Microwave Transistor Modeling Using Machine Learning Techniques

An Analysis of Cross-Validation, Optimizer Performance, and Extra Randomness in Tree-Based Models for GaN HEMT Modeling in Comparison with Artificial Neural Networks

Capstone Report

Rauan Ibragim

Nazarbayev University
Department of Electrical and Computer Engineering
School of Engineering and Digital Sciences

This project report was created on TexStudio editing platform using LaTeX. All the figures were drawn using draw.io online software tool.

NAZARBAYEV UNIVERSITY

Electrical and Computer Engineering
Nazarbayev University
http://www.nu.edu.kz

Title:
Microwave Transistor Modeling using Machine Learning Techniques

Theme:
An Analysis of Cross-Validation, Optimizer Performance, and Extra Randomness in Tree-Based Models for GaN HEMT Modeling in Comparison with Artificial Neural Networks

Project Period:
Fall 2024 - Spring 2025

Project Group:
Lab's Name

Participant(s):
Rauan Ibragim

Supervisor(s):
Mohammad Hashmi

Copies: 1

Page Numbers: 43

Date of Completion:
April 24, 2025

Abstract:

The current research assesses machine learning methods for modeling GaN HEMT, specifically interpolation and extrapolation issues within higher-frequency circuits. S-parameters prediction based on GaN HEMT data is evaluated by comparing Neural Networks (ANN) and Tree-Based Models (Extra Trees - ET, Random Forests - RF). The methodology involved splitting data by V_{DS} bias and comparing validation strategies (80/20 split vs 5-fold CV) and various hyperparameter optimizers. Results show the 80/20 split is significantly faster ($\sim 9\times$) with comparable or better performance than CV=5. ET models generally outperformed RF. ET GA outperformed in interpolation, whereas ANN Optuna had superior stability and robustness with extrapolation. Models based on trees are even simpler and much quicker to apply and adjust (ET GA tuning $\sim$14 times quicker compared to ANN Optuna). The research confirms the trade-offs between types of models and validation strategies for accurate and efficient microwave transistor modeling.

Contents

Preface

This capstone project is born from the ever-grown need for more accurate models within the framework of RF and microwave design. The work explores how congenial machine learning approaches can be for interpolation and extrapolation, hence addressing some the challenging issues in high-frequency transistor modeling. Through this project, I have been able to combine two of my areas of interest - RF engineering and machine learning with valuable insights and practical skills.

I would like to thank Dr. Mohammad Hashmi and Mr. Saddam Hussain for invaluable guidance in the whole process. Their expertise and encouragement have been a significant boost to the development of this work.

Nazarbayev University, April 24, 2025

Rauan Ibragim
<Rauan.Ibragim@nu.edu.kz>

Chapter 1

Introduction

Microwave transistors have some unique features that differentiate them from conventional silicon-based transistors. These features are high-frequency operation, power amplification, and low-noise generation [1, 2, 3]. Due to these characteristics, modern communication systems depend heavily on microwave transistors [4, 5, 6]. The rapid development of wireless communications increased the demand for high-performance microwave transistors, making them more complex. Therefore, conventional modeling techniques for these devices, which are primarily physics-based, have grown computationally expensive and challenging to scale [7, 8]. This is especially true when considering process differences, bias conditions, and varied frequency ranges [7, 9]. Data-driven approaches have become promising solutions to these challenges. Artificial Neural Networks, Decision Trees, and Support Vector Regression use massive datasets to forecast device performance. The computational efficiency and flexibility of these models achieved faster simulations and design iterations than physics-based models [10, 11, 12, 13].

One of the key stages in microwave transistor design is small-signal modeling as it makes it possible to predict how a transistor will behave under different operating conditions. The conventional process for modeling transistors involves selecting the appropriate transistor technology, such as GaN HEMTs or SiGe HBTs, measuring scattering parameters (S-parameters), and then fitting these parameters into a model [1]. Besides, the procedure also entails extensive data preprocessing, model selection, parameter extraction, and validation [14, 15, 16, 17].

A similar pipeline was used in developing small-signal machine learning (ML) models in this work. The research compares the capabilities of tree-based models with neural networks – how accurately they forecast the S-parameters of microwave transistors under various operating scenarios. It is discovered in many papers that complex patterns in data can be captured by ANNs thanks to their multilayered design and weight modifications [10, 12, 18, 15, 19]. Random forests (RFs) – an ensemble of decision trees – offer better accuracy and robustness by

minimizing overfitting [15, 16, 17, 20, 19]. However, there is no work that utilizes extra trees (ETs) in microwave transistor modeling.

