NeuroTF, we propose a novel operator network that allows us to learn transfer functions for different geometrical parameters implicitly.

2.3 Deep Operators Learning

Neural operator learning has emerged as a powerful tool for learning maps between inputs of a dynamic system and its state. This line of work extends the use of Neural Networks as surrogate models for a solution operator for various dynamic systems. Representative research works in this direction include the DeepONet architecture [18] and the integral neural operator architectures [19–21].

The generalizability to different input instances is one of the most notable advantages of neural operators. Once the neural operator is trained, solving for a new instance of the input parameter requires only a forward pass, thus different from traditional ANNs that only approximate the solution for a single instance.

An earlier effort in this field is DeepONet [18]. The main idea is to create a two-branches neural network, one for learning the underlying function mapping through the sampled point from given functions and the other for taking various instances as input and returning the weight scalar vector of basis functions. Although DeepONet was originally proposed for modeling dynamic systems described by partial differential equations in the time domain, the introduced neural

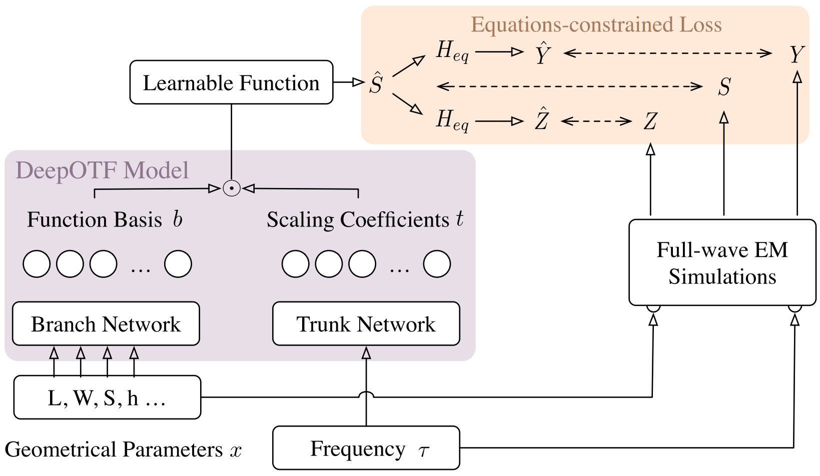

Fig. 3. The overview of our proposed DeepOTF framework.

operator structure has demonstrated advantages and potential applications in linear dynamic systems modeling [22].

Neural operators provide an alternative framework for operator learning [19–21]. Typical neural operators compose multiple hidden layers, with each hidden layer composing an affine operator with a scalar nonlinear activation operator [20, 21]. More recently, Reference [19] proposed to use convolution-based integral kernels in Fourier space to address the drawback of existing operators that require using integral kernels within the neural operators. This approach results in significant performance accelerations when solving the unknown partial differential equation system.

Motivation. When designing a predictor for EM simulation, it is crucial to consider the unique properties of EM behavior prediction. Specifically, there are two important aspects to take into account: (1) The distributed parameters corresponding to different frequencies, for the same geometric parameters, should originate from the same mapping function. (2) Different geometric parameters should correspond to distinct mapping functions. Unfortunately, simple ANN predictors often overlook these properties, leading to subpar generalization performance. However, DeepONET offers significant advantages in this regard. DeepONET naturally aligns the frequency in the frequency domain with the location part of the input. Moreover, the use of a branch net enables the generation of different transfer functions for different size parameters, ensuring that the same geometric parameters map to the same mapping function.

3 Algorithms

This section presents the proposed DeepOTF framework flow and details the algorithms. Figure 3 and Figure 5(a) show the overall flow of our modeling framework. Our modeling differs from the existing NeuroTF methods that rely on the fixed format of transfer functions. Instead, we design a customized deep operator network for EM prediction and enforce constraints among the predictions.

The whole framework can be summarized with three major components:

— **DeepOTF and ComplexTF models for learning transfer functions:** The existing NeuroTF methods are limited by the specific form of the transfer function with discontinuity issues or high sensitivities issues. In DeepOTF, we propose to learn a neuro operator that produces the desired transfer functions for different data. In ComplexTF, we utilize

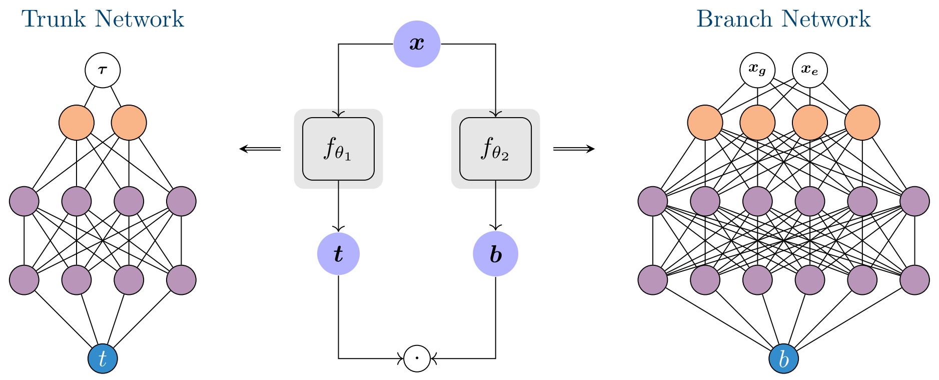

Fig. 4. The network structure of our proposed DeepOTF predictor.

complex-valued neural networks to fit feasible transfer functions for different geometric parameters in the complex domain while maintaining causality and passivity.

— **Equations Constrained Learning Scheme (ECL):** The existing prediction methods pursue each distributed parameter's accuracy and bring about large consistency errors. In DeepOTF, we further develop an equations-constrained learning scheme to address the issue.

— **Dynamic Weighting Strategy:** To balance the potential tradeoff in ECL scheme, we adopt a dynamic weighting strategy considering the uncertainty during optimization.

In the remainder of this section, we first introduce the design of our DeepOTF predictor in Section 3.1, our ComplexTF predictor in Section 3.2, and then establish the ECL learning scheme in Section 3.3 and provide a dynamic weight strategy in Section 3.3.2.

3.1 DeepOTF: Data-driven Operator Learning for Identifying Neuro-transfer Functions

Existing NeuroTF methods are limited by the specific form of the transfer function, resulting in discontinuity issues or high sensitivities [14, 15]. Thus, we need to identify the appropriate transfer function for specific data, which is often task-independent. A different approach is to learn an operator that produces the desired transfer functions for different data.

Regarding the unique properties of prediction for EM behavior, designing a predictor for EM simulation should take two aspects into account: (1) For the same geometric parameters, the distributed parameters corresponding to different frequencies come from the same mapping function; (2) for different geometric parameters, they correspond to a different mapping function. Simple ANNs predictors ignore these two properties, leading to poor generalization performance. Although the NeuroTF method satisfies these two properties by introducing transfer functions, it is limited by the specific form of the transfer function.

Inspired by DeepONet [18], we propose a customized deep operator network architecture that targets implicitly learning the transfer functions for different geometric parameters, namely, functional **deep operator neuro-transfer functions (DeepOTF)**. Our proposed DeepOTF follows an asymmetrical network architecture with a branch and trunk network, as shown in the Figure 4.

In a general setting, the inputs, i.e., $\boldsymbol{x}$, of our proposed network consist of two separate components: $[\boldsymbol{x}_g, \boldsymbol{x}_e]$ and $\boldsymbol{\tau}$, where $\boldsymbol{x}_g$ and $\boldsymbol{x}_e$ are geometric features of the input vector $\boldsymbol{x}$, and $\boldsymbol{\tau}$ is the input frequency vector. The branch network takes $[\boldsymbol{x}_g, \boldsymbol{x}_e]$ as the input and outputs $\boldsymbol{b} = [b_1, b_2, \ldots, b_k] \in \mathbb{R}^p$, where p is the dimension of prediction. The trunk network takes frequency vector $\boldsymbol{\tau}$ as the input and outputs a vector $\boldsymbol{t} = [t_1, t_2, \ldots, t_k] \in \mathbb{R}^p$. In the last stage, we may add a bias $b_0 \in R$.

