Received 6 March 2024, accepted 4 April 2024, date of publication 8 April 2024, date of current version 15 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3386166

RESEARCH ARTICLE

Detection of Modal Numbers From Field Configurations in Rectangular Waveguides via Machine Learning Models of Noisy Datasets

RASUL CHOUPANZADEH, (Graduate Student Member, IEEE), AND ATA ZADEHGOL, (Senior Member, IEEE)
Electrical and Computer Engineering Department, University of Idaho, Moscow, ID 83844, USA
Corresponding author: Ata Zadehgol (azadehgol@uidaho.edu)

ABSTRACT We propose a *machine learning* (ML) modeling methodology to predict the propagation mode number of electromagnetic (EM) fields inside a metallic rectangular waveguide based on the field configuration in the waveguide cross-section, in the presence of noise. We consider the *Transverse Electric* (TE_{mn}) modes and assume m and n in the range of 0 to 2 inside the waveguides, where the magnitude and phase of the noiseless field configurations are obtained from the analytical solution of the electric vector field $\vec{E}$. We generate training/testing datasets that includes 64,000 plots of the magnitude and phase of $\vec{E}$ over the waveguide cross-section, spanning various TE modes in the frequency range of 13-17 GHz. Our methodology for training and evaluation is based on the classification model, and relies primarily on *Stochastic Gradient Descent (SGD)* and *k-Nearest Neighbors*. For real-world scenarios which include noise, we introduce two random distributions in the datasets; specifically, the exponential and the Gaussian distributions are added onto the computed E-fields to further challenge the ML model. We discuss the limitations of the proposed ML modeling approach and the challenges in finding the optimal ML model for these types of problems. The proposed methodology may be generalized to predict both the TE and *Transverse Magnetic* (TM_{mn}) mode numbers with a wide ranges of m and n, as well as for other types of waveguides; e.g., circular, elliptical, etc.

INDEX TERMS Classification, electromagnetic (EM) field, exponential distribution, Gaussian distribution, k-Nearest neighbors, machine learning, mode number, noise, rectangular waveguide, stochastic gradient descent, transverse electric (TE), transverse magnetic (TM).

I. INTRODUCTION

Machine Learning (ML) is the science of programming the computers so they can learn from data to do the desired tasks. The ML has been around for decades in many specialized applications and hundreds of products and features that we use regularly [1]. ML has shown great potential for modeling for various complex tasks such as pattern recognition in domains ranging from computer vision [2] over speech recognition [3] and text understanding [4] to Game AI [5]. Additionally, due to the data-based nature from a huge number of examples, ML is being increasingly important and successful in other areas such as electrical engineering and electromagnetics (EM) [6], [7], [8], [9], [10].

There are various types of ML systems which may be categorized in broad categories such as regression versus classification in terms of the problem type, supervised, unsupervised, semi-supervised, and reinforcement learning

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengqing Yun.

in terms of the need to human supervision, online versus batch-learning in terms of the leaning approaches. Further details about these categories are provided in [1].

In the field of microwaves, one of the typical tasks are calculating electric and magnetic fields inside the waveguides and obtaining the cross-sectional plots due to a given propagation mode number [11]. However, in some cases, we may need to perform the reverse process, meaning that we are seeking to detect the modal configuration according to given plots of electric and magnetic fields. Although, there exist many previous works throughout literature that predicts the modal configuration through various algorithms, the authors were seeking to find a faster methodology to predict the modal configuration inside rectangular waveguide, which can be obtained through ML based algorithms.

In this paper, we will study and work on two different classification models, such as *Stochastic Gradient Descent (SGD)* and *K-nearest Neighbors* to predict the TE propagation mode number (i.e., modal configuration) inside a rectangular waveguide in the presence of exponentially distributed noise in comparison with Gaussian noise. This paper presents the main steps of proposed algorithm, as follows. **(1)** Framing the problem and determining type of ML algorithm, **(2)** generating and preparing the data to train ML models, **(3)** exploring proposed models, training and evaluating them to select the best one, **(4)** testing the model on training dataset for generalization and launching the system.

The remainder of this paper is organized as follows. In section II we present the methodology for prediction of TE mode number inside the rectangular waveguide by describing the structure of problem, noise distributions, data generation procedure, data preparation, proposed ML models, performance measuring methods, error analysis procedure, and testing process to launch the proposed models. In section III we present numerical calculations and the results of proposed models for this problem, and discuss the challenges and limitations of the proposed models. In section V we conclude with some closing remarks.

II. METHODOLOGY

A. STRUCTURE

As we noted previously, the main objective of this work is predicting TE^z_{mn} propagation mode number of EM wave due to the given noisy plots of magnitude and phase of E_x inside an air-filled rectangular waveguide, where $0 \leq m, n \leq 2$, and m and n cannot be zero simultaneously (i.e., $m = n \neq 0$), assuming the waveguide dimensions $a = 1.07$ cm, $b = 0.43$ cm, operating at frequencies from 13 GHz to 17 GHz. The structure of described waveguide is shown in Fig. 1.

