IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier

Detection of Modal configurations in Rectangular Waveguides via Machine Learning Algorithms for Noisy Datasets

RASUL CHOUPANZADEH
Electrical and Computer Engineering Department, University of Idaho, Moscow, USA (e-mail: chou1121@vandals.uidaho.edu)
Corresponding author: Rasul Choupanzadeh (chou1121@vandals.uidaho.edu)

ABSTRACT This paper presents an application of Machine Learning (ML) algorithms in the field of Electromagnetics, or particularly in Microwave. In this work, we use ML models to predict the propagation mode number of a given Electric (E) field inside a waveguide in the presence of noise. To restrict the problem, we only consider predicting Transverse Electric (TE_{mn}) mode numbers; assuming m and n in the range of 0 to 2, inside a rectangular waveguide due to the given magnitude and phase plots of E field in $\hat{x}$ direction (i.e., $|E_x|$ and $\angle E_x$), while assuming propagates in $\hat{z}$ direction, and having a noise with Exponential distribution. This problem will be treated as a classification task of ML. We generate a dataset including 64,000 plots for each of $|E_x|$ and $\angle E_x$, for various propagation TE modes and frequency range 13-17 GHz, to train our ML models. This paper provides training and evaluation of two classification models such as *Stochastic Gradient Descent (SGD)* and *k-Nearest Neighbors* for a noisy dataset with Exponential distribution, and compares to the Gaussian noise distribution. Finally, we discuss the challenges, solutions, assumptions, and limitations of the models to find most appropriate model for this particular problem.

INDEX TERMS classification, dataset, Exponential distribution, Gaussian noise, k-Nearest neighbors, machine learning, mode number, propagation, stochastic gradient descent, training, transverse electric, waveguide.

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]. In fact, ML has shown great success in building models for various tasks such as pattern recognition in domains ranging from computer vision [2] over speech recognition [3] and text understanding [4] to Game AI [5]. In addition to these domains, machine learning is increasingly important and successful in engineering and sciences, which is due to the data-based nature of ML from a huge number of examples.

There are various types of ML systems which may be categorized in broad categories such as regression versus classification in terms of the type of problem, supervised, unsupervised, semi-supervised, and reinforcement learning in terms of the need to human supervision, online versus batch-learning in terms of the leaning ways, and instance-based versus model-based learning in terms of learning methods. Further details are about these categories are provided in [1].

In the field of microwaves, one of the typical tasks are calculating Electric (E) and Magnetic (H) fields inside the waveguides and obtaining the cross-sectional plots due to a given propagation mode number [6]. However, in some cases, we may need to do the reverse process, meaning that we need to detect the modal configuration due to given plots of E and H fields. Although, there exist many previous works throughout literature that predicts the modal configuration through various algorithms, the author was unable to find an ML based algorithm that predicts the modal configuration inside rectangular waveguide.

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 comparing to 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 different 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, proposed methods for performance measuring, 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 IV we conclude with some closing remarks.

II. METHODOLOGY

A. STRUCTURE

As we noted previously, the main objective of this work is predicting the TE^z_{mn} propagation mode number of 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 can not 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.Fig. 1 shows The structure of described waveguide.

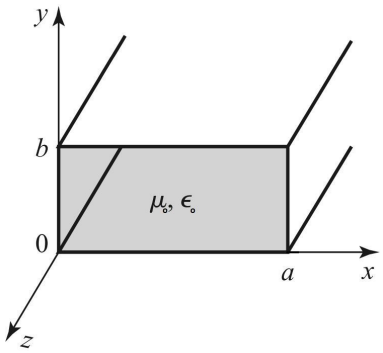


FIGURE 1. Geometry of rectangular waveguide

For data generation purposes, we use (1) to calculate the Electric filed in $\hat{x}$ direction (E_x) for TE mode configuration [6]. Accordingly, magnitude and phase datasets for each mode number m and n can be obtained.

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

where, $\mu_0 = 4\pi \times 10^{-7}$ is free-space permeability, $A = 1$, a and b are the given 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, $k = \omega\sqrt{\mu_0\epsilon_0}$ is wave number in unbounded medium, and $\epsilon_0 = 8.854 \times 10^{-12}$ is free-space permittivity.