In the DeepOTF framework, we utilize the first layer of the branch network as the function encoding layer. Our objective is to learn the operator from the input function using the optimal transfer function. This approach can be seen as a specific instance of operator learning. We have the first layer of the branch and a default function encoding layer denoted as u_{θ^1}. The input function in this case is $u_{\theta^1}(\boldsymbol{x}_g, \boldsymbol{x}_e)$. However, since the optimal transfer function is unknown, we aim to solve this problem using an end-to-end approach. Through experimental analysis, we have validated the performance advantage of this architecture in EM parameter learning.

We merge them to approximate the desired mapping functions for different geometric parameters:

$$G(u(\boldsymbol{x}_g, \boldsymbol{x}_e))(\tau) \approx \boldsymbol{b} \cdot \boldsymbol{t} + b_0, \tag{2}$$

where G is an operator taking the output of the inner function u that take geometric features $[\boldsymbol{x}_g, \boldsymbol{x}_e]$ as input, and then outputs $G(u(\boldsymbol{x}_g, \boldsymbol{x}_e))$, w.r.t. τ, as the corresponding implicit transfer functions. For any specific frequency τ in the domain of $G(u(\boldsymbol{x}_g, \boldsymbol{x}_e))(\tau)$, the output $G(u(\boldsymbol{x}_g, \boldsymbol{x}_e))(\tau)$ will give prediction of the corresponding distributed parameter. Due to the multiplication of outputs from two differentiable network modules, the continuity of the transfer functions can naturally be ensured.

The Universal Approximation Theorem for Operator from Reference [18] shows that the DeepOTF structure can effectively identify the appropriate transfer function from data by fitting the appropriate transfer function from data. This approximation theorem indicates the potential application of neural networks to learn nonlinear operators from data. We can learn a functions producer here, similar to learning functions from data.

The next step is to demonstrate that both the set of rational-based and pole-residue-based transfer functions are proper subsets of the function space that our proposed DeepOTF can express. By establishing this, we can confidently assert that DeepOTF surpasses traditional methods that rely on predetermined transfer function forms. This is because DeepOTF can automatically learn the most suitable transfer function format from its function space.

To prove that DeepOTF possesses greater expressive power than the fixed-form NeuroTF method, it suffices to demonstrate that DeepOTF can represent both the rational-based and pole-residue-based formats of transfer functions are the proper subsets of the DeepOTF function set $G(u)$. We present the following proposition to clarify the expressive power of DeepOTF in comparison to the fixed-form NeuroTF method:

Proposition 1. *Let σ be a continuous non-polynomial function, $\boldsymbol{x}$ be a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$, V be a compact set in $C(K_1)$, and G be a nonlinear continuous operator mapping V into $C(K_2)$. Note that the function set of DeepOTF $G(u)$ is defined on $C(K_2)$. Both the set of continuous functions with rational-based transfer functions, denoted as $\mathcal{F}_R$, and the set of continuous functions with pole-residue-based transfer functions, denoted as $\mathcal{F}_P$, are proper subsets of $C(K_2)$.*

Proof. To begin, it should be noted that both continuous functions derived from rational-based transfer functions, denoted as $f_r \in \mathcal{F}_R$, and continuous functions derived from pole-residue-based transfer functions, denoted as $f_p \in \mathcal{F}_P$, belong to the set $C(K_2)$.

Formally,

$$f_r \in \mathcal{F}_R, \quad f_p \in \mathcal{F}_P \implies f_r, f_p \in C(K_2). \tag{3}$$

Next, we aim to demonstrate the existence of functions $f_o \in C(K_2)$ that do not belong to either $\mathcal{F}_R$ or $\mathcal{F}_P$. Consider the function $f_o(x) = e^{1/(x-c)}$, where c is a constant parameter. This function is not in $\mathcal{F}_P$, since it is not a ratio of two polynomial functions, and it is not in $\mathcal{F}_R$, since it is not the summation of ratios of poles and residues.

Consequently, we have established the existence of functions that lie outside the sets $\mathcal{F}_R$ and $\mathcal{F}_P$, thus completing the proof. $\square$

With the help of neural operators, our proposed DeepOTF implements implicit transfer function learning and automatically approximates different transfer functions for different geometric parameters. Thus, our proposed method omits the discontinuity issues and high sensitivities problems. In addition, due to the elimination of the complex computation of transfer function, our proposed DeepOTF also has further improved running speed than NeuroTF.

3.2 ComplexTF: Complex-valued Neuro-transfer Method with Causality and Passivity

When predicting the EM parameters of the device's electromagnetic behavior, two key issues need to be considered. First, the EM parameters are represented in complex form. However, most of the previous methods and architectures were based on real-valued operations and representations. Therefore, we have designed a series of elementary complex activation functions and layers to assemble a **Complex Feedforward Neural Network ($\mathbb{C}$FNN)** in Section 3.2.1. Second, previous NN-based techniques have primarily focused on numerically matching S-parameters, neglecting the underlying physical phenomena and lacking physical consistency (causality and passivity). Consequently, these techniques are unsuitable for broader applicability. Therefore, in this article, we utilize the **passivity enforcement layer (PEL)** in Section 3.2.3 to learn a physically meaningful representation between input parameters and S-parameters.

3.2.1 Complex Feedforward Neural Networks. The $\mathbb{C}$FNNs involve complex numbers, where a complex number $z = a + ib$ consists of a real component denoted as $\Re(z) = a$ and an imaginary component denoted as $\Im(z) = b$.

To perform the equivalent of a traditional real-valued linear layer in the complex domain, we multiply a complex matrix $\boldsymbol{W} = \boldsymbol{A} + i\boldsymbol{B}$ by a complex vector $\boldsymbol{h} = \boldsymbol{x} + i\boldsymbol{y}$ where $\boldsymbol{A}$ and $\boldsymbol{B}$ are real matrices and $\boldsymbol{x}$ and $\boldsymbol{y}$ are real vectors, since we are simulating complex arithmetic using real-valued entities.

$$\mathbb{C}Linear(\boldsymbol{h}) = \boldsymbol{W}\boldsymbol{h} = (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{y}) + i(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{y}).$$

If we utilize matrix notation to represent the real and imaginary components of the complex-valued linear operation, then we can express it as follows:

$$\mathbb{C}Linear(\boldsymbol{h}) = \begin{bmatrix} \Re(\boldsymbol{W}\boldsymbol{h}) \\ \Im(\boldsymbol{W}\boldsymbol{h}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B} \\ \boldsymbol{B} & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}.$$

The activation function used is the **complex rectified linear unit (CReLU)** defined as follows:

$$\mathbb{C}ReLU(z) = \mathrm{ReLU}(\Re(z)) + i\,\mathrm{ReLU}(\Im(z)). \tag{4}$$

As depicted in Figure 5(b), we construct the $\mathbb{C}$FNN sequentially as:

$$\mathbb{C}FNN : \mathbb{C}Linear \circ \mathbb{C}ReLU \circ (\mathbb{C}Linear \circ \mathbb{C}ReLU) \times N \ldots \circ \mathbb{C}Linear,$$

where $\times N$ indicates that there are N hidden blocks consisting of a complex linear layer followed by a CReLU.

3.2.2 Complex Neuro-transfer Method. With the assistance of $\mathbb{C}$FNN, we can naturally derive the Neuro-transfer Function method in the complex domain. In this method, let $\boldsymbol{y}$ represent the model's output. $\boldsymbol{x}$ represents the inputs of the model, i.e., the geometrical parameters. $\mathcal{P}$ denotes a mapping function between inputs and the output vector containing all the transfer function parameters (e.g., poles/residues).

