Towards Physics-Based Generalizable Convolutional Neural Network Models for Indoor Propagation

Aristeidis Seretis and Costas D. Sarris

***Abstract*—A fundamental challenge for machine learning models for electromagnetics is their ability to predict output quantities of interest (such as fields and scattering parameters) in geometries that the model has not been trained for. Addressing this challenge is a key to fulfilling one of the most appealing promises of machine learning for computational electromagnetics: the rapid solution of problems of interest just by processing the geometry and the sources involved. The impact of such models that can "generalize" to new geometries is more profound for large-scale computations, such as those encountered in wireless propagation scenarios. We present generalizable models for indoor propagation that can predict received signal strengths within new geometries, beyond those of the training set of the model, for transmitters and receivers of multiple positions, and for new frequencies. We show that a convolutional neural network can "learn" the physics of indoor radiowave propagation from ray-tracing solutions of a small set of training geometries, so that it can eventually deal with substantially different geometries. We emphasize the role of exploiting physical insights in the training of the network, by defining input parameters and cost functions that assist the network to efficiently learn basic and complex propagation mechanisms.**

***Index Terms*—Machine learning, Convolutional Neural Networks, Radiowave Propagation, Ray Tracing**

I. Introduction

Radio propagation modeling is a prerequisite for the efficient planning and management of any communication system [1]. In indoor environments, propagation modeling can be utilized for the efficient placement of wireless access points (WAPs), for localization purposes, as well as for ensuring appropriate quality of service (QoS) to users. Typically, extensive measurement campaigns are performed to derive a propagation model. However, these are both labor intensive and site-specific; new measurements have to be taken to model a different environment.

As a result, site-general, empirical propagation models have been formulated, enabling approximate, yet fast propagation modeling in a variety of different indoor environments [2], [3]. These models are based on a small number of input parameters, flexible and helpful for preliminary propagation studies. However, site-general models do not capture the wide range of wave propagation mechanisms present in an environment. In such cases, site-specific physics-based solvers, such as ray tracing (RT) and full-wave methods, can be used instead [4], [5]. Such methods utilize detailed information about the geometry of the environment, to model all propagation mechanisms present. Thus, these models are highly accurate, especially when calibrated with measured data [6]. However, their development requires a high level of relevant expertise and computational resources that can be prohibitive for real-time application. Moreover, small changes to the communication channel or the geometry necessitate new runs of the solver.

Machine learning (ML) methods offer an alternative route to formulating propagation models that can potentially combine the efficiency of empirical models with the accuracy of physics-based models [7]–[9]. These methods are based on training a model, such as an artificial neural network (ANN), with measured and/or simulated data corresponding to various quantities of interest, such as path loss and received signal strength (RSS) [10]–[12]. A fundamental challenge for such models is their ability to "generalize": to rapidly and accurately solve problems with specifications (geometry, position and type of transmitter/receiver (T_x/R_x) antennas, frequency of operation) that are beyond those included in the training set. Such "generalizable" models will clearly have a profound impact on propagation modeling studies, overcoming the classical dichotomy between computational efficiency and accuracy that has dominated this area.

To achieve this goal, the model has to essentially "learn" the physics of radiowave propagation during training. Therefore, the challenge of generalizability includes three interconnected problems. The first problem is the choice of input parameters, in addition to the geometry itself, that can assist the training and generalizability of the model. For example, empirical path loss models include parameters such as the number of obstructions between the transmitter and the receiver that can be defined in any new environment [2]. Naturally, such parameters are good candidates as inputs to a learned model. The second problem is the selection of a training set that is relatively compact, yet effective at "teaching" propagation physics to the model in a generalizable manner, without using an exhaustively large training set. Third, the model itself (the type of neural network employed) has to be computationally efficient.

In an effort to meet this fundamental challenge, we present ML models for indoor propagation that can generalize with respect to the geometry, position of transmitter and receiver and operating frequency. Our end goal is to produce models with "electromagnetic vision" [13], i.e. the ability to recognize the propagation characteristics of an environment based on its geometry and the source specification (transmitting antennas and frequency).

The most difficult component of this effort is the generalization with respect to the geometry. Networks that generalize with respect to the position to transmitters and receivers in a fixed environment have been demonstrated before [14], [15]. To that end, we consider the three problems that are associated with the challenge of generalizability. First, we select a small set of input parameters inspired by the physics of

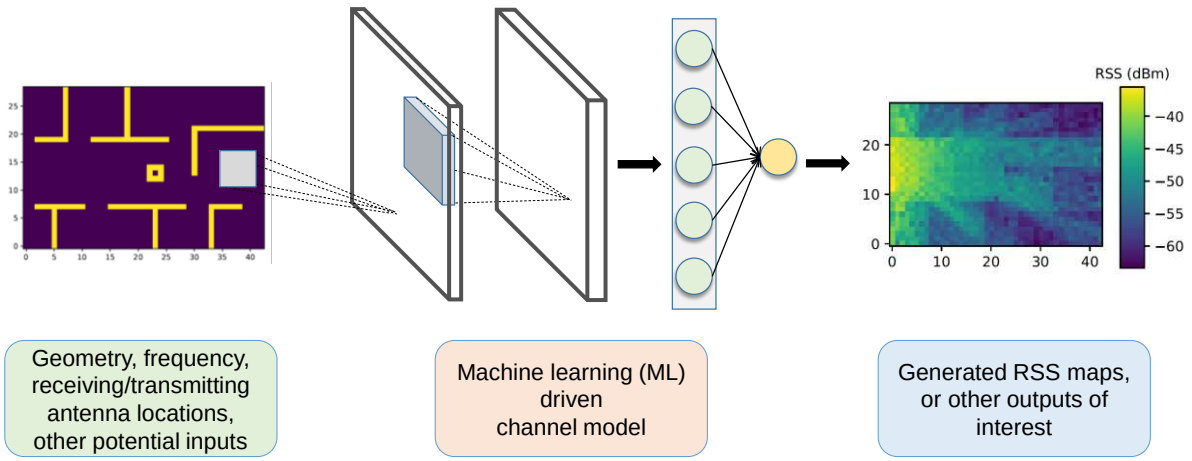


Fig. 1: The main function of the trained CNN-based propagation model. Given a geometry, a transmitting antenna location and receiving points at a given frequency, the CNN is tasked with generating RSS predictions at the specified receiving points.

radiowave propagation. For example, one of these parameters indicates whether the receiver is at line of sight (LoS) or not from the transmitter; another indicates the proximity of a receiver to a diffracting edge. We demonstrate that using these parameters as inputs to the model, along with the geometry, is a key to achieving generalizable models. Second, we build a training set, consisting of basic and complex geometries, that helps the model learn basic and complex indoor propagation phenomena. Third, we use a convolutional neural network (CNN) as our learning structure, driven by cost functions that are physics-informed. A flowchart of our work is shown in Fig. 1. For training, we use data generated by a shooting and bouncing ray (SBR) ray-tracer incorporating the imaging method to adjust the ray paths [16].

The structure of the paper is as follows. In Section II, we give a brief overview of previous work on ML-based propagation models with emphasis on their generalizability, which is the main challenge that our paper is focused on. In Section III, we introduce our approach for the formulation of generalizable indoor propagation models and describe the data generation and training process. In Section IV, we present results regarding the generalizability of our ML model with respect to new transmitter/receiver positions and frequencies, yet within the training geometries. The generalizability of the trained CNN to new geometries is demonstrated successfully in Section V. We also show the important role of the physics-inspired parameters (added as inputs to the model) in the successful generalizability of the model. Summary and conclusions of our work are included in the final section.

II. ML-based Propagation Models

A. Generalizability

ML-based models, such as ANNs and CNNs, have been extensively used for propagation modeling over the past few years, in a variety of different environments [7], [17]. ML models are usually trained by RSS measurements [18]–[20]. Hence, their scope is often limited to propagation modeling in small-scale areas, where it is easier to collect an adequate volume of measured data. Most measurement-based ML models generalize with respect to different transmitter/receiver locations inside an already known geometry. This is the case in user localization applications [18], [21]. Others are trained using data collected from specific user trajectories, and then tested on data from different routes that a user follows in user tracking scenarios [12], [20], [22], [23]. The geometry remains fixed in these cases as well; the test routes are selected inside the same geometry. An exception can be found in [24], where measurements from many different types of outdoor areas (urban, semi-urban, rural) were used to train the ML model.

Training ML models requires a large volume of training data. Hence, creating highly generalizable propagation models using only measured data is an expensive and time-consuming process. Instead, the use of simulated data can enlarge training datasets at a cost that is significantly smaller to that of a measurement campaign. Thus, data generated by physics-based methods, such as RT [14], [25]–[27], or other sources of data that are readily available (like satellite images in outdoor environments [28], [29]) have been used to train ML models. In indoor environments, where RT-based training is more extensively used, the availability of synthetic data offers more possibilities for geometry generalization. However, research so far has been restricted to specific geometries. For example, [25] presented an ML model trained with RT-generated RSS values in urban environments of uniformly and non-uniformly distributed buildings. However, the model was evaluated at different receiver points than those used for training, rather than in new geometries. A similar procedure was followed in [15], where the network was trained to generalize to new receiving points and frequencies in a fixed geometry. Likewise, in [14], the designed ANN was trained using RT-generated data inside two indoor environments. Then, it was tested on different receiving points in those specific geometries.

