Deep Neural Networks for Rapid Simulation of Planar Microwave Circuits Based on Their Layouts

Shutong Qi\textsuperscript{a}, Graduate Student Member, IEEE, and Costas D. Sarris\textsuperscript{b}, Senior Member, IEEE

Abstract—This article demonstrates a deep learning (DL)-based methodology for the rapid simulation of planar microwave circuits based on their layouts. We train convolutional neural networks (CNNs) to compute the scattering parameters of general, two-port circuits consisting of a metallization layer printed on a grounded dielectric substrate, by processing the metallization pattern along with the thickness and dielectric permittivity of the substrate. This approach harnesses the efficiency of CNNs with pattern recognition tasks and extends previous efforts to employ neural networks for the simulation of parameterized circuit geometries. Furthermore, we integrate this CNN in a hybrid network with a long-short term memory (LSTM) module that uses coarse mesh finite-difference time-domain (FDTD) simulation data as an additional input. We show that this hybrid network is computationally efficient and generalizable, accurately modeling geometries well beyond those that the network has previously seen.

Index Terms—Artificial neural networks, finite difference methods, microwave circuits.

I. INTRODUCTION

D eepl learning (DL) techniques have shown great potential in the area of computational physics for solving partial differential equations (PDEs) [1] and accelerating traditional numerical methods [2]. For microwave circuit modeling and design, DL has also been used for linear and nonlinear microwave component modeling [3], [4], [5], [6], [7], [8], [9], [10] and optimization. In these applications, neural networks are trained to map a set of input variables to a desired output function of interest. Hence, trained networks act as inexpensive surrogates of complex circuits and system models. These can dramatically accelerate design optimization and uncertainty quantification studies [11]. Applications of neural networks to microwave circuit design have focused on models based on input parameters that describe fixed geometries, such as the length of stubs in microstrip filters [12]. More recently, physics-informed neural networks for Maxwell’s equations have been presented [13], aimed at solving simple electromagnetic wave interaction problems.

In this article, we propose a DL-based methodology for the rapid simulation of a wide range of printed microwave circuit geometries based on their layouts. Our method combines a geometry pattern recognition stage that is enabled by a convolutional neural network (CNN) and an error compensation stage that is enabled by a recurrent neural network (RNN) with long-short term memory (LSTM) structure. Fig. 1 shows the flowchart of our approach.

For the CNN-based prediction, the network is constructed with convolutional layers for feature extraction and fully connected layers for data generation. A pattern generator is used to produce various metallization patterns printed on grounded dielectric substrates of varying height and permittivity. These patterns are aimed at “teaching” the solution of general printed circuit board (PCB) problems to the CNN. Unlike previous work in this area, our approach does not rely on the parameterization of a specific class of geometries. Therefore, it is applicable to a wide range of PCB problems. “Ground truth” S-parameter data for training and testing the CNN are produced by full-wave, finite-difference time-domain (FDTD) simulations. These simulations are computationally expensive, because of the requirement that the Yee cell size should be kept electrically small and typically smaller than one-tenth of a wavelength [14]. Coarse mesh FDTD simulations are faster, yet corrupted by pronounced numerical errors. This limitation has been addressed with various error compensation methods [15] and with the formulation of high-order finite difference schemes, such as the scaling multi-resolution time domain (S-MRTD) scheme of [16].

We propose a powerful route to numerical error compensation in FDTD, building a hybrid neural network consisting of a CNN and an RNN with LSTM structure. The hybrid network is trained with coarse and dense mesh FDTD simulations, with the goal of “learning” the pattern of numerical errors by comparing solutions of various problems at coarse and dense grids. This ultimately enables the fast generation of high fidelity data from coarse mesh FDTD simulations. We demonstrate that the hybrid CNN/LSTM structure we introduce in this article (in detail, for the first time) has significant accuracy and efficiency advantages. In particular, the proposed hybrid network is more lightweight and faster to train than a standalone CNN, yet it achieves excellent accuracy in S-parameter prediction for a wide range of structures, well beyond those included in the training set.

A brief introduction of the proposed hybrid structure has been included in [17]. This article provides a comprehensive presentation of our methodology. Moreover, we demonstrate...
the accurate prediction of complex S-parameters (magnitude and phase) and discuss the computational savings that have been achieved.

The rest of the article is organized as follows. In Section II, we present the CNN part of our hybrid network that infers the S-parameters of the two-port, planar microwave circuits from their layouts. We explain the structure and hyperparameters of the network we use for this task and an adaptive sub-division scheme applied to the simulated frequency band to improve the accuracy of the network. In Section III, we demonstrate the accuracy of this approach for a wide range of microwave circuit geometries, instead of just a specific class of parameterized geometries. In Section IV, we embed this CNN in a hybrid network that leverages an LSTM structure to significantly advance the efficiency of the standalone CNN. The results of this hybrid network are shown in Section V.

II. CNN-BASED COMPUTATION OF S-PARAMETERS OF PLANAR MICROWAVE TWO-PORT CIRCUITS: METHOD

We first consider the direct computation of the S-parameters of planar, two-port circuits from their layouts. Given the layouts of the circuits along with the substrate thickness and permittivity, the trained networks are expected to produce the S-parameters, over a targeted frequency band. In the following, we explain the structure of the network and introduce an adaptive sub-division scheme for the simulated bandwidth, to facilitate training and improve the accuracy of the network.