We use the complex feedforward neural networks ($\mathbb{C}$FNN with weights Θ) as the mapping function between inputs $\boldsymbol{x}$ and transfer function parameters $\mathcal{P}$:

$$\boldsymbol{y}(x, w, s) = H(\boldsymbol{p}, s) = H(\mathbb{C}FNN(\boldsymbol{x}, \Theta), s), \tag{5}$$

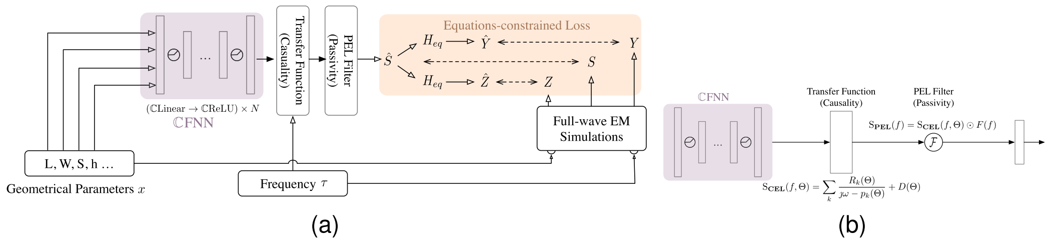
Equations-constrained Loss
H_{eq} → $\hat{Y}$ ↔ Y
$\hat{S}$ ↔ S
H_{eq} → $\hat{Z}$ ↔ Z
Transfer Function (Causality)
PEL Filter (Passivity)
(ℂLinear → ℂReLU) × N
ℂFNN
Full-wave EM Simulations
L, W, S, h ...
Geometrical Parameters x
Frequency τ
(a)
ℂFNN
Transfer Function (Causality)
PEL Filter (Passivity)
$S_{\mathrm{PEL}}(f) = S_{\mathrm{CEL}}(f,\Theta) \odot F(f)$
$S_{\mathrm{CEL}}(f,\Theta) = \sum_k \frac{R_k(\Theta)}{j\omega - p_k(\Theta)} + D(\Theta)$
(b)

Fig. 5. (a) The overview of our proposed ComplexTF framework; (b)The network structure of our proposed ComplexTF predictor.

where $H(\boldsymbol{p}, s)$ represents the transfer function and s represents the frequency in the Laplace domain.

In traditional NeuroTF methods, the complex-valued nature of the output's EM parameters is often not taken into account [15]. Instead, a simple approach of separating the real and imaginary parts is used to fit and learn with a real-valued neural network as in References [14–16]. However, neglecting the complex nature of the EM parameters in this fitting process may lead to two issues: First, the optimal mapping function may exist in the complex domain, and second, accuracy errors caused by disregarding geometric properties are overlooked.

By introducing complex-valued neural networks, the first issue can be naturally addressed, allowing us to identify an appropriate mapping function in the complex-valued functions space. As for the consideration of the second geometric information, we will introduce a complex-specific geometric loss function in Section 3.2.4 to tackle this problem.

3.2.3 Causality and Passivity Enforcement. For causality enforcement, if complex poles appear in the form of complex conjugate pairs in the left half-plane of the complex plane, then the model naturally satisfies causality and stability. For example, we can fix the real part of these poles to be negative. Since complex conjugate pairs have the same real part, it ensures that the pole pairs are located in the left half of the complex plane.

Specifically, here, we choose to model the transfer function using the Pole-Residue form, assuming S-parameters without loss of generality. Therefore, we have the following equation:

$$S_{\mathrm{CEL}}(f, \Theta) = \sum_k \frac{R_k(\Theta)}{j\omega - p_k(\Theta)} + D(\Theta). \tag{6}$$

In modeling the real part of the poles, we employ the following treatment at the output of the neural network:

$$\Re(p_k) = -e^{ax}, \tag{7}$$

where $a = 0.01$. This treatment ensures that the poles are located in the left half-plane of the complex plane while effectively mitigating the issue of the excessively large dynamic range of the pole's real part.

For passivity enforcement, the S-parameters output at frequency point f are denoted as $S(f)$ in Neuro-TF methods. The passivity condition for S-parameters is that the maximum singular value of $S(f)$ should not exceed 1. Therefore, we can satisfy this condition by introducing a passivity correction module, which is a frequency-dependent filter that ensures the maximum singular value of the output S-parameters does not exceed 1.

We utilize the frequency-dependent passivity enforcement operation in Reference [23] as filtering in the frequency domain, which is written as:

$$S_{\mathrm{PEL}}(f) = S_{\mathrm{CEL}}(f, \Theta) \odot F(f), \tag{8}$$

where the complex-valued passivity enforcement filter, denoted as $F(f)$, is applied to each data point along the frequency. To inflict minimal changes to $\mathrm{S}_{\mathrm{CEL}}(f)$, the desired magnitude spectrum of $\Sigma(f)$ at each data point is expressed as:

$$|\Sigma(f)| = \begin{cases} \frac{1}{\hat{\sigma}_{max}(f)}, & \text{for } \hat{\sigma}_{max}(f) > 1 \\ 1, & \text{for } \hat{\sigma}_{max}(f) \leq 1. \end{cases} \tag{9}$$

There are two approaches to calculating the maximum singular value of the S matrix $\hat{\sigma}_{max}(f)$. One approach is to directly compute the singular values using the SVD function. However, the drawback is that it has low computational efficiency. Another approach is to estimate the maximum singular value of the S matrix using its trace, as referenced in Reference [24]:

$$\partial = \sqrt{\frac{c}{n} + \sqrt{\frac{n-1}{n}\left(d - \frac{c^2}{n}\right)}}, \tag{10}$$

where n is the number of ports, c and d are the trace estimation:

$$\begin{gathered} c = \operatorname{tr}\left(S^H S\right) = \sigma_1^2 + \ldots + \sigma_n^2, \\ d = \operatorname{tr}\left(\left(S^H S\right)^H \left(S^H S\right)\right) = \sigma_1^4 + \ldots + \sigma_n^4. \end{gathered} \tag{11}$$

Previous studies have demonstrated that it is possible to construct a minimum-phase passivity enforcement filter based solely on the magnitude spectrum of a minimum-phase frequency response [23, 25], which can be interpreted as the frequency response of a causal system. A minimum-phase passivity enforcement filter is formulated using the desired magnitude spectrum as:

$$\begin{aligned} \Sigma(f) &= |\Sigma(f)| e^{j\theta(f)}, \\ \theta(f) &= \mathscr{H}\{\log|\Sigma(f)|\}, \end{aligned} \tag{12}$$

where $\log(\cdot)$ is the natural logarithm operator and the Hilbert transform $\mathscr{H}\{\cdot\}$ is taken using the FFT-based approach.

3.2.4 Geometrical Loss Function for Complex Numbers. In traditional Neuro-TF methods, the output S-parameters are often represented using a separated matrix representation due to the absence of neural networks that can handle complex numbers. When designing the loss function, fitting is typically performed separately for the real and imaginary parts, neglecting the geometric properties of complex numbers.

Therefore, we propose a loss function that leverages the properties of complex numbers, specifically combining the squared error functions of both the magnitude and the angle of the complex numbers. Specifically, In complex analysis, the complex numbers are customarily represented by the symbol z, which can be separated into its real (x) and imaginary (y) parts:

$$z = x + iy.$$

In this customary notation, the complex number z corresponds to the point (x, y) in the Cartesian plane. In the Cartesian plane the point (x, y) can also be represented in polar coordinates as:

$$(x, y) = (r\cos\theta, r\sin\theta) \quad \longrightarrow \quad (r, \theta) = \left(\sqrt{x^2 + y^2}, \quad \arctan\frac{y}{x}\right). \tag{13}$$

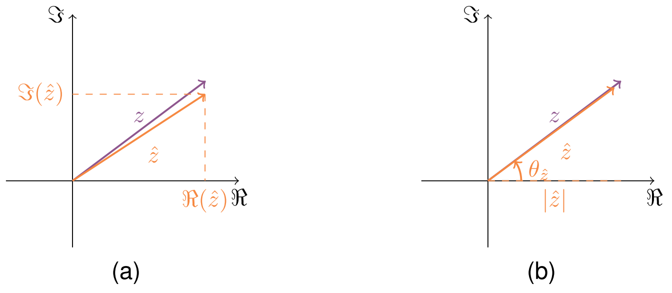

Fig. 6. A comparison of different loss functions for complex numbers: (a) Loss function for separate representation; (b) Loss function for geometric representation.

