CHALMERS
UNIVERSITY OF TECHNOLOGY

Comparison of Circuit Models for ML-Assisted Microwave Circuit Design

Citation for the original published paper (version of record):
Sjödin, M., Talcoth, O., Chang, H. et al (2025). Comparison of Circuit Models for ML-Assisted Microwave Circuit Design. IEEE Journal of Microwaves, 5(6): 1358-1369.
http://dx.doi.org/10.1109/JMW.2025.3610923

(article starts on next page)

IEEE Journal of Microwaves

Received 10 March 2025; revised 18 August 2025; accepted 12 September 2025. Date of publication 29 September 2025; date of current version 7 November 2025.

Digital Object Identifier 10.1109/JMW.2025.3610923

Comparison of Circuit Models for ML-Assisted Microwave Circuit Design

MARTIN SJÖDIN [1] (Member, IEEE), OSKAR TALCOTH [1], HAOJIE CHANG [2] (Member, IEEE), HAN ZHOU [2] (Member, IEEE), AND KRISTOFFER ANDERSSON [1,3]

(Regular Paper)

[1]Ericsson Research, 417 56 Gothenburg, Sweden
[2]Chalmers University of Technology, 412 96 Gothenburg, Sweden
[3]Saab Surveillance, 412 76 Gothenburg, Sweden

CORRESPONDING AUTHOR: Martin Sjödin (e-mail: martin.sjodin@ericsson.com).

This work was supported by Innostar (Innovative Systems and Automated Design for 5G/6G Connectivity and Radar Applications).

ABSTRACT Machine-learning (ML) assisted microwave circuit design is an interesting complement to traditional topology-based design since it opens up previously unexplored design spaces that in some cases may offer better performance, or similar performance with a different form factor. A key part is the circuit model, i.e., the set of discrete building blocks used to create circuits. In the work published so far circuit models encompassed a single element type in the form of metal pixels. In this paper we propose a circuit model with additional elements that facilitates diagonal connections and provides higher robustness to variations in the manufacturing process. A comparison with the pixel model shows that the new model results in more accurate ML-models for S-parameter prediction with a 9.5% reduction in root mean-square error (RMSE) on the testset, which translates to more accurate results for circuit synthetization. In addition, we demonstrate that circuits built with the new model has a higher tolerance to manufacturing imperfections, with 33% smaller RMSE penalty with respect to the original S-parameters when adding a width perturbation of 50 μm to diagonal connections, and 50/40% smaller RMSE penalty when shrinking/expanding the size of elements forming diagonal connections with 2.5% . We also use both the pixel model and the newly proposed model to design low-pass filters with competitive performance.

INDEX TERMS Circuit optimization, convolutional neural networks, electronic circuits, genetic algorithm, machine learning, surrogate models.

I. INTRODUCTION

Passive circuits, i.e., circuits consisting entirely of passive components, play an important role in microwave electronics, and are for example used to realize matching networks, power splitters, power combiners, filters and hybrid couplers. Traditionally, such circuits are created by pre-selecting a circuit topology consisting of lumped transmission lines and/or discrete components to satisfy a set of performance criteria that often are related to the scattering parameters (S-parameters) between a set of input/output (I/O) ports. The performance is then optimized through parameter sweeps of transmission line lengths and/or the impedance of discrete components.

The traditional approach has the drawback that pre-selecting a circuit topology limits the space of achievable S-parameters and consequentially, one may miss out on more advantageous electromagnetic (EM) structures. To address this, Liu et al. [1] proposed a deep-learning based *inverse design method* for passive EM structures with two ports, which was later expanded to planar antenna structures and more complex multi-port circuits by Karahan et al. [2], [3], [4]. The main principle in inverse design is to first define a set of targeted properties, and then use an algorithm to find a structure providing such properties. Inverse design methods have been used for many different purposes, including in nanophotonics design of devices such as multiplexers and lenses [5], [6], [7], in material design to create materials with specific properties [8], [9], [10], [11], [12], and in chemical engineering optimization of distillation column networks [13].

In the inverse design method of Liu et al., the first step is to choose an area of a chip or printed circuit board (PCB) to which circuits are confined, in addition to defining a set of ports. The selected circuit area is discretized such that it can be described in matrix form, for example by an $N \times M$ matrix, where the elements can be either "1" or "0", representing presence and absence of conducting material in the corresponding location on the chip/PCB, respectively. Through EM simulations of random circuits described in this manner, a dataset with circuit arrays and their corresponding S-parameters is created, which can be used to train machine learning (ML) models to predict the S-parameters of circuits based on the binary matrices. Once such a model has been obtained, it can be used as a surrogate model for Maxwell's equations to vastly improve the speed of genetic optimization algorithms for synthetization of new circuits. In contrast, solving Maxwell's equations as part of the optimization process was for example done in [14], but will in many cases be exceedingly time consuming and computationally demanding. Using a surrogate model is therefore the preferable option, provided it is sufficiently accurate.

In the work published so far the circuit models has used only square-shaped metal pieces. This might lead to issues both with high impedance diagonal connections and the manufacturing process, if the connections are very narrow. We therefore propose a new circuit model which apart from square-shaped metal pieces uses additional elements that enable diagonal connections of the same width as the straight paths. The proposed model is compared with a model using square-shaped and we find that it enables more accurate ML-models to be trained, which in turn lead to a more accurate process for circuit synthetization. Furthermore it results in circuits with slightly smaller loss compared to the known model and we demonstrate that it is more robust to imperfections in the width of diagonal circuit connections, which indicates it should be more robust to manufacturing imperfections.

The paper is structured as follows: in Section II we introduce the circuit models we consider, and describe how they are used to build circuits. In Section III we describe the generation of datasets with circuit arrays and corresponding S-parameters for the circuit models, that can subsequently be used to train ML-models for S-parameter prediction. Using these datasets we also compare the circuit models in terms of loss and tolerance to manufacturing imperfections. In Section IV we discuss the structure and the performance of the ML-models trained on the datasets and Section V presents the genetic optimization procedure for synthesizing new circuits, as well as how we generate low-pass filters with competitive performance. Finally, we conclude the paper in Section VI.

II. CIRCUIT MODEL DESCRIPTIONS

We use two circuit models which we call the *Square-Model* and the *Octagon-Model*, which sometimes will be referred to as *SqMod* and *OctMod*, respectively. The geometrical structures of the models are identical. The stack-up is the same as in [15], from bottom to top, a copper ground plane, a 508 μm Rogers-4350B substrate with $\epsilon_r = 3.66$, and a 17 μm copper layer for the circuits and port connections. The circuits are confined to a square-shaped area discretized with a grid of 13×13 square-shaped cells with 900 μm $\times$ 900 μm size, and a pair of ports placed symmetrically on opposite sides. Each circuit is represented by a 13×13 binary matrix which describes the placement of elements in the circuit layer. The choices regarding grid size and number of elements are motivated by initial experimenting where we find that the circuit size and resolution allow us to, e.g., design low-pass filters, while at the same time enable training of accurate ML-models with reasonable EM-simulation efforts.

The positions of the circuit elements are described using a Cartesian coordinate system with origin in the middle of the circuit area, the x-axis pointing from port 1 to port 2, and the y-axis perpendicular to the x-axis in the circuit plane. It can also be said that the x- and y-directions correspond to the rows and columns of a circuit matrix, respectively.

A. THE SQUARE-MODEL

The Square-model utilizes only square-shaped metal elements to form circuit patterns, and was for example utilized in [1], [2], [3], [4]. Fig. 1(a) shows an example circuit created using the circuit matrix shown in Fig. 1(c). The grid onto which the metal elements are placed is indicated with dotted lines. Each grid element corresponds to an element of the circuit array, and the placement of metal elements is straightforward: for every matrix element which is "1", a metal element is placed onto the corresponding grid element such that the center points coincide.

The Square-model has narrow diagonal connections, in particular when the elements have identical size as the grid such that only the corners of elements on a diagonal touch each other. Narrow connections imply large impedance, and will also be more difficult to manufacture with good precision, thus increasing the risk for discrepancy in performance between simulated and fabricated designs. Narrow connections in circuits may also increase the simulation time required to create a dataset since a finer mesh typically is required. One way to enable diagonal connections for a circuit model with only square-shaped elements is to expand the size of the discrete elements relative to the circuit grid. In the work by Karahan et al., an expansion of 25% was used [2] In our work we chose an expansion of 20% which gives the circuit elements of the square-model a length of 1080 μm and a characteristic impedance of 50 Ω with the given stack-up [16]. The corresponding width of diagonal connections is 255 μm.