B. Input features

Many physics-based features have been used in ML-based propagation modeling in the past. Most of them are implementation-dependent [7]. For example, if the model has

to be used in a wide range of frequencies, a feature should account for the relationship of the output of the model to frequency. Others may indirectly connect the output of the ML model to a generalizable feature. For example, a feature defining whether a receiving point is at the line of sight (LoS) of the transmitter can indirectly account for the geometry (generalizable feature) or the complexity of the environment.

Most measurement-based papers use simple input features, such as the distance between T_x and R_x or the height of transmitting/receiving antennas [7]. The same applies to papers that rely on path loss models to augment their dataset or use them as a benchmark to evaluate the accuracy of their ML model. The features used are similar to the ones used by the site-general, path loss models [1]–[3]. These include the frequency, the distance and the number of floors between T_x and R_x and others. One of the features that can give useful information about the propagation environment is the path loss exponent, that can be calculated from tables based on the type of the indoor environment, such as an office or a corridor. However, this feature does not account for specific layouts of a given environment.

ML propagation models that are trained by simulated data usually incorporate additional input features that are solver-specific [26]. For example, the model of [30] included parameters such as the number of reflected and diffracted rays that reach the receiver according to a ray-tracer. Moreover, a variety of different input features can be used to account for the geometry of the modeling environment. Those include features describing LoS conditions, transmission losses through walls, diffraction losses over edges, the position and the height of obstacles between the transmitter and the receiver [11], [27], [31], [32]. While some of them are straightforward and easy to compute, such as the LoS points, others are more involved. For example, the authors of [31] used the reflection losses from walls and the orientation of walls relative to the receiving points, as computed via the appropriate reflection coefficients. Computing these parameters requires significant computation time that compromises the advantages of such a model compared to a standard ray-tracer.

C. ML model

Next, we discuss ML models that have been used in propagation modeling studies. Older propagation modeling papers used ANNs or simple ANN-based models, such as radial basis function (RBF)-ANNs [7]. However, these networks do not exploit the parameter-sharing scheme of CNNs [33]. The CNNs use 2-dimensional filters that share parameters, i.e. the same filter is applied to different parts of the input [33]. This is an efficient way of learning mappings between geometry-encoding features and the output of the CNN. CNNs are more efficient than ANNs [18]. Recently, they have been used to encode geometry-based information into their input channels [11], [18], [23], [34]. Other ML models, such as RNNs, are more suitable and often used when modeling data with temporal dependencies [12]. Finally, more complex ML models can be used, such as models that are based on ensemble methods [10]. However, these models are more computationally intensive than CNNs.

In this paper we use the CNN as our ML model for two reasons. First, the parameter-sharing scheme of the CNN fits our goal of encoding the geometry in our input parameters. Second, CNNs strike an appealing balance between accuracy and efficiency.

III. Proposed Approach For Generalizable Models

This section describes our approach for building generalizable ML-based indoor propagation models. To that end, we address the following challenges:

1) Determining a set of input features that leads to generalizable propagation models, that can also deal with new geometries.
2) Choosing training geometries that can help the ML model in its task of geometry generalization.
3) Choosing an ML model that is computationally efficient and accurate.
4) Choosing cost functions that account for the physics of propagation in an environment.

In the following, we elaborate on these directions, introduce the solver used for generating our dataset, and explain our evaluation metrics. To begin with, we discuss the general function of an ML-based propagation model.

A. ML propagation model

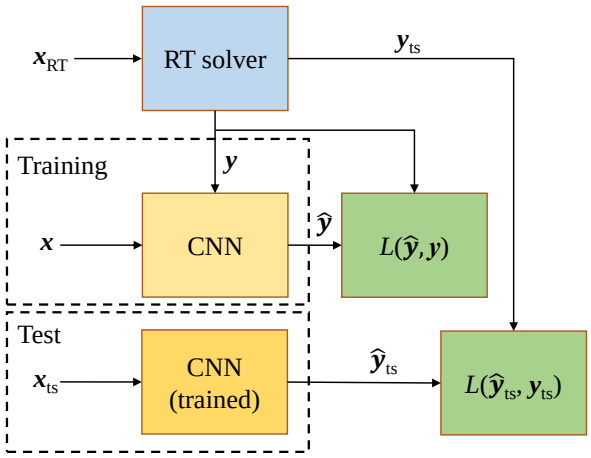
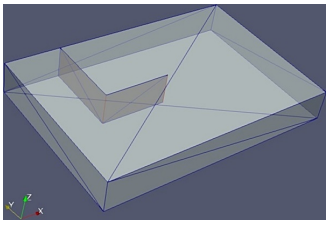
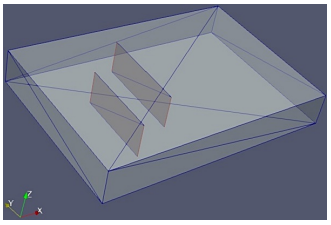


Fig. 2: The flowchart of the proposed CNN-based propagation model.

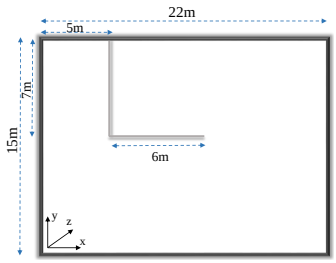
The flowchart of the CNN-based propagation model is shown in Fig. 2. The ray-tracer is first configured based on input parameters, $\mathbf{x}_{\text{RT}}$, that include the coordinates of the receiving and transmitting points, the frequency, and other RT-related parameters, such as the number of allowed reflections, diffractions and transmissions that a ray can undergo. Then, the solver computes RSS values, $\mathbf{y}$, at the specified receiving points. These values are used as target values for the CNN, along with appropriately chosen input features $\mathbf{x}$ that are used as its input. The CNN is trained on the generated pairs of $(\mathbf{x}, \mathbf{y})$, aiming to minimize its loss function $L(\hat{\mathbf{y}}, \mathbf{y})$, where $\hat{\mathbf{y}}$ are its predicted RSS values. After training, the CNN



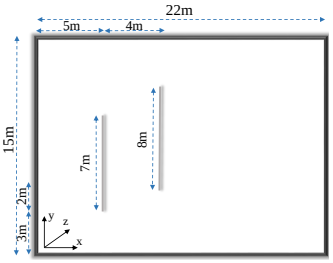
(a) 3D view of the 1st geometry



(b) 3D view of the 2nd geometry

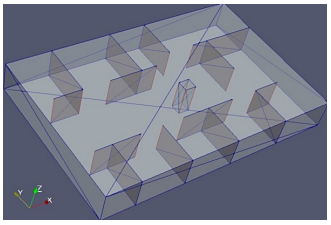


(c) 2D view of the 1st geometry

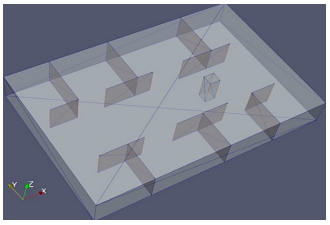


(d) 2D view of the 2nd geometry

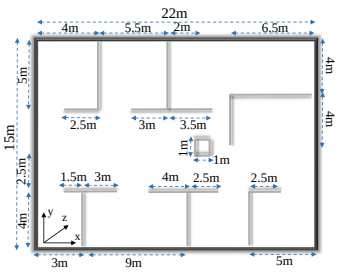
Fig. 3: 3- and 2-dimensional figures for two of the basic geometries. Also visible in the upper figures are the diffracting edges (in red) and the different type of materials of the inner and outer walls, in each one of them.



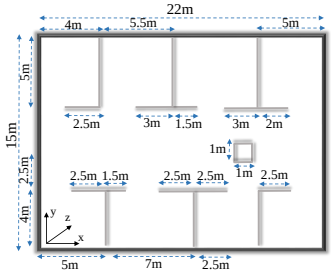
(a) 3D view of the 1st geometry



(b) 3D view of the 2nd geometry



(c) 2D view of the 1st geometry



(d) 2D view of the 2nd geometry

Fig. 4: 3- and 2-dimensional figures for two of the complex geometries. Also visible in the upper figures are the diffracting edges (in red) and the different type of materials of the inner and outer walls, in each one of them.

is tested on different input features $\mathbf{x}_{\text{ts}}$, corresponding to different frequency and transmitting/receiving points, or new geometries altogether, not used during the training process. The predictions of the CNN on its test set, $\mathbf{y}_{\text{ts}}$, are evaluated by computing the test loss function, $L(\hat{\mathbf{y}}_{\text{ts}}, \mathbf{y}_{\text{ts}})$.

B. Geometry and RT configuration

The modeling environment consists of a small office, 15 m wide, 22 m long and 3 m in height. These dimensions remain fixed among the different geometries that are used to train and test the CNN. However, the placement of the inner walls differs in each one of them. The outer walls of the office, including the floor and ceiling, are concrete. The inner walls, separating the rooms, are made of plasterboard. They have a thickness of 0.02 m, a dielectric permittivity of 2.5 F/m and a conductivity of 0.03 S/m, while the outer walls are 0.02 m thick, have a permittivity of 9 F/m and a conductivity of 0.09 S/m. These values are typical of materials used in office environments.