Based on the definitions of angle and magnitude, we have a geometric loss function for EM Parameters as follows:

$$\mathcal{L}_{geo}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{i=1}^{N} \| |y_i| - |\hat{y}_i| \|_2 + \| \arg(y_i) - \arg(\hat{y}_i) \|_2 \,, \tag{14}$$

where N is number of Port for EM parameters, $|\cdot|$ is the mod function, and $\arg(\cdot)$ is the angle for complex numbers. By minimizing this loss function, we can effectively fit complex-valued predictions and targets while maintaining its geometrical meanings and obtain more accurate results. We visualize the loss functions for separate representation and geometrical representation, as shown in Figure 6. The loss function for geometrical representation aims to preserve the phase and magnitude of the predicted EM parameters as much as possible, resulting in lower errors during the testing phase in practice.

3.3 ECL: Equations-constrained Learning Scheme with Dynamic Weighting Strategy

In this work, we attempt to address the consistency error problem in the EM simulation from a new perspective. It is easy to see that the ultimate goal of EM prediction is to make accurate and consistent predictions to estimate the simulation behavior—however, blindly pursuing the accuracy of a single prediction of each distributed parameter is not feasible if thinking about the consistency error. Towards this, *learning the optimal predictor while maintaining consistency is vital to the prediction of EM behavior*. While reducing the consistency error does increase training time, it also enhances the consistency of the output results, making them follow the physical characteristics.

However, the existing soft-constraint approaches are not a cure, so we propose an equations-constrained learning scheme to solve this issue. While regularization and other soft-constrained methods, such as shared representation like multi-task learning [26], could alleviate consistency errors, those methods fall short of maintaining consistency among predictions.

As for solving consistency issues, our proposed equations-constrained scheme enforces hard constraints among the predictions of distributed parameters, as shown in the Figure 8. The core idea is that we incorporate prior transformation equations among distributed parameters into the model's output. We derive the prediction of Y-parameter and Z-parameter from the basic prediction, i.e., S-Parameter. Therefore, we eliminate the inconsistency between different prediction distributed parameters. The error between target distributed parameters will be reduced to the error of basic prediction. Moreover, the most important advantage of our ECL scheme is that our predictions are fully consistent for the equations conversion. Since it secures the full consistency among predictions, it is more meaningful for modeling EM behavior compared with previous methods.

We implement equation-conversion modules in our approach, denoted as H_{eq}. We exploit the pre-existing conversion formulas between distributed parameters. These equation-conversion modules contain forward and backward functions, enabling them to be easily inserted into existing neural network predictors. The equation-conversion modules allow us to build predictions for other distributed parameters using the S parameters as the base prediction. It is also possible to use Y-parameter or Z-parameter as the base prediction, but we found in our empirical experiments that the S parameter performs better than others.

Concretely, we can consider using an parametric model $f_{\boldsymbol{\theta}}$ with network parameters $\boldsymbol{\theta}$ to predict the base distributed parameter S-parameter:

$$\hat{\boldsymbol{y}}_S = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{15}$$

where $\boldsymbol{x}$ is the input feature vector, which consists of $\boldsymbol{\tau}$ as input frequency vector, and $[\boldsymbol{x}_g, \boldsymbol{x}_e]$ as the geometric features.

Subsequently, we obtain the Y-parameter and Z-parameter conversion results through the equation-conversion modules and define them as the predicted values of the Y-parameter and Z-parameter.

$$\begin{aligned} \hat{\boldsymbol{y}}_Y &\coloneqq \hat{\boldsymbol{y}}_{Y\leftarrow S}, \hat{\boldsymbol{y}}_{Y\leftarrow S} = H_{S\rightarrow Y}(\hat{\boldsymbol{y}}_S), \\ \hat{\boldsymbol{y}}_Z &\coloneqq \hat{\boldsymbol{y}}_{Z\leftarrow S}, \hat{\boldsymbol{y}}_{Z\leftarrow S} = H_{S\rightarrow Z}(\hat{\boldsymbol{y}}_S), \end{aligned} \tag{16}$$

where $H_{S\rightarrow Y}$ is the equation-conversion module transforming S-parameter to Y-parameter, and $H_{S\rightarrow Z}$ is the equation-conversion module transforming S-parameter to Z-parameter. The transformer module $H_{S\rightarrow Y}$, which converts from S-Parameter to Y-Parameter, can be expressed as:

$$\begin{cases} Y_{11} &= Y_0 \frac{(1-S_{11})(1+S_{22})+S_{12}S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}, \\ Y_{12} &= Y_0 \frac{-2S_{12}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}, \\ Y_{21} &= Y_0 \frac{-2S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}, \\ Y_{22} &= Y_0 \frac{(1+S_{11})(1-S_{22})+S_{12}S_{21}}{(1+S_{11})(1+S_{22})-S_{12}S_{21}}, \end{cases}$$

where $Y_0 = \frac{1}{Z_0}$ and $Z_0 = 50$. Similarly, the transformer module $H_{S\rightarrow Z}$, which converts the S-Parameter to Z-Parameter, can be expressed as:

$$\begin{cases} Z_{11} &= Z_0 \frac{(1+S_{11})(1-S_{22})+S_{12}S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}, \\ Z_{12} &= Z_0 \frac{2S_{12}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}, \\ Z_{21} &= Z_0 \frac{2S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}, \\ Z_{22} &= Z_0 \frac{(1-S_{11})(1+S_{22})+S_{12}S_{21}}{(1-S_{11})(1-S_{22})-S_{12}S_{21}}, \end{cases}$$

where $Z_0 = 50$.

Our goal is to obtain the network $f_{\boldsymbol{\theta}}$ that predicts the base distributed parameter, i.e., S-parameter, with high accuracy while minimizing the error of other distributed parameters derived from the predicted parameters by the equations beforehand, which can be described as follows:

$$\min_{\theta} \mathcal{L} = ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_S||_2 + ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_{Y\leftarrow S}||_2 + ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_{Z\leftarrow S}||_2, \tag{17}$$

where $\boldsymbol{y}_S, \boldsymbol{y}_Y, \boldsymbol{y}_Z$ are target values of S-Parameter, Y-Parameter, Z-parameter, respectively.

Optimizing the equations-constraint loss can result in an equations-constrained learning process. With the aid of equation-conversion modules, this learning approach enforces the constraint among distributed parameters and thus obtains accurate and consistent predictions, as shown in Figure 7.