This study compares the performance of ANNs, RFs, and ETs, and determines how they perform with different optimizers such as random search, grid search, bayes search with gaussian process, bayes search with tree-structured Parzen estimator, and genetic algorithms. We systematically explore these models and factors that affect their performance such as hyperparameter tuning techniques, cross-validation, random seed, parallelism, and regularization. Model training and testing are done under the same conditions with the same dataset and data split. Gallium Nitride (GaN) high electron mobility transistor's (HEMT) data was used for all models. The background section elaborates on transistor modeling techniques and stages, the structure of GaN HEMTs, and importance of data-driven models. The methodology section explains the implemented machine learning pipeline, ML and hyperparameter tuning algorithms, and performance metrics used in the study. The results and discussions sections provide the model performance results in different forms, explaining how each factor affected the output. The conclusion section highlights the findings, implications, and limitations of the study.

Microwave transistor design has advanced significantly with the use of machine learning models instead of conventional physics-based methods. This method creates new opportunities for quicker, more effective device development by cutting down on simulation times and improving model adaptability [21]. The application of these techniques can spur innovation in wireless communication technologies like 5G, radar systems, and satellite communications, therefore the consequences of this research go beyond the technological sphere [22]. Designing the next generation of microwave transistors will require more and more data-driven techniques as the need for high-frequency devices grows. Figure 1.1 depicts example of GaN HEMT amplifier in a test board circuit [23].

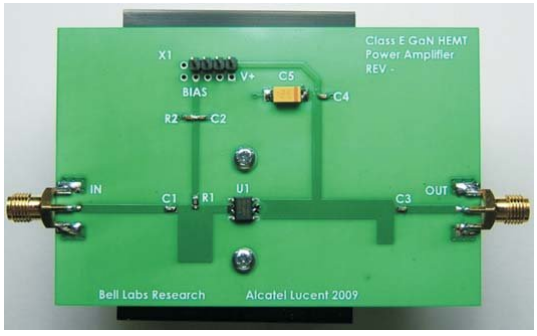


Figure 1.1: GaN HEMT Power Amplifier Test Board

1.1 Ethical and Professional Responsibilities

- **Ethical Responsibility:**
  There are several ethical considerations that should the addressed during the development and implementation of the project. The first potential issue is data privacy and integrity. To prevent risks and issues in this regard, the datasets will be obtained from legitimate sources, with use adhering to the terms of any applicable licenses.

  The next ethical concern arises when the model unintentionally favors one set of conditions over others. The algorithmic bias comes from specific cases of training data that give way to inaccurate predictions under conditions where the data is underrepresented. This potential issue will be mitigated with validation and testing.

  The results of this project must, in the end, be presented completely transparently to mitigate misinterpretation of data and results. Otherwise, it could lead to incorrect conclusions causing long-term technical and ethical ramifications.

- **Informed Judgments:**
  Informed and responsible decision-making is the key priority in the implementation of this project. Optimal machine learning techniques must be judiciously chosen after having had some idea about their performance in terms of accuracy, computational efficiency, and applicability regarding real-world modeling of transistors. Each decision will be data-driven and will be supported by well-established performance metrics such as mean squared error, accuracy, and generalization capability.

  Besides, the societal implications of the project will be put into consideration in informing the possibly global impact that improved transistor models could have to bear on communications, medical technologies, and defense systems, among other areas that directly impinge on society. Consequently, decisions will result from consultations with experts in machine learning and RF engineering, as well as reviews of the literature concerning wider needs and consequences at the levels of society in general.

- **Global Context:**
  While the modeling techniques being developed in this project have potential applications throughout most regions of the world, especially in countries with growing telecommunications and electronic industries, the global context also presents a range of variables that may impact upon the way in which these models are executed and results arising. For instance, regulatory environments and technological or economic capability may vary between coun-

tries, which would impact the reception of relevant machine learning models for microwave transistors.

Also, the implementation of such technologies in regions with limited access to advanced technologies could be done perhaps at a slow pace to widen further the gap in technology between industrialized and developing nations. On the other hand, for highly developed regions with strong RF and microwave industries, such models will enhance the efficiency of communications systems, making technologies more available and at cheaper costs.