A. Proposed Network Structure

To obtain accurate S-parameters from an input metallization pattern, the architecture of a data-dependent neural network has to be carefully designed. In this work, we use convolutional layers to transform the metallization pattern of a planar microwave circuit into high-dimensional features, along with fully connected layers to map these features to S-parameters. These two DL architectures are discussed in the following.

1) Fully Connected Neural Network: A neuron is the fundamental unit of a DL model. In a fully connected neural network, the number of neurons in each layer is different; neurons in one layer are connected to all neurons in the next layer. Let $x_i \in \mathbb{R}^m$ be the $i$th input of the fully connected layer and $y_i \in \mathbb{R}^n$ be the $i$th output. The mapping of $x_i$ to $y_i$ can be expressed as

$$y_i = f\left(\sum_i w_i x_i + b\right)$$

where $w_i$ is the weight for the $i$th connection of $x_i$ to $y_i$, $b$ is the bias set to improve network learning, and $f(x)$ is a nonlinear activation function. Notably, the fully connected structure fits with discrete data well. In this work, FDTD is used to generate multiple frequency samples of scattering parameters in a single run for every geometry. The process of S-parameter generation can be replaced by the fully connected structure.

2) Convolutional Neural Network: Unlike fully connected layers, a convolutional layer is formulated by locally weighted summations of the previous layer, as shown in Fig. 2. For a single weighted calculation, the inner product between one local batch and a convolution filter is performed, followed by a nonlinear activation function [18], [19]. We set $X^{(l-1)}(m \times m)$ as the input of the $(l-1)$th layer, and $K^{(l)}(n \times n)$ as the
corresponding filter kernel. The output of this layer, \( Z^{(l)}(m-n+1) \times (m-n+1) \), consists of elements

\[
z_{u,v}^{(l)} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x_{i+u,j+v}^{(l-1)} \cdot K_{i,j}^{(l)} \cdot \chi(i,j) + b^{(l)} \quad (2)
\]

After a convolution operation, the output also needs to be processed by the activation layer as

\[
a_{u,v}^{(l)} = g(z_{u,v}^{(l)}) \quad (4)
\]

where \( g(x) \) is a nonlinear activation function.

Generally, the convolution filter is smaller than the size of the feature map. It incorporates a matrix of trainable weights, which are shared by two locally connected layers, allowing the network to include more layers without the burden to store a large number of parameters. With these features, the CNN is very efficient in processing data structures with large size and multiple dimensions, such as 2-D and 3-D images. Therefore, in our proposed network, convolutional layers are applied to treat the geometry and substrate thickness of the given planar circuit layouts.

To directly predict S-parameters from their layouts and substrate properties, several convolutional layers are used to extract the features of the input pattern. Then, a dimension conversion function transforms the 2-D features to 1-D. Finally, fully connected layers generate the S-parameters based on these 1-D features. This procedure can be expressed as

\[
a_{\text{feature1}}(x) = \text{CNN}(x)
\]

\[
a_{\text{feature2}}(x') = \text{TRANS}(x')
\]

\[
g_{\text{S-parameter}}(y) = \text{ANN}(y)
\]

where CNN, TRANS, and ANN represent the convolutional layers, dimension conversion function, and the fully connected layers in the network, respectively. The structure of our network is shown in Fig. 3.

3) Number of Convolutional Layers: The number of convolutional layers is of great importance in DL. If the number of layers is small, the network does not adequately learn from the layouts of the circuits and substrate properties in the training data. On the other hand, too many layers will lead to overfitting, which should also be avoided. Since overfitting can be observed at the beginning of the training process, we use pre-tests to determine the appropriate number of layers. Initially, we train a small number of convolutional layers for 100 rounds, checking for overfitting. We continue adding layers until the loss trend shows that the network is overfitted. If there are \( N \) layers in the network where overfitting is detected, we use \( N-1 \) layers in the final network.

B. Adaptive Sub-Division of Simulated Frequency Band

To improve the accuracy of our results, we recursively divide the simulated frequency band according to the mean square error (MSE) of the network. This approach parallels the sub-division of the range of training parameters of the artificial neural network for microwave filter modeling presented in [20]. In particular, we set an accuracy criterion of the form

\[
C : M_{i,j} < \varepsilon \quad (6)
\]

where \( M_{i,j} \) represents the MSE for the sub-band \([F_i, F_j]\) and \( \varepsilon \) is the threshold we set. We start by training a single neural network for the whole frequency band \([0, F_{\text{max}}]\) GHz. If (6) is met by this single network, the training is terminated. If not, the frequency band \([0, F_{\text{max}}]\) GHz is divided in two equal sub-bands \([0, F_m]\) and \([F_m, F_{\text{max}}]\) GHz, where \( F_m = F_{\text{max}}/2 \). Two sub-networks are trained for each sub-band individually. After checking the MSE of these two sub-networks, we continue the frequency-division process on the sub-network that fails to meet (6) recursively, until the required criterion is achieved. Finally, after \( k \) divisions, all sub-bands also meet (6). This recursive process is expressed as a pseudo-code in Algorithm 1. Since the input of each sub-network is the same 2-D pixel basis planar circuit layout and all the sub-networks have the same structure except the output dimension, they can be trained in parallel. The outputs from each sub-network are assembled to form the final result. To keep the final results continuous, we take the average of two adjacent predictions as the final result for the boundary points of adjacent sub-bands. In this way, there is still one input and one output for each case, while the accuracy of the trained network is significantly improved.
### Algorithm 1 ADAPTIVE FREQUENCY BAND SUB-DIVISION