To generate the training data for TE mode configuration, we calculate the electric filed in $\hat{x}$ direction (E_x) using (1). Accordingly, magnitude and phase datasets for each mode number m and n can be obtained.

$$E_x = \frac{j\omega\mu_0 n\pi}{{k_c}^2 b} A \cos\frac{m\pi x}{a} \sin\frac{n\pi y}{b} e^{-j\beta z} \tag{1}$$

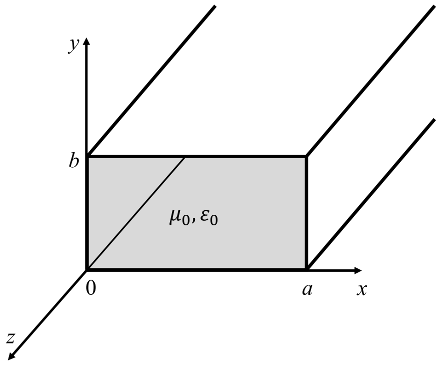


FIGURE 1. Geometry of rectangular waveguide.

where, μ_0 and ϵ_0 are free-space permeability and permittivity, respectively, $A = 1$, a and b are waveguide dimensions, x and y are magnitude and phase calculation points, m and n indicate mode number, $z = 0$, $\omega = 2\pi f$, f is the operating frequency in range 13-17 GHz, $k_c = \sqrt{(\frac{m\pi}{a})^2 + (\frac{n\pi}{b})^2}$ is cutoff wave number, $\beta = \sqrt{k^2 - {k_c}^2}$ is wave number in $\hat{z}$ direction, and $k = \omega\sqrt{\mu_0\epsilon_0}$ is wave number in unbounded medium.

B. NOISE DISTRIBUTIONS

In a real-world waveguide problem, the measured fields inside the waveguides may be affected by internal or external noises. Many natural noises exhibit statistical properties that can be well approximated by Gaussian or exponential distributions. For example, Gaussian noise is often used to model random fluctuations in measurements due to the Central Limit Theorem [12, p. 10], which states that the sum of many independent random variables tends to follow a Gaussian distribution. Exponential noise, on the other hand, can represent phenomena with exponential decay or growth rates, such as radioactive decay or signal attenuation. Therefore, to have a realistic problem, we inject the noises to the training dataset generated by (1). The Gaussian distribution, which is also known as normal distribution, is expressed as follows [13].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

where, μ is mean, and σ is standard deviation. Furthermore, the exponential distribution is formulated as

$$f(x) = \lambda e^{-\lambda x} \tag{3}$$

where, $\lambda > 0$ is the rate parameter. It is notable that in exponential distribution $\mu = \sigma = 1/\lambda$. Fig. 2 illustrates particular cases for exponential and Gaussian noise distributions.

Initially, we create a noisy dataset by introducing exponentially distributed noise to the clean data. Subsequently, we utilize the noisy dataset to train the ML models and compare it with the scenario where the dataset is noisy with a Gaussian distribution.

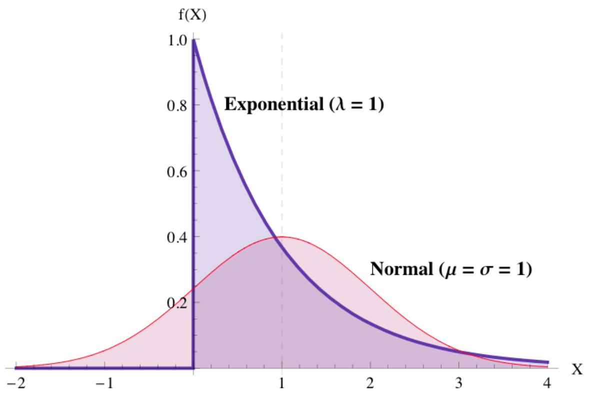


FIGURE 2. Gaussian (normal) distribution with $\mu = \sigma = 1$ versus exponential distribution with $\lambda = 1$.

C. FRAMING THE PROBLEM

Before selecting and training a model, we must frame and categorize the problem. The provided data includes propagation mode numbers (i.e., m and n) as labels, and noisy values of magnitude and phase at any point (x, y) inside the waveguide as features. Our objective is to build a model that learns from this data and is able to predict m and n (between 0 and 2) for any instances, given the magnitude and phase plots of E field. Obviously, it is a *supervised* task, since the instances are labeled with m and n values. Since we are classifying m and n to the values from 0 to 2, it is a *classification* task. It is also a *multi-class* classification, since we are categorizing two labels m and n in different classes. There is no need to update the data continuously, and the data is small enough to fit in a computer memory, so we can use batch learning technique.

D. DATA GENERATING

As our problem is particularly designed with specified fixed values, we are required to generate our own dataset for training the ML models. This dataset is generated by calculating magnitude and phase values of E_x in (1), considering previously mentioned waveguide dimensions and frequencies, using the Python programming language [14]. In order to capture a diverse range of field distributions across various frequencies, we have generated a dataset comprising 64, 000 instances spanning 8, 000 frequency points between 13 and 17 GHz. Each instance includes both magnitude and phase variations in the x and y directions, resulting in a total of 5, 000 features for every instance. Subsequently, in order to introduce noise into our dataset, we applied random variations using Gaussian and exponential distributions to both the magnitude and phase features.

E. DATA PREPARATION

We should perform a few steps to prepare the data before training ML models. To begin, it should be mentioned that the generated data are not shuffled, thus, if we use them directly, our ML models may recognize undesired patterns and make a decision bias toward them. Therefore, we shuffle the dataset as the first step of data preparation. The next step is applying the mentioned noises. Typically, everyone is seeking to eliminate the noises from input dataset, however, in this work we want to add the noise intentionally, to evaluate the effect of noise on ML prediction. The final step is setting aside a part of the data randomly as the test set, which is normally 20% of whole dataset. We also visualize the training dataset to gain insights about them. For example, since we are working with magnitude and phase of E_x, we plot them as an image illustrating the modal configuration and corresponding characteristics.