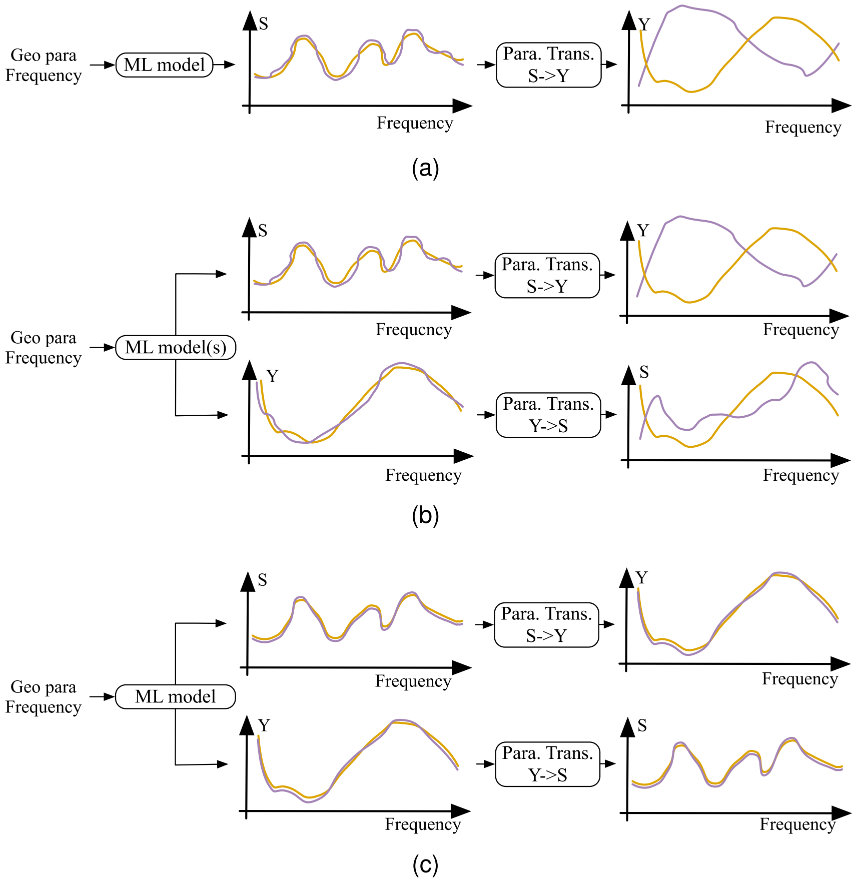

Fig. 7. A comparison illustration of predicted magnitude for different methods: (a) ANN/NeuroTF methods; (b) MTL methods; (c) Our method.

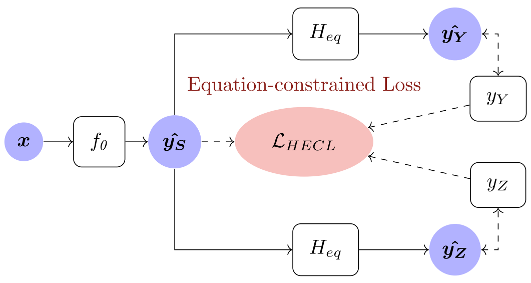

Fig. 8. The overview of our proposed ECL scheme.

3.3.1 Gradients Extraction Process. To perform gradient-based optimization, we need to calculate the gradients of the Equation-contrained Loss function $\mathcal{L}$, as defined in Equation (17), w.r.t. predicted parameters. We use the chain rule to compute the gradients, which have three consecutive parts as follows:

(1) Compute gradient w.r.t. the derived parameter $\hat{\boldsymbol{y}}_Y$ and $\hat{\boldsymbol{y}}_Z$: $\nabla_{\hat{\boldsymbol{y}}_Y}\mathcal{L}$ and $\nabla_{\hat{\boldsymbol{y}}_Z}\mathcal{L}$;
(2) Extract gradients $\nabla_{\hat{\boldsymbol{y}}_Y}\mathcal{L}$ and $\nabla_{\hat{\boldsymbol{y}}_Z}\mathcal{L}$ through the equation transformation module $H_{S\to Y}(\hat{\boldsymbol{y}}_S)$ and $H_{S\to Z}(\hat{\boldsymbol{y}}_Z)$ to obtain the gradient w.r.t. the basis prediction $\nabla_{\hat{\boldsymbol{y}}_S}\mathcal{L}$;
(3) Back-propagate $\nabla_{\hat{\boldsymbol{y}}_S}\mathcal{L}$ through the DeepOTF/ComplexTF model $G_\theta(\cdot)$ to obtain the gradient w.r.t. weights Θ: $\nabla_\Theta\mathcal{L}$.

3.3.2 Dynamic Weighting Strategy. In ECL, we have introduced auxiliary tasks to decrease consistency errors among predictions. However, as informed by the multi-task learning research [27, 28], improving a model's performance may negatively affect performance on a different task with different requirements without adjustment. A suitable loss function must be established to balance the preference between tasks in Equation (17).

The basic method of balancing the multi-objective losses is a weighted linear sum of the losses for each separate task:

$$\mathcal{L}_{\text{total}} = \sum_i \omega_i \mathcal{L}_{\text{total}}. \tag{18}$$

However, we adopt the dynamic weighting technique in Reference [26] by considering the uncertainty of each task. Specifically, our objective function has the following form:

$$\min_\theta \mathcal{L} = \mathcal{L}^1 + \mathcal{L}^2 + \mathcal{L}^3, \quad \text{where} \begin{cases} \mathcal{L}^1 = ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_S||_2 \\ \mathcal{L}^2 = ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_{Y\leftarrow S}||_2 \\ \mathcal{L}^3 = ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_{Z\leftarrow S}||_2 \end{cases}. \tag{19}$$

After introducing additional noise parameter $\boldsymbol{\sigma} = \sigma_1, \sigma_2, \sigma_3$ to capture how much noise we have in the outputs, we then have the following minimization objective for our multi-output model after applying maximum likelihood inference:

$$\min_{\theta,\sigma} \frac{1}{2\sigma_1^2}\mathcal{L}^1_\theta + \frac{1}{2\sigma_2^2}\mathcal{L}^2_\theta + \frac{1}{2\sigma_3^2}\mathcal{L}^3_\theta + \log \sigma_1\sigma_2\sigma_3. \tag{20}$$

The minimization of the loss terms w.r.t. $\boldsymbol{\sigma}$ is to dynamically weigh the loss term $\mathcal{L}^1$, $\mathcal{L}^2$, and $\mathcal{L}^3$. If a specific noise parameter increases, then the weight of the corresponding loss term decreases. Thus, we can apply a dynamic weighting strategy according to each task's uncertainty, i.e., the noise parameter.

4 Experimental Results

We evaluate the performance and properties of the proposed method to answer the following research questions:

(1) How do the proposed ComplexTF and DeepOTF methods perform compared to the classical baseline methods (Section 4.2)?
(2) How does the training speed of DeepOTF compared to the baseline methods (Section 4.3)?
(3) How does the ECL scheme affect other methods (Section 4.4)?
(4) How well does the Dynamic Loss Weighting technique boost the final performance (Section 4.5)?

4.1 Experimental Setup

Enviroment. We implement all experiments based on Python (version 3.9.13) and PyTorch(1.12.1) on a Linux Server with 48 Intel Xeon Silver 4212 cores (2.20 GHz), 1 GeForce RTX 2080 Ti GPU, and a 32 GB main memory.

Table 2. The Statistics of Datasets

	Case					
	1	2	3	4	5	6
Dimensions L (mm)	6	6	6	6	10	16
Dimensions W (mm)	3	3	3	3	5	8
Material Code	HQ	HW	TG	TN	HQ	HQ
Inductance (nH)	[0.3,39]	[0.3,18]	[0.3,39]	[0.6,22]	[0.3,150]	[1,470]
Tolerance (%)	[3, 10]	[3, 10]	[3, 10]	[3, 10]	[3, 10]	[3, 10]

Datasets. Due to significant differences in electronic parameters and geometrical parameters, we restrict our research scope to the prediction of inductors. Inductors are a classic passive two-terminal electrical component of RF circuits and have a wide range of applications. We conduct our experiments on six dataset cases with different dimensions and materials. Table 2 shows all of the geometric parameters and highlights their main characteristics, including the dimensions, the material codes, inductance, and tolerance.

These datasets contain the analysis results of high-frequency chip ceramic inductors with different geometrical and electrical parameters. The sizes of dataset cases are 31,800, 29,401, 32,401, 32,401, 27,001, and 19,801, respectively. All analysis results are measured with the same commercial standard RF impedance material analyzer, where the measured frequency ranges from 10 MHz to 10 GHz.