**Input:** $[0, F_{\text{max}}]$: the whole frequency band; $\varepsilon$: threshold;  
**Output:** $[0, F_1], [F_1, F_2], \ldots, [F_{n-1}, F_{\text{max}}]$: divided sub-bands; $Net_{0,1}, Net_{1,2}, \ldots, Net_{n-1,n}$: trained networks for the corresponding sub-bands;

```plaintext
function FREQUENCY_DIVISION($F_i, F_j, \varepsilon$) 
    Train the network $Net_{i,j}$ for frequency band $[F_i, F_j]$;  
    Compute $MSE$ of $Net_{i,j}$;  
    if $MSE < \varepsilon$ then  
        return $Net_{i,j}, [F_i, F_j]$  
    else  
        Divide $[F_i, F_j]$ in two equal frequency band $[F_i, F_j], [F_j, F_{max}]$, $F_t = (F_i + F_j)/2$;  
        FREQUENCY_DIVISION($[F_i, F_t], \varepsilon$);  
        FREQUENCY_DIVISION($[F_t, F_j], \varepsilon$);  
    end if  
end function
```

**FREQUENCY DIVISION**($[0, F_{\text{max}}], \varepsilon$);

---

**III. CNN-BASED PREDICTION OF S-PARAMETERS OF PLANAR MICROWAVE TWO-PORT CIRCUITS: NUMERICAL RESULTS**

#### A. Data Format

In this study, the FDTD method is applied to generate ground-truth data. The S-parameters are extracted by FFT as in [21]. The FDTD computation region is partitioned into $100 \times 100 \times 16$ cells of dimensions $\Delta x = 0.406$ mm, $\Delta y = 0.423$ mm. The thickness of the substrate is three cells. To generalize the substrate thickness with respect to frequency, we introduce a variable $h' = h/\lambda_{\text{min}}$ to represent the relative substrate thickness, where $h$ is the substrate thickness and $\lambda_{\text{min}}$ is the shortest simulated wavelength. The simulated frequency band is 0–20 GHz. Besides, the relative permittivity of the substrate varies from 2 to 5, and the relative substrate thickness ranges from 0.05 to 0.1. Therefore, $\Delta z$ varies from 0.112 to 0.354 mm. Fig. 4 shows the details of the simulation region, excluding the air-filled area.

The inputs of the network are layouts of various common microwave circuits, as opposed to a single parameterized geometry. To demonstrate our approach, we include generic stub and stepped-impedance filters, along with radial stubs in our training and testing cases. Two matrices are used to represent an input dataset. The first matrix contains the metallization area and the relative permittivity of the substrate, as shown within the dashed line in Fig. 4. The second matrix provides the relative thickness of the substrate $h'$. For example, the relative permittivity of the substrate is 5 and the thickness is 0.603 mm. The first matrix has entries that show whether a pixel on the metallization layer is covered by metal or not. In this case, if the pixel is covered by metal, the entry is set to 0, otherwise the entry is set to 5. The second matrix has constant entries of the relative thickness. Since the maximum frequency is 20 GHz, the smallest simulated wavelength is 6.7 mm in the dielectric. If the actual thickness of the PCB is 0.603 mm, the relative thickness is 0.09.

To fully exploit the performance of the network, we gradually increase the number of data from 1000 to 5000 to 8000 to 10 000 in each topology. We found that the network converges well without overfitting till 8000 samples. There is no difference between 8000 samples and 10 000 samples in terms of training and testing loss. Therefore, we conclude that 10 000 samples are sufficient to represent one topology. About 80% of them are randomly selected for training the network and the rest are used to test the performance of the network. The output of the network is S-parameters. In this work, the 0–20 GHz frequency band is represented by 120 points uniformly.

#### B. Model Parameters

1) **Convolutional Layers:** Following the method we used to choose the number of convolutional layers in Section II, we determined that seven layers were sufficient in our final model. Fig. 5 shows the model parameters in detail. The input layers are connected with convolutional layers. There are max-pooling layers after the first two convolutional layers to reduce the size of the inputs. The kernel size for max-pooling is $2 \times 2$ and the stride is 2. The kernel size is $7 \times 7$ for the first convolution layer, and $5 \times 5$ for the second convolution layer. These large kernels are used to collect overall features from the inputs. The following five convolutional layers have $3 \times 3$ kernels which capture the details of the input. After each convolutional layer, there is a rectified linear unit (ReLU) layer to increase the nonlinear capabilities of the network.

---

Fig. 4. Details of the FDTD simulation domain.