- **Economic Impact:**
  The proposed study will have a short-term economic impact because of the potential to decrease design and manufacturing costs for microwave transistors. As a result of machine learning models, companies can cut the time used and resources spent on traditional trial-and-error methods to arrive at transistors. This can lead to significant cost reductions, especially in those industries where RF components play an integral part in assembly, such as telecommunications and aerospace.

  In the long term, economic effects would be even more drastic. Increased accuracy in transistor behavior predictions by machine learning models will go a long way in ensuring these designs get completed in record time and hence minimize both production costs and time-to-market for new technologies. Anyhow, this can accelerate economic development because businesses could innovate much more quickly and effectively than ever.

  On the other side, this also has a couple of possible drawbacks: initial investments to train personnel on their use and integration with existing workflows, and if all such technologies were widely implemented, as many of the low-skilled manual jobs out of place, especially the need for recently updated workforces to equitably share the economic benefits.

- **Environmental Impact:**
  This project should be expected to promote an overall positive environmental impact by offering the possibility of reduction in resource consumption during the design and testing stages of microwave transistors since the normal modeling methods would require a great deal of time and material for prototyping and testing, which usually results in higher energy consumption and material wastes.

  With the machine learning models that can predict the behavior of the transistor, less physical prototyping can be done. These ML models will further optimize the design process, which will make more energy-efficient transistors and can have positive cascading effects across industries. For example, in telecommunication, more efficient RF components can lower the energy con-

sumption of communication networks that could, over time, lower carbon emissions.

However, machine learning also has an important impact on the environment due to the energy required for model training, especially large neural networks. One way of mitigating this will be through the use of energy-efficient algorithms and cloud services that have lower carbon footprints. In such ways, the project can guarantee maximum environmental benefit with a minimum in energy costs associated with model development.

- **Societal Impact:**
  Improvements in microwave transistor modeling, however, could potentially have near-term benefits for industries in telecommunications, healthcare, and defense-all of which would have definite societal benefits wherein a better model eventually translates to much better-performing devices such as smartphones, medical apparatus, and communication systems that improve people's lives millions of folds worldwide.

  Indirectly, such a project may imply a reduction in the digital gap by allowing cheaper and more efficient technologies. With increases in accuracy and reliability in transistor modeling, potential production costs for high-performance RF systems may decrease, allowing access to advanced technologies for the resisted populations. A societal risk of such rapid advancement in technology is the displacement of jobs amidst automation. The present project also recognizes such risks and underlines that policies should be used to help distribute the societal benefits from technological advancement equitably.

Chapter 2

Background

In preparation for this project, we conducted extensive background reading in the areas of microwave transistor operation, small-signal models, and machine-learning techniques. Specifically, we reviewed the textbook by Jianjun Gao called "RF and Microwave Modeling and Measurement Techniques for Compound Field Effect Transistors" [1] and the article written by Frank Schwierz titled "Microwave Transistors – The last 20 years [24]." Besides, the materials from MATLAB and the Sci-kit-learn websites were crucial for understanding the machine learning techniques and models.

2.1 Features and structure of Microwave Transistors

After reading these works, we can say that microwave transistors are the workhorses of modern wireless communication. They have some unique features that differentiate them from conventional transistors. These features include high-frequency operation, power amplification, and low-noise generation. The microwave transistors consist of heterostructures which makes them different than the homogenous structure of the silicon-based transistors. The use of several materials with different conduction bands creates mechanisms like the formation of a two-dimensional electron gas – 2DEG. This process is discussed in more detail for GaN HEMTs below.

Thanks to their unique features, microwave transistors are widely used in satellite communication, radar systems, mobile phones, and wireless networks. The most commonly used microwave transistors are Gallium Nitride High Electron Mobility Transistors (GaN HEMT), Silicon Germanium (SiGe) Heterojunction Bipolar Transistors (HBT), and Gallium Arsenide MESFET (Metal-Semiconductor Field-Effect Transistor). Each of them has quite different characteristics and is therefore used in various areas. In our case, the chosen microwave transistor is known for its high-power handling capability and excellent frequency performance. Therefore,

GaN HEMTs are very good for high-power amplifiers in cellular base stations and radar systems.