We start by creating some basic geometries, horizontally and/or vertically placing up to 4 single walls. The goal of these simple geometries is to help the CNN learn basic wave propagation mechanisms, such as diffraction from an edge, reflection from and transmission through one or more walls. As an example, Fig. 3 illustrates two of these basic geometries. In total, 16 such geometries are used to train the CNN. We also add more complex geometries to the training dataset, such as the ones shown in Fig. 4. These geometries represent more realistic and complex scenarios for the CNN, aiming to help it learn multipath features of the communication channel. In total, 12 such geometries are created and added to the 16 basic ones. All training geometries are provided in Appendix I.

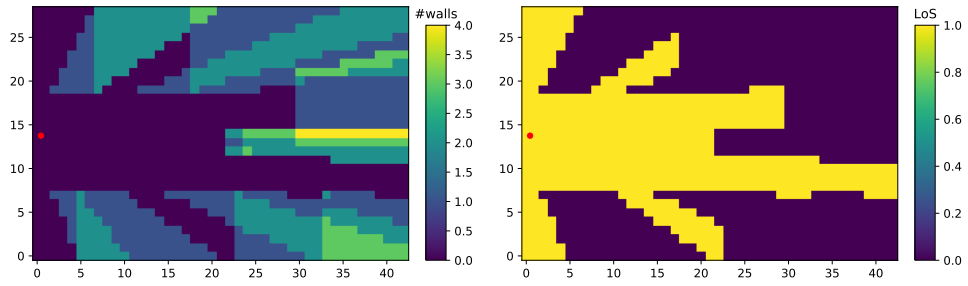
An in-house RT solver [16] is used to model radiowave propagation inside the various geometries created. The solver is based on the SBR method. It uses triangular cross-section ray tubes and the imaging method to adjust the ray paths, to determine whether rays may reach a receiver [35]. The ray-tracer is configured after performing a convergence study. This is done by monitoring the convergence of the RSS values computed by the solver, as we increase the number of reflections, transmissions and diffractions accounted for per ray. The simulated RSS values converge after a number of six reflections and four transmissions. The same procedure is followed for the number of allowed diffractions, where using more than one has negligible impact on the simulated RSS values.

Both receiving and transmitting antennas are assumed to be vertically polarized half-wave dipoles. The transmit power is 15 dBm. The simulated RSS points lie along 2-dimensional grids (RSS maps) that can be defined as the set of points $A = \{R(x, y, z) : x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}, z = z_o\}$, where z_o is a constant value and $x_{min}, x_{max}, y_{min}, y_{max}$ are the sampling limits along the x and y axis of the geometry, respectively. The dimensions of the grid are $(n_w, n_l) = (29, 43)$, where n_w is the number of grid points along the width of the office, and n_l is the number of grid points along its length. The resulting RSS maps that the solver generates are used as a benchmark to evaluate the performance of the CNN.

C. Input Features

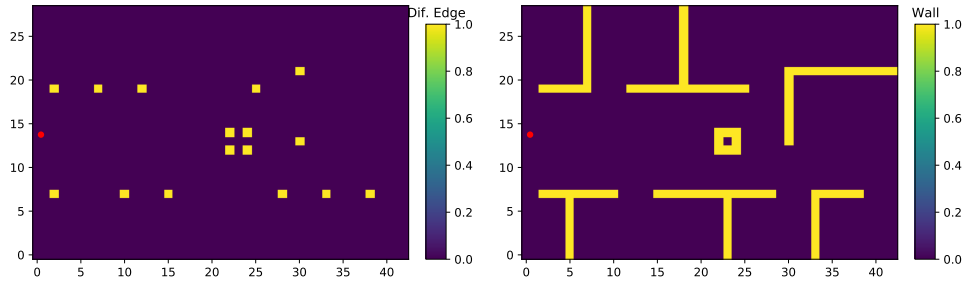
Propagation from the transmitter to a receiver is possible through direct, reflected, transmitted and diffracted waves. We include input features that are connected to these propagation mechanisms. In particular, we use the following input features:

1) the coordinates of the receiving and transmitting antennas inside the office;
2) the frequency of operation;



(a) The number of obstructions

(b) The LoS points



(c) The closest points to the diffracting edges

(d) The geometry

Fig. 5: An example of the 2-dimensional representation of the geometry-based features on a horizontal plane for the geometry of Fig 4a. In clockwise order, the number of obstructions, the LoS points, the location of diffracting edges, and the geometry of the environment. The location of T_x is noted with red.

3) the number of obstructions between T_x and R_x;
4) the presence of LoS at each receiving point;
5) the presence of a diffracting edge close to a receiving point;
6) the layout of the geometry of the environment.

The first feature corresponds to as many input features as the number of transmitting or receiving coordinates that are trainable (not constant). Regarding the third feature, obstructions are the inner walls of each geometry. Generally, they can represent any object that intersects the line from T_x to R_x. Regarding the fifth feature, the closest to each diffracting edge receiving point is used. However, this is a hyperparameter that can be tuned. The first two features are fixed for each run of the solver, while the other four features are computed for each receiving point. As an example, a plot of these four features for the geometry of Fig. 4a is shown in Fig. 5. It should be noted that these features are computed on each horizontal plane, $z = z_o$. Moreover, they are not part of the input to the ray-tracer, $\boldsymbol{x}_{\text{RT}}$, unlike the first two features.

Regarding the first two features, the transmitter is located at $(x_T = 0.25\text{m}, y_T, z_T = 2.5\text{m})$, while the receiver is at (x_R, y_R, z_R). The range of their coordinates, as well as the range of the frequencies used, are listed in Table I. The features that vary, $\xi = f, x_R, y_R, z_R, y_T$, are sampled according to the following formula:

$$\xi_i = \xi_{min} + i \times \frac{\xi_{max} - \xi_{min}}{N_\xi} \tag{1}$$

where $i = 0, 1, 2, ..., N_\xi$ is an index to the i-th sample, ξ_i, of a given input feature, N_ξ is the total number of sampling points, and ξ_{min} and ξ_{max} is the minimum and maximum value of each feature, respectively. The increments of Table I are computed by setting the required number of sampling points N_ξ for each feature in the fraction of Eq. (1). For the validation and test set, we use random samples of the input features, instead of selecting the samples in a deterministic way (as we did in Eq. (1)). The validation set is used for tuning the hyperparameters of the CNN, such as the number of layers, nodes per layer, and the learning rate. The test set is used to evaluate the CNN; it represents various propagation scenarios outside its training set.

TABLE I: RT-used variable input features

Input Features	Min. Value	Max. Value	# Samples
f (GHz)	5.25	5.35	5
x_R (m)	0.5	21.5	43
y_R (m)	0.5	14.5	29
z_R (m)	0.5	2	4
y_T (m)	4.5	13.5	10

Each input feature is represented by a 2-dimensional input channel of the same size as that of the simulated grid of our geometry. Stacking the input channels on top of each other creates a 3-dimensional input tensor x_t for a single sample to the CNN, i.e. for a sample that has constant input channels of $z = z_o, f = f_o$ and $y_T = y_{To}$. Its dimensions are $(n_w, n_l, n_c) = (29, 43, 9)$, where n_c is the number of different input channels used. Hence, a vertical slice along the input channels represents the input features for a single receiving point in space. An example of the 3-dimensional input to the CNN is shown in Fig. 6. The input tensor x_t to the CNN is paired with its corresponding RSS map, a vector $\boldsymbol{y}$ containing

$n_w \times n_l = 1247$ RSS entries, one for each point in the 2-dimensional grid. Then, all pairs of $(x_t, \boldsymbol{y})$ are passed on to the CNN for its training.

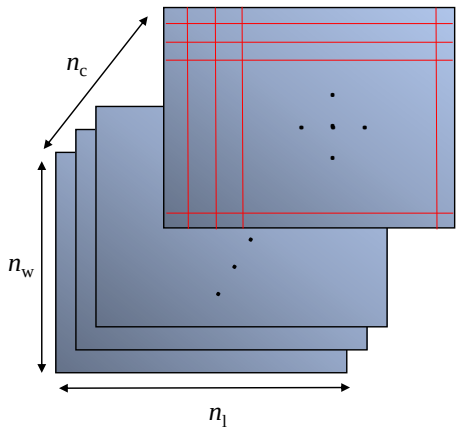


Fig. 6: The 3-dimensional representation of the input features. Also visible is the simulation grid in the topmost channel.

D. CNN

The specific architecture of the CNN is based on the computed error on the validation samples, while evaluating different sets of hyperparameters. These included varying numbers of convolutional layers and filters per layer, and the convolutional and padding stride [33]. We selected 4 convolutional layers for the CNN, since the validation error of that architecture was very close to the error of deeper CNNs we tried, while also being faster to train. The architecture of the CNN is depicted in Fig. 7. Each hidden layer consists of a convolutional and an average pooling layer [33]. The initial input to the CNN, a batch of RSS maps, is convolved with the filter parameters (weights) of the first layer. Batch-normalization [36], which normalizes the input into each layer based on the statistics (mean and variance) of each training batch, is applied to the convolutional output, followed by a $tanh$ activation. The resulting activation (feature map) is passed through an average pooling layer, that is used to reduce the size of the input data. This is done by only keeping the average value of the sections of the feature map that the pooling layer is applied to. The output of the first layer is used as an input for the second layer. This process is repeated for every hidden layer, to generate the output of the forward pass of the CNN. Then, a backward pass will generate the weight increments for the filters of the network, to minimize its cost function. As shown in Fig. 7, the number of channels increase as we move deeper in the network, by using an increased number of filters. More specifically, we use 16, 32, 64 and 128 filters for each of the hidden layers, respectively. Finally, the output of the convolutional part of the CNN is flattened and passed through a single feed-forward layer of 23,760 nodes, which produces the final output of the network $\hat{\boldsymbol{y}}$. The output has the same dimensions as its target $\boldsymbol{y}$.