Fig. 5. Model parameters for CNN-based prediction.
After seven convolutional layers, the inputs are transformed into $12 \times 12 \times 512$ element arrays.

2) Fully Connected Layers: A dimension conversion function maps the 3-D outputs of convolutional layers into 1-D. Therefore, the input of the first fully connected layer has dimensions $73728 \times 1$. After each fully connected layer, there is a nonlinear ReLU layer and a dropout layer. Dropout layers [22] are included to prevent overfitting in the training process. The percentage of dropout is set to be 0.5. The output dimension equals the number of points in the corresponding sub-band, which depends on how the sub-bands are divided.

C. Numerical Results

After determining the structure of the proposed network and the size of the dataset, we begin with training for the multi-stub filters first. However, it is observed that the generalization ability of the trained network is limited. The network trained with multi-stub filters cannot predict the S-parameters of stepped impedance filters or radial stubs accurately. To address this problem, we can increase the complexity of the network and train all the data in one network. Alternatively, we can also develop a sub-model for each topology and combine them in parallel with a classifier identifying the kind of topology. We choose the latter method for the following two advantages. First, it is easier to train compared to building a more complex network, because the network only updates the part of neurons that are responsible for one topology in the training process. Second, it can be readily extended to new topologies. For a new topology, we only need to train a sub-model and retrain the classifier. There is no need to retrain the entire network. In this work, the classifier is a CNN with three layers. It is trained with 300 layouts (100 for each category) and tested with 2700 layouts (900 for each topology). The trained classifier can achieve an accuracy of 100% for these three topologies on both datasets. It takes 198.12 s to finish training the classifier on an Intel i5 CPU.

All the training and testing has been performed on a Compute Canada server with AMD Milan 7413 @ 2.65 GHz CPU and NVidia A100 GPU. Training is terminated when

\[ \text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 < 0.002 \]  

where $N$ is the number of points in the corresponding sub-band, $y_i$ is the ground truth of $|S_{21}|$ and $\hat{y}_i$ is the predicted $|S_{21}|$. If this condition is not met after 800 rounds, the training will be terminated by default. Following the frequency division method stated in Section II, we first train a single network for the whole frequency band (0–20 GHz). The MSE is 0.0087 for the entire frequency band. Then, the simulated band is split in two and two networks are trained separately for the two new sub-bands. The MSE is 0.0010 for the 0–10 GHz band and 0.0065 for the 10–20 GHz band. Since (7) is not met for the 10–20 GHz band, another round of frequency band division is performed. After three divisions, all sub-bands meet (7). At this time, the whole frequency band is divided into four parts (0–10, 10–15, 15–17.5, and 17.5–20 GHz). The history of the MSE at each sub-band is shown in Table I. For the whole frequency band, the MSE is 0.0014 for the training data and 0.0015 for the testing data. For more complex cases or a stricter threshold, more sub-bands may be needed. Notably, the proposed method can be adapted to any given frequency band and accuracy threshold. The similarity between the training and testing MSE shows that the network is well trained. Fig. 6(a)–(e) are testing data comparing the prediction from the network and FDTD results. To further explore the ability of the network to generalize to new geometries, we randomly generate validation cases, which are not included in the training or testing data. Fig. 6(f)–(h) show the prediction results of these validation data by the CNN-based model. From these results, we can see that the trained network works well when the number of stubs increases from 4 to 6. This indicates that the network recognizes patterns rather than specific parameters of the geometry, such as the number of stubs in a filter.

The model is trained with four NVidia A100 GPUs. It takes 241.24 s for one round of training. The dataset is generated with AMD Milan 7413 CPU. Generating a set of data takes 53.74 s in average, while the trained model only takes 2.10 s to infer one set of data.

TABLE I

| Frequency band | Round 0 | Round 1 | Round 2 | Round 3 |
|----------------|---------|---------|---------|---------|
| 0-10 GHz       | 0.0087  | 0.0010  | 0.0010  | 0.0010  |
| 10-15 GHz      | 0.0065  | 0.0012  | 0.0012  |         |
| 15-17.5 GHz    | 0.0045  | 0.0013  |         | 0.0017  |
| 17.5-20 GHz    |         |         |         |         |

IV. INTEGRATION OF CNN IN AN EFFICIENT HYBRID NETWORK: METHOD

The CNN of Section III demonstrated excellent accuracy and generalization ability, but it was computationally expensive to train. This motivates us to integrate the CNN in a hybrid network of improved computational efficiency that can still generalize to new geometries. To this end, we employ the results of coarse mesh FDTD simulations of the geometry of interest, as an additional input, processed by an RNN with LSTM structure. The structure of the resulting hybrid network is discussed in the following.

A. Recurrent Neural Networks

The major shortcoming of fully connected layers is that they treat each input element individually, so they cannot “remember” past inputs. RNNs address this issue by introducing loops, as shown in Fig. 7. In this diagram, a neural network $A$, processes a series of inputs $X_t$, and outputs a value $h_t$. A loop allows information to be passed from one step of the network to the next, preserving the information included in the input data.