F. SELECTING AND TRAINING THE MODEL

So far we have framed the problem, prepared the data by shuffling and adding noise to generated dataset, separated the training and test sets. Now, we are ready to select and train the ML models. We have several options for classification such as *Stochastic Gradient Descent (SGD)*, *K-Nearest Neighbors*, *Support Vector Machine (SVM)*, *Softmax Regression*, *Random Forest*, *Naive Bayes*, etc. While other models like SVM, Random Forest, and Softmax Regression are indeed popular and effective, they were not included in this study due to constraints such as time, computational resources, and the objectives of this research. For example, methods like SVM, Softmax Regression, and Random Forest are primarily effective when the data is linearly separable. However, for problems including non-linear relationships, they require large kernel matrices, additional feature engineering, or large number of trees, which can be computationally expensive and may not scale well to large datasets. In such cases, simpler and computationally efficient methods like SGD or k-Nearest Neighbors might be preferred for their efficiency in optimizing large-scale datasets, convex objectives, effectiveness in handling non-linearity, handling datasets with large number of features, and robustness to noisy data. Therefore, we have opted SGD and k-Nearest Neighbors to be trained, evaluated, and compared as the classification models for this problem. It is worth noting that for modeling electromagnetic structures, there are more complex methodologies available, such as deep neural networks (DNNs) [6] and physics-informed neural networks [15]. However, due to the intricacy of these models, simpler classical ML models are typically preferred for simple problems. If these simpler models are not effective, DNNs and physics-informed neural networks can then be utilized.

For training purposes, we may treat the problem in two approaches. As the first approach, we may consider the problem as a multi-label multi-class classification for predicting m and n labels. In the second approach, we may assign a class for each pair of (m,n) to consider the problem as a single-label multi-class problem and find the class representative of m and n labels, instead of directly finding m and n. Since the m and n values are limited from 0 to 2, we have 8 possible pairs of (m,n) such as {(0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)}. Therefore, in the second approach, we assign a class from 0 to 7 for each of this possible states, respectively, to convert our problem from multi-label to single-label classification. The relationship between labels (i.e., m, n) and classes are shown as follows:

$$\begin{array}{lcccccccccc} Labels & = & \{ & (0,1), & (0,2), & (1,0), & (1,1), & (1,2), & (2,0), & (2,1), & (2,2) \; \} \\ & & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ Classes & = & \{ & 0, & 1, & 2, & 3, & 4, & 5, & 6, & 7 \; \} \end{array}$$

In this work, we use the second approach, and compare the execution time, cross-validation scores, precision, recall, and F1 scores of the classifier, which are described with details in the following sections.

G. PERFORMANCE MEASURE

In following sections, we use cross-validation scores to measure the accuracy of models, which can be computed using Scikit-Learn's K-fold *Cross-Validation* feature. The aforementioned cross-validation feature randomly splits the training set into k distinct subsets called folds, then trains and evaluates the Decision Tree k times, picking a different fold for evaluation every time and training on the other $k-1$ folds. The result will be an array containing the k evaluation scores. Another method to evaluate the performance of a classifier is evaluating the confusion matrix. To compute the confusion matrix, we require a set of predictions so that they can be compared to the actual targets, and it can be obtained simply using the confusion matrix function using Scikit-Learn. Each row in a confusion matrix represents an actual class, while each column represents a predicted class. For example, in this problem we have 8×8 confusion matrix, representing 8 actual and 8 predicted classes from 0 to 7. It should be noted that, in a confusion matrix, we have four definitions such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), where a TP is an outcome where the model correctly predicts a positive class, TN is an outcome where the model correctly predicts the negative class, FP is an outcome where the model incorrectly predicts a positive class, and FN is an outcome where the model incorrectly predicts the negative class. A perfect classifier would have only TP and TN values, leading to a diagonal confusion matrix. After obtaining the confusion matrix, we are able to calculate the values of precision and recall (sensitivity or true positive rate) as follows.

$$Precision = \frac{TP}{TP + FP}, \tag{4}$$

which shows the accuracy of true predictions (i.e., percentage of true values that are predicted correctly).

$$Recall = \frac{TP}{TP + FN}, \tag{5}$$

which shows what percentage of actual trues could be detected as true positive.

It is notable that there is a trade of between precision and recall, thus, neither of them can show the performance independently. Therefore, they should be used together to evaluate the performance. To this end, it is often convenient to combine precision and recall into a single metric called F_1 score, which is a simple way to compare two classifiers.

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{6}$$

In sum, we have several options to measure the performance such as cross-validation, precision, recall, and F_1 scores.

H. EVALUATING THE SYSTEM ON TEST SET

Once we have achieved a well-performing final model on the training dataset, it is essential to assess the model's performance on the test dataset, which is known as the generalization procedure. In some cases, the generalization error on test set may not be low enough to launch the system. In these cases, there may be many reasons such as small training dataset, selecting a wrong model, poor-quality data, etc. The main reason and a solution for such generalization errors can be found by error analysis of confusion matrix.

I. ERROR ANALYSIS

Suppose we have identified a promising model through evaluation on both the training and test datasets, but we observe a notable generalization error. In such a case, it becomes imperative to analyze the error and identify its underlying causes. The best approach involves assessing the confusion matrix and visually representing it through images. Frequently analyzing the confusion matrix can offer valuable insights, providing opportunities for enhancements to our models.

III. NUMERICAL RESULTS

In this section we show numerical results of proposed classification models for prediction of TE mode number.