B. THE OCTAGON-MODEL

To address the Square-model's issue with narrow diagonal connections we propose the *Octagon-Model* which utilizes metal elements of both square and octagonal shape. The Octagon-model uses the same types of circuit grid and circuit arrays as the Square-model, but a different set of circuit elements. While the square models use a single discrete building block to form metal paths, the Octagon-model uses both

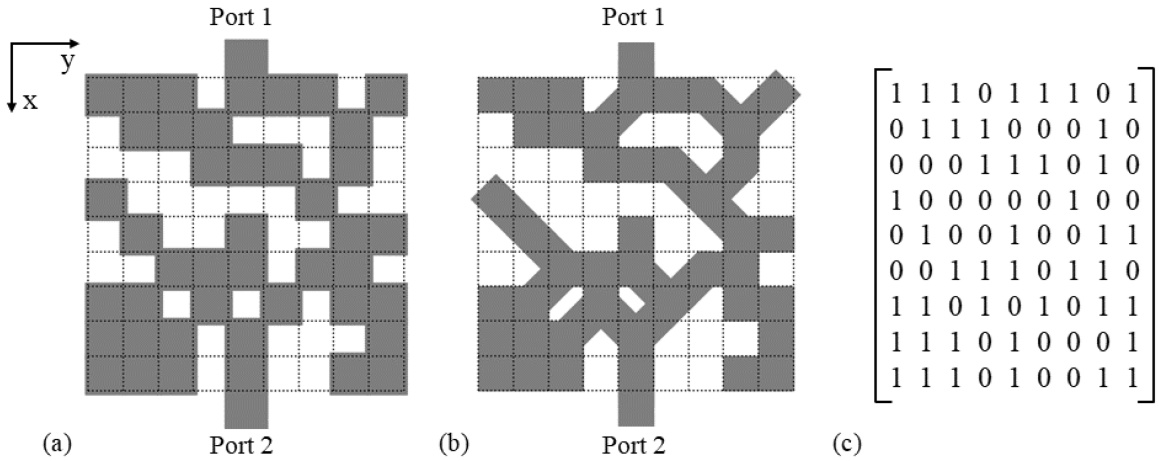

FIGURE 1. Two example circuits resulting from the same circuit matrix. The circuit grid onto which elements are placed is indicated with dotted lines. (a) Square-model with metal elements enlarged with 20% relative to the grid. (b) Octagon-model utilizing a combination of square-shaped pieces with either 0° or 45° rotation and octagons. (c) The circuit matrix with 9 × 9 elements. Port 1 is connected to element (0,4) and port 2 to element (8,4).

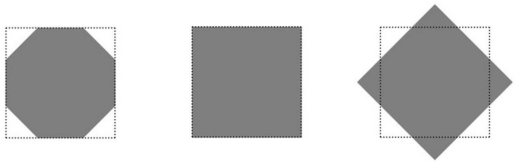

FIGURE 2. The main building blocks of the Octagon-model with, from left to right: octagon, square without rotation, square rotated 45°.

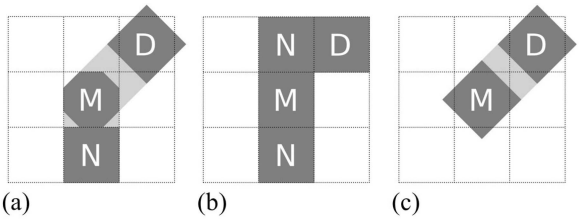

FIGURE 3. Examples of main elements (M) surrounded by nearest neighbors (N) and/or diagonal neighbors (D). (a) Type 1 main element with a single nearest neighbor and a single diagonal neighbor that does not share any nearest neighbors with the main elements. (b) Type 2 main element with two nearest neighbors and a single diagonal neighbor sharing one nearest neighbor with the main element. (c) Type 3 main element with a single diagonal neighbor.

square-shaped elements and octagons. Fig. 1(b) shows an example circuit created using the circuit matrix in Fig. 1(c). The length of the sides of the square-shaped elements, and the distance between opposite edges of the octagons, is the same as the length of the sides of the grid cells, i.e. 900 μm. This gives the element of the Octagon-model a characteristic impedance of 56 Ω [16].

The final part of this section contains a description of the Octagon-model, which we divide into two parts. Part 1 is dedicated to the main circuit elements and how to select them, and part 2 to secondary circuit elements used to bridge gaps and/or to smoothen paths between main elements.

1) MAIN CIRCUIT ELEMENTS

The main elements of the Octagon-model are placed such that their center points coincide with the center points of the grid elements. There are three types of main elements:

Oct: octagon without rotation,
Sq-0: square without rotation,
Sq-45: square with 45° rotation.

Fig. 2 shows the main elements of the Octagon-model and how they are placed relative to their corresponding grid elements. For a given entry in the circuit matrix, the circuit element to use depends on the surrounding matrix entries, i.e. those whose row and/or column indexes differ with at most 1.

We refer to these elements as *nearest neighbors* and *diagonal neighbors*, defined as:

Nearest neighbor: a "1" in the circuit matrix with ±1 difference in row or column index,
Diagonal neighbor: a "1" in the circuit matrix with ±1 difference in row and column index.

Consider an element ϵ with a set of nearest neighbors $\mathbf{N}_\epsilon$, a set of diagonal neigbors $\mathbf{D}_\epsilon$, and let $\mathbf{A}_\epsilon = \mathbf{N}_\epsilon \cup \mathbf{D}_\epsilon$. We choose the element type for ϵ by using the following rules:

$$\textit{Oct: } (|\mathbf{N}_\epsilon| \geq 1) \wedge (|\mathbf{D}_\epsilon| \geq 1) \wedge (\exists \delta \in \mathbf{D}_\epsilon, \mathbf{N}_\delta \cap \mathbf{N}_\epsilon = \emptyset),$$
$$\textit{Sq-0: } ((|\mathbf{N}_\epsilon| \geq 1) \wedge (\forall \delta \in \mathbf{D}_\epsilon, \mathbf{N}_\delta \cap \mathbf{N}_\epsilon \neq \emptyset)) \vee |\mathbf{A}_\epsilon| = 0,$$
$$\textit{Sq-45: } (|\mathbf{N}_\epsilon| = 0) \wedge (|\mathbf{D}_\epsilon| \geq 1)$$

The math symbols used for the element placement rules are found in [18].

Fig. 3(a)–(c) shows examples of a main element (indicated by "M") surrounded by nearest neighbors (indicated by "N") and/or diagonal neighbors (indicated by "D"). To provide an explanation of how the placement rules, we choose the case

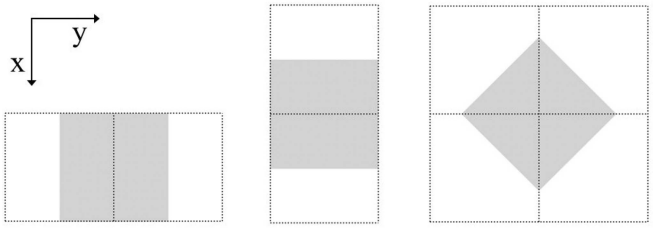

FIGURE 4. The secondary building blocks of the Octagon-model with, from left to right: square without rotation forming connections in the y-direction, square without rotation forming connections in the x-direction, and square rotated 45° forming diagonal connections.

displayed in Fig. 3(a) where the main element has a single nearest neighbor and a single diagonal neighbor. Selecting an octagon for this element requires three conditions to be satisfied. Firstly, the number of nearest neighbor elements must be at least 1, secondly the number of diagonal neighbor elements must be at least one, and thirdly, there must be an element δ among the set of diagonal neighbor elements that does not share any nearest neighbors with the main element. All three conditions are satisfied for the main element in Fig. 3(a), which consequently is an octagon. On the other hand, as seen in the rule for *Sq-0*, selecting such an element requires either that there are no neighbor elements at all, or that all diagonal neighbor elements share a nearest neighbor elements with the main element, which is not true in this case.