B. NOISE DISTRIBUTIONS

As we mentioned, the objective of this work is to find the mode numbers of a noisy dataset. Therefore, we must add noises to the clean dataset generated through (1). We have several options for the type of noise distribution. One of the popular distributions is normal distribution, which is also known as Gaussian distribution. The Gaussian noise formula is written as follows [7].

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

where, μ is mean, and σ is standard deviation. Another well-known case is Exponential distribution, which is formulated as follows [8].

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

where, $\lambda > 0$ is the rate parameter. It is notable that in Exponential distribution $\mu = \sigma = 1/\lambda$. Fig. 2 shows a particular case of Exponential and Gaussian noise distributions for comparison.

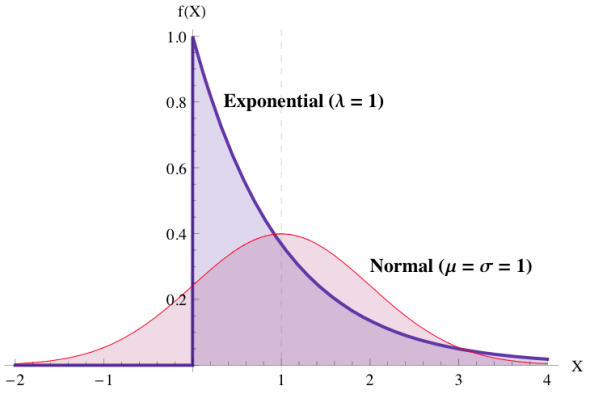


FIGURE 2. Gaussian Distribution

Therefore, we will add an Exponentially distributed noise to the clean data, and provide a noisy dataset. We will use this noisy dataset to train the ML models, and compare it to the case of noisy dataset with Gaussian distribution.

C. FRAMING THE PROBLEM

Before selecting and training a model, we should frame and categorize the problem. The provided data includes propagating mode numbers (i.e., m and n) as labels, and noisy values of magnitude and phase at any point (x, y) inside the waveguide with operating frequency range 13-17 GHz as features. Our objective is building a model that learns from this data and is able to predict m and n (between 0 and 2) for any instances, given the plots of magnitude and phase of E_x. Obviously, it is a *supervised* task, since the instances are labeled. Since we are classifying m and n for any instances from 0 to 2, it is a *classification* task. It is also a *multiclass* classification, since we are categorizing two labels m and n in different classes from 0 to 2. 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

Since this problem is a particularly designed problem with specified fixed values, we are unable to find a pre-generated data, therefore, we must generate our own data to train the ML models. The data are generated with the written *data_generation.py* python script; which is provided in the folder of this article, by considering the mentioned values, dimensions, and frequencies. Therefore, we can load our generated dataset to train our models instead of generating them for each model.

E. DATA PREPARATION

We should perform 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 must shuffle the dataset as the first step of data preparation. The next step is applying noise. Mainly, 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. Therefore, we add a Exponentially distributed noise to the clean data generated by *data_generation.py* python script. The final step is setting aside a part of the data randomly as the test set, which is normally 20% of whole dataset, and name the remaining as training dataset. As our brains may bias any model, it is crucial to create the test set before knowing more about the data. We also may need to discover and visualize the training dataset to gain insights about them. For example, since we are working with magnitude and phase of E_x, we may 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 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. We selected the first two options, trained, evaluated, and compared them. Finally, after comparison of the results, we selected the model which performs better for this particular type of problem.

It is notable that we may treat the problem in two ways. Firstly, consider the problem as a multi-label multiclass classification for predicting m and n labels. Secondly, assigning a class for each pair of (m,n) to consider the problem as a single-label multiclass 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, 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:

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

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 will use cross-validation score 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 [1]. Another way to evaluate the performance of a classifier is evaluating the confusion matrix. To compute the confusion matrix, we need to have 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 thanks to 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 [1]. After obtaining the confusion matrix, we are able to calculate the values of precision and recall (sensitivity or true positive rate) as follows [1].

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

which shows the accuracy of true predictions (i.e., how many of values 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 [1].