Using the mentioned Python script in section II-D, we generated a clean dataset with 64,000 instances for 8000 frequency points between 13-17 GHz. In data generation process, we split the axis of length and width of rectangular waveguide (x, y) into 50 incremental points ($incr = 50$), and calculate the magnitude and phase values on these points. Thus, each of magnitude and phase datasets (i.e., plots) have a dimension of 50×50, which leads to 2500 features. Therefore, by assuming $incr = 50$, the total number of features for each instance is:

$$\text{Features} = \text{Magnitude features} + \text{Phase features} = 5000 \tag{7}$$

In prepared dataset, each instance has three labels m, n, and a pre-assigned class number representing m and n. Therefore, for $incr = 50$, the generated data has 64,000 rows (number of instances) and 5003 columns (5000 features and 3 labels), where, the first 2500 columns indicate the magnitude features, second 2500 columns indicate the phase features, column 5001 indicates m label, column

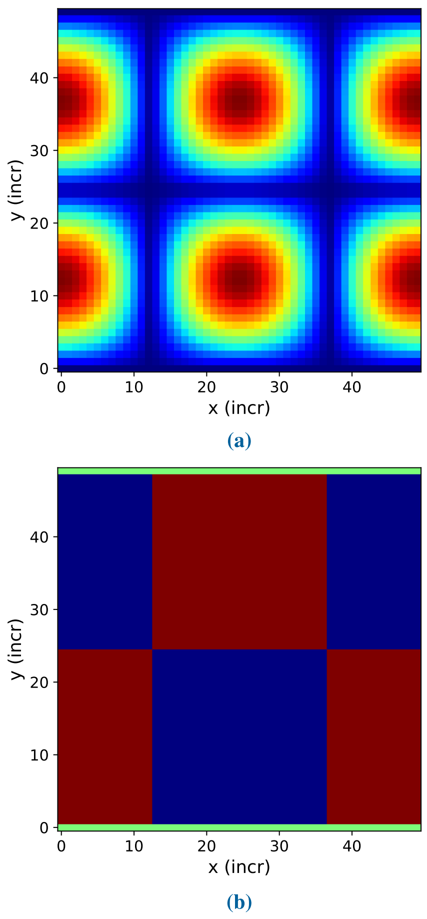


FIGURE 3. Plots of E_x for random instance 3 without noise: (a) Magnitude plot, (b) Phase plot.

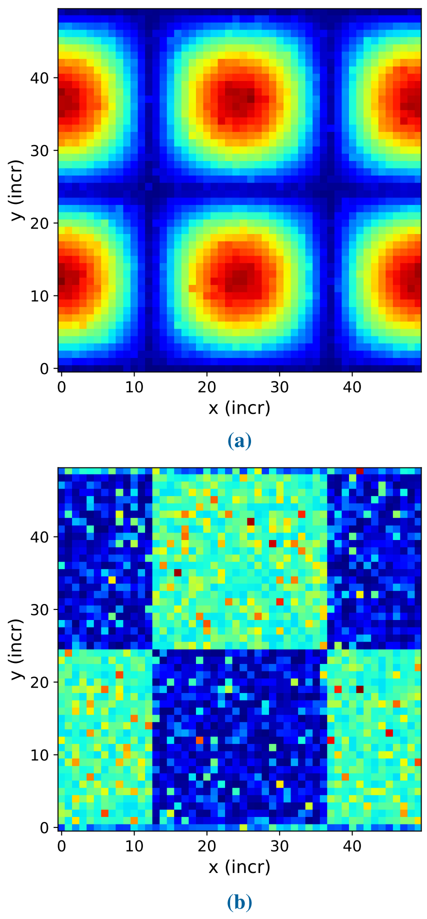


FIGURE 4. Plots of E_x for random instance 3 after adding noise with exponential distribution:(a) Magnitude, (b) Phase.

5002 indicates n label, and column 5003 indicates the class number from 0 to 7. It is notable that the generated data are not shuffled, thus, we must shuffle them before using for training models.

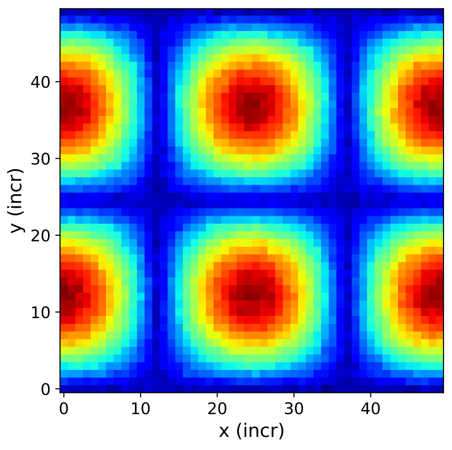
In order to produce a noisy dataset, we added an exponentially distributed noise with rate parameter $\lambda = 1$, and a Gaussian noise with standard normal distribution (i.e., $\mu = 0, \sigma = 1$) and then shuffled the data. In the next step, we split the dataset (i.e., noisy dataset) into two parts of training set (50,000) and testing set (14,000). To visualize the training dataset, we may pick an instance, reshape its feature vector to an array of 50×50, and visualize it using Python functions. Accordingly, to visualize the effect of noises on a clean dataset, we picked a random instance (instance 3) and plotted the magnitude and phase figures, before and after adding the noises. Fig. 3 shows the magnitude and phase plots of the random instance in dataset without noise. This instance is labeled as $m = 2$ and $n = 2$ (i.e., eighth class). It is notable that in Fig. 3(b), red, green, and blue colors indicate $+\frac{\pi}{2}, 0, -\frac{\pi}{2}$, respectively.