We also experiment with 3 different cost functions. First, we use an L_1 cost function:

$$L_1 = \frac{1}{N} || \, |\mathbf{P}| - |\hat{\mathbf{P}}| \, ||_1 \tag{2}$$

where $|\mathbf{P}|$ is a vector containing the target RSS values of the batch, $|\hat{\mathbf{P}}|$ is the prediction vector of the network, $|| \cdot ||_n$ corresponds to the n-norm, and N is the total number of receiving points of the batch. An L_2 cost function is also used:

$$L_2 = \frac{1}{N} || \, |\mathbf{P}| - |\hat{\mathbf{P}}| \, ||_2 \tag{3}$$

Finally, the third cost function is a modified L_2 cost function that penalizes differently LoS and nLoS receiving points. The L_2 cost function for the LoS points is computed as follows:

$$L_{2,\,\text{LoS}} = \frac{1}{N_{\text{LoS}}} || \, |\mathbf{P}_{\text{LoS}}| - |\hat{\mathbf{P}}_{\text{LoS}}| \, ||_1 \tag{4}$$

where N_{LoS} is the number of LoS receiving points. Likewise, the L_2 cost function of the nLoS points is:

$$L_{2,\,\text{nLoS}} = \frac{1}{N_{\text{nLoS}}} || \, |\mathbf{P}_{\text{nLoS}}| - |\hat{\mathbf{P}}_{\text{nLoS}}| \, ||_2 \tag{5}$$

where N_{nLoS} is the number of nLoS points. Then, a modified cost function, $L_{2,\,\text{mod}}$ is defined as a weighted sum of the LoS and nLoS L_2 cost functions:

$$L_{2,\,\text{mod}} = w_{\text{LoS}} * L_{2,\,\text{nLoS}} + (1 - w_{\text{LoS}}) * L_{2,\,\text{LoS}} \tag{6}$$

This type of cost function is selected based on our intuition that nLoS points may be more difficult to optimize for. Finally, an L_2 regularization penalty term is applied to every cost function to combat overfitting. As an example, the final form of the L_2 cost function is:

$$L_2 = \frac{1}{N} || \, |\mathbf{P}| - |\hat{\mathbf{P}}| \, ||_2 + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \tag{7}$$

where λ is a hyperparameter that controls the regularization and $\mathbf{w}$ is a vector containing all the weights of the network. Finally, the CNN's training took less than an hour on a GTX 1080, whereas during test time, predictions were generated at real-time (< 1s).

E. Evaluation metrics

The predictions of the network are evaluated on the test set by computing the L_1 error (or mean absolute error (MAE)) of Eq. (2) over all test samples. Due to the strong multipath in the office environment, a comparison of fast-varying RSS values solely on an L_1 norm is not reliable. Hence, as an additional metric to quantify not only the deviations in the magnitude between the predicted and target RSS values, but also their features, the feature selective validation (FSV) algorithm is used [37]. Succinctly, its three metrics are the amplitude difference measure (ADM), the feature difference measure (FDM) and the global difference measure (GDM). From a propagation perspective, a good ADM score corresponds to a good prediction of the slow fading profile of the signal and relevant parameters such as the path loss exponent. The FDM score represents the accuracy of the model with respect to fast fading effects. Apart from a single numerical value for each

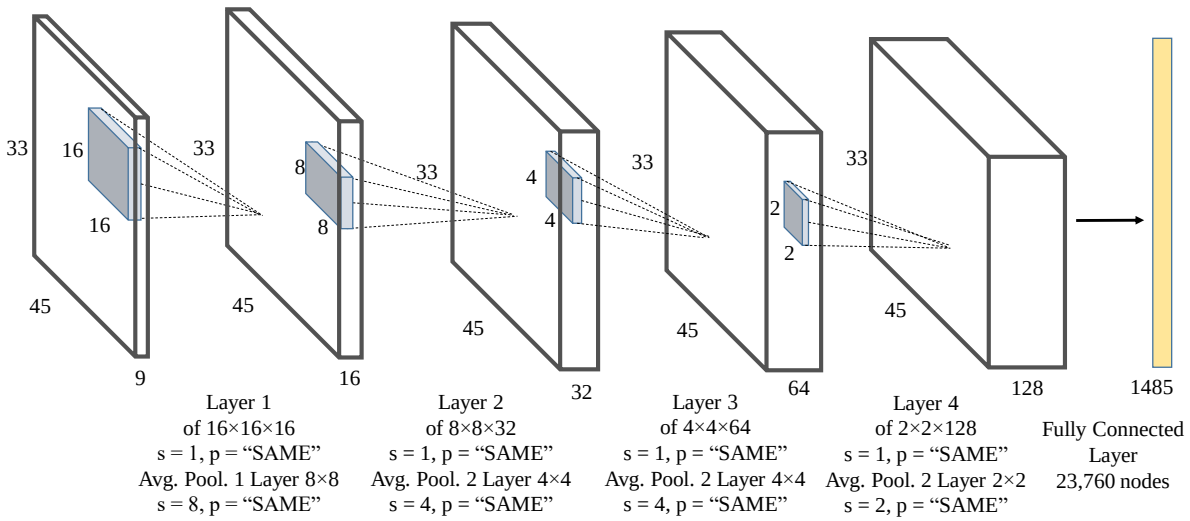


Fig. 7: The high-level CNN architecture from its input to the output. Regarding notation; s stands for stride, and p is the type of padding. Filter parameters are specified according to the form $(f_w \times f_h \times f_n)$, where f_w is the width of the filter, f_h its height, and f_n the number of filters used.

of the three metrics, the method includes a natural language interpretation of the ADM, FDM and GDM scores, as shown in Table II. More information about the FSV algorithm is provided in Appendix II.

TABLE II: FSV interpretation

FSV metric (M) numeric value	FSV interpretation
$M \leq 0.1$	Excellent
$0.1 \leq M \leq 0.2$	Very Good
$0.2 \leq M \leq 0.4$	Good
$0.4 \leq M \leq 0.8$	Fair
$0.8 \leq M \leq 1.6$	Poor
$1.6 \leq M$	Very Poor

IV. Simulation Results: Training Geometries

A. Impact of cost functions

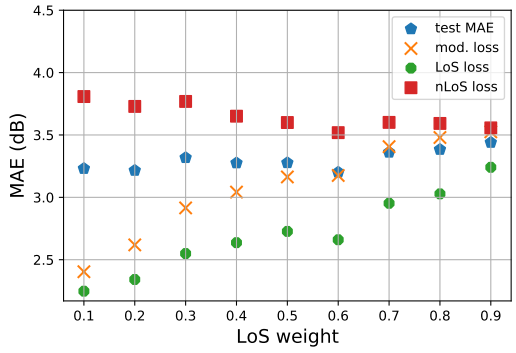
We first investigate how the different cost functions impact the accuracy of the CNN. This is done by training the CNN using the three cost functions described in Section III; the L_1, L_2 and $L_{2,\text{mod}}$ cost functions. Then, we evaluate the generalization abilities of the CNN at the 28 fixed geometries of Appendix I. This is done by computing the test MAE at different frequencies, receiving and transmitting points than those that the CNN is trained on. The results when minimizing the three cost functions are presented in Table III.

TABLE III: Error metrics for the training geometries

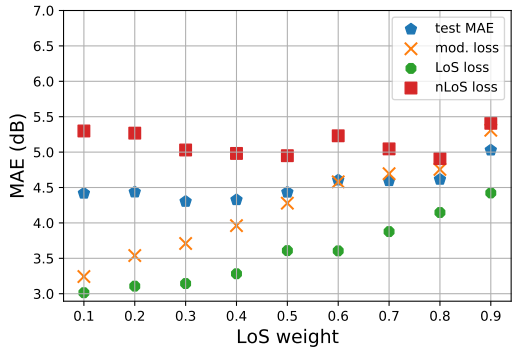
Cost Function	Samples	MAE (dB)	ADM	FDM	GDM
L_1	Training	3.6	0.07	0.38	0.386
L_1	Test	4.14	0.076	0.39	0.397
L_2	Training	3.24	0.069	0.352	0.359
L_2	Test	4.27	0.079	0.366	0.374
$L_{2,\text{mod}}$	Training	3.2	0.068	0.345	0.352
$L_{2,\text{mod}}$	Test	4.24	0.072	0.355	0.362

The resulting differences between the test MAE when using the L_1 and L_2 cost functions are not large. However, the L_2 cost results in smaller FDM and subsequently GDM errors for the test samples. This shows that the CNN actually tries to learn more of the fast fading profile of the communication channel. This is expected, since the L_2 cost function penalizes larger errors more than the L_1. For the case of the $L_{2,\text{mod}}$ cost function, we vary w_{LoS} to see how it affects the accuracy of the CNN. Figs. 8a, 8b display the mean MAE, modified, LoS and nLoS L_2 errors for various values of w_{LoS}, for the training and test samples, respectively. Theoretically, increasing w_{LoS} penalizes the L_2 norm of the nLoS points, reducing their corresponding error. The LoS error increases on the other hand, as the network starts overfitting to the nLoS points. Hence, for the training cases, a trade-off exists between these two errors, as can be seen in Fig. 8a. However, the trade-off relationship between LoS and nLoS errors is not that clear for the test cases, as can be observed in Fig. 8b. The error of LoS points is easier for the network to be minimized. This can be deduced by the small reduction in the nLoS error for the training samples of Fig. 8a, as we increase w_{LoS}. Moreover, as the network learns to minimize the nLoS error, it learns in the process helpful latent features (at its filters) that are also applicable for the LoS points.