2) SECONDARY CIRCUIT ELEMENTS

Apart from the main circuit elements listed here, secondary square-shaped elements are placed in the circuits to bridge gaps between elements, or to form smoother paths. The types of secondary elements are:

Sq-0-y: square without rotation for connections along the y-direction (column direction in circuit matrix),
Sq-0-x: square without rotation for connections along the x-direction (row direction in circuit matrix),
Sq-45-diag: square with 45° rotation for diagonal connections.

Fig. 4 illustrates the secondary elements of the Octagon-model. Given two main elements ϵ_1 and ϵ_2, the rules for when to use them are as follows:

Sq-0-y: ϵ_1 and ϵ_2 are nearest neighbors in the same row. At least one is of type *Oct*,
Sq-0-x: ϵ_1 and ϵ_2 are nearest neighbors in the same column. At least one is of type *Oct*,
Sq-45-diag: ϵ_1 and ϵ_2 are diagonal neighbors. Both are of type *Oct* or *Sq-45*.

All secondary elements are placed with their center points between the center points of the main elements. Fig. 5 shows three examples of uses of secondary elements. In Fig. 5(a) a square is used to form a smoother circuit path along the x-direction when two octagons are next to each other, in

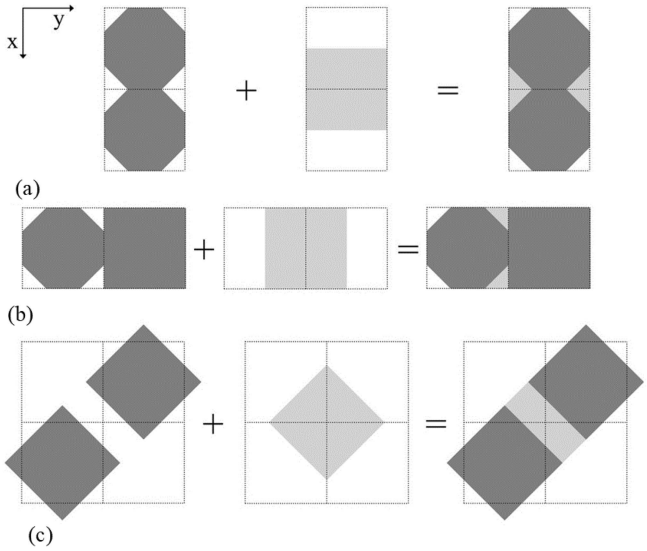

FIGURE 5. Examples of how the secondary elements of the Octagon-model can be used to form smoother paths, or to form diagonal connections. (a) a metal square smoothens the path between two octagons in the x-direction, (b) a metal square smoothens the path between main elements of type 1 and 2 in the y-direction, and (c) a square rotated 45° forms a connection between two main elements of type 3.

Fig. 5(b) a square smoothens the path along the y-direction when an octagon is placed next to a square without rotation, and in Fig. 5(c) a square rotated 45° is required to form a connection between two main elements that also are squares with 45° rotation.

III. COMPARISON OF CIRCUIT MODELS

In this section we compare the circuit loss and tolerance to manufacturing imperfections of the circuit models based on the datasets we generate through simulations of random circuits in HFSS.

A. DATASET CREATION

We create two datasets, one for the Octagon-model and another for the Square-model. 60,000 unique circuit matrices are generated as a first step. The corresponding circuits are then simulated with an EM-solver for both circuit models and added to the respective dataset together with the S-parameters. The circuit matrices of the two datasets are hence identical, but the S-parameters are different due to the usage of different circuit models.

A circuit matrix is created by first drawing a value for the probability of ones, p_{metal}, from a normal distribution with mean of 0.50 and standard deviation of 0.15 and with lower and upper limits of 0.20 and 0.80, respectively. In the next step a 13×13 matrix with random numbers drawn from a uniform distribution between 0 and 1 is generated. Matrix elements smaller than p_{metal} are set to 1 and the remaining elements to 0. In the final step we check if a circuit resulting from the matrix has a metal path between the ports. If true, the matrix is kept, otherwise new matrices are generated using the same

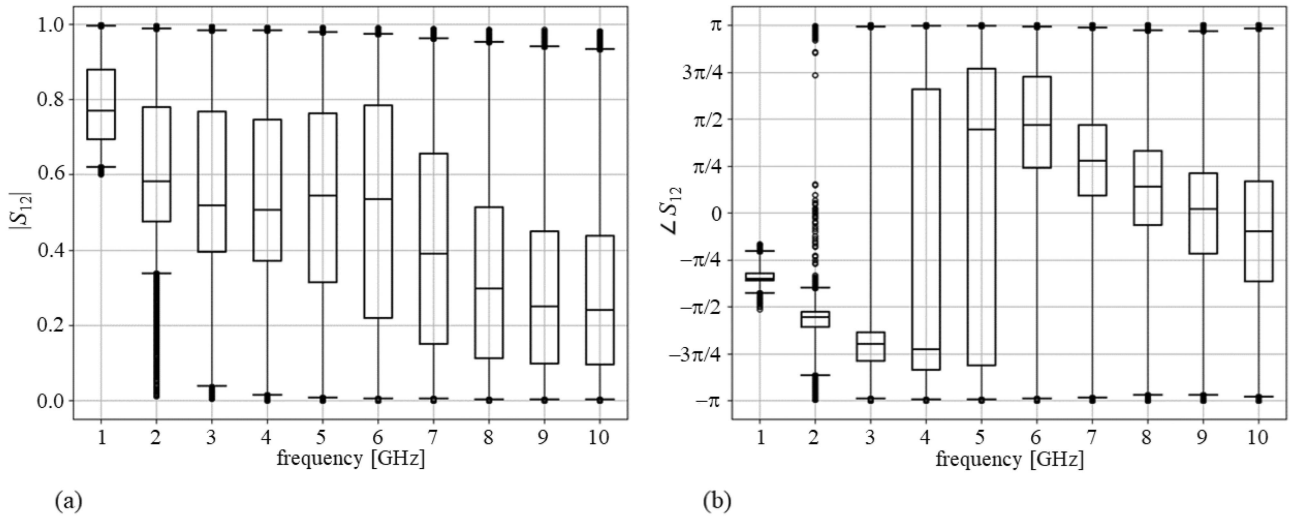

FIGURE 6. Boxplots of S_{12} versus frequency obtained using the dataset of the Octagon-model. (a) $|S_{12}|$, (b) $\angle S_{12}$ in radians.

p_{metal} value until we find one that satisfies the port connection condition.

As in [1] and [2], the number of data examples in the dataset is increased with data augmentation. With two symmetrically placed ports, and the coordinate system defined as in II, the data can be quadrupled with the following operations:

- flip array around the x-axis,
- flip array around the y-axis,
- 180° rotation in the xy-plane.

Simulating 60,000 circuits with the 3D FEM EM solver HFSS [17] thus provides us with 240,000 circuits in total. The augmentation operation only needs to be performed a single time and takes less than a minute for our datasets. In the HFSS simulations we use transmission lines with a length of 4.5 mm between the ports and the circuits, and the S-parameters are de-embedded such that the reference plane is 0.9 mm from the circuits.

If the datasets are to be expanded, a method for active learning similar to the work in [19] can be used, where new variations of unit cells for optical metasurface creation are added to an existing dataset based on the uncertainty of a ML-model in predicting their transfer functions.

We first compare the variation of the S-parameters in the datasets for the circuit models. Fig. 6(a) and 6(b) show boxplots of the magnitude and phase of S_{12} from 1 to 10 GHz for the Octagon-model. The edges of each box marks the first and third quartile, hence each box contains 50% of the data points at that frequency. Furthermore, the horizontal line within each box represents the median, the whiskers (the horizontal lines above and below each box) indicate the range that contains 99% of the data points, and outliers are indicated with black circles below or above the whiskers. Visualizing the S-parameters values in the dataset in this way is useful to get an idea of the S-parameters that are supported by our circuit model and hence can be achieved in circuit synthetization through genetic optimization. As an example, with our dataset, it is easy to see that both the phase and amplitude ranges of S_{12} at the lower frequencies in the plot are limited, and it would be difficult to achieve circuits with $\angle S_{12} = \pm\pi$ at 1 GHz. Looking at Fig. 6(a), we also note that the whiskers containing 99% of the points span almost the whole possible range from 3 GHz and upwards, which indicates that circuits with filter characteristics could be generated at those frequencies. We explore this in Section V-A. The achievable S-parameter range for a given model is determined by its material and geometrical properties. If we want to achieve more S-parameter variations one option is to use a larger chip area and another to increase ϵ_r.