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

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

H. EVALUATING THE SYSTEM ON TEST SET

After obtaining a final model that performs well on training dataset, we must evaluate the model on test dataset, which is called 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

Assume that we have found a promising model evaluated by training data and test dataset, and we found a significant generalization error, therefore, we should analyze the errors to find the reason. The best way to do this is evaluating the confusion matrix and plotting the image representation of confusion matrix. Analyzing the confusion matrix often can give us insights into ways to improve our models [1].

III. NUMERICAL RESULTS

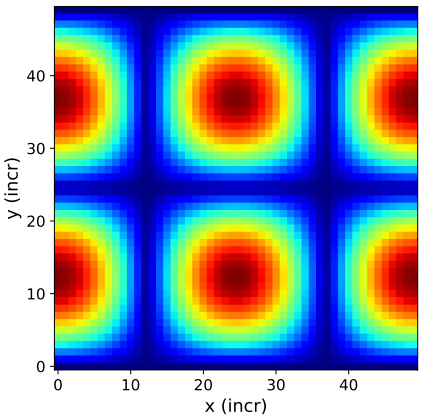
In this section we show numerical results of proposed classification models for prediction of TE mode number.

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 (x, y) into $incr = 50$ points, and calculate the magnitude and phase values on these points. Thus, each of magnitude and phase datasets (plots) have a dimension 50*50, which leads to 2500 features. Therefore, by assuming $incr = 50$, the total number of features for each instance is:

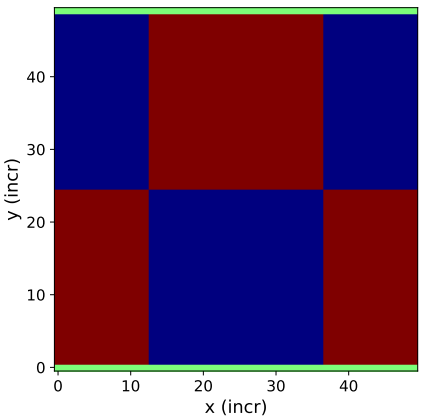
$$\text{Features} = \text{Magnitude features+Phase features} = 5000 \quad (7)$$

In generated 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 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 training the models on them.

In order to produce a noisy dataset, we added a Exponentially distributed noise with rate parameter $\lambda = 1$, and Gaussian noise with standard normal distribution ($\mu = 0$, $\sigma = 1$) to generated clean and shuffled dataset. 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 a 50*50 array, and display it using Matplotlib's *imshow()* function. 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).



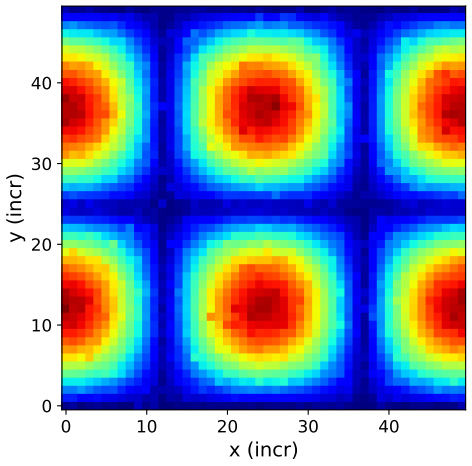
(a)



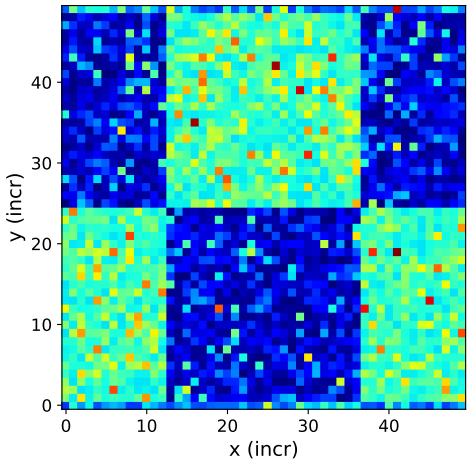
(b)

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

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 more features, but also, requires more execution time and memory. 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