The best results for the test geometries are achieved for $w_{\text{LoS}} = 0.3$, i.e. assigning more weight for LoS than nLoS points. That may seem surprising, given the fact that nLoS points incur larger errors than LoS points, as can be seen in Fig. 8. However, the error of LoS points is easier for the network to minimize. Moreover, we are interested in the total MAE, irrespective of whether the point is LoS or not, and in the calculation of that specific metric, the ratio of LoS to nLoS points does not matter, contrary to the computation of the other losses. Finally, even if the differences between the observed MAE between the three cost functions are small, using the $L_{2,\text{mod}}$ improves the FSV metrics even more. We will revisit the topic of the cost functions in the next section, when we will evaluate the CNN on unknown geometries.



(a) Training losses.



(b) Test losses.

Fig. 8: Comparison between the MAE, $L_{2,\text{mod}}$, $L_{2,\text{LoS}}$ and $L_{2,\text{nLoS}}$ errors for various values of w_{LoS} for the training and test samples.

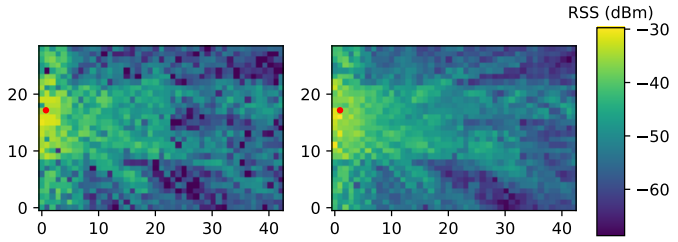
B. Generalization for the training geometries

We now present additional results to showcase the generalization abilities of the CNN that was trained on minimizing the $L_{2,\text{mod}}$ cost function for the case of $w_{\text{LoS}} = 0.3$. The mean training and test MAE for that case have already been presented in Table III. As expected, the test error is higher than the training error, by about 1 dB. However, the network does not overfit to the training cases. This can be validated in Fig. 9 that shows the predicted versus the target RSS maps for two sample test cases. Both of these correspond to test scenarios simulated in the geometry of Fig. 4a. For the first, $y_T = 9.18\,\text{m}$, $f = 5.313\,\text{GHz}$ and $z_R = 1.81\,\text{m}$. For the second, $y_T = 6.67\,\text{m}$, $f = 5.282\,\text{GHz}$ and $z_R = 1.38\,\text{m}$.

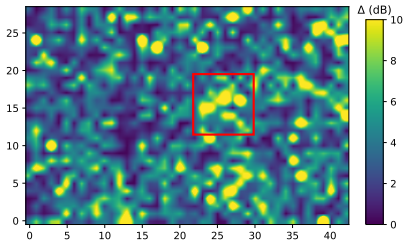
With respect to the changing transmitter location, the network is able to generalize its RSS predictions by capturing the dominant ray paths that are associated with the new locations, as shown in the predicted RSS maps of Figs. 9a and Figs. 9c. The absolute deviation:

$$\Delta = |P_{\text{map}} - \hat{P}_{\text{map}}| \tag{8}$$

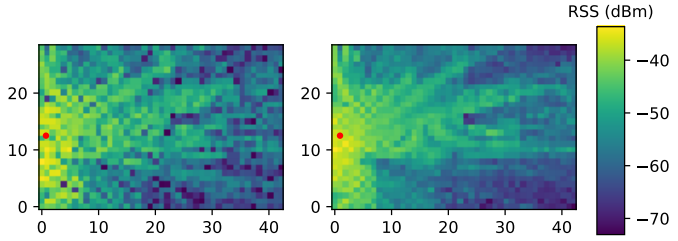
between the target RSS map, P_{map}, and the predicted map, $\hat{P}_{\text{map}}$, for these two test cases is shown in Figs. 9b and Figs. 9d, respectively. High-error points are usually located in areas where nLoS conditions exist. This observation motivated the search for another cost function to treat nLoS and LoS receiving points differently; in our case the $L_{2,\text{mod}}$ cost function. One of the areas of high-error points is also enclosed by a red



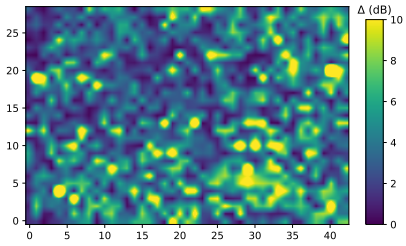
(a) RSS maps of the 1st test case



(b) Δ-map of the 1st test case



(c) RSS maps of the 2nd test case

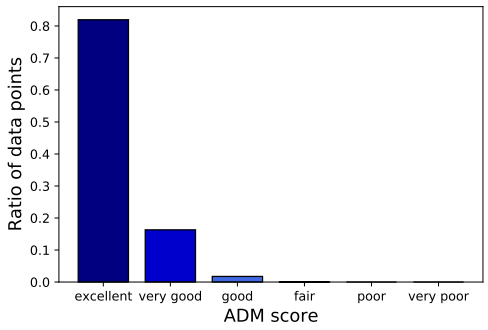


(d) Δ-map of the 2nd test case

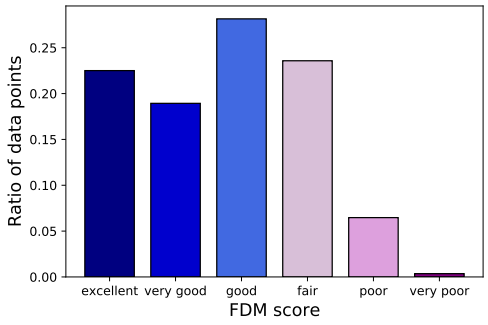
Fig. 9: A comparison between predicted and target RSS maps for two arbitrarily selected test cases utilizing the training geometries. The left plots represent the target RSS maps, while the right ones are the CNN's predictions. The location of T_x is noted with red.

rectangle for the 1st test case. These error points are within a pillar, as can be seen in Fig. 4c, where RSS predictions are not needed.

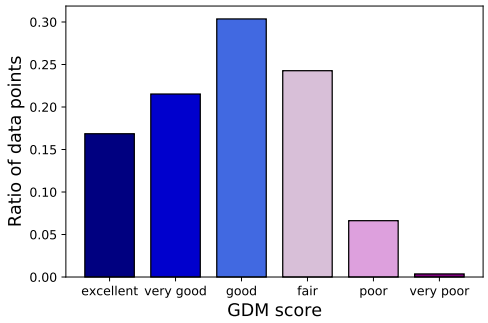
To further evaluate the goodness of fit between the predicted and target data, we use the FSV method. Histograms of the FSV metrics are shown in Fig. 10. The mean ADM is very good, with most of the points having an "excellent" score. That is an indication that the predicted slow fading characteristics of the channel are very accurate. The FDM score, although not as low as the ADM, is labeled as "good". The network does not capture all the variations of the signal, hence, the worse performance of the CNN for this metric. A similar score is



(a) ADM confidence histogram



(b) FDM confidence histogram



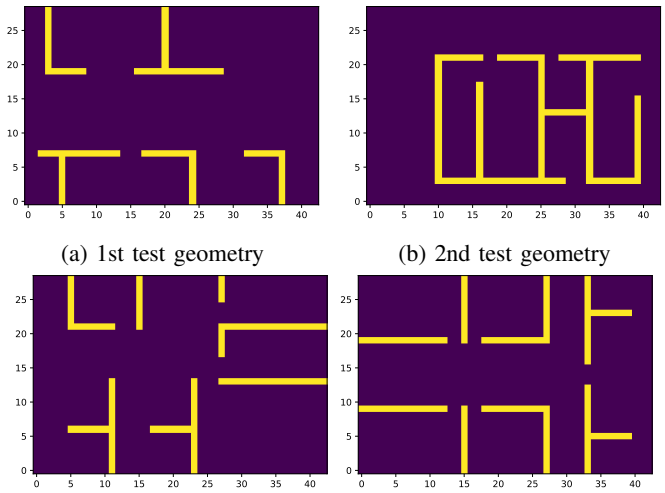
(c) GDM confidence histogram

Fig. 10: The FSV confidence histograms computed over all test samples.

achieved for the GDM, further confirming an overall "good" agreement between the predicted and target RSS values. It is also important to note that the percentage of "poor" and "very poor" points is less than 7%. Moreover, the test ADM metrics are very close to the training ones. The difference between training and test MAE is due to fast fading effects, as shown by the difference between the training and test FDM scores.