As it can be seen in Fig. 3, magnitude and phase of E_x at the bottom and top of plots are zero, this is simply due to the boundary conditions (B.C), which states that the tangential filed (i.e, E_x) is zero on the *perfect electric conductor* (PEC). To increase the quality of Fig. 3a, we may increase the number of *incr* points, leading to larger matrix dimension and higher number of features, and also requiring higher execution time and memory usage. It is notable that generating the data with $incr = 50$ takes about 70 minutes and 2.5 GB storage. Therefore, increasing the *incr* may not be a good idea unless having a super fast computer. We used the data with $incr = 50$ to train our models, which resulted a good performance.

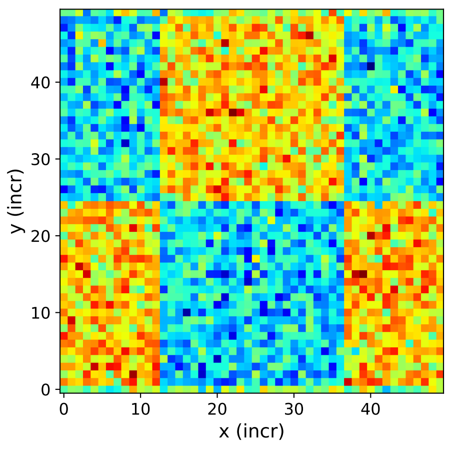
Moreover, the Fig. 4 and Fig. 5 illustrate magnitude and phase plots of the random instance 3 in dataset, after adding noises with exponential and Gaussian distributions.

The rates of noise effect can be observed clearly by comparison of Figs. 3-5. It is notable that the effect of noise can be increase or decreased by increasing or reducing the rate parameter and standard deviation in exponential and Gaussian distributions, respectively. Fig. 6 shows the histogram of generated noises using exponential and Gaussian distributions.

To prevent redundancy, we exclusively outlined the training and evaluation procedures for models using datasets featuring only exponential distribution noise. However, we provide a comparison of results between the two specified noise distributions.



(a)



(b)

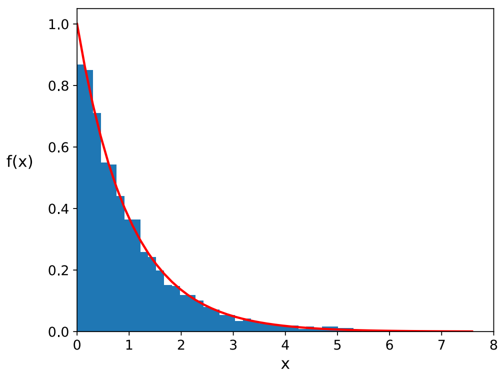
FIGURE 5. Plots of E_x for random instance 3 after adding noise with Gaussian distribution: (a) Magnitude, (b) Phase.

To avoid repetition, we presented the training and evaluation process of models for noisy datasets only with exponential distribution, but we provide the result comparison of two mentioned noise distributions. So far, we have generated a noisy datasets which is divided into training and test datasets, therefore, we must select and train the proposed ML models using training dataset.

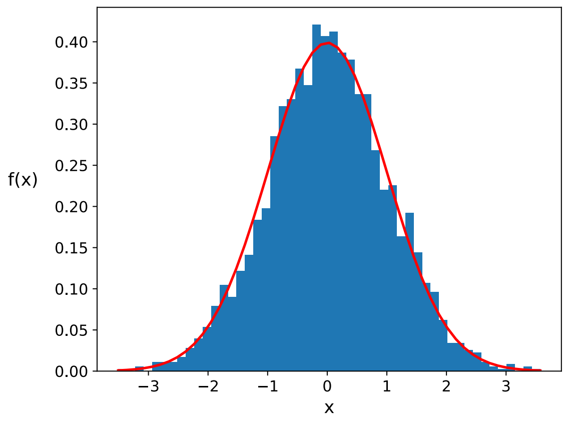
A. SGD CLASSIFIER

As the first model, we selected and trained a multi-class SGD classifier. This classifier has the advantage of being capable of handling very large datasets efficiently. After successfully training the SGD model by fitting the features of training data with target values, we correctly predicted the previously mentioned random instance 3 as eighth class, which represents $m = 2$ and $n = 2$. To see the corresponding scores and explain how SGD classifier resulted eighth class, we may call the decision function of this classifier. The resulted scores for instance 3, using SGD classifier, are shown as follows.

$$\begin{aligned} scores\ :&[-202737.63, -9162.73, -3200.45, -167285.61, \\ &-51402.26, -3107.66, -38067.77, 38919.18] \end{aligned}$$



(a)



(b)

FIGURE 6. Histogram of noise generated by: (a) exponential distribution, (b) Gaussian distribution.

In a decision tree function, highest score indicates the correct corresponding class. For example, in the previous scores, the highest score belongs to the eighth class. Thus, it predicts the mode number of instance 3 as $m = 2$ and $n = 2$, which is a correct prediction.

Although we had a correct prediction, we may require to measure the performance of the model on other instances in training dataset. One of the methods to evaluate the performance of a model, is calculating confusion matrix. The confusion matrix of SGD classifier on our training dataset is calculated as (8).

$$confusion = \begin{bmatrix} 6244 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6228 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3187 & 0 & 0 & 3109 & 0 & 0 \\ 0 & 0 & 0 & 6221 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6235 & 0 & 0 & 0 \\ 0 & 0 & 3170 & 0 & 0 & 3090 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6257 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6259 \end{bmatrix} \quad (8)$$

Comparing to the confusion matrix of a perfect classifier, which must be a diagonal matrix, confusion matrix (8) illustrates a good classification for this problem, except for the third and fifth class. We will show this exception and explain the reason, then provide a solution by the error analysis. We calculate *Precision*, *Recall*, and F_1 scores using the confusion matrix (8) and based on the definitions in (4), (5), and (6). The mentioned scores of SGD classifier are calculated as:

$$\begin{aligned} Precision &= 0.874975575 \\ Recall &= 0.874975579 \\ F_1score &= 0.874972627 \end{aligned}$$

A more popular method to evaluate the performance of a model is employing cross-validation. To this end, we used 3-fold cross-validation to evaluate the performance of proposed SGD classifier. The accuracy scores (ratio of correct predictions) of 3-fold cross validation are as follows.

$$\text{accuracy} = [0.87556249, 0.87280254, 0.874895]$$

As it can be seen, the SGD model has above 87.28% accuracy for data with exponential noise distribution. Repeatedly, the accuracy of SGD model for noisy dataset with Gaussian distribution is above 87.39%.