B. CIRCUIT LOSS

We expected circuits built with the Octagon-model to have smaller insertion loss in general due to its improved diagonal connections. To assess this we plot the cumulative distribution function (CDF) of $|S_{12}|$ in dB at different frequencies for each dataset. The results are shown in Fig. 7(a) and 7(b), for $|S_{12}|$ ranges of −20 dB to 0 dB and −1 dB to 0 dB, respectively, at frequencies of 1 GHz, 5.5 GHz and 10 GHz. Fig. 7(a) shows that the curves for the two models look qualitatively similar, but it is noteworthy that for all three frequencies the CDF for the Octagon-model is shifted to the right with respect to the corresponding curve for the Square-model, indicating lower circuit loss. For example, looking at Fig. 7(b) we note that at a cumulative probability of 0.9 and a frequency of 1 GHz $|S_{12}|$ is approximately 0.1 dB smaller for the Square-model.

C. ROBUSTNESS TO MANUFACTURING IMPERFECTIONS

Due to its narrow diagonal connections we assume that the Square-model is more sensitive than the Octagon-model to manufacturing imperfections. To investigate this we randomly select 1000 circuit matrices from the datasets which contain at least one diagonal connection, and for the corresponding

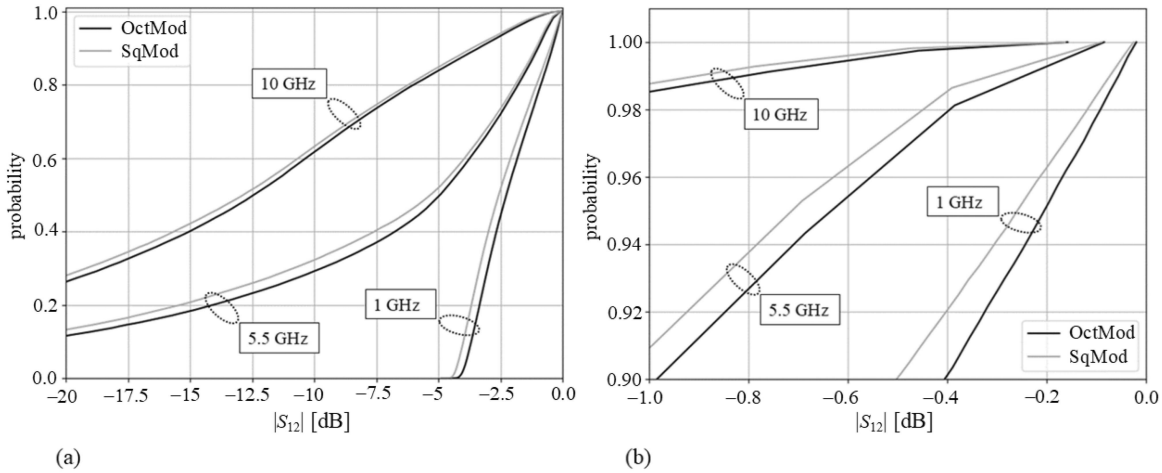

FIGURE 7. Cumulative distribution functions of $|S_{12}|$ at 1 GHz, 5.5 GHz and 10 GHz for the dataset of each circuit model. (a) −20 dB to 0 dB, and (b) −1 dB to 0 dB and cumulative probabilities above 0.9.

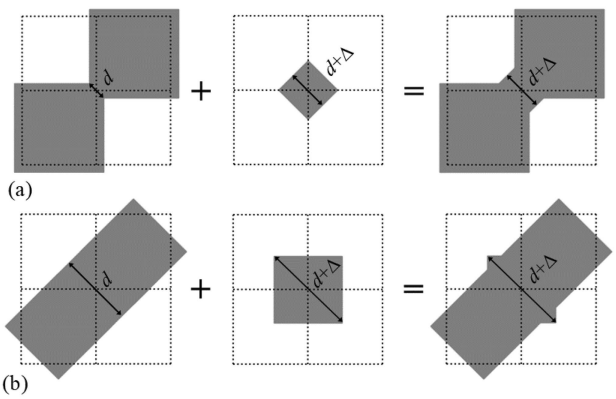

FIGURE 8. Diagonal connections with perturbaxtion elements increasing the normal width d with Δ. Here $\Delta = 250$ μm for illustrative purposes but in our test we use $\Delta = 50$ μm. (a) Square-model, $d = 254.6$ μm. (b) Octagon-model, $d = 900$ μm.

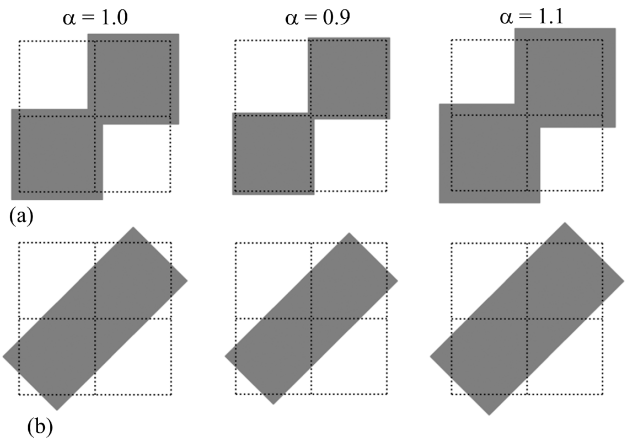

FIGURE 9. Circuit elements forming diagonal connections are expanded with a factor α. The leftmost column shows the case $\alpha = 1.0$ (normal element size), the column in the middle the case $\alpha = 0.9$ and the rightmost the case $\alpha = 1.1$. The values of 0.9 and 1.1 were chosen for illustrative purposes. In the tests, we compared the normal case with $\alpha = 0.975$ and $\alpha = 1.025$. (a) Square-model. (b) Octagon-model.

circuits we run two different tests where we perturb or modify the width of diagonal connections.

1) TEST 1

In the first test, a small perturbation element is added to all diagonal connections, which increases the normal width d with $\Delta = 50$ μm. The reason for using an expansion of 50 μm is that this is the largest fabrication error for the process we use for circuit manufacturing. These perturbations can be easily implemented in HFSS simulations by adding a square-shaped element with appropriate dimensions, as illustrated in Fig. 8. Note that in the examples the width expansion $\Delta = 250$ μm is used to better illustrate the perturbations. The normal width of diagonal connections are 254.6 μm and 900 μm for the square-model and the octagon-model, respectively. The main benefit with this test is that it provides a fair comparison between the two circuit models in the sense that the perturbations will have the same area.

2) TEST 2

In the second test we expand or shrink the size of circuit elements forming diagonal connections with a factor α and compare with the normal case of $\alpha = 1.0$. We use α values of 0.975 (2.5% shrinkage) and 1.025 (2.5% expansion). Fig. 9 visualizes the impact of different α values for both circuit models. In the leftmost column the normal case is shown, while the center and rightmost columns show example connnections for $\alpha = 0.9$ and $\alpha = 1.1$, respectively. For the purpose of illustration, we use larger shrinkage/expansion in the figure.

3) RESULTS

Table 1 shows the results for the two tests when we calculate the RMSE with respect to the S-parameters of the circuits used for the tests without modifications. For Test 1 where

TABLE 1. RMSE With Respect to Circuits Without Perturbed or Modified Diagonal Elements

	Octagon-model	Square-model
Test 1	1.6e-2	2.4e-2
Test 2, $\alpha = 0.975$	2.3e-2	4.7e-2
Test 2, $\alpha = 1.025$	2.8e-2	4.6e-2

a small perturbation is added to diagonal connections, the average RMSE of the Octagon-model is 1.6e-2 and for the Square-model it is 2.4e-2. The Square-model thus has a 50% RMSE increase on average in this test.

For Test 2 with $\alpha = 0.975$, the Octagon-model circuits have an average RMSE of 2.3e-2, while the RMSE is 4.7e-2 for the Square-model circuits, i.e. more than a factor of 2 larger. With $\alpha = 1.025$ the RMSE is 2.8e-2 and 4.6e-2 for the Octagon-model circuits and the Square-model circuits, respectively. Using the Octagon-model hence gives 40% smaller penalty.