V. Simulation Results: Generalization to New Geometries

In this Section, we investigate the generalization abilities of the network with respect to unknown geometries. To that end, we study 4 new geometries, shown in Fig. 11. All these geometries have the same outer size as the training geometries. However, the placement of the inner walls is different. The first geometry is similar to those included in the training set. Hence, we expect the test error for this geometry to be close to the training error. Geometries 2-4 are significantly different from those that were used to train the CNN. For example, geometry 2 includes multiple small partitions adjacent to each other, with no main hallway. Also, geometries 3 and 4 have a combination of vertical and horizontal partitions that is not encountered in the training set. Our goal is to show that the proposed CNN deals with these new geometries successfully. Moreover, we determine the key reasons behind this success and whether they are related to the input features we selected and the training geometries (basic and/or complex).



(a) 1st test geometry

(b) 2nd test geometry

(c) 3rd test geometry

(d) 4th test geometry

Fig. 11: The test geometries.

A. Generalization results for the new geometries

We test the trained CNNs on the 4 test geometries of Fig. 11. The mean MAE for all test cases, for the three different cost functions, is presented in Table IV. It is evident that using the $L_{2,\text{mod}}$ improves the accuracy of the ANN for generalizing to new geometries. In our case, it improves the test MAE by about 0.4 dB compared to the L_1 cost, while having better FSV metrics than both L_1 and L_2 cost functions.

TABLE IV: Error metrics for the test geometries

Cost Function	test MAE (dB)	ADM	FDM	GDM
L_1	4.65	0.061	0.395	0.408
L_2	4.49	0.059	0.383	0.388
$L_{2,\text{mod}}$	4.35	0.058	0.374	0.378

Also, the CNN that uses the $L_{2,\text{mod}}$ cost function exhibits the smallest test MAE increase, compared to the test MAE of the training geometries of the previous section. The same is observed for the FSV metrics. Moreover, the maximum increase in the test MAE of about 0.5 dB, is observed for the $L_{1,\text{mod}}$ cost function. This is a modest increase, considering that the geometries are very different from those of the training set.

Table IV presents the MAE for each test geometry separately, when the $L_{2,\text{mod}}$ cost function is used. Regarding each test geometry, the 1st has the lowest MAE among the four. This is a result of that geometry being similar to the training

ones. The test MAE for the 4rd geometry, 4.21 dB, is also comparable to the mean test error of the training geometries. The test MAE of the remaining geometries is within 0.4 dB of the error measured for the test geometries of Section IV.

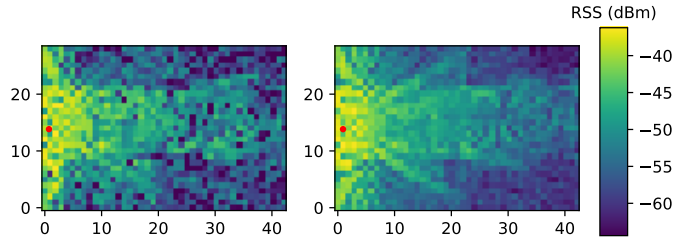
TABLE V: Error metrics: test geometries for the $L_{2,\text{mod}}$ case

Geometry	test MAE (dB)	ADM	FDM	GDM
1st test	4.02	0.054	0.34	0.344
2nd test	4.54	0.054	0.359	0.371
3rd test	4.21	0.062	0.389	0.402
4th test	4.64	0.064	0.38	0.395

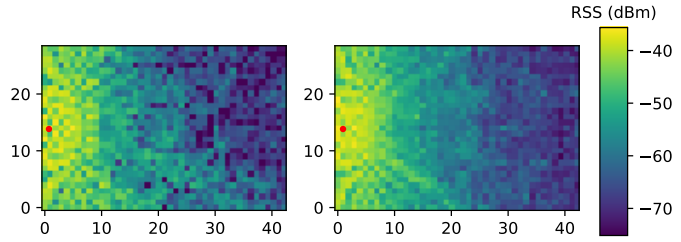
The predicted versus target RSS maps for one test sample from each of the four geometries can be seen in Fig. 12. There is a generally good agreement between the actual and generated RSS maps, although some of the main ray paths are not captured. For example, in the actual RSS map of the 4th test geometry, there is strong reception in the rightmost upper and lower rooms of Fig. 11d. However, in the predicted RSS maps, the predicted RSS is overall lower. That is one of the reasons why this test geometry experiences the highest MAE among the 4 geometries. Like before, most high-error points are nLoS points behind walls. This can be also seen in Fig. 13, that contains maps of the absolute deviation of the predicted RSS map compared to the target RSS map. It is especially apparent for the 2nd test geometry, where most of the LoS points exhibit small errors, contrary to the nLoS points.

B. Impact of geometry-encoding input features

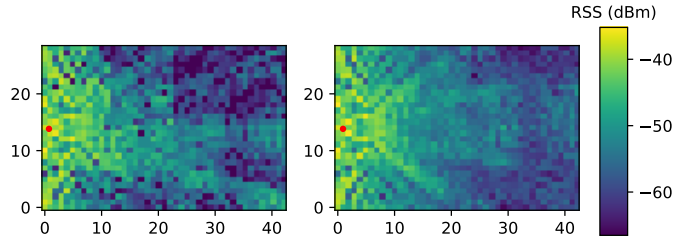
Features describing the environment and the physics of wave propagation in it, improve the ability of the ML model to generalize to unknown geometries. For that reason, we evaluate the relative contribution of each input feature, i.e. the features regarding the number of obstructions, LoS, diffraction and the geometry, to the accuracy of the CNN. To do so, we compute the absolute value of the weights among all the filters that multiply each of the corresponding input channels. Here, we use the CNN that is trained to minimize the $L_{2,\text{mod}}$ cost function. The largest weights are assigned by the network to the filter channels that multiply the channel that encodes the number of obstructions. Second in importance is the channel defining the LoS receiving points. Interestingly, the importance of the input channel that encodes the geometry comes third, for two reasons. First, the outline of the inner walls can be indirectly deduced by the two highest in importance input features (LoS points and number of obstructions). Second, the network is asked to "learn" only 28 geometries during its training, whereas there are hundreds of different configurations of simulated receiver points for a single geometry. Regarding what the filters learn, the small size of them makes it difficult to interpret their values. However, some conclusions can still be deduced for the filters of the first layer that are larger in size. One such example is presented in Fig. 14 for eight of the filters of the first convolutional layer that are associated with the LoS input feature. These filters seem to learn transitions between high and low RSS values that are indicative of the presence of a wall. That, in turn, influences the LoS conditions of the receiving points.



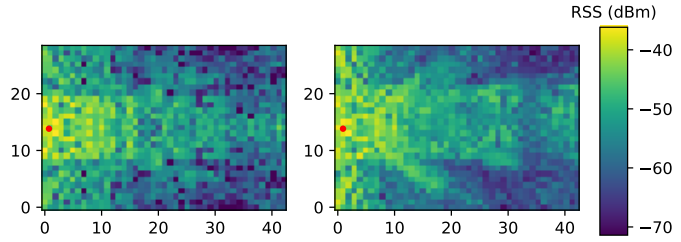
(a) 1st test geometry



(b) 2nd test geometry



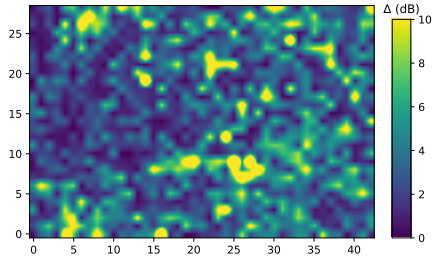
(c) 3rd test geometry



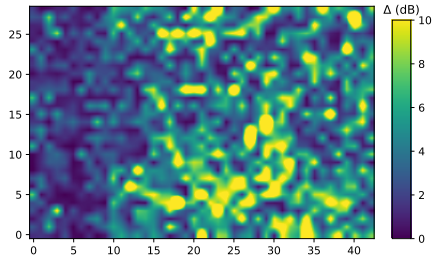
(d) 4th test geometry

Fig. 12: A comparison between predicted and target RSS maps for the four test geometries. The left plots represent the target RSS maps, while the right ones are the predictions of the CNN. The location of T_x is noted with red.

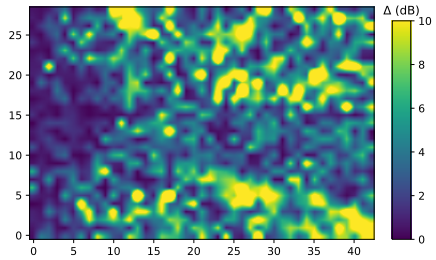
We also test how important the physics-based features are for geometry-generalization tasks. To that end, we remove them altogether and train the network only on the remaining features, on all 28 geometries. Then, the network is evaluated on the 4 test geometries. The results are summarized in Table VI. The test MAE and ADM for all cases are now higher than the respective cases using the additional geometry-encoding features of Table V. The test MAE of the 2nd test geometry has especially seen a sharp increase. The FDM scores are also worse. Consequently, GDM metrics are worse. More importantly, the network is incapable of differentiating between the different propagation scenarios. This is shown in Fig. 15. Even though the two cases correspond to two different



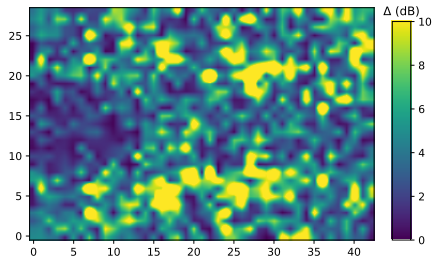
(a) 1st test geometry



(b) 2nd test geometry



(c) 3rd test geometry



(d) 4th test geometry

Fig. 13: Error maps (Δ) for the four test cases of Fig. 12.

test geometries, the network cannot distinguish them. Thus, its output for the two test cases looks identical, and resembles only the target RSS map of the first geometry of Fig. 15a. This shows that geometry-encoding features are important in learning the various wave-propagation characteristics in an environment.