A chain of repeating modules exists in all RNNs. In the standard RNNs, the repeating modules have a very simple
Fig. 6. Randomly selected testing and validation cases for the CNN-based prediction experiment. The substrate thicknesses for each case are (a) $h' = 0.07$, (b) $h' = 0.06$, (c) $h' = 0.10$, (d) $h' = 0.10$, (e) $h' = 0.07$, (f) $h' = 0.10$, (g) $h' = 0.06$, and (h) $h' = 0.08$.

Fig. 7. Loops in the RNN.

structure, such as a single layer of an activation function. However, this simple structure can hardly “remember” long-term information. There is a special kind of RNN, the LSTM network [23], which is capable of learning long-term dependencies. In an LSTM module, four layers interact in a special way (Fig. 8). There are two horizontal lines in each module. The top one, representing the cell state, is the key to LSTM. It runs straight down the entire chain, with only some minor linear interactions. This property allows information to flow along the chain. The bottom line represents the output from each module, which is the same as that of a normal RNN.

The four yellow blocks (gates) enable the LSTM to remove or add information to the cell state. The role of each of the four gates is discussed next.

The first step in an LSTM is to decide what information to remove from the cell state. The decision is made by a
sigmoid function ("forget gate layer"). The input of this layer is a new input $x_t$ and the output of the previous module $h_{t-1}$. The output of this gate is represented by

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

where $W_f$ are weights, $b_f$ are biases and $f_t$ represents the output of this gate.

The next step is to decide what new information will be stored in the cell state. This has two parts. First, a sigmoid layer called "input gate layer" decides which values to update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we combine these two parts to update the state

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Then, we update the cell state, $C_{t-1}$. As shown in (10), the old cell state $C_t$ is multiplied by the output of the "forget gate" $f_t$. Next, we add $i_t \cdot \tilde{C}_t$, which is the new candidate value

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t.$$  

Finally, the last step is to design the output of the module. The output is based on the input of the module, output from the previous module and the updated cell state. In this step, a sigmoid layer is used to decide which parts of the cell state to output. We normalize the cell state by a tanh layer and multiply it by the output of the sigmoid gate. In this way, we only output the parts we decide to:

$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \cdot \tanh(C_t).$$

Many repeating LSTM modules form an RNN (Fig. 7) that is capable of remembering long term information. In a standard RNN, only neighboring modules are connected. The input of each module cannot be kept for a long time. On the other hand, LSTM modules can preserve the input information for a long time along the chain.

From the previous analysis, we can see that the properties of the RNN are suitable for sequence tasks. When we deal with S-parameters in the frequency domain, it is observed that S-parameters at different frequency bands share some common features, such as the sequence of peaks and dips. Therefore, we treat different frequency bands as strongly connected series. In this way, our model can benefit from the LSTM structure in the learning process, thus overcoming the problem we encountered with the standalone CNN where we needed to separately train over frequency sub-bands.

**B. Proposed Hybrid Network Structure**

It is observed that the S-parameters from coarse mesh and dense mesh simulations share some common features, such as the general shape of the envelope of their magnitude. Since RNNs, especially with LSTM structure, have good performance in dealing with sequence data [24], [25], we apply LSTM networks to compensate for the numerical errors in coarse mesh FDTD simulations, due to numerical dispersion and inaccurate modeling of boundary conditions (e.g., staircasing errors). To obtain the dense mesh solution at a frequency $f_i$, the LSTM uses the information from coarse solutions at the frequency $f_i$, and at lower previous frequencies $f_{i-1}, f_{i-2}, f_{i-3}, \ldots$. The S-parameters of the whole frequency band are treated as a "paragraph" that contains sub-bands with strong correlation and each sub-band can be seen as a different "word". Besides, we previously showed that convolutional layers can process the layouts of planar circuits. Therefore, a CNN is used here to provide the geometry and substrate properties for input circuits to each LSTM module. To be specific, there are two inputs to each LSTM module. First, the whole frequency band is divided into sub-bands; each sub-band is an input for an LSTM module. Moreover, the geometry and substrate properties for each circuit, extracted from the CNN, are additional inputs. In this way, each LSTM module receives both space and frequency domain information. These two inputs are combined to train an LSTM network.

The structure of this hybrid network is shown in Fig. 9. In this work, the output of the CNN is processed with two fully connected layers and transformed into a $120 \times 1$ vector, which is presented as $x_C$ in Fig. 9. The vector $x_C$ is a common input for different $t$ of the LSTM, where $t = 1, 2, 3, \ldots$ represents the index of the frequency sub-band. The 0–20 GHz band is divided into five frequency sub-bands, so there are five corresponding LSTM modules. Each LSTM module produces an output with dimension of $24 \times 1$. The total number of solutions from the five LSTM modules is 120 representing...
Fig. 10. Convergence of the MSE of the hybrid network for different numbers of CNN layers.