Both tests indicate that the Square-model is much more sensitive to manufacturing uncertainties, due to its narrow diagonal connections. The higher robustness of the Octagon model should be taken into account when deciding which circuit model to use.

IV. MACHINE LEARNING MODELS

Similar to [1] and [2], we use the datasets to train convolutional neural network (CNN) models to act as surrogate models, thus enabling circuit synthesis with genetic optimization without requiring time consuming EM simulations for evaluating the candidate circuits. The CNN, originally proposed in [20], is a natural choice as ML-model for ML-assisted circuit design due to the image-like circuit arrays being used. We train models to predict the real and imaginary parts of S_{11}, S_{12} and S_{22} from 0.5 GHz to 10 GHz with 0.5 GHz resolution and the total number of output parameters is hence 120 (real and imaginary parts of three parameters at twenty frequency points). The ML-models are trained for a maximum of 300 epochs, with learning rate starting at 0.05 and gradually decreasing to 1×10^{-4}. Early stopping is used in case model performance stops to improve based on the validation data. We use Keras with Tensorflow with GPU support, and the GPU is a Tesla V100. The time to train a single model is approximately 3 hours and 15 minutes if the 300 epochs limit is reached.

For all ML-model tests, the dataset is split into one part for the training, another part for the validation during the training, and a third part for testing after training was completed, with identical splits for the datasets for the different circuit models. The shares are 70%, 15% and 15% for training, validation and testing respectively. It is ensured that augmented data examples are put in the same set as their corresponding originals, i.e. if one of the original circuits is placed in the training set, so are its three augmentations. To assess the performance of the ML-models we calculate the average RMSE on the testset.

It is worth mentioning that the complexity of training or using an ML-model does not depend on the choice of circuit model, since both the Octagon-model and the Square-model represent circuits with a single binary matrix.

TABLE 2. Convolutional Layers of the Baseline ML-Model

Layer	Kernel size	$n_{filters}$
Conv_1	12×12	64
Conv_2	10×10	64
Conv_3	8×8	64
Conv_4	6×6	64
Conv_5	5×5	64
Conv_6	5×5	64
Conv_7	4×4	64
Conv_8	4×4	64
Conv_9	4×4	64
Conv_{10}	3×3	64
Conv_{11}	3×3	64
Conv_{12}	3×3	64

TABLE 3. RMSE of Baseline ML-Model on Testsets

	Octagon-model	Square-model
RMSE	0.106	0.114

A. FINDING A SUITABLE MODEL STRUCTURE

As a starting point we use a model with essentially the same structure as the one presented in [2] as a baseline. The baseline model has 12 convolutional layers with kernel sizes and number of filters $n_{filters}$ as shown in Table 2, with batch normalization after each layer, which is a technique known to improve convergence speed [21]. After the convolutional layers are four fully-connected layers with 500 neurons in each and batch normalization after each layer, followed by an output layer for the complex S-parameters. In [2], dropout of 50% was used in the fully connected layers and L2-regularization with a regularization factor of 0.0001 applied to all learnable coefficients. The purpose of utilizing these techniques is to prevent overfitting [22], [23]. However, these regularization settings did not work well for us as the models had difficulties to learn, and for our baseline model we instead opted for 15% dropout in both the convolutional layers and the fully-connected layers, and omitted the L2 regularization. As in [2], leaky ReLU activations [24] are used in both the convolutional layers and the fully-connected layers.

For each circuit model we use its dataset to train three ML-models with the baseline structure. The average RMSE score on the testset for each case is shown in Table 3. With the baseline structure as a starting point we make a large number of tests to see how the performance is affected by the number of convolutional layers and fully-connected layers (also known as dense layers), convolutional filter kernel sizes and the number of filters per layer, number of neurons in the dense layers, activation functions, regularizations, and utilizing pooling operations after the convolutional layers.

As it turns out, similar ML-model structures perform equally well for the circuit models, which is not surprising considering that the input data is identical and although there

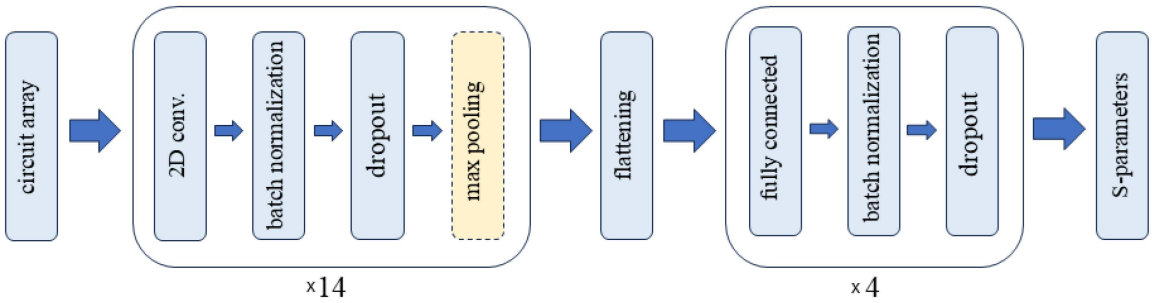

FIGURE 10. Overview of the structure of the ML-model for S-parameter prediction. Kernel sizes from 8 × 8 to 2 × 2 are used for 2D convolutions. Max-pooling is only used after twelfth convolutional layer.

TABLE 4. Convolutional Layers of the Final ML-Model

Layer	Kernel size	$n_{filters}$
$Conv_1$	8×8	128
$Conv_2$	8×8	128
$Conv_3$	7×7	128
$Conv_4$	7×7	128
$Conv_5$	6×6	128
$Conv_6$	6×6	128
$Conv_7$	5×5	128
$Conv_8$	5×5	128
$Conv_9$	4×4	128
$Conv_{10}$	4×4	128
$Conv_{11}$	3×3	128
$Conv_{12}$	3×3	128
$Conv_{13}$	2×2	64
$Conv_{14}$	2×2	64

TABLE 5. RMSE on Testsets of Final ML-Models Predicting Complex S_{11}, S_{12} and S_{22}

	Octagon-model	Square-model
average mean(*RMSE*)	6.39e-2	6.89e-2
best model mean(*RMSE*)	6.14e-2	6.78e-2
best model median(*RMSE*)	5.66e-2	6.27e-2
best model min(*RMSE*)	0.67e-2	0.71e-2
best model max(*RMSE*)	0.29	0.36

is variation in the output data it can at least be considered to be very similar. For the later tests and comparisons we select a final ML-model structure which has 14 convolutional layers, with kernel sizes and number of filters $n_{filters}$ as shown in Table 4. Increasing the number of convolutional layers further gives diminishing returns in terms of performance but on the other hand it increases the training time and the time required for circuit synthetization. This may change however for a larger dataset. An overview of the model structure is shown in Fig. 10. The final model also has four fully-connected layers, with 512 neurons in each. Dropout of 10% is used for all hidden layers and every hidden layer is followed by batch normalization. We use exponential linear unit (ELU) [25] activations instead of leaky RELU for all hidden layers, since this results in approximately a 10% reduction in RMSE for otherwise identical structures. Max pooling [26] is used after the twelfth convolutional layer. Apart from a small reduction in RMSE the max pooling reduces both the training time and the time required for genetic optimization, due to the smaller number of model parameters. We test removing and adding convolutional layers, but 14 layers provides the best performance.

B. PERFORMANCE ON TEST SETS

To get a final RMSE score for comparing ML-models trained on the datasets for the different circuit models, we train new models using a different random seed to split the data into different sets for training, testing and validation. We train three ML-models for each circuit model and the performance in terms of RMSE is shown in Table 5. The top row contains the average mean RMSE for all three ML-models trained per circuit model, while the remaining rows show the performance of the best ML-model for each case. ML-models trained using the dataset of the Octagon-model has 7.2% smaller RMSE on average, and the difference between the best ML-model for each circuit model is 9.4% . We assume the reason for the performance difference is that the Octagon-model, with its broader diagonal connections, results in circuit behavior that is more easily captured by an ML-model.

V. CIRCUIT SYNTHETIZATION

Our genetic optimization process for circuit synthesis utilizes the algorithm described in Algorithm 1.