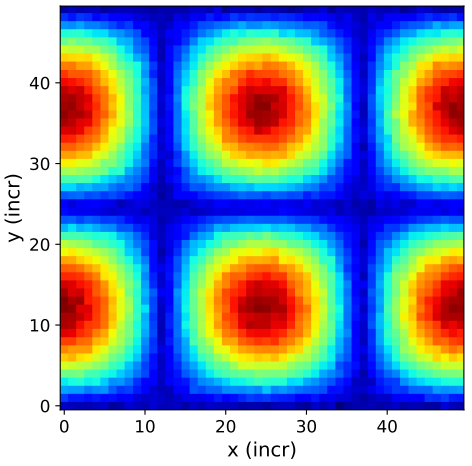


(a)

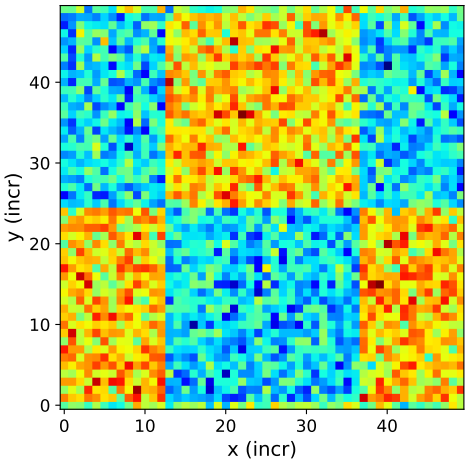


(b)

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



(a)



(b)

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

train our models, which resulted a good performance.

Now, lets see the noise effects on magnitude and phase plots. The Fig. 4 and Fig. 5 shows magnitude and phase plots of the same instance in dataset after adding noises with Exponential and Gaussian distributions.

The rates of noise effect can be observed clearly by comparison of Fig. 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. Now, we have a noisy datasets which is divided into training and test datasets. We should select and train ML models using training datasets. Fig. 6 shows the histogram of generated noises using Exponential and Gaussian distributions.

Du to page limitation, we provided 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.

A. SGD CLASSIFIER

As we mentioned before, we had several options to select a model. First, we used a multiclass SGD classifier and trained it. This classifier has the advantage of being capable of handling very large datasets efficiently. Training the SGD classifier on training set is very simple, and all we need to do is using the *fit* function and put the all features of training dataset (excluding labels) and a vector of target labels (i.e.,classes). We successfully trained the SGD model, and predicted instance 3 correctly 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 this classifier. The resulted scores for instance 3, using SGD classifier, are shown as follows.

scores: [-202737.63, -9162.73, -3200.45, -167285.61, -51402.26, -3107.66, -38067.77, 38919.18]

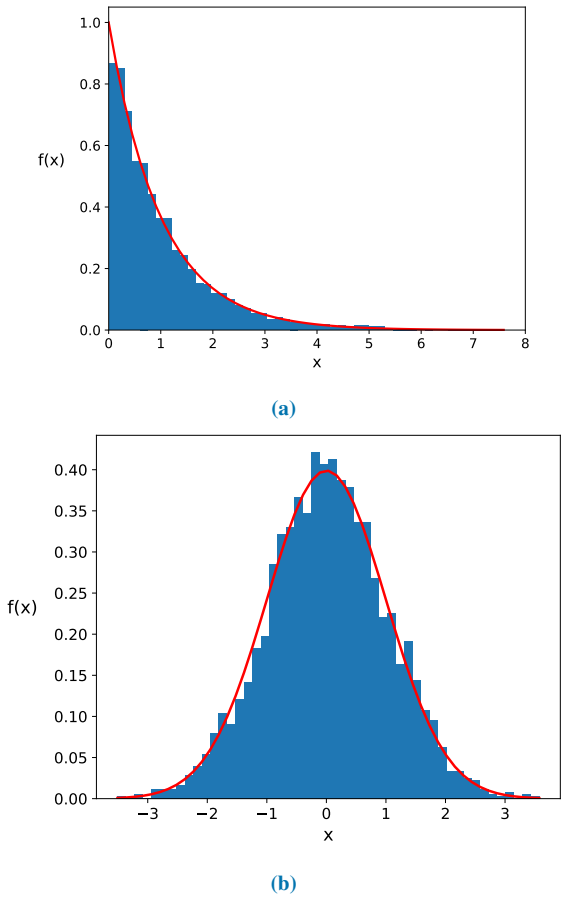


FIGURE 6. Histogram of noise generated by: (a) Exponential distribution, (b) Gaussian distribution

In decision tree function, highest score indicates the correct corresponding class. For example, in the previous scores, the highest score belong 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 need to measure the performance of our 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 training dataset is calculated as (8).