Furthermore, we can improve the model by analyzing the types of errors it makes. To this end, we may check the confusion matrix, or plot it as an image representation. For example, the image representation of confusion matrix (8) is shown in Fig. 7.

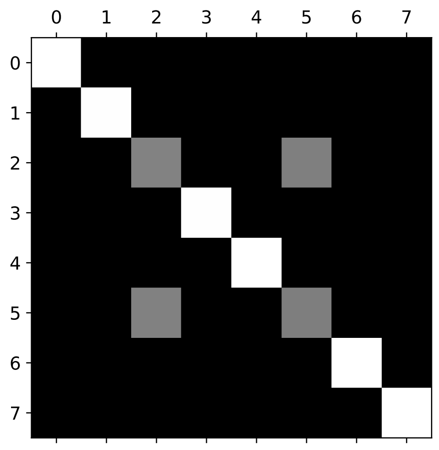


FIGURE 7. Image representation of confusion matrix in SGD classifier.

In Fig. 7, most values which are denoted by white image pixels (except the third and fifth classes) are on main diagonal, which means that they were classified correctly and confusion matrix looks good. To focus the plot only on the errors, we divide each value in confusion matrix by number of total values in corresponding class to calculate the error rates instead of absolute errors numbers. Then, we fill the diagonal with zeros to keep only the errors. Fig. 8 shows the error rates of confusion matrix (8).

Analysis of Fig. 8 reveals that we are facing some errors in predicted third and fifth classes which belong to $\{m = 1, n = 0\}$ and $\{m = 2, n = 0\}$, respectively. We may find the reason of error for these classes, by looking at the equation of generated data (i.e., (1)), where we notice that both $\{m = 1, n = 0\}$ and $\{m = 2, n = 0\}$ results $E_x = 0$, meaning that they have the same magnitude and phase plots equal to zero. Therefore, the ML model may not classify them correctly. It is notable that even a human brain cannot classify the zero input with 100% accuracy, because a zero plot may belong to either of these two modes. Therefore, the SGD model classifies the zero input with only 50% accuracy, which is observable in (8). We found the reason, therefore, as we stated before, our efforts should be spent on reducing the error by doing actions like gathering more training data for these types of errors so that the classifier can learn to distinguish them from actual values. However, adding more training data generated by (1) will not improve the model. The only solution is to remove our limitations and use additional data such as E_y, H_x, H_y, and H_z in for these classes.

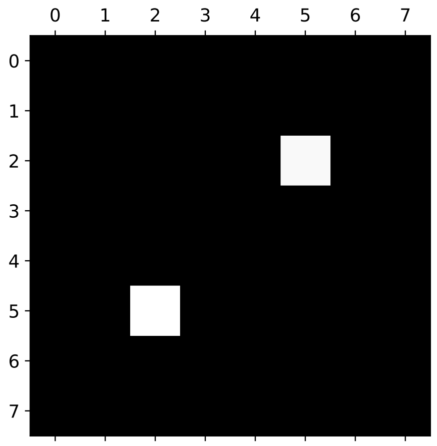


FIGURE 8. Image representation for error rates of confusion matrix in SGD classifier.

Besides the inevitable errors shown in Fig. 8, the proposed SGD model has a good performance in general. It is notable that for $incr = 50$, the execution time to train and evaluate the SGD model on exponentially distributed noisy dataset is about 547 seconds, which is more than the execution time of this model for a noisy dataset with Gaussian distribution (393 sec).

B. K-NEAREST NEIGHBORS (KNN) CLASSIFIER

To train and evaluate a second model, we used K-Nearest Neighbors (KNN) classifier. We trained a KNN model and correctly predicted the random instance 3 as eighth class ($m = 2$ and $n = 2$). To measure the performance of KNN classifier, we obtained the confusion matrix of KNN as follows.

$$confusion = \begin{bmatrix} 6244 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6228 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3643 & 0 & 0 & 2653 & 0 & 0 \\ 0 & 0 & 0 & 6221 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6235 & 0 & 0 & 0 \\ 0 & 0 & 3585 & 0 & 0 & 2675 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6257 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6259 \end{bmatrix} \quad (9)$$

The obtained confusion matrix of KNN classifier (9) is almost same as the confusion matrix of SGD classifier (8). The calculated values of *Precision*, *Recall*, and F_1 scores using

KNN classifier are as follows.

$$Precision = 0.87577959$$
$$Recall = 0.87574220$$
$$F_1 score = 0.87505380$$

Subsequently, 3-fold cross-validation scores of KNN classifier for exponentially distributed noisy dataset are:

$$\text{accuracy} = [0.8729825, 0.8782024, 0.8745349]$$

As we see, KNN model for exponentially distributed noisy dataset has above 87.29% accuracy, which is the almost same accuracy of SGD model for this dataset (87.28%), and is slightly less than accuracy of KNN model for noisy dataset with Gaussian distribution (87.47%).