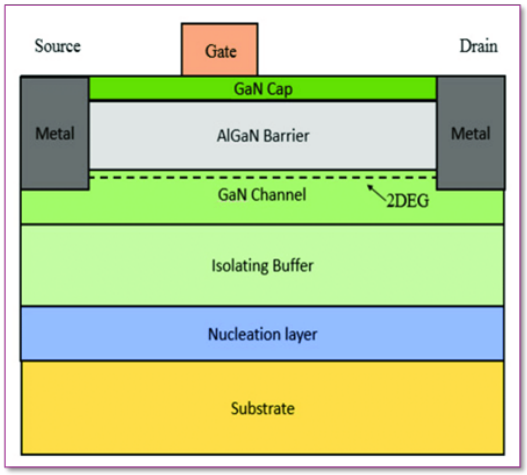


Figure 2.1: The basic structure of GaN HEMT

Figure 2.1 demonstrates the basic structure of the Gallium Nitride High Electron Mobility Transistor [17]. Metal gate contacts are for the controlling flow of the current. Other layers are for the 2DEG formation. The process of 2DEG formation can be divided into four stages:

1. **Band Bending:** Due to the difference in bandgaps between AlGaN and GaN, a natural energy band offset forms at their interface. The conduction band of AlGaN is higher than that of GaN. This causes a bending of the energy bands.

2. **Polarization Charges:** Both AlGaN and GaN are piezoelectric materials. That means the lattice mismatch at the interface induces a strong polarization field, creating a sheet of positive charges at the AlGaN/GaN interface.

3. **Electron Attraction** The positive polarization charges attract free electrons from the AlGaN barrier layer. These electrons get confined into a very thin triangular potential well at the AlGaN/GaN interface.

4. **2DEG Formation:** The high density of electrons confined in this thin sheet-like region is known as the two-dimensional electron gas (2DEG).

Electrons in the 2DEG experience very little scattering. This translates directly into exceptionally high electron mobility, which is the key to high-frequency operation. The 2DEG is spatially separated from the ionized donors in the AlGaN, further minimizing scattering and enhancing mobility. The 2DEG can achieve very high charge density, allowing for transistors with remarkable current-carrying capabilities. This is what leads to high power handling.

2.2 Transistor Modeling

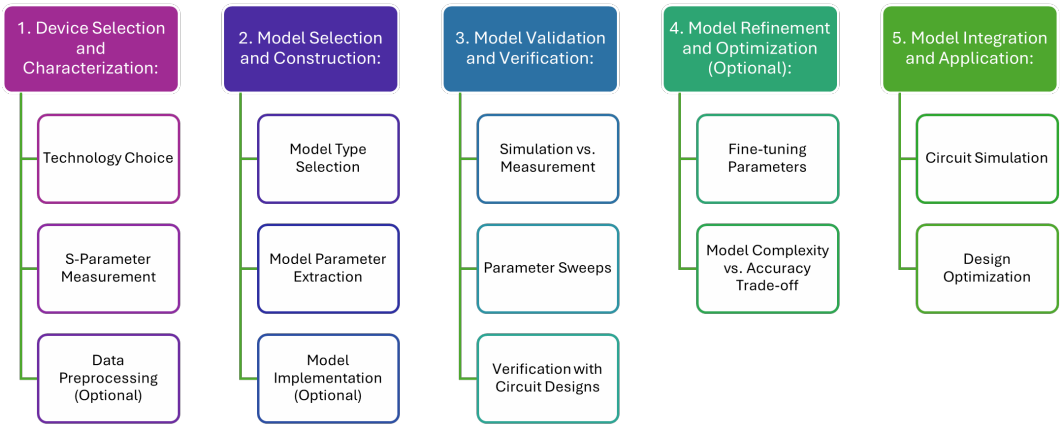


Figure 2.2: Transistor modeling stages

Transistor modeling is a complex process that consists of several stages as shown in Figure 2.2. Technology choice is about the selection of a suitable transistor technology like GaN HEMT or SiGe HBT based on application requirements. Scattering parameter measurement is usually done on-wafer using a Vector Network Analyzer (VNA). Data preprocessing suggests cleaning, formatting, and filtering S-parameter data. The next stage which is model type selection is based on the desired level of accuracy and complexity. Models can be empirical, physics-based, data-driven, and a combination of several. After choosing the right model, parameters should be extracted to fit the selected model. Then the model can be implemented which might involve coding the model equations in a simulation environment like ADS or using existing model libraries. After the model construction, its results are compared with the original S-parameter measurements for model accuracy assessment. After the parameter sweeps stage, the model is tested and integrated into simulated or actual microwave circuits. The model can be optimized by fine-tuning its parameters. It means iteratively adjusting model parameters within acceptable bounds until the accuracy level satisfies the criteria. Finally, the model can be utilized to simulate microwave circuits before their fabrication.