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

Comparing to the confusion matrix of a perfect classifier, which is a diagonal matrix, confusion matrix (8) illustrates a good classification for this model, except for the third and fifth class. We will show this exception and explain the reason, and provide a solution, in the error analysis. We are able to 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:

$$\begin{aligned} \textit{Precision} &= 0.874975575 \\ \textit{Recall} &= 0.874975579 \\ F_1\ \textit{score} &= 0.874972627 \end{aligned}$$

Another method to evaluate the performance of model, which is more common, is using cross-validation. To this end, we used 3-fold cross-validation for performance evaluation of SGD classifier. 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. However it is a good accuracy, but it is slightly lesser than accuracy of this model for noisy dataset with Gaussian distribution which is above 87.39%.

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

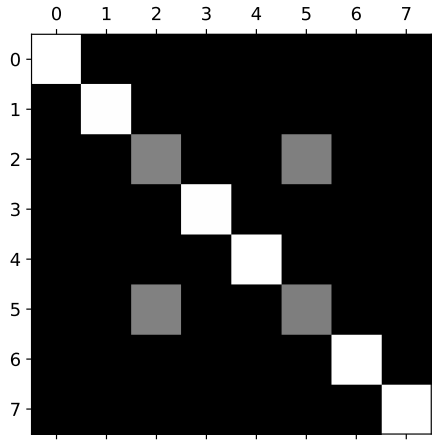


FIGURE 7. Image representation of confusion matrix of SGD classifier

fifth classes) are on main diagonal, which means that they were classified correctly and confusion matrix looks good. To focus the plot on errors, we need to divide each value in confusion matrix by number of images in corresponding class to calculate the error rates instead of absolute errors numbers. Then, we must 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. But why this classes? What is the reason? If we look at the equation of generated data (i.e., (1)), we notice that both $\{m = 1, n = 0\}$

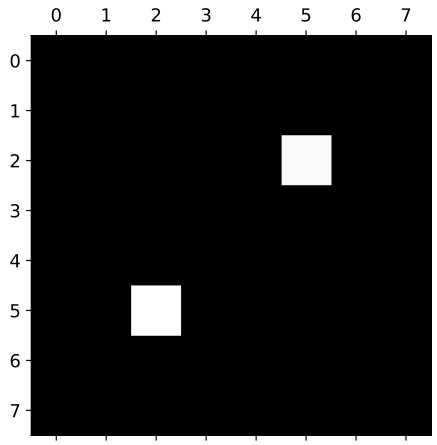


FIGURE 8. Image representation for error rates of confusion matrix of SGD classifier

and $\{m = 2, n = 0\}$ results $E_x = 0$, which means that they have the same magnitude and phase plots equal to zero. Accordingly, when ML model see the plots of zero, it cannot classify it. Even, a human brain cannot classify the zero input correctly by 100%, because a zero plot may belong to any of these two modes. Therefore, this model classifies the zero input correctly by 50%, 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 other data such as E_y, E_z, H_x, H_y, and H_z.

Although we had some inevitable errors, shown in Fig. 8, but 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 the KNN model and it correctly predicted instance 3 as eighth class ($m = 2$ and $n = 2$). To measure the performance of KNN classifier, we obtained its confusion matrix as follows.

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

The 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.

$$\begin{aligned} \textit{Precision} &= 0.87577959 \\ \textit{Recall} &= 0.87574220 \\ F_1\ \textit{score} &= 0.87505380 \end{aligned}$$

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 lesser 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.

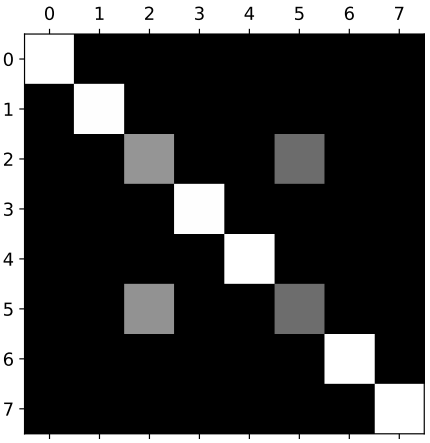


FIGURE 9. Image representation of confusion matrix of KNN classifier

Analysis of Fig. 10 reveals that we are again facing errors in predicting between third fifth class, and our efforts should 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 shorter execution time of SGD classifier, we select SGD classifier as our desired model.