We provided the image representation of confusion matrix (9) and its error rate in Fig. 9 and Fig. 10.

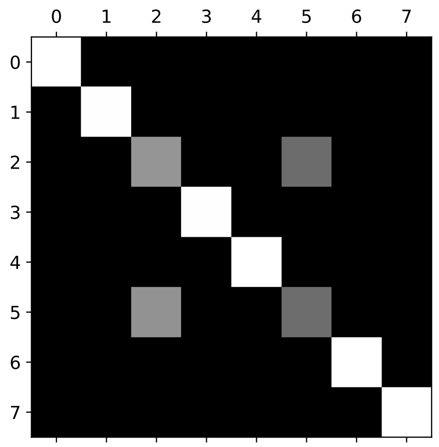


FIGURE 9. Image representation of confusion matrix of KNN classifier.

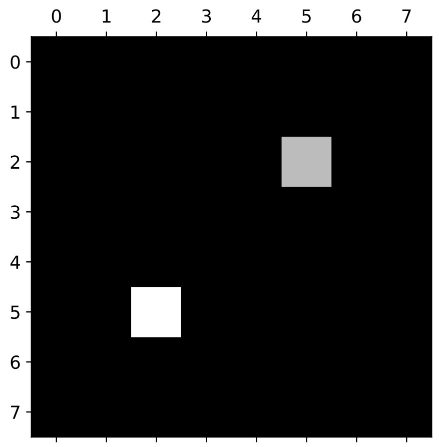


FIGURE 10. Image representation for error rates of confusion matrix of KNN classifier.

Analysis of Fig. 10 reveals that we are again facing errors in predicting between third fifth class, and our efforts must be spent on reducing these types of errors.

The results of this model for exponentially distributed noisy dataset is almost same as the SGD classifier, however, the execution time of KNN (855 sec) is more that SGD classifier (547 sec) for this type of noise, and is almost same as execution time of KNN for dataset with Gaussian noise (819 sec). As we noticed, there is no significant difference between the accuracy and F_1 scores of SGD and KNN. Therefore, due to lower execution time of SGD classifier, we select SGD classifier as the optimal model.

In summary, we provided the performance measure of two models for noisy dataset with exponential distribution in Table. 1.

TABLE 1. Performance measurements of SGD and KNN classifiers for exponentially distributed noisy dataset.

Model	Precision	Recall	F_1	3-fold cross-validation			Accuracy
				1st fold	2nd fold	3rd fold	
SGD	0.87497	0.87497	0.87497	0.8755	0.8728	0.8748	87.28
KNN	0.87577	0.87574	0.87505	0.8729	0.8782	0.8745	87.29

As it can be seen from Table. 1, the performance of two models for exponentially distributed noisy dataset are almost same. Therefore, we compare the execution time to determine the optimal model. Table. 2 shows the summary of F_1 scores, accuracy, and execution time of two models for two types of noise distributions.

TABLE 2. Comparison of SGD and KNN classifiers for different noisy datasets.

Model	Noise distribution	Execution time (s)	Accuracy
SGD	Exponential	547	87.28
SGD	Gaussian	393	87.39
KNN	Exponential	855	87.29
KNN	Gaussian	819	87.47

According to Table. 2, it seems that both the SGD and KNN models require more effort to detect the modal configuration in the case of noisy datasets with exponential distributions. Additionally, Table 2 illustrates that the exponentially distributed noise has been affected the dataset and the performance of models as it results in lower accuracy and requires more computational time for both SGD and KNN classifiers.

C. SYSTEM ACCURACY ON TEST DATASET

As we noted in previous section, due to less execution time of SGD, we select SGD classifier as our optimal model. This classifier performed well on training dataset. Therefore, we must evaluate its performance on the test dataset, which has never been seen by this model. The accuracy score of SGD classifier on the test dataset which includes 14,000 unseen instances, was 0.87835. This means that we have generalization accuracy above 87.83%, which is a good accuracy, however, we should remember that this model have a problem in predicting modes with $n = 0$, and requires a change in problem limitations. It is notable that the corresponding algorithms for the proposed ML models are released on GitHub and are accessible through [16].

IV. LIMITATIONS AND APPLICABILITY

Although we developed a final model performing well with an accuracy exceeding 87.83%, it remains inefficient in predicting modes with $n = 0$. This limitation stems from our training data, which was generated only based on the electric fields in the $\hat{x}$ direction as in (1). Since the mode with $n = 0$ always yield $E_x = 0$, regardless of the m value, their features become identical, making them indistinguishable for the machine. To address this issue, we might consider incorporating an additional field component such as E_y or magnetic field components into the training data generation process. This would ensure distinct input features for each unique combination of m and n. However, introducing an extra equation would increase the number of features significantly, potentially posing new challenges for modeling the problem.

The provided methodology is not restricted to predicting TE mode numbers in rectangular PEC waveguides. It is also applicable to predicting TM mode numbers in rectangular waveguides, as well as any other waveguide that can be described by the two-dimensional (2-D) Helmholtz equation. For instance, to predict the TE mode numbers of a circular waveguide, (1) can be substituted by E_ρ and/or E_ϕ as described in [17, p. 486] for data generation purposes. Moreover, while our current approach relies on analytical formulas for generating training data, for more complex waveguides, training data may be generated using solutions obtained through advanced numerical methods or EM simulators such as Ansys HFSS [18], CST Studio Suite [19], Altair Feko [20], and others. Therefore, the development of an efficient model for mode numbers prediction in broader structural configurations with more complex geometries could be explored as a potential future work.