We let the mutation probability p_{mut} vary with the iteration count as shown in Fig. 11. The initial value of p_{mut} at the first iteration is 0.15, and then it decreases to a minimum value of 0.01 if the iteration limit of 1000 is reached. This profile for p_{mut} was chosen after some experimentation indicated that it provides good performance. Starting off with a quite large mutation probability that subsequently is decreasing is intuitive, as in the beginning of the process one wants to explore many options, while only doing small refinements at the end, when a good candidate hopefully has been obtained. The number of candidate matrices N in Algorithm 1 is set to 8192, and we use $M = 256$. With these settings, it takes approximately 10 minutes to run a maximum number of 1000 iterations, using the same Tesla V100 GPU that was used to train the ML-models.

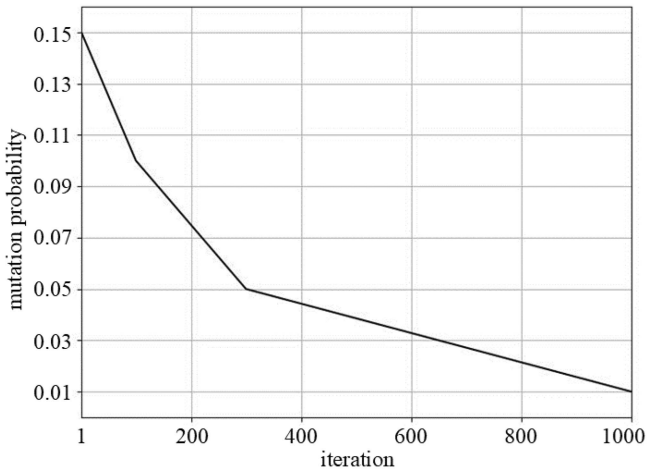

FIGURE 11. The dependency of the mutation probability p_{mut} on the iterations of the genetic optimization algorithm.

Algorithm 1: Algorithm for genetic optimization.

1) Select a target profile for the S-parameters,
2) Generate N random binary matrices as the initial set of candidate circuits,
3) Let the ML-model predict the S-parameters of the candidate circuits and use the target profile to calculate a score such as MSE or RMSE for each.
4) Check if there are circuits that satisfies the performance criteria or if the iteration limit has been reached. If neither is true, go to the next step.
5) Keep the M best candidates and make copies to form a new set of N candidate arrays.
6) Randomly permute the copies of the candidates in the new set of candidate circuits through bit-flip mutations with a probability p_{mut}, and return to step 3.

A. GENERATING CIRCUITS FROM TEST SETS

We perform genetic optimization tests of the circuit models and their corresponding ML-models by selecting 100 identical circuit arrays from each testset not used during training, and using the S-parameters as the targets to optimize for. We use complex S_{11}, S_{12} and S_{22} for this test and the RMSE of the S-parameters of the candidate circuits with respect to the target parameters as the score function in the genetic optimization process. After obtaining candidate circuits for the test examples we verify the results with HFSS simulations. The main purpose of this test is to see if there are differences between the circuit models in the accuracy of the genetic optimization procedure. Since the S-parameter targets comes from the testset, we know they should be possible to obtain, provided the ML-model is accurate enough and the optimization procedure works as intended. The results of this comparison is shown in Table 6. Looking at this table, we first note for both circuit models a significant discrepancy between the mean RMSE values for the testset examples predicted by the ML-models in the genetic optimization process, and the corresponding values from the HFSS verification. The reason for this is that in both cases there are nine circuits without a metal path between the ports, which causes a large discrepancy between prediction and verification. If we instead consider the median RMSE, which is impacted by these examples to a much smaller extent, the agreement between the predicted performance and the verification results is better. Considering the verification results there is a small RMSE improvement for the Octagon-model circuits. The final four rows in Table 6 show the results when excluding the nine circuits without port-to-port connections, which results in a large reduction in the mean RMSE of the HFSS verification. The median RMSE of the HFSS verification is reduced with almost 10% for the Octagon-model but for the Square-model the reduction is only about 2% . While this test was performed on a limited number of test examples, the results indicate that the Octagon-model provides more accurate genetic optimization results since it results in a more accurate ML-model.

TABLE 6. RMSE When Generating 100 Circuits Using S-Parameters From the Testset Not Seen During Training

	Octagon-model	Square-model
mean($RMSE_{gen-opt}$)	6.13e-2	6.61e-2
mean($RMSE_{HFSS}$)	17.2e-2	17.7e-2
median($RMSE_{gen-opt}$)	5.13e-2	5.89e-2
median($RMSE_{HFSS}$)	9.26e-2	9.52e-2
mean($RMSE_{gen-opt}$)*	5.90e-2	6.37e-2
mean($RMSE_{HFSS}$)*	10.03e-2	10.55e-2
median($RMSE_{gen-opt}$)*	4.97e-2	5.55e-2
median($RMSE_{HFSS}$)*	8.44e-2	9.35e-2

*excluding the nine circuits without port-to-port connection.

TABLE 7. Optimization Criteria for Low-Pass Filter Generation

	0.5–6.0 GHz	6.5 GHz	7.0–10.0 GHz
Criteria 1	$\lvert S_{12}\rvert = 1$	-	$\lvert S_{12}\rvert = 0$

B. LOWPASS FILTER GENERATION

As another circuit generation test, we use the ML-models and the genetic optimization algorithm to generate low-pass filters within 0.5-10 GHz. As a reference we use the commercially available filter ALF-6000+ from Mini-Circuits with typical and maximum insertion loss from DC up to 6 GHz of 0.7 dB and 1.2 dB, respectively, typical insertion loss at 6.93 GHz of 3 dB, and rejection higher than 37 dB between 8.2 GHz and 10 GHz [27]. In an attempt to generate circuits replicating, or coming close to, this performance, we use the simple genetic optimization criterion shown in Table 7. S_{11} and S_{22} are not used in the lowpass filter generation test and neither is the phase of S_{12}. From 0.5 to 6.0 GHz we aim for $|S_{12}| = 1$ and from 7.0 to 10 GHz the target is $|S_{12}| = 0$. We don't use any condition at 6.5 GHz, meaning that $|S_{12}|$ at this frequency will adapt to achieve the best tradeoff between the other requirements. When selecting a target profile one needs to consider what can be realistically achieved with the given

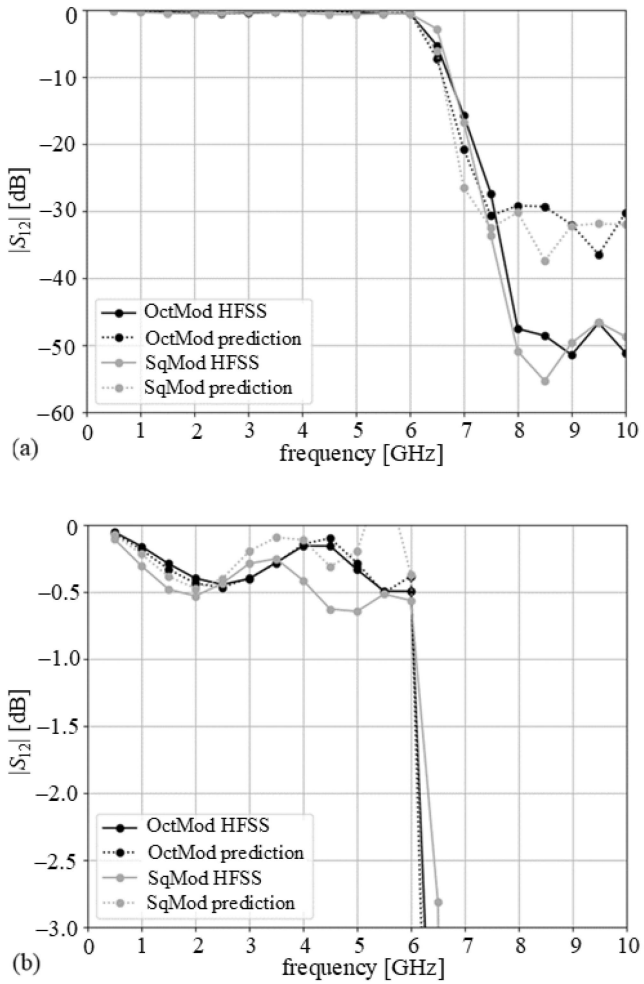

FIGURE 12. $|S_{12}|$ in dB of the low-pass filters with best results in HFSS simulations for each circuit model, together with the ML-model predictions from the genetic optimization. (a) The performance is shown between 0–60 dB. (b) Zoom-in to better visualize the performance in the pass-band.

circuit geometry, the resolution of the discrete elements, and other factors that influence the performance. In our case we find that aiming for $|S_{12}| = 0$ at 6.5 GHz impacts the performance in the pass-band negatively, with a premature drop to smaller $|S_{12}|$ values near the upper band-edge. With a larger circuit geometry and/or a circuit using multiple layers such as in [28], it might be possible to achieve a steeper transition between the pass-band and the stop-band.