During the training process, we find that because of the LSTM structure, there is no need to use as many as seven convolutional layers to build a deep CNN. Three convolutional layers are sufficient. Fig. 10 shows how the network converges with a different number of CNN layers. From this comparison, we can deduce two conclusions. First, convolutional layers are necessary for the network. Second, there is no significant difference between three and seven convolutional layers regarding the training and testing loss of the trained network. This is attributed to the fact that the coarse mesh simulation has provided useful information about the S-parameters to the network; the role of the CNN is supplementary. We benefit from the decreased number of convolutional layers in the following two aspects. First, the model is much lighter than before. With convolutional layers only, one model has $7.60 \times 10^8$ parameters, requiring 3.03 GB. However, the hybrid model only has $1.01 \times 10^9$ parameters, requiring 404.74 MB. Second, training the hybrid model requires significantly reduced computational time and memory resources.

V. INTEGRATION OF CNN IN AN EFFICIENT HYBRID NETWORK: NUMERICAL RESULTS

A. Data Format

For the following experiments, the set of geometries used to train the standalone CNN in Section III for CNN-based prediction is reused. Additionally, the FDTD program used for generating data is modified; each geometry is re-simulated, with coarse mesh partitioning. In particular, the FDTD computational domain is partitioned into $33 \times 33 \times 16$ cells of dimensions $\Delta x = 1.232$ mm and $\Delta y = 1.270$ mm this time. The cell size $\Delta z$ in the $z$-direction is kept the same. For the dense mesh simulation, the number of cells per wavelength at 20 GHz varies from 16.32 to 25.81. For the coarse mesh simulation, the number of cells per wavelength at 20 GHz varies from 5.44 to 8.60.

The results from dense mesh FDTD are used as ground truth for this work. The inputs of the network contain simulation results of coarse mesh FDTD, metallization pattern, dielectric permittivity, and normalized substrate thickness. The topologies include various multistub filters, stepped impedance filters and radial stubs. Each input layout is represented by two arrays, one for the metallization pattern and the substrate dielectric permittivity and the other for the substrate thickness with respect to the smallest simulated wavelength ($\lambda_{\text{min}}$). The frequency band for the coarse mesh simulation is divided into five sub-bands. Each part is an input to one LSTM module. Note that the number of sub-bands in the hybrid network is a hyperparameter for the LSTM. Unlike the CNN-based prediction, there is no need for separate trained network for each sub-band. From the convergence of the MSE, we conclude that this choice avoids overfitting. There are 10,000 sets of data for each category. About 80% of the data are used for training, and the rest is used for testing.

B. Model Parameters

The parameters for the complete model are shown in Fig. 11. On the left column, the first three convolutional layers and the dimension conversion function are the same as the CNN model of Section II. Then, three fully connected layers transform the dimension from $12,800 \times 1$ to $120 \times 1$. There is no dropout layer following each fully connected layer since we apply $L^2$ regularization in the optimization function to avoid overfitting. With $L^2$ regularization, the loss function becomes:

$$\text{Loss}_{\text{reg}} = \text{Loss} + \frac{wd}{2} \sum w_i^2$$

where $\text{Loss}_{\text{reg}}$ is the loss with regularization, $\text{Loss}$ is the MSE between prediction and ground truth, $wd$ is the hyperparameter weight decay, and $w_i$ represents the weight and bias in each neuron. The weight decay is set to $10^{-4}$. 

Fig. 11. Model parameters for the hybrid network. The coarse mesh solutions at 120 frequencies are divided into five subsets corresponding to five sub-bands each with 24 frequency samples. Each sub-band is processed by one of the five LSTM modules.
On the right column, the coarse mesh simulation results are inputs to an RNN with five LSTM modules. The input for each LSTM module is an $144 \times 1$ array. This consists of the output from fully connected layers ($120 \times 1$) and the coarse mesh results ($24 \times 1$). The outputs of the fully connected layers are duplicated five times. They are the first part of an LSTM module. The coarse mesh simulation results are uniformly divided into five parts. Each one is the second part of an LSTM module. The LSTM modules have 64 hidden layers. The outputs of an LSTM module are expected to be dense mesh results for a frequency band of dimensions $24 \times 1$. All the outputs from these five LSTM modules form the complete dense mesh results. The CNN and the LSTM of the hybrid network in Fig. 11 are both trained together instead of being trained separately. Both coarse mesh FDTD simulation results (as input data for the LSTM) and dense mesh FDTD simulation results (as target output data for the hybrid network) are needed for the training.

C. Numerical Results

In this section, we compensate the numerical errors for both amplitude and phase of the S-parameters deduced by coarse mesh FDTD. Fig. 12 shows results from the test set. Table II shows the comparison of the aggregate MSE between the coarse mesh simulation, training set, and testing set. Results from the dense mesh simulation are chosen as ground truth. From this comparison, we can see that the hybrid network can decrease the MSE of amplitude by an order, and MSE of phase by two orders of magnitude compared to the coarse mesh simulation.

Moreover, we explore the generalization ability of the hybrid network, by applying it to composite geometries that are well beyond those included in the training set. Representative results are shown in Fig. 13. Notably, the hybrid network is able to leverage the coarse mesh FDTD results, in addition to the layout learning of the CNN, to accurately estimate the dense FDTD results for new geometries.

This model was also trained with four NVidia A100 GPUs. One round of training needed 122.39 s. The average execution time for generating one coarse mesh dataset was 6.76 s on AMD Milan 7413 CPU. It took the trained network 0.67 s to compensate for one set of coarse mesh results using the same CPU.