2.3 Data-Driven Models

Data-driven models use machine learning algorithms to analyze large datasets and learn relationships between input and output features. As the ML techniques advance, the accuracy and efficiency of data-driven models also increase compared

to traditional methods. For instance, deep neural networks (DNNs) and support vector machines (SVMs) effectively model the behavior of GaN HEMTs, capturing the nonlinearities common in semiconductor devices [25, 26].

These approaches can adapt to varying operational conditions, making them suitable for real-time applications in power electronics and RF circuits. By integrating machine learning with traditional modeling techniques, compact models can be developed for easy implementation in circuit simulators like SPICE. This is particularly useful for high-frequency applications where accurate modeling of parasitic capacitances and resistances is essential [27].

Recent studies show that data-driven models can outperform conventional physics-based models by accurately predicting device behavior under different conditions, such as temperature variations and biasing scenarios [28]. These models can be checked against experimental data to confirm their reliability in real-world applications.

Chapter 3

Methodology

3.1 Computational Environment and Device Under Test

Computational Environment and Software Tools

All the experiments in this study were done on a single workstation with these characteristics:

- 64-bit system on Windows 11 Enterprise, version 22H2.
- Intel® Core™ i7-12700F processor with 12 physical cores and 20 logical processors. The base clock speed is 2.10 GHz.
- 32 GB of random-access memory (RAM), where 31.8 GB are usable.
- High-speed solid-state drive (SSD) with 1 TB of memory.

The workflow, which consists of data analysis, data preprocessing, model development, model training and tuning, as well as results visualization, was carried out using Python as the sole programming language and Jupyter Notebook as the interactive computing environment. The main libraries used in the study are sci-kit learn, tensorflow keras, pandas and numpy.

Device Under Test (DUT)

The study was conducted on a GaN HEMT dataset. The data was obtained from the device under test (DUT), which has 10 fingers, each 50 µm long, resulting in a total gate periphery of 500 µm. Figure 2.1 illustrates the simplified epitaxial structure of the DUT. As stated earlier in the background section, a two-dimensional electron gas (2DEG) is generated due to polarization effects between the barrier and channel layers. The DUT's epitaxial structure consists of a 2 nm GaN cap

layer, a 20 nm AlGaN barrier layer, a 2 µm GaN buffer layer, a 1 nm AlN nucleation layer, and a 500 µm diamond substrate.

On-wafer measurements of the device were performed at seven different gate-to-source voltages (VGS) ranging from −3 V to 0 V in increments of 0.5 V, thirteen different drain-to-source voltages (VDS) ranging from 0 V to 30 V in increments of 2.5 V, and at 400 unique frequencies starting from 0.1 GHz to 40 GHz in increments of 0.1 GHz. These measurements were repeated at two distinct temperatures, 25 °C and 85 °C, resulting in a total of 72,800 data samples.

3.2 Data Preparation and Splitting

Data Preparation

There are four input and eight output features in the dataset. Input features are frequency, VDS, VGS, and temperature. Output features are eight S-parameters (S11, S12, S21, and S22) given in magnitude and angle values, which requires sine and cosine transformations due to cyclical nature. Due to non-linear relationships, it complicates the development of ML models. That's why eight S-parameters were converted from polar representations into real and imaginary components. Then, the converted dataset was checked for missing values and outliers. Figure 3.1 (a) and (b) show the distribution of frequency values across bias points for 25°C and 85°C, respectively. We can see that these plots are identical, meaning that the frequency sampling is consistent across both temperature conditions. There are 91 unique bias points (multiply 13 VDS values to 7 VDS values), each of which has 400 different frequencies, resulting in 36400 samples per temperature.