For each circuit model we generate 10 different candidate circuit arrays that are simulated in HFSS. We then select the filter for each circuit model with the lowest maximum insertion loss in the pass-band while simultaneously achieving at least the typical rejection of the ALF-6000+ of 37 dB between 8.2 GHz and 10 GHz. The results are displayed in Fig. 12(a) and (b) for the ranges of −60 dB to 0 dB and −3 dB to 0 dB, respectively and for both the HFSS verifications and the predictions by the ML-models after the genetic optimization. Looking at the HFSS simulation results we note that the maximum and average pass-band insertion loss for the square-model filter is 0.64 dB and 0.43 dB, respectively, and the corresponding values for the Octagon-model filter are 0.49 dB and 0.30 dB. Both filters achieve a rejection of more than 45 dB within 8–10 GHz. When comparing with the ML-model predictions we observe that both models overestimate $|S_{12}|$ in the stop-band. which was the case for several of the generated filters. The reason for this should be that it is difficult for the models to learn to distinguish between very small S-parameter values since such errors are not penalized much during training. A solution for this could be to train ML-models used for filter generation also on S-parameters in log-scale. We also observe a better agreement between simulation and model prediction for the Octagon-model. The circuit matrices of the two filters are shown in Fig. 13(a)–(b).

We believe these results show that low-pass filters with competitive performance can be achieved with ML-assisted circuit design, even with modest efforts. Better performance might be achieved by generating more filter candidates in the genetic optimization, improving the datasets or using larger circuit area and/or finer discretization resolution. One way to improve the dataset could be to include the simulation results from this optimization process, which could enable re-trained ML-models to produce even better filters. For example, we note that for some of the generated filters the stop-band rejection get significantly worse at 10 GHz, and the simulated data for these filter tests could help the ML-models to improve on such details that are important for specific circuit types. Another suggestion is to use ML-models and optimization criteria that are more sensitive to very small $|S_{12}|$ values.

C. FABRICATION AND MEASUREMENTS OF LOW-PASS FILTERS

The best performing low-pass filter for each circuit model is fabricated with the help of EuroCircuits. For the circuit created using the Octagon-model we use a taper to achieve an impedance match with 50 Ω as shown in Fig. 13(c) for an HFSS model of the filter. Fig. 13(d)–(e) shows photographs of the fabricated circuits for the Square-model and the Octagon-model, respectively.

We remove the impact of the feeding lines from the measurements by using a de-embedding process [29] where the S-parameters S_c of the circuits of interest are extracted from the S-parameters S_m measured for the whole fixture and S_f of the feeding lines. The measured S-parameters are converted to ABCD-parameters and the ABCD parameters of a circuit of interest are obtained using (1) and subsequently converted to S-parameters to yield S_c.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}_c = \begin{bmatrix} A & B \\ C & D \end{bmatrix}_f^{-1} \begin{bmatrix} A & B \\ C & D \end{bmatrix}_m \begin{bmatrix} A & B \\ C & D \end{bmatrix}_f^{-1} \quad (1)$$

Fig. 14 shows the measured S-parameters for the complete fixture for both circuits as well as the S-parameters for the circuits extracted using the de-embedding procedure. The transmission lines between the ports and the circuits

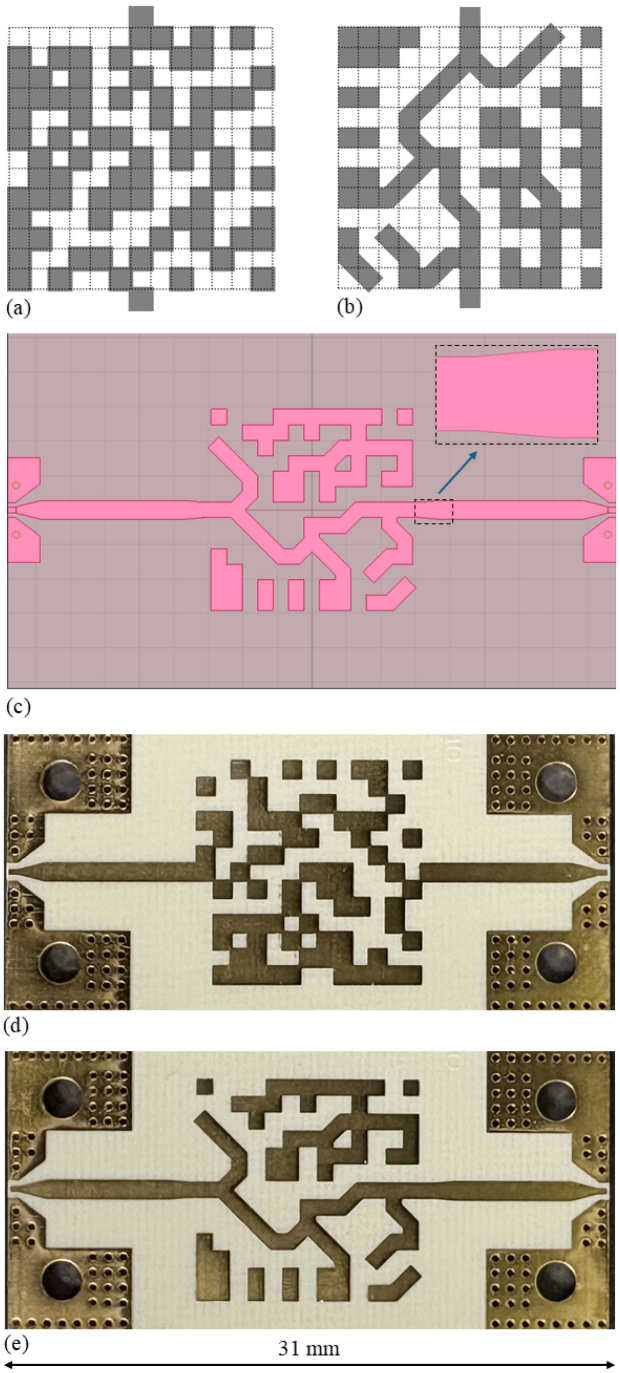

FIGURE 13. The best low-pass filter circuits for the two circuit models. (a) Square-model. (b) Octagon-model. (c) Top view of HFSS model of the Octagon-model filter with taper added to provide match to 50 Ω. (d) Fabricated Square-model filter. (e) Fabricated Octagon-model filter.

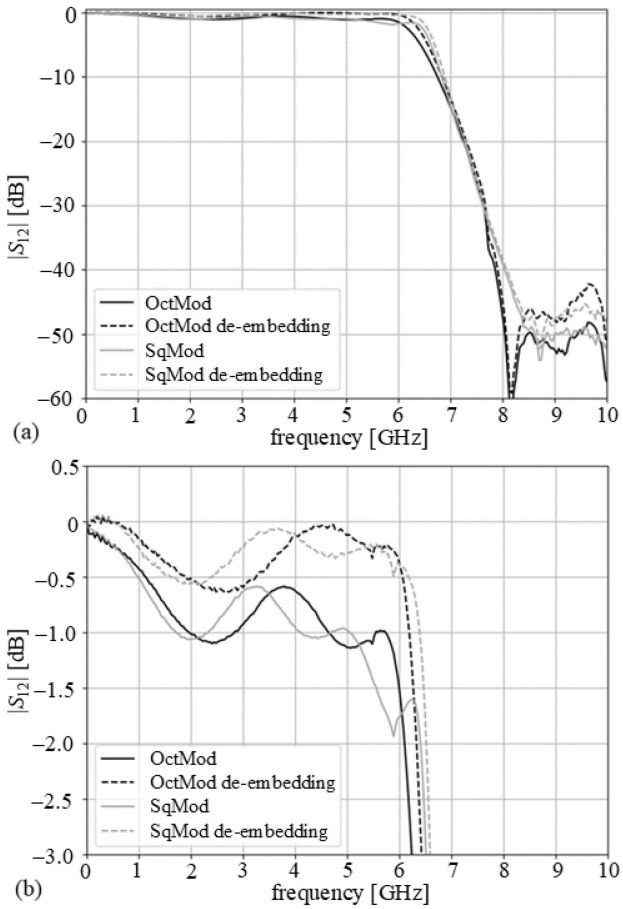

FIGURE 14. Measured $|S_{12}|$ in dB of the low-pass filters with and without de-embedding of the S-parameters of the transmission line feeds. (a) Using y-axis scale of 0–60 dB. (b) Zoom-in with clearer view of the pass-band.