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

As we can see in Table. 1, the performance of two models for Exponentially distributed noisy dataset are almost same.

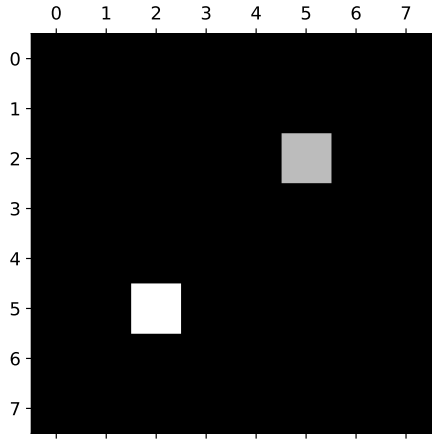


FIGURE 10. Image representation for error rates of confusion matrix of KNN classifier

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

Model	Precision	Recall	F_1	3-fold cross-validation			Accuracy
				1^{st} fold	2^{nd} fold	3^{rd} 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

We should compare the execution time to determine the model. Table. 2 shows the F_1 scores, accuracy, and execution time of two models for two types of noisy datasets.

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 models need more effort to detect the modal configuration for noisy datasets with Exponential distributions.

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 final model. This classifier works good on training dataset, therefore, we must evaluate its performance on test dataset, which has never been seen by the SGD classifier. 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.

IV. CONCLUSION

We achieved excellent results in this work, in which we performed a rigorous and comprehensive analogy of ML models to predict the propagation TE mode number inside a rectangular waveguide in the presence of Exponentially distributed noise comparing to Gaussian distributed noise, as follows. First, we trained two types of classifiers such as *Stochastic Gradient Descent* and *K-nearest Neighbors* to predict the propagating TE mode numbers inside a rectangular waveguide due to the given magnitude and phase plots of E_x and adding an Exponentially distributed noise. To measure the performance of classifiers, we calculated the confusion matrix, precision, recall, and F_1 scores. Additionally, evaluated performance of these models using cross-validation technique to find their corresponding accuracy scores. Finally, analyzed the model errors and provide solutions. In comparison with other classifier, *SGD* was fastest, while both of the models had almost same accuracy. Therefore, we select *SGD classifier* as our final model. As an additional work, we provided a comparison of these two models for different noisy dataset with Gaussian distribution, in terms of time and accuracy.

V. ACKNOWLEDGMENT

This paper was written for a class project in the course entitled ECE 504, ST:"Machine Learning for Electromagnetics" taught in Spring 2022 by Dr. Ata Zadehgol at the University of Idaho in Moscow. The results of this paper was generated with the help of [1] and lectures of Dr. Ata Zadehgol .Special thanks to Aurélien Géron and Dr. Ata Zadehgol for providing such an amazing book and interesting course, respectively.

REFERENCES

[1] A. Geron. Hands on Machine Learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems, 2nd edition. O'Reilly, Sebastopol, California, 2019.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In 2012 Neural Information Processing Systems (NIPS).
[3] G. Hinton, L. Deng, D. Yu, G. Dahl, and et al. Deep neural networks for acoustic modeling in speech recognition. Signal Processing Magazine, 29, 2012.
[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. arxiv:1606.01781, 2016.
[5] D. Silver, A. Huang, C. G. Maddison, A. Guez, and et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587), 2016.
[6] David. M. Pozar. Microwave Engineering, 4th edition. John Wiley and Sons, 2011.
[7] A. Zadehgol. Reduced-order stochastic electromagnetic macro-models for uncertainty characterization of 3-d band-gap structures, in FDTD. In 2017 IEEE International Symposium on Antennas and Propagation and USNC/URSI National Radio Science Meeting, San Diego, CA, USA.
[8] N. Balakrishnan and Asit P. Basu. Exponential Distribution: Theory, Methods and Applications, 1st edition. CRC Press, 1996.