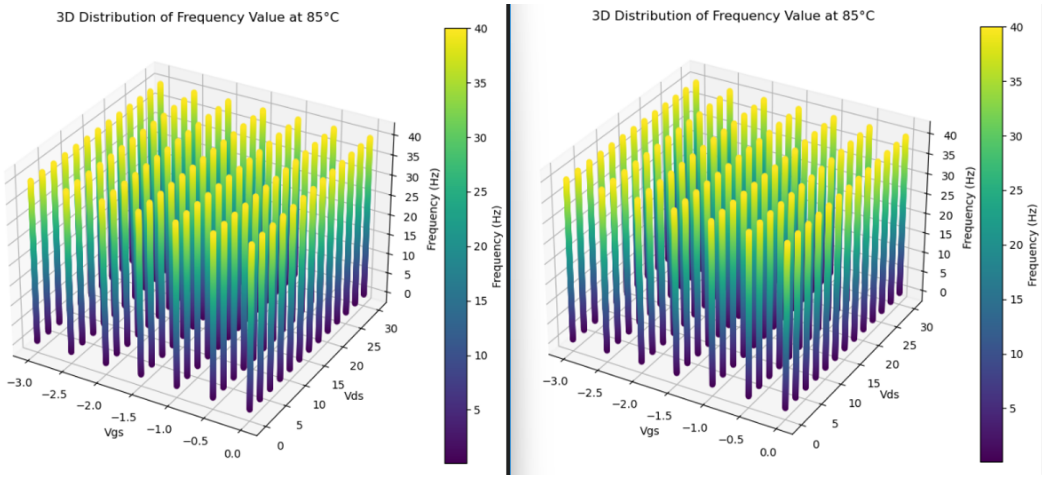


Figure 3.1: 3D distribution of input parameters for two different temperatures.

Dataset Split

To develop an ML model, the dataset should be split into training and testing sets, then the training set partitioned as a training subset and validation subset. In standard ML cases, the split is generally carried out using a ratio of 80/20 or 70/30, and data points are shuffled in order to achieve generality and avoid any kind of bias. In this research, however, the partitioning of data was carried out using VDS bias, not random shuffling. It is because splitting data by VDS bias preserves the physical relevance of the operating conditions during model development [29]. It is also stated that this approach prevents overfitting to mixed bias data and helps capture nonlinear device behavior with better accuracy [30, 31].

On-wafer measurements of the device were performed at seven different gate-to-source voltages (VGS) ranging from $-3\,\mathrm{V}$ to $0\,\mathrm{V}$ in increments of $0.5\,\mathrm{V}$, thirteen different drain-to-source voltages (VDS) ranging from $0\,\mathrm{V}$ to $30\,\mathrm{V}$ in increments of $2.5\,\mathrm{V}$, and at 400 unique frequencies starting from $0.1\,\mathrm{GHz}$ to $40\,\mathrm{GHz}$ in increments of $0.1\,\mathrm{GHz}$. These measurements were repeated at two distinct temperatures, $25\,^{\circ}\mathrm{C}$ and $85\,^{\circ}\mathrm{C}$, resulting in a total of 72 800 data samples.

Of the 13 distinct VDS values, 4 were assigned to the testing set and the other 9 to the training set. This resulted in a distribution of roughly a 70/30 ratio. For the test set, the highest VDS value of $30\,\mathrm{V}$ was initially selected to test the ability of the model to extrapolate beyond the training data. With one VDS value already reserved for extrapolation, three more had to be defined to create the rest of the test set. This resulted in 220 possible combinations of three VDS values from the remaining twelve. To efficiently identify a robust combination for the test set, a simple RF model (hyperparameters set to default) was trained and evaluated on each of these 220 combinations across eight S-parameters. The performance of each combination was assessed using the coefficient of determination (R^2). This revealed the combination of VDS values $\{17.5, 22.5, 25.0, 30.0\}\,\mathrm{V}$ to be repeatedly among the best-performing across all eight S-parameters.