VI. Discussion

The integration of the CNN of Sections II and III into the hybrid network along with the LSTM modules has led to a model that combines as follows.
Fig. 13. Composite geometries designed to test the generalizability of the hybrid network. The relative substrate thicknesses for each case are (a) $h' = 0.06$, (b) $h' = 0.08$, (c) $h' = 0.07$, (d) $h' = 0.09$, (e) $h' = 0.10$, and (f) $h' = 0.09$.

1) The ability of the CNN to directly process single layer PCB layouts.
2) Computational efficiency and excellent generalizability to new geometries, due to the augmentation of the input dataset with coarse mesh FDTD data. This additional input, processed by the LSTM modules of the hybrid network, leads to dramatic computational savings, retaining the accuracy of the CNN and generalizing well to geometries beyond those included in the training set.

In terms of computational cost, we note the following. For the standalone CNN, training data generation took 447.83 core hours on our CPU and model training took 321.65 core hours on our GPU. The hybrid network required an additional 56.33 core hours on our CPU for coarse mesh data generation, but its training time was reduced to just 40.8 core hours on our GPU. Core hour means the equivalent of using one CPU/GPU core continuously for an hour. Since CPU/GPU cores can run in parallel, the actual time for generating the data and training the network is much shorter.

The results of Figs. 12 and 13 demonstrate that our approach advances the state of the art compared to neural network models based on parameterized geometries (e.g., [6], [7], [12]). The hybrid network learns to correlate PCB layouts to S-parameters. As a result, once it learns 2–3 stub filters, it can also make accurate predictions for stub filters of different numbers of stubs and other composite geometries. The network achieves this because the learning process focuses on layouts rather than number of stubs or other geometric and material parameters.

**VII. CONCLUSION**

We showed the feasibility of applying deep neural networks to simulate two-port, single-layer planar microwave circuits. Unlike previous approaches, we have modeled a wide range of two-port circuits based on their layouts, rather than specific classes of parameterized geometries. A CNN was used to directly predict the S-parameters of planar circuits from their geometries and substrate properties, including thickness and permittivity. Moreover, the CNN was integrated with LSTM modules in a hybrid network that produced high-resolution S-parameter predictions, based on coarse mesh FDTD data. These input data were produced by inexpensive simulations at sampling rates down to 5.5 points per wavelength. Numerical experiments have shown the accuracy, efficiency and generalization ability of the proposed approach. This work is the first step toward applying machine learning algorithms to model multilayer/multiport PCB geometries directly from their layouts.

**REFERENCES**

[1] G. E. Karniadakis et al., “Physics-informed machine learning,” Nat. Rev. Phys., vol. 3, no. 6, pp. 422–440, 2021.
[2] S. Qi, Y. Wang, Y. Li, X. Wu, Q. Ren, and Y. Ren, “Two-dimensional electromagnetic solver based on deep learning technique,” IEEE J. Multiscale Multiphys. Comput. Techn., vol. 5, pp. 83–88, 2020.
[3] Q. J. Zhang, K. C. Gupta, and V. K. Devabhaktuni, “Artificial neural networks for RF and microwave design—from theory to practice,” IEEE Trans. Microw. Theory Techn., vol. 51, no. 4, pp. 1339–1350, Apr. 2003.
[4] J. Jin, C. Zhang, F. Feng, W. Na, J. Ma, and Q.-J. Zhang, “Deep neural network technique for high-dimensional microwave modeling and applications to parameter extraction of microwave filters,” IEEE Trans. Microw. Theory Techn., vol. 67, no. 10, pp. 4140–4155, Oct. 2019.

[5] J. Jin, F. Feng, W. Zhang, J. Zhang, Z. Zhao, and Q.-J. Zhang, “Recent advances in deep neural network technique for high-dimensional microwave modeling,” in IEEE MTT-S Int. Microw. Symp. Dig., Dec. 2020, pp. 1–3.

[6] F. Feng, C. Zhang, J. Ma, and Q.-J. Zhang, “Parametric modeling of EM behavior of microwave components using combined neural networks and pole-residue-based transfer functions,” IEEE Trans. Microw. Theory Techn., vol. 64, no. 1, pp. 60–77, Jan. 2016.

[7] Z. Zhao, F. Feng, W. Zhang, J. Zhang, J. Jin, and Q.-J. Zhang, “Parametric modeling of EM behavior of microwave components using combined neural networks and hybrid-based transfer functions,” IEEE Access, vol. 8, pp. 93922–93938, 2020.

[8] H. Ma, E.-P. Li, J. Schutt-Aine, and A. C. Cangellaris, “Deep learning method for prediction of planar radiating sources from near-field intensity data,” in Proc. IEEE Int. Symp. Electromagn. Comput., Signal Power Integrity (EMC-SIPI), Jul. 2019, pp. 610–615.

[9] W. Zhang, F. Feng, S. Yan, Z. Zhao, and W. Na, “Multiphysics parametric modeling of microwave components using combined neural networks and transfer function,” IEEE Access, vol. 8, pp. 5383–5392, 2020.