Methods. We compare our DeepOTF with the following strong baselines: (1) **Single MLP Predictor (MLP)** [29]; (2) **Multi-task Learning with Uncertainty (MTL)** [26], and (3) **NeuroTF Model with Pole-Residue-Based transfer functions (NeuroTF)** [15]. Among all baselines, MLP [29] uses a single Multi-Layer Perceptron as the plain predictor, which takes feature vectors, including geometrical and frequency, as input and outputs the predicted value of each distributed parameter. The pole-residue-based NeuroTF approach [15] is one of the popular NeuroTF approaches that adopts the pole-residue-based technique to address the discontinuity problem. The Multi-task Learning with Uncertainty method [26] is a popular multi-task learning method. In the MTL method, we adopt a share representation among the predictions of different distributed parameters, thus enforcing a soft constraint on those predictions.

Evaluation Metrics. The proposed approaches aim to boost the quality of the learned predictions for distributed parameters. We adopt the ***Sum of Absolute Errors (SAE)*** and ***Absolute Error (AE)*** to measure the model performance and efficiency, respectively. The SAE metric is defined as the summation of the prediction error and the consistency errors:

$$\begin{aligned} SAE_S &= ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_S||_1 + ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_{Y\leftarrow S}||_1 + ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_{Z\leftarrow S}||_1, \\ SAE_Y &= ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_Y||_1 + ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_{S\leftarrow Y}||_1 + ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_{Z\leftarrow Y}||_1, \\ SAE_Z &= ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_Z||_1 + ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_{S\leftarrow Z}||_1 + ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_{Y\leftarrow Z}||_1. \end{aligned} \tag{21}$$

The AE metric is defined as the absolute distance of the prediction error or consistency error:

$$\begin{aligned} AE_S = ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_S||_1, AE_Y = ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_Y||_1, \\ AE_Z = ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_Z||_1, AE_{S\leftarrow\cdot} = ||\boldsymbol{y}_S - \hat{\boldsymbol{y}}_{S\leftarrow\cdot}||_1, \\ AE_{Y\leftarrow\cdot} = ||\boldsymbol{y}_Y - \hat{\boldsymbol{y}}_{Y\leftarrow\cdot}||_1, AE_{Z\leftarrow\cdot} = ||\boldsymbol{y}_Z - \hat{\boldsymbol{y}}_{Z\leftarrow\cdot}||_1. \end{aligned} \tag{22}$$

The reported results are the average of the top-1 results in 3 runs from different random seeds. DeepOTF and MTL are trained with 50 epochs to obtain the general predictor for all distributed parameters. MLP and NeuroTF are trained with 50 epochs for obtaining the separate predictors of each distributed parameter.

Table 3. The Comparisons between Baseline Models and the Proposed Method

	MLP [29]			NeuroTF [15]			MTL [26]			ComplexTF (ours)	DeepOTF (ours)
Case	SAE (S)	SAE (Y)	SAE (Z)	SAE (S)	SAE (Y)	SAE (Z)	SAE (S)	SAE (Y)	SAE (Z)	SAE (S/Y/Z)	SAE (S/Y/Z)
1	1,173.43	68.52	7.63	520.06	1,328.70	88.67	71.61	4.40	8.98	**4.32**	**2.85**
2	1,263.73	71.17	8.72	335.21	2,530.98	92.22	67.58	3.91	9.11	**4.11**	**2.84**
3	1,115.61	77.33	7.28	409.45	5,308.14	89.63	72.04	4.47	8.81	**4.22**	**3.10**
4	1,381.91	56.00	6.72	856.16	572.35	87.99	23.78	5.26	7.87	**4.20**	**4.25**
5	459.69	59.56	8.52	204.96	3,706.76	50.32	42.53	3.86	4.77	**3.02**	**2.14**
6	247.21	53.87	7.65	189.24	1,542.85	31.39	47.22	3.27	4.73	**2.43**	**3.16**

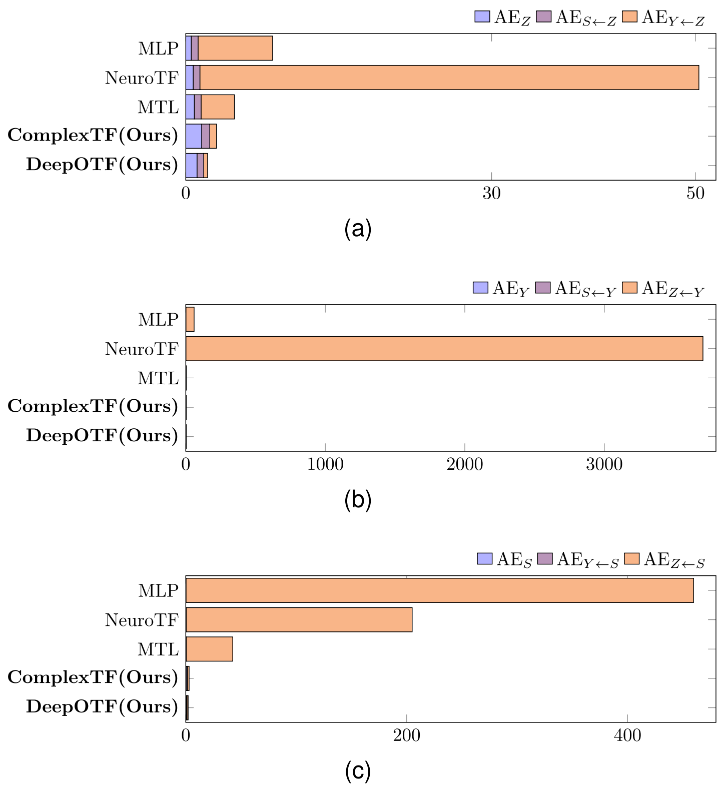

Fig. 9. The analysis of the prediction results of different methods on dataset Case5. (a) Each item in SAE ($_Z$); (b) Each item in SAE ($_Y$); (c) Each item in SAE ($_S$).

4.2 Comparison Results

The results in Table 3 and Figure 9 show our DeepOTF attains superior performance among all baselines regarding *accuracy*. More specifically, we observe the following:

— In terms of model performance, our DeepOTF beats all baselines. It attains up to 25× improvement over the best baseline, i.e., MTL, while attaining up to $1,700\times$ improvement over the NeuroTF method. Our equations-constraint learning scheme enforces consistency among the outputs to make distributed parameters consistent and reliable predictions instead of biased learning in others.

— From Figure 9, it can be seen that consistency errors take up a large percentage of the overall errors, especially for $AE_{Y\leftarrow Z}$, $AE_{Z\leftarrow Y}$, $AE_{Z\leftarrow S}$. Although the prediction error of MLP and NeuroTF are acceptable, their consistency error is unbearable. It also explains why it performed the

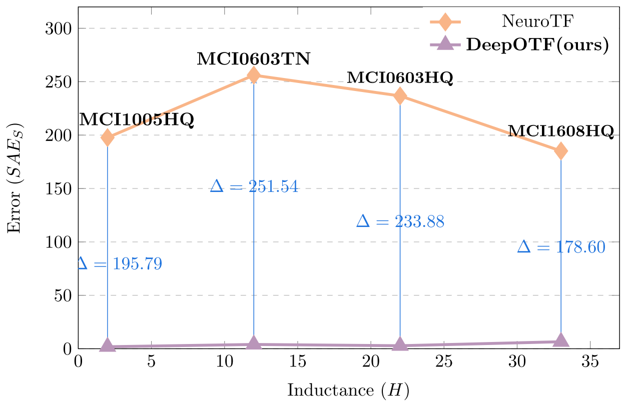

Fig. 10. Testing errors of DeepOTF trained to learn the optimal transfer function compared with NeuralTF [15]; MCI1005HQ, MCI0603TN, MCI0603HQ, MCI1608HQ are different inductors from four datasets cases 1, 4, 5, 6, respectively).

worst across all methods. On the contrary, our method has the same errors for each item, because it maintains consistency among the predicted values. Therefore, our method obtains the best performance.