Alternatively, the choice of test set VDS values was informed by a derivative-based analysis of the S-parameters versus VDS. This involved examining the rate of change—the first and, in some cases, the second derivatives—of the average trend of each S-parameter. The goal was to identify VDS regions where device behavior underwent the largest changes. Using threshold- and percentile-based criteria in derivative plots, "critical" points were found. This analysis showed that the largest changes in device behavior tended to occur below a VDS of 10 V, while intermediate voltages had more gradual changes. Figure 3.2 shows the derivative plot for Real S12 versus VDS values with identified critical points. We can see that 0.0 V, 2.5 V, 5 V, and 27.5 V are identified as critical points, at which significant shifts occur. Thus, it became evident why VDS values within the intermediate range (17.5 V, 22.5 V, and 25.0 V) yielded better model performance compared to combinations incorporating these lower critical voltage points. Finally, the test set

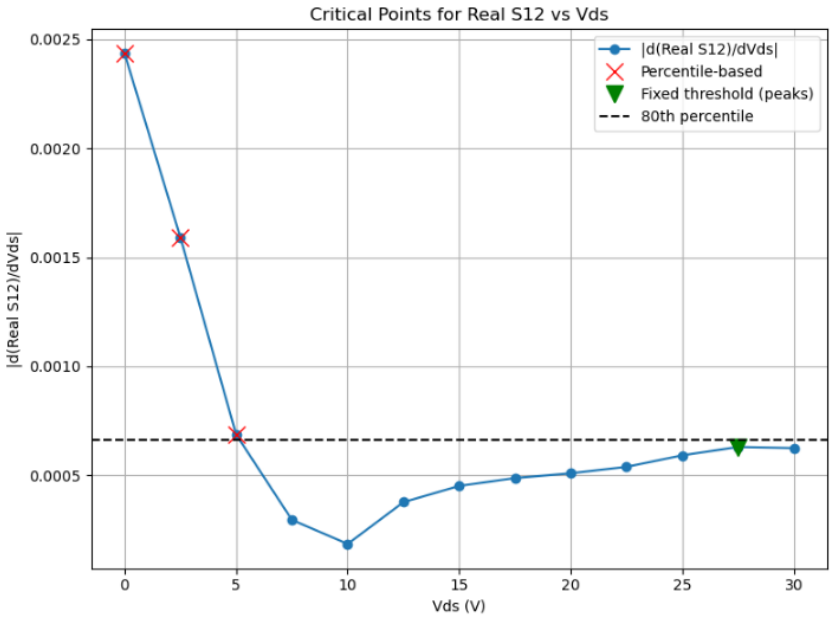


Figure 3.2: Derivative of Real S12 vs. Vds: Critical Point Identification

included {17.5 V, 22.5 V, 25.0 V, 30.0 V}, while the remaining values were selected for the training set. This data split is used in all models developed in this study.

Validation Methods

After dividing the dataset into training and testing sets, the models will be trained only on the training set, holding out the testing set. Subsequently, the hyperparameter tuning will be carried out using the training dataset, a crucial step to achieve the best possible performance of the models. For this reason, the training set is further divided into training and validation subsets. The former is used to fit the model with a given set of hyperparameters being evaluated, while the latter is used to assess the model's performance with those hyperparameters during the tuning process.

There are various methods validation methods. This study utilized two of them: a simple 80/20 split and 5-fold cross-validation. In the first case, training set partitioned by 80/20 ratio. In the second case, the training set was divided into five folds of the same size, wherein the model will be trained on four folds and validated using the fifth one, and this will be repeated five times, each time using a different fold for validation. The choice between these methods often involves a trade-off between computational cost and the stability of the performance estimate, with k-fold cross-validation generally providing a more reliable assessment. This work compares results from both methods to determine the effect of cross-

validation on model results.

Random Seed and Random State

In order to ensure the reproducibility of our findings and to enable fair comparison of models, a random seed and random state were both set to the same value, 42, across all implemented machine learning models. This approach guarantees that the stochastic processes involved in model building and testing—such as weight initialization and data shuffling—produce consistent results when the code is executed multiple times.

Evaluation Metrics

Model performances were evaluated using three metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and the Coefficient of Determination (R^2). MAE measures the average absolute difference between predicted and actual values, making it less sensitive to outliers [32]. It is defined as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|.$$

MSE calculates the average of the squared differences between actual and predicted values, heavily penalizing larger errors [33]. It is given by

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2.$$

The Coefficient of Determination quantifies how well the independent variables explain the variability of the dependent variable [32, 33]. Its formula is

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}.$$

In addition to these accuracy metrics, we recorded the time taken for hyperparameter tuning, training, and testing. To ensure a fair comparison across models, the `n_jobs` parameter was set to `-1` to utilize all available CPU cores. During grid searches, slight reductions in `n_jobs` were necessary to remain within memory limits, yet CPU utilization remained near 100%. All ANN models ran on CPU (GPU was unavailable); due to the iterative nature of neural network training, these did not fully saturate the CPU as tree-based models did. Therefore, runtime comparisons between ANN and tree-based models should be interpreted with these practical considerations in mind.