V. CONCLUSION

We proposed an ML-based modeling methodology for predicting the propagation mode number inside a rectangular waveguide, based on the noisy field configurations in metallic rectangular waveguides, We trained two types of classifiers: SGD and KNN to predict the TE mode numbers, based on datasets containing magnitude and phase plots of $\vec{E}$. We added random noise to $\vec{E}$ datasets with exponential and Gaussian distributions. To measure the performance of classifiers, we calculated the *confusion matrix*, *precision*, *recall*, and F_1 scores. Additionally, we evaluated the performance of these models using cross-validation techniques to find their corresponding accuracy scores. We analyzed the model errors and provided solutions.

For this problem, the *SGD* classifier was found to be more efficient in terms of execution time, while both models had almost the same accuracy. Therefore, we selected *SGD* classifier as our final model. Additionally, we provided a comparison of these two models for different noisy datasets with exponential and Gaussian distributions, in terms of time and accuracy.

REFERENCES

[1] A. Geron, *Hands on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly, 2019.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1–9.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," 2016, *arXiv:1606.01781*.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[6] R. Choupanzadeh and A. Zadehgol, "A deep neural network modeling methodology for efficient EMC assessment of shielding enclosures using MECA-generated RCS training data," *IEEE Trans. Electromagn. Compat.*, vol. 65, no. 6, pp. 1782–1792, Dec. 2023.

[7] V. K. Devabhaktuni, M. C. E. Yagoub, and Q.-J. Zhang, "A robust algorithm for automatic development of neural-network models for microwave applications," *IEEE Trans. Microw. Theory Techn.*, vol. 49, no. 12, pp. 2282–2291, 2001.

[8] X. Ding, V. K. Devabhaktuni, B. Chattaraj, M. C. E. Yagoub, M. Deo, J. Xu, and Q. J. Zhang, "Neural-network approaches to electromagnetic-based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization," *IEEE Trans. Microw. Theory Techn.*, vol. 52, no. 1, pp. 436–449, Jan. 2004.

[9] V. K. Devabhaktuni, C. Xi, F. Wang, and Q.-J. Zhang, "Robust training of microwave neural models," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 12, no. 1, pp. 109–124, Jan. 2002.

[10] A. Sohani, C. Cornaro, M. H. Shahverdian, S. Hoseinzadeh, D. Moser, B. Nastasi, H. Sayyaadi, and D. Astiaso Garcia, "Thermography and machine learning combination for comprehensive analysis of transient response of a photovoltaic module to water cooling," *Renew. Energy*, vol. 210, pp. 451–461, Jul. 2023.

[11] D. M. Pozar, *Microwave Engineering*, 4th ed. Hoboken, NJ, USA: Wiley, 2011.

[12] M. Rouaud, *Probability, Statistics and Estimation: Propagation of Uncertainties in Experimental Measurement*. Chico, CA, USA: Lulu, 2017.

[13] A. Zadehgol, "Reduced-order stochastic electromagnetic macro-models for uncertainty characterization of 3-D band-gap structures, in FDTD," in *Proc. IEEE Int. Symp. Antennas Propag. USNC/URSI Nat. Radio Sci. Meeting*, San Diego, CA, USA, Jul. 2017, pp. 2421–2425.

[14] Python Software Foundation. *Python Language Reference, Version 3.11.4*. Accessed: Feb. 20, 2024. [Online]. Available: https://www.python.org

[15] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[16] R. Choupanzadeh. *Machine Learning for Detection of Modal Configurations in Rectangular Waveguides*. Accessed: Jan. 8, 2024. [Online]. Available: https://github.com/RasulChoupanzadeh/ML-for-Rectangular-Waveguide-Modes
[17] C. A. Balanis, *Advanced Engineering Electromagnetics*, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.
[18] Ansys. *Ansys HFSS 2023*. Accessed: May 9, 2023. [Online]. Available: https://www.ansys.com/products/electronics/ansys-hfss
[19] CST-Computer Simulation Technology AG. (2022). *CST Studio Suite*. Accessed: May 9, 2023. [Online]. Available: https://www.3ds.com/products-services/simulia/products/cst-studio-suite
[20] Altair Engineering. (2022). *FEKO Electromagnetic Simulation Software*. Accessed: May 9, 2023. [Online]. Available: https://altair.com/feko/

RASUL CHOUPANZADEH (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering from the K. N. Toosi University of Technology, Tehran, Iran, and the M.S. degree in electrical engineering from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Applied Computational Electromagnetics and Signal/Power Integrity (ACEM-SPI) Group, University of Idaho, Moscow, ID, USA. He is also a Research Assistant with ACEM-SPI Group, University of Idaho. His research interests include electromagnetic field theory, computational electromagnetics, electromagnetic modeling, scattering and wave propagation, macromodeling, equivalent circuits, signal and power integrity, and electromagnetic compatibility.

ATA ZADEHGOL (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Washington, in 1996, the M.S. degree in electrical and computer engineering (ECE) from the University of California at Davis, in 2006, and the Ph.D. degree in ECE from the University of Illinois at Urbana–Champaign, in 2011. With more than a decade of experience in advanced microelectronics industry, in 2014, he joined the University of Idaho, where he is currently an Associate Professor in ECE and the Director of the Applied Computational Electromagnetics and Signal/Power Integrity (ACEM-SPI) Group. His research interests include computational electromagnetics and its various applications in low-frequency to THz electronic systems. His work has been recognized through research grants from the National Science Foundation, NASA, Micron Technology Inc., and Schweitzer Engineering Laboratories, the IEEE TCPMT Best Poster-Paper Award, and University of Idaho's Presidential Mid-Career Award. He is a Professional Engineer Licensed in Idaho State.