— Regarding the performance details of the testing on various devices shown in Figure 10, we selected four different devices (MCI1005HQ, MCI0603TN, MCI0603HQ, MCI1608HQ) from four datasets (case1, case4, case5, case6), respectively and make detailed analysis. The NeuroTF and our proposed method are compared separately, and for each different device, our method demonstrates a prediction accuracy that far exceeds that of NeuroTF, and the prediction is at most 200 times better than that for NeuroTF.
— We notice that MTL is considerably better than other baseline methods, which corroborates that the equations-constrained scheme is preferred. However, these methods rely on the shared representation of predictions and thus fail to keep the consistency between predictions due to the high complexity of the optimization process.

4.3 Comparison between DeepOTF Predictor and NeuroTF Predictor

To test our proposed model's generalization ability and robustness against the NeuroTF method, we performed an experiment on their performance comparison, in which we implemented a DeepONet predictor without ECL, shorted as DeepOTF*. Experiments were conducted on six dataset cases to compare these two methods, and the experimental results are summarized in Table 4.

The table shows that our proposed framework shows substantial performance superiority over the baseline NeuroTF method across all the cases. Remarkably, without elaborately designed transfer functions, our method, i.e., DeepOTF, attains near 1.5× improvement in terms of AE_S. This comparison verifies the superior generalization ability and robustness of our proposed DeepONet on different data. We also compare the number of parameters and inference time of the two methods. Regarding efficiency, our DeepOTF outperforms the NeuroTF method, around 5% faster than NeuroTF in training time. It is worth noting that, even though the NeuroTF method has fewer parameters, the reason for the increase in inference time is that the computation of the transfer function in NeuroTF takes more time compared to the matrix operations in DeepOTF.

Without the heavy computation of the transfer function, our DeepOFT method has a noticeable advantage in reducing the computation cost.

Table 4. The Analysis of the NeuroTF Method and the DeepOTF Method

	NeuroTF [15]			DeepOTF* (ours)		
Case	AE (S)	Inference (s)	#Params (M)	AE (S)	Inference (s)	#Params (M)
1	0.3082	0.2986	9.616	**0.1944**	**0.2843**	18.064
2	0.3087	0.2922	9.616	**0.1831**	**0.2869**	18.064
3	0.3115	0.2935	9.616	**0.1870**	**0.2851**	18.064
4	0.3169	0.2924	9.616	**0.1830**	**0.2865**	18.064
5	0.3127	0.2909	9.616	**0.2082**	**0.2865**	18.064
6	0.3024	0.2961	9.616	**0.1890**	**0.2842**	18.064

Table 5. Comparison between MLP and MLP* (w/o ECL Scheme)

	MLP [29]			MLP* (ours)		
Case	SAE (S)	SAE (Y)	SAE (Z)	SAE (S)	SAE (Y)	SAE (Z)
1	1,173.43	68.52	7.63	**2.96**	**2.96**	**2.96**
2	1,263.73	71.17	8.72	**2.91**	**2.91**	**2.91**
3	1,115.61	77.33	7.28	**3.14**	**3.14**	**3.14**
4	1,381.91	56.00	6.72	**3.10**	**3.10**	**3.10**
5	459.69	59.56	8.52	**7.40**	**7.40**	**7.40**
6	247.21	53.87	7.65	**8.67**	**8.67**	**8.67**

4.4 Effectiveness of Equations-constrained Scheme

We evaluate our equations-constrained scheme's effectiveness with other predictors to further validate it. Since the MLP method is the basic method, we test the equations-constrained scheme on MLP as MLP*, i.e., MLP with the equations-constrained scheme. Results in Table 5 show that MLP* outperforms the one without the equations-constrained scheme.

This result implies that we can improve other prediction methods' performance by simply adopting the equations-constrained scheme into their learning process.

4.5 Effectiveness of Dynamic Weighting Strategy

Recall that we adopt a dynamic loss weighting strategy in Section 3.3.2 and hope to strike a compromise between prediction error and consistent error. A set of ablation experiments is conducted on the Case 6 dataset to investigate the effect of dynamic loss weighting. We evaluate the fixed weighting version of our DeepOTF method as DeepOTF$_{\text{fixed}}$, i.e., DeepOTF without the dynamic loss weighting strategy.

Figure 11 shows that the DeepOTF outperforms the DeepOTF$_{\text{fixed}}$, which identifies the importance of the dynamic weighting strategy.

4.6 Experiments on the Solenoidal Inductor Dataset

To further validate the effectiveness of our proposed method, we use a more complex case to demonstrate its efficacy. In this case, we utilized the experimental dataset from Reference [30] for a solenoidal inductor using a **Nickel-Zinc (NiZn)** ferrite magnetic integrated on the top metal layer of a silicon interposer-based 2.5D heterogeneously integrated system score. The geometry of the inductor is defined by eight parameters, and the corresponding ranges for each parameter are shown in Table 6.

After sampling 1,000 instances using **Latin Hypercube Sampling (LHS)**, the measured complex permeability of the NiZn core [31] was imported into the full-wave electromagnetic solver

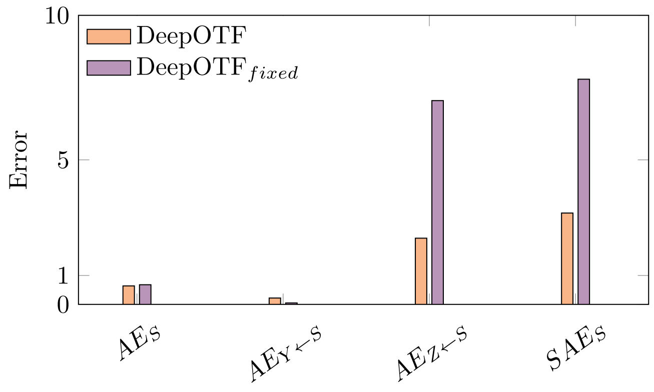

Fig. 11. Ablation study on dynamic weighting strategy.

Table 6. Control Parameters of Solenoidal Inductor

Parameter		Unit	Min	Max
Gap between windings	g	mil	2	20
Number of windings	N		3	13
Size of via	s_r	μm	S0	103
Copper Trace Width	w_c	mil	2	20
Copper Thickness Bottom	t_{ch}	μm	35	170
Copper Thickness Top	$t_{L,t}$	μm	35	170
Magnetic Core Thickness	4	μm	S0	630
Magnetic Core Width	w_d	μm	50	350

Table 7. The Comparisons between the NeuroTF Model and the Proposed Method on the Solenoidal Inductor Dataset

	NeuroTF [15]	ComplexTF (ours)	DeepOTF (ours)
SAE (S)	121.07	**13.10**	68.32
SAE (Y)	201.81	**13.10**	68.32
SAE (Z)	203.33	**13.10**	68.32

Ansys HFSS [32] to simulate the EM parameters at 200 frequency points ranging from 10 MHz to 500 MHz. In the training stage, 800 out of the 1,000 samples served as training data for the experiment.

Based on the experimental results, we can see that ComplexTF and DeepOTF have significant advantages over NeuroTF on the more complex Solenoidal Inductor Dataset (Table 7). On the Solenoidal Inductor Dataset, ComplexTF achieves SAE (S), SAE (Y), and SAE (Z) values of 13.10, which are much lower than NeuroTF's values of 121.07, 201.81, and 203.33, respectively. DeepOTF also achieves SAE (S), SAE (Y), and SAE (Z) values of 68.32, which is also lower than NeuroTF's values. This indicates that ComplexTF can more accurately predict and model the outputs consistently, resulting in lower consistent errors. Furthermore, DeepOTF and ComplexTF obtain the same low error value of 13.10 across all three metrics, demonstrating better stability and consistency. In contrast, NeuroTF exhibits significant variations in transformation errors across different outputs.

Our DeepOTF and ComplexTF perform better on the complex dataset, indicating their ability to handle different types of input data effectively. In comparison, NeuroTF shows significantly higher errors on this dataset, suggesting limited modeling capability for complex datasets.