TABLE VI: Error metrics without geometry-encoding features

Geometry	MAE (dB)	ADM	FDM	GDM
Trained	3.58	0.074	0.376	0.395
Test	5.7	0.089	0.418	0.442
1st test	4.82	0.088	0.419	0.442
2nd test	7.64	0.108	0.454	0.484
3rd test	5.07	0.081	0.399	0.42
4th test	5.69	0.079	0.4	0.421

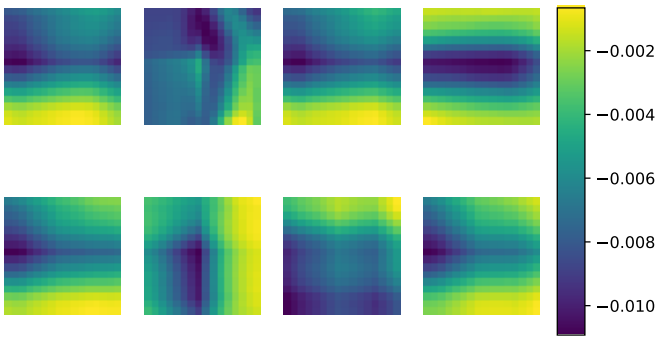
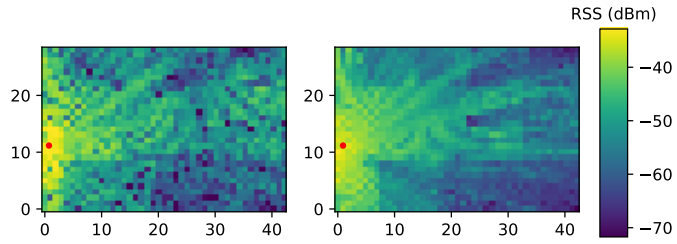
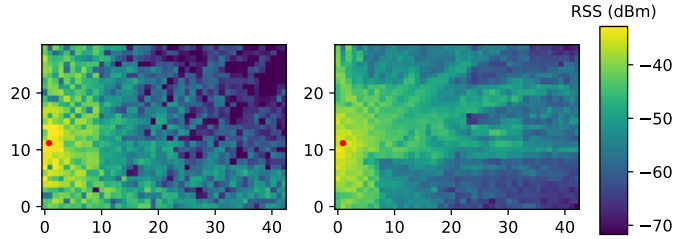


Fig. 14: Eight of the learned filters of the first hidden layer, that are associated with the LoS input feature.



(a) Test case 1.



(b) Test case 2.

Fig. 15: A comparison between the predicted and target RSS maps for two test samples at different geometries, with no physics-based featuers. The right plots are the CNN's predictions, while the left ones represent the target RSS maps. The location of T_x is noted with red.

VI. Conclusion

We presented a CNN-based indoor propagation model that can generalize with respect to geometries defined by arbitrarily repartitioning an indoor space. In addition to generalizing with respect to new geometries, our model can deal with new transmitter/receiver locations and frequencies within a fixed bandwidth. Hence, this work advances the state of the art with respect to the key challenge of formulating ML models that can learn the physics of indoor propagation and rapidly solve new problems, well beyond those that they were trained for. An important aspect of our approach was that we used physical insights as we built the CNN model, selecting its input parameters and cost functions. Instead of relying on just the geometry, we added physics-inspired input parameters that proved to be critical in enabling the development of

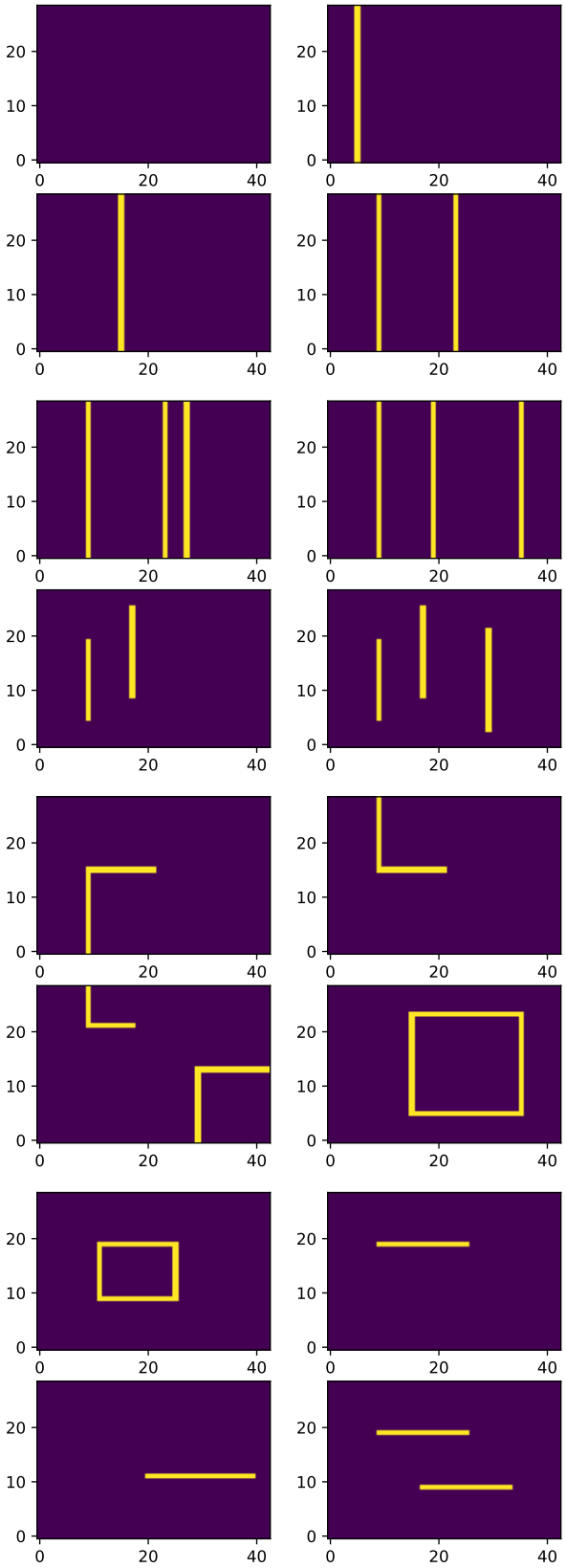


Fig. 16: The simple training geometries.

generalizable models. Likewise, cost functions were modified to account for the difference between LoS and nLoS points.

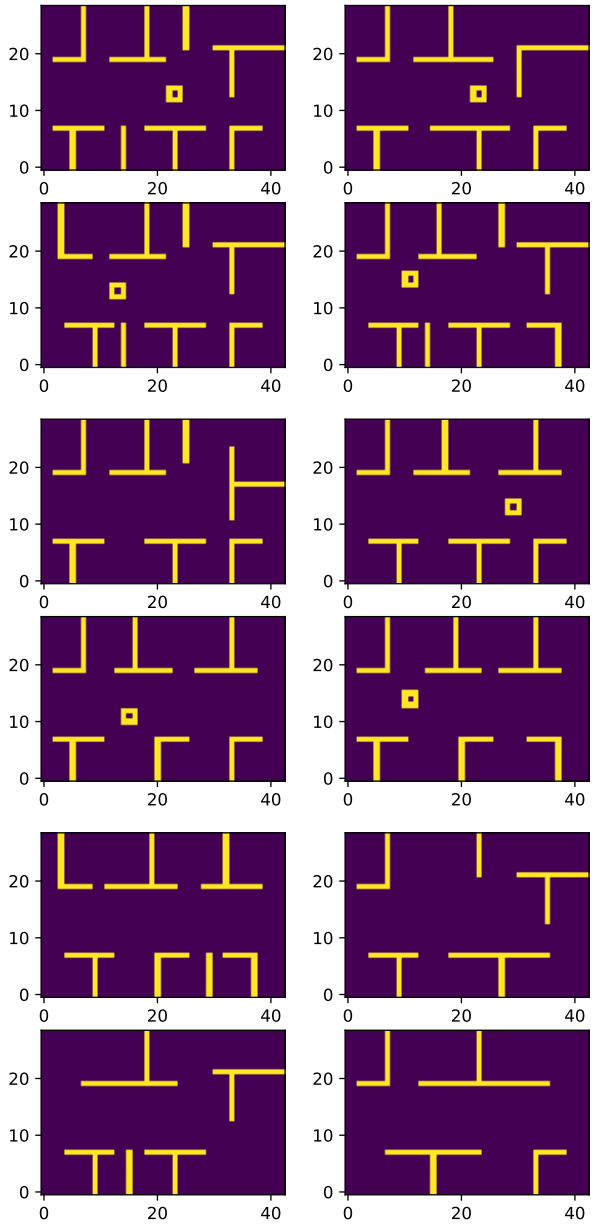


Fig. 17: The complex training geometries.