have lengths of 7.8 mm which degrades the performance considerably and the maximum insertion-loss in the pass-band is now 1.5 dB and 1.9 dB for the Octagon-model filter and the Square-model filter, respectively. The performance is as exected much better when de-embedding of the transmission line feeds is used and the worst pass-band performance is now 0.63 dB and 0.57 dB for the two filters, with the Octagon model filter being slightly worse. We note however that the de-embedding results in S_{12} being slightly larger than 1 close to DC which indicates some inaccuracy. It is also worth emphasizing that the performance of the Octagon-model filter has a small degradation due to the taper used for impedance matching. Regardless, the performance is quite similar to the HFSS verification results displayed in Fig. 12.

VI. CONCLUSION

We compare two circuit models for ML-assisted circuit design, one newly proposed model utilizing a combination of octagonal elements and square shaped elements with different rotations and translations with respect to the circuit grid, and another with only square-shaped metal pieces enlarged 20% with respect to the grid. Datasets with 60,000 circuit matrices and their corresponding S-parameters are generated for each circuit model, and we show that more accurate ML-models can be trained for the circuit models with wider diagonal connections, which in turn translates to better performance for the genetic optimization process used to design new circuits.

In addition, the cumulative distribution functions of $|S_{12}|$ indicate that circuits created using the Octagon-model typically has smaller loss than circuits created using the Square-model. Furthermore we demonstrate with two different tests that circuits built with the new Octagon-model can be expected to have higher tolerance to manufacturing imperfections. In a test where a width perturbation of 50 μm was added to all diagonal connections, Octagon-model circuits had 33% smaller RMSE penalty than Square-model circuits on average, with respect to the S-parameters of the corresponding circuits without permutations. In another test where circuit elements forming diagonal connections were scaled in size with $\pm$2.5%, the RMSE penalty for the Octagon-model circuits were reduced with 40–50% compared to Square-model circuits. We also demonstrate generation of low-pass filters with competitive performance compared to a commercially available 6 GHz filter. The filter created using the Octagon-model performs best in terms of insertion-loss, with maximum and average loss of 0.49 dB and 0.30 dB in HFSS verification, respectively, while the filters for both circuit models has stop-band rejection of at least 45 dB. We also fabricate filters and find good agreement between measurements and simulations.

REFERENCES

[1] Z. Liu, E. A. Karahan, and K. Sengupta, "Deep learning-enabled inverse design of 30–94 GHz $P_{sat,3dB}$ SiGe PA supporting concurrent multiband operation at multi-Gb/s," *IEEE Microw. Wireless Compon. Lett.*, vol. 32, no. 6, pp. 724–727, Jun. 2022.

[2] E. A. Karahan, Z. Liu, and K. Sengupta, "Deep-learning-based inverse-designed millimeter-wave passives and power amplifiers," *IEEE J. Solid-State Circuits*, vol. 58, no. 11, pp. 3074–3088, Nov. 2023.

[3] E. A. Karahan, A. Gupta, U. K. Khankhoje, and K. Sengupta, "Deep learning based modeling and inverse design for arbitrary planar antenna structures at RF and millimeter-wave," in *Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting*, 2022, pp. 499–500. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9887077

[4] E. A. Karahan et al., "Deep-learning enabled generalized inverse design of multi-port radio-frequency and sub-terahertz passives and integrated circuits," *Nature Commun.*, vol. 15, no. 10734, Dec. 2024. [Online]. Available: https://doi.org/10.1038/s41467-024-54178-1

[5] S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez, "Inverse design in nanophotonics," *Nature Photon.*, vol. 12, no. 11, pp. 659–670, Nov. 2018.

[6] S. So, T. Badloe, J. Noh, J. Rho, and J. Bravo-Abad, "Deep learning enabled inverse design in nanophotonics," *Nanophotonics*, vol. 9, no. 5, pp. 1041–1057, May. 2020.

[7] R. Pestourie, Y. Mroueh, T. V. Nguyen, P. Das, and S. G. Johnson, "Active learning of deep surrogates for PDEs: Application to metasurface design," *NPJ Comput. Mater.*, vol. 6, no. 164, Oct. 2020.

[8] C. Xie et al., "Deep learning assisted inverse design of metamaterial microwave absorber," *Appl. Phys. Lett.*, vol. 123, no. 18, pp. 7607–7622, Oct. 2023.

[9] Y. Huang, X. Liu, M. Liu, J. Chen, W. Du, and Z. Liu, "Deep learning-based inverse design of multi-functional metasurface absorbers," *Opt. Lett.*, vol. 49, no. 10, pp. 2733–2736, 2024.

[10] S. Thakkar, L. Szymanski, J. Ruiz-Garcia, G. Gok, and A. Grbic, "Inverse design of perfectly-matched metamaterials," in *Proc. IEEE/MTT-S Int. Microw. Symp.*, 2023, pp. 347–350.

[11] K. Sattari, Y. Xie, and J. Lin, "Data-driven algorithms for inverse design of polymers," *Soft Matter*, vol. 17, no. 33, pp. 7607–7622, Jul. 2021.

[12] M. Röding, V. Wåhlstrand Skärström, and N. Lorén, "Inverse design of anisotropic spinodoid materials with prescribed diffusivity," *Nature*, vol. 12, no. 1, pp. 2045–2322, Oct. 2022.

[13] S. B. Kim and A. A. Linninger, "Optimization of complex column networks with hybrid genetic algorithm," *Comput. Aided Chem. Eng.*, vol. 31, pp. 1597–1601, Jan. 2012.

[14] H. Lv, L.-Y. Xiao, H.-J. Hu, and Q. H. Liu, "A spatial inverse design method (SIDM) based on machine learning for frequency-selective-surface (FSS) structures," *IEEE Trans. Antennas Propag.*, vol. 72, no. 3, pp. 2434–2444, Mar. 2024.

[15] H. Zhou, J.-R. Perez-Cisneros, S. Hesami, K. Buisman, and C. Fager, "A generic theory for design of efficient three-stage Doherty power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 70, no. 2, pp. 1242–1253, Feb. 2022.

[16] "Microstrip impedance calculator - engineering calculators & tools," 2003. [Online]. Available: https://www.allaboutcircuits.com/tools/microstrip-impedance-calculator/

[17] *Ansys, Inc.*, "ANSYS HFSS, High Frequency Structure Simulator," Release 2024 R2, Canonsburg, PA, USA: Ansys Inc.. [Online]. Available: https://www.ansys.com/products/electronics/ansys-hfss

[18] "List of LaTeX mathematical symbols," 2010. [Online]. Available: https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols

[19] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 6402–6413.

[20] Y. Le Cun et al., "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 396–404.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[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, 2014.

[23] H. Drucker and Y. Le Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 991–997, Nov. 1992.

[24] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, 2013, vol. 30, pp. 3–11.

[25] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. 4th Int. Conf. Learn. Representations*, Nov. 2016, pp. 1–14.

[26] M. Ranzato, Y.-L. Boureau, and Y. Le Cun, "Sparse feature learning for deep belief networks," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2007, pp. 1185–1192.

[27] Mini-Circuits, "Thin film surface mount low pass filter ALF-6000," 2023. [Online]. Available: https://www.minicircuits.com/pdfs/ALF-6000.pdf

[28] V. Chenna and H. Hashemi, "Topology-optimized nonintuitive multilayered mm-Wave power amplifiers," in *Proc. IEEE Radio Freq. Integr. Circuits Symp.*, 2025, pp. 279–282, doi: 10.1109/RFIC61188.2025.11082875.

[29] A. Pham, J. Laskar, and J. Schappacher, "Development of on-wafer microstrip characterization techniques," in *Proc. 47th ARFTG Conf. Dig.*, 1996, pp. 85–94.