[10] A. Khan, V. Ghorbaniyan, and D. Lowther, “Deep learning for magnetic field estimation,” IEEE Trans. Magn., vol. 55, no. 6, pp. 1–4, Jun. 2019.

[11] X. Ma, M. Raginsky, and A. C. Cangellaris, “A machine learning methodology for inferring network S-parameters in the presence of variability,” in Proc. IEEE 22nd Workshop Signal Power Integrity (SPI), May 2018, pp. 1–4.

[12] O. W. Bhatti and M. Swaminathan, “Design space extrapolation for power delivery networks using a transposed convolutional net,” in Proc. 22nd Int. Symp. Quality Electron. Design (ISQED), Apr. 2021, pp. 7–12.

[13] O. Noakoosteen, S. Wang, Z. Peng, and C. Christodoulou, “Physics-informed deep neural networks for transient electromagnetic analysis,” IEEE Open J. Antennas Propag., vol. 1, pp. 404–412, 2020.

[14] A. Talleve, S. C. Hagness, and M. Piket-May, “Computational electromagnetics: The finite-difference time-domain method,” in The Electrical Engineering Handbook, vol. 3, Boston, MA, USA: Elsevier, 2005.

[15] C. D. Moss, F. L. Teixeira, and J. A. Kong, “Analysis and compensation of numerical dispersion in the FDTD method for layered, anisotropic media,” IEEE Trans. Antennas Propag., vol. 50, no. 9, pp. 1174–1184, Sep. 2002.

[16] M. Krumpholz and L. P. Katehi, “MRTD: New time-domain schemes based on multiresolution analysis,” IEEE Trans. Microw. Theory Techn., vol. 44, no. 4, pp. 555–571, Apr. 1996.

[17] S. Qi and C. D. Sarris, “Numerical dispersion compensation for FDTD via deep learning,” in Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting (AP-USRSI), Jul. 2022, pp. 671–672.

[18] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS), 2011, pp. 315–323.

[19] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” 2015, arXiv:1505.00853.

[20] W. Zhang, F. Feng, J. Zhang, Z. Zhao, J. Ma, and Q.-J. Zhang, “Parallel decomposition approach to wide-range parametric modeling with applications to microwave filters,” IEEE Trans. Microw. Theory Techn., vol. 68, no. 12, pp. 5288–5306, Dec. 2020.

[21] D. M. Sheen, S. M. Ali, M. D. Abouzahra, and J. A. Kong, “Application of the three-dimensional finite-difference time-domain method to the analysis of planar microstrip circuits,” IEEE Trans. Microw. Theory Techn., vol. 38, no. 7, pp. 849–857, Jul. 1990.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[23] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[24] A. Graves, N. Jaitly, and A.-R. Mohamed, “Hybrid speech recognition with deep bidirectional LSTM,” in Proc. IEEE Workshop Autom. Speech Recognit. Understand., Dec. 2013, pp. 273–278.

[25] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence—Video to text,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 4534–4542.

Shutong Qi (Graduate Student Member, IEEE) received the B.S. degree (Hons.) in electronic and information engineering from Beihang University, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree in electrical engineering at the University of Toronto, Toronto, ON, Canada. From 2018 to 2020, he was a Research Assistant at the Institute of EMC Technology, Beijing. His research interests include scientific machine learning, computational electromagnetics, multiphysics simulation, numerical methods, and microwave circuit modeling.

Costas D. Sarris (Senior Member, IEEE) received the Diploma degree (Hons.) in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1997, and the Ph.D. degree in electrical engineering and the M.Sc. degree in applied mathematics from the University of Michigan, Ann Arbor, MI, USA, in 2002. He is currently a Full Professor with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. His research interests include computational electromagnetics, with an emphasis on time-domain modeling, adaptive mesh refinement, enhanced stability, and higher order methods. He also works on physics-based wireless propagation models (with full-wave, asymptotic, and hybrid techniques), uncertainty quantification, and scientific machine learning.

Dr. Sarris was a recipient of the IEEE Microwave Theory and Technology Society (MTT-S) Outstanding Young Engineer Award in 2013 and the Early Researcher Award from the Ontario Government in 2007. He received the University of Toronto Faculty of Applied Science and Engineering Teaching Award in 2021, and four teaching awards for his work on undergraduate electromagnetics courses. He is the Co-Chair of the MTT-S Working Group on artificial intelligence (AI)/machine learning (ML) Based Technologies for Electromagnetics, and the Chair of the MTT-S Technical Committee (USNC) Joint Meeting and the 2019 Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO) Conference, the Technical Program Committee (TPC) Vice-Chair of the 2012 IEEE MTT-S International Microwave Symposium, and the Chair of the MTT-S Technical Committee on Field Theory and Numerical Electromagnetics from 2018 to 2020. He is the Editor-in-Chief of the IEEE JOURNAL ON MULTISCALE AND MULTIPHYSICS COMPUTATIONAL TECHNIQUES. He was a Guest Editor for the Special Issue on Machine Learning for Microwave Engineering of the IEEE Microwave Magazine in October 2021 and an Associate Editor of the IEEE MICROWAVE AND WIRELESS COMPONENTS LETTERS from 2007 to 2009 and the IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES from 2009 to 2013.