Future work will extend to further generalizing the geometries that the CNN can simulate (removing the need for generating geometries by repartitioning a fixed frame) and to integrating measured data in the training process.

Appendix I - The Training Geometries

Here we present the top-down view of the training geometries, both simple and complex, in Figs. 16 and 17, respectively.

Appendix II - The FSV Algorithm [37]

The FSV algorithm quantifies the difference between two datasets. The algorithm compares the datasets by decomposing their data into amplitude and feature information. This is done by Fourier transforming the data and then passing the transformed data through a low-pass and a high-pass filter. The

cut-off point of these two filters can be determined empirically. The comparison between the low-pass filtered data of the two datasets, $x_{\text{lp},1}, x_{\text{lp},2}$ determines the ADM score for a datapoint k according to:

$$\text{ADM}(k) = \frac{(|x_{\text{lp},1}(k)| - |x_{\text{lp},2}(k)|)}{\frac{1}{N}\sum_{i=1}^{N}(|x_{\text{lp},1}(i)| - |x_{\text{lp},2}(i)|)} \tag{9}$$

where N is the total number of datapoints. The mean ADM is computed by averaging the ADM scores over all datapoints. The FDM score is computed in a similar fashion, by involving the first derivatives of $x_{\text{lp},1}, x_{\text{lp},2}$, as well as the first and second derivatives of the high-pass filtered data $x_{\text{hp},1}, x_{\text{hp},2}$. Finally, the GDM combines the ADM and FDM for a datapoint k as:

$$\text{GDM}(k) = \sqrt{\text{ADM}(k)^2 + \text{FDM}(k)^2} \tag{10}$$

The final GDM score is the mean of $\text{GDM}(k)$ over all k points. Generally, the ADM metric gauges the slow-changing features (envelopes) of the two signals. On the other hand, the FDM metric scores their fast-changing features.

Finally, the algorithm assumes two 1-dimensional datasets. In our case, where the output RSS maps are by definition 2-dimensional, we extract the corresponding FSV scores by using line segments along x, of the form $y = y_o$. There exist 29 such segments in total, containing 43 receiving points each (the number of sample points used along the length of the office). Hence, the reported values of ADM, FDM and GDM throughout the paper represent mean values over all those segments and across all samples.

REFERENCES

[1] A. F. Molisch, *Wireless Communications*. Wiley Publishing, 2nd ed., 2011.

[2] T. S. Rappaport, *Wireless communications: principles and practice*. Prentice Hall, New Jersey, 1996.

[3] ITU Radiocommunication Bureau, "Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 450 GHz," *Recommendation ITU-R P.1238-10*, 2019.

[4] M. Catedra and J. Perez, *Cell planning for wireless communications*. Artech House, Inc., 1999.

[5] X. Zhang and C. D. Sarris, "A gaussian beam approximation approach for embedding antennas into vector parabolic equation-based wireless channel propagation models," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 3, pp. 1301–1310, 2017.

[6] F. Fuschini, E. M. Vitucci, M. Barbiroli, G. Falciasecca, and V. Degli-Esposti, "Ray tracing propagation modeling for future small-cell and indoor applications: A review of current techniques," *Radio Science*, vol. 50, no. 6, pp. 469–485, 2015.

[7] A. Seretis and C. D. Sarris, "An overview of machine learning techniques for radiowave propagation modeling," *IEEE Transactions on Antennas and Propagation*, 2021.

[8] A. Seretis, X. Zhang, K. Zeng, and C. Sarris, "Artificial neural network models for radiowave propagation in tunnels," *IET Microwaves, Antennas & Propagation*, vol. 14, 2020.

[9] G. P. Ferreira, L. J. Matos, and J. M. M. Silva, "Improvement of outdoor signal strength prediction in UHF band by artificial neural network," *IEEE Transactions on Antennas and Propagation*, vol. 64, no. 12, pp. 5404–5410, 2016.

[10] J. Wen, Y. Zhang, G. Yang, Z. He, and W. Zhang, "Path loss prediction based on machine learning methods for aircraft cabin environments," *IEEE Access*, vol. 7, pp. 159251–159261, 2019.

[11] J.-Y. Lee, M. Y. Kang, and S.-C. Kim, "Path loss exponent prediction for outdoor millimeter wave channels through deep learning," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, 2019.

[12] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate RSSI indoor localization," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10639–10651, 2019.

[13] A. Malone and C. D. Sarris, ""electromagnetic vision": Machine intelligence models of radiowave propagation in tunnels," in *2018 IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting*, pp. 557–558, 2018.

[14] L. Azpilicueta, M. Rawat, K. Rawat, F. M. Ghannouchi, and F. Falcone, "A ray launching-neural network approach for radio wave propagation analysis in complex indoor environments," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 5, pp. 2777–2786, 2014.

[15] A. Seretis, T. Hashimoto, K. Zeng, and C. D. Sarris, "Ray-tracing driven ann propagation models for indoor environments at 28 GHz," in *2020 IEEE International Symposium on Antennas and Propagation and North American Radio Science Meeting*, pp. 1029–1030, 2020.

[16] T. Hashimoto, X. Zhang, and C. D. Sarris, "Heuristic UTD diffraction coefficient for three-dimensional dielectric wedges," *IEEE Transactions on Antennas and Propagation*, vol. 69, no. 8, pp. 4816–4826, 2021.

[17] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando, "A survey of machine learning for indoor positioning," *IEEE Access*, vol. 8, pp. 214945–214965, 2020.

[18] J. Jang and S. Hong, "Indoor localization with WiFi fingerprinting using convolutional neural network," in *10th Int. Conf. Ubiquitous and Future Netw. (ICUFN)*, pp. 753–758, 2018.

[19] R. Wang, H. Luo, Q. Wang, Z. Li, F. Zhao, and J. Huang, "A spatial–temporal positioning algorithm using residual network and LSTM," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 11, pp. 9251–9261, 2020.

[20] X. Gan, B. Yu, L. Huang, and Y. Li, "Deep learning for weights training and indoor positioning using multi-sensor fingerprint," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, 2017.

[21] M. I. AlHajri, N. T. Ali, and R. M. Shubair, "Classification of indoor environments for IoT applications: A machine learning approach," *IEEE Antennas and Wireless Propagation Letters*, vol. 17, no. 12, pp. 2164–2168, 2018.

[22] A. Belmonte-Hernández, G. Hernández-Peñaloza, D. Martín Gutiérrez, and F. Álvarez, "Recurrent model for wireless indoor tracking and positioning recovering using generative networks," *IEEE Sensors Journal*, vol. 20, no. 6, pp. 3356–3365, 2020.

[23] Q. Li, H. Qu, Z. Liu, N. Zhou, W. Sun, S. Sigg, and J. Li, "AF-DCGAN: Amplitude feature deep convolutional GAN for fingerprint construction in indoor localization systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 468–480, 2021.

[24] M. Ayadi, A. Ben Zineb, and S. Tabbane, "A UHF path loss model using learning machine for heterogeneous networks," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 7, pp. 3675–3683, 2017.

[25] S. P. Sotiroudis, S. K. Goudos, K. A. Gotsis, K. Siakavara, and J. N. Sahalos, "Modeling by optimal artificial neural networks the prediction of propagation path loss in urban environments," in *2013 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, pp. 599–602, 2013.

[26] S. Sotiroudis and K. Siakavara, "Mobile radio propagation path loss prediction using artificial neural networks with optimal input information for urban environments," *AEU - International Journal of Electronics and Communications*, vol. 69, 2015.

[27] W. Chen, Y. Lin, and J. Yang, "Hybrid prediction model for field strength with ray tracing and artificial neural networks," in *2012 IEEE 14th International Conference on Communication Technology*, pp. 301–305, 2012.

[28] O. Ahmadien, H. F. Ates, T. Baykas, and B. K. Gunturk, "Predicting path loss distribution of an area from satellite images using deep learning," *IEEE Access*, vol. 8, pp. 64982–64991, 2020.

[29] J. Thrane, D. Zibar, and H. L. Christiansen, "Model-aided deep learning method for path loss prediction in mobile communication systems at 2.6 GHz," *IEEE Access*, vol. 8, pp. 7925–7936, 2020.

[30] G. Cerri, M. Cinalli, F. Michetti, and P. Russo, "Feed forward neural networks for path loss prediction in urban environment," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 11, pp. 3137–3139, 2004.

[31] G. Wolfle and F. Landstorfer, "Field strength prediction in indoor environments with neural networks," in *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, vol. 1, pp. 82–86 vol.1, 1997.

[32] R. Fraile, L. Rubio, and N. Cardona, "Application of RBF neural networks to the prediction of propagation loss over irregular terrain," in *Vehicular Technology Conference Fall 2000. IEEE VTS Fall VTC2000.*

52nd Vehicular Technology Conference (Cat. No.00CH37152), vol. 2, pp. 878–884 vol.2, 2000.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[34] T. Imai, K. Kitao, and M. Inomata, "Radio propagation prediction model using convolutional neural networks by deep learning," in *13th European Conf. Antennas and Propag. (EuCAP)*, pp. 1–5, 2019.

[35] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.

[37] A. Duffy, A. Martin, A. Orlandi, G. Antonini, T. Benson, and M. Woolfson, "Feature selective validation (FSV) for validation of computational electromagnetics (CEM). part i-the FSV method," *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, no. 3, pp. 449–459, 2006.