5 Conclusion

In this work, we presented the DeepOTF framework, a novel framework that integrates recent neuro-operator learning advances with physical relationships embedded for EM behavior prediction. Our DeepOTF framework consists of a customized deep operator network as a predictor, an equations-constrained learning scheme, and a dynamic weighting strategy to enable the accurate and consistent prediction of different distributed parameters of passive devices. Extensive experiments demonstrate that our DeepOTF framework can achieve higher accuracy (up to $1,700\times$) than the baseline methods.

With the help of DeepOTF, we can provide faster and more accurate answers for time-consuming electromagnetic simulations and analysis.

References

[1] Yang-Ki Hong and Jaejin Lee. 2013. Ferrites for RF passive devices. In *Solid State Physics*. Vol. 64. Elsevier, 237–329.

[2] A. S. Royet, J. P. Michel, B. Reig, J. L. Pornin, M. Ranaivoniarivo, B. Robain, P. de Person, and G. Uren. 2016. Design of optimized high Q inductors on SOI substrates for RF ICs. In *IEEE International Conference on Electronics Circuits and Systems (ICECS'16)*. 324–327.

[3] Seok Bae, Yang-Ki Hong, Jae-Jin Lee, Won-Mo Seong, Jun-Sig Kum, Won-Ki Ahn, Sang-Hoon Park, Gavin S. Abo, Jeevan Jalli, and Ji-Hoon Park. 2010. Miniaturized broadband ferrite T-DMB antenna for mobile-phone applications. *IEEE Trans. Magnet.* 46, 6 (2010), 2361–2364.

[4] I. Alexander and F. Aitken. 1975. Broadband VHF/UHF circulators with tailored filter characteristics. *IEEE Trans. Magnet.* 11, 5 (1975), 1267–1269.

[5] Stéphane Capraro, Thomas Rouiller, Martine Le Berre, Jean-Pierre Chatelon, Bernard Bayard, Daniel Barbier, and Jean Jacques Rousseau. 2007. Feasibility of an integrated self biased coplanar isolator with barium ferrite films. *IEEE Trans. Compon. Packag. Technol.* 30, 3 (2007), 411–415.

[6] Hsiao-Miin Sung, Chi-Jen Chen, Wen-Song Ko, and Hong-Ching Lin. 1994. Fine powder ferrite for multilayer chip inductors. *IEEE Trans. Magnet.* 30, 6 (1994), 4906–4908.

[7] G. K. L. Wong, Myeong Soo Kang, H. W. Lee, F. Biancalana, C. Conti, T. Weiss, and P. St J. Russell. 2012. Excitation of orbital angular momentum resonances in helically twisted photonic crystal fiber. *Science* 337, 6093 (2012), 446–449.

[8] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic analysis: Concrete results. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES'11)*. 251–261.

[9] Jiefu Chen and Qing Huo Liu. 2012. Discontinuous Galerkin time-domain methods for multiscale electromagnetic simulations: A review. *Proc. IEEE* 101, 2 (2012), 242–254.

[10] James C. Rautio. 2007. Synthesis of compact lumped models from electromagnetic analysis results. *Trans. Microw. Theor. Techniq.* 55, 12 (2007), 2548–2554.

[11] Sayed Alireza Sadrossadat, Yazi Cao, and Qi-Jun Zhang. 2013. Parametric modeling of microwave passive components using sensitivity-analysis-based adjoint neural-network technique. *Trans. Microw. Theor. Techniq.* 61, 5 (2013), 1733–1747.

[12] Vittorio Rizzoli, Alessandra Costanzo, Diego Masotti, Alessandro Lipparini, and Franco Mastri. 2004. Computer-aided optimization of nonlinear microwave circuits with the aid of electromagnetic simulation. *Trans. Microw. Theor. Techniq.* 52, 1 (2004), 362–377.

[13] Michael B. Steer, John W. Bandler, and Christopher M. Snowden. 2002. Computer-aided design of RF and microwave circuits and systems. *Trans. Microw. Theor. Techniq.* 50, 3 (2002), 996–1005.

[14] Yazi Cao, Gaofeng Wang, and Qi-Jun Zhang. 2009. A new training approach for parametric modeling of microwave passive components using combined neural networks and transfer functions. *IEEE Trans. Microw. Theor. Techniq.* 57, 11 (2009), 2727–2742.

[15] Feng Feng, Chao Zhang, Jianguo Ma, and Qi jun Zhang. 2016. Parametric modeling of EM behavior of microwave components using combined neural networks and pole-residue-based transfer functions. *Trans. Microw. Theor. Techniq.* 64 (2016), 60–77.

[16] Zhihao Zhao, Feng Feng, Wei Zhang, Jianan Zhang, Jing Jin, and Qi-Jun Zhang. 2020. Parametric modeling of EM behavior of microwave components using combined neural networks and hybrid-based transfer functions. *IEEE Access* 8 (2020), 93922–93938.

[17] Zhiyu Guo, Jianjun Gao, Yazi Cao, and Qi-jun Zhang. 2012. Passivity enforcement for passive component modeling subject to variations of geometrical parameters using neural networks. In *IEEE International Microwave Symposium (IMS'12)*. 1–3.

[18] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* 3, 3 (2021), 218–229.

[19] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2021. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR'21)*.

[20] Anima Anandkumar, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Nikola Kovachki, Zongyi Li, Burigede Liu, and Andrew Stuart. 2020. Neural operator: Graph kernel network for partial differential equations. In *International Conference on Learning Representations (ICLR'20)*.

[21] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew M. Stuart, Kaushik Bhattacharya, and Anima Anandkumar. 2020. Multipole graph neural operator for parametric partial differential equations. In *Annual Conference on Neural Information Processing Systems (NIPS'20)*.

[22] Lizuo Liu, Kamaljyoti Nath, and Wei Cai. 2022. A causality-DeepONet for causal responses of linear dynamical systems. *arXiv preprint arXiv:2209.08397* (2022).

[23] Hakki Mert Torun, Ahmet Cemal Durgun, Kemal Aygün, and Madhavan Swaminathan. 2020. Causal and passive parameterization of S-parameters using neural networks. *Trans. Microw. Theor. Techniq.* 68, 10 (2020), 4290–4304.

[24] Jorma Kaarlo Merikoski, Humberto Sarria, and Pablo Tarazaga. 1994. Bounds for singular values using traces. *Linear Algeb. Appl.* 210 (1994), 227–254.

[25] Piero Triverio. 2013. Robust causality check for sampled scattering parameters via a filtered Fourier transform. *IEEE Microw. Wirel. Compon. Lett.* 24, 2 (2013), 72–74.

[26] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. 7482–7491.

[27] Yu Zhang and Qiang Yang. 2022. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.* 34, 12 (2022), 5586–5609. DOI: http://dx.doi.org/10.1109/TKDE.2021.3070203

[28] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[29] Feng Feng, Weicong Na, Jing Jin, Wei Zhang, and Qi-Jun Zhang. 2021. ANNs for fast parameterized EM modeling: The state of the art in machine learning for design automation of passive microwave structures. *IEEE Microw. Mag.* 22, 10 (2021), 37–50.

[30] Hakki Mert Torun, Huan Yu, Nihar Dasari, Venkata Chaitanya Krishna Chekuri, Arvind Singh, Jinwoo Kim, Sung Kyu Lim, Saibal Mukhopadhyay, and Madhavan Swaminathan. 2019. A spectral convolutional net for co-optimization of integrated voltage regulators and embedded inductors. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'19)*. 1–8.

[31] M. L. F. Bellaredj, S. Mueller, A. K. Davis, P. Kohl, M. Swaminathan, and Y. Mano. 2017. Fabrication, characterization and comparison of FR4-compatible composite magnetic materials for high efficiency integrated voltage regulators with embedded magnetic core micro-inductors. In *IEEE Electronic Components and Technology Conference*. 2008–2014.

[32] Erdogan Madenci and Ibrahim Guven. 2015. *The Finite Element Method and Applications in Engineering using ANSYS®*. Springer.

Received 18 September 2023; revised 11 January 2024; accepted 15 April 2024