3.3 Model Development

Artificial Neural Networks

Artificial Neural Networks (ANNs) are a family of machine learning models inspired by the structure and organization of the human brain [34]. Fundamentally, ANNs are composed of interconnected layers of artificial neurons, which process and transmit information. A standard ANN design encompasses an input layer, one or more hidden layers, and an output layer. Each connection between nodes has a weight, and each node typically possesses a bias [34]. The nodes compute an activation function of their weighted sum of inputs to yield an output. Through training, ANNs learn sophisticated non-linear relationships in the data by updating these weights and biases to reduce the discrepancy between the predicted values from the model and actual values. Depth (hidden layer count) and width (number of nodes in each layer) are important architectural hyperparameters that impact the capacity and performance of the model, as is also the selection of activation functions. The structure and organization of a neural network is illustrated in Figure 3.3.

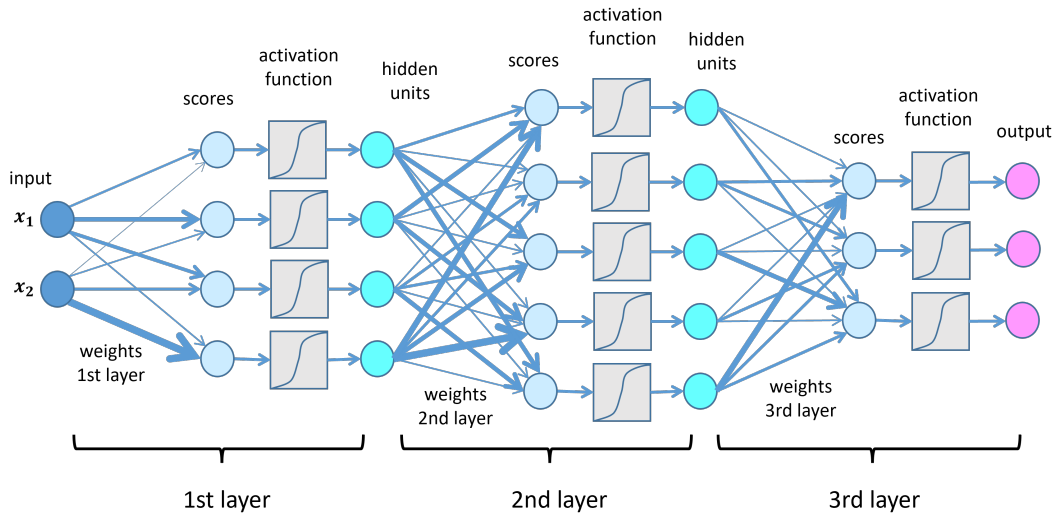


Figure 3.3: Deep neural network structure [34] © Lamarr Institute

In this study, an optimal ANN structure for each output target (S-parameter) was determined via Optuna, a library that performs Bayesian optimization with tree-structured Parzen estimator (TPE). Hyperparameters tuned by Optuna are the number of hidden layers, units per layers, activation functions, learning rate, batch size, and training epochs. The hyperparameter space is given below:

- Activation function: relu, tanh, leaky_relu, elu, swish, selu.

- Number of hidden layers: 1 to 3.
- Units per layer: 10 to 50 in steps of 10.
- Learning rate: from 1e-4 to 1e-2 on a log scale.
- Batch size: 16 to 64 in steps of 16.
- Epochs: 50 to 200 in steps of 50.
- Scaler: MinMaxScaler, MaxAbsScaler, StandardScaler.

Optuna iteratively evaluates candidate configurations and selects those producing minimal validation error. The chosen neural network configuration was then compiled with Adaptive Moment Estimation (ADAM). ADAM is the low-level training optimizer that updates the weights of the neural network in each epoch. It performs backpropagation and parameters updates during training.

Random Forests

Random Forests (RF) are ensembles of Decision Trees. Decision Tree (DT) is a tree-like model with if-then-else rules. DTs recursively split the dataset into smaller subsets based on feature values until a stopping criterion is met [35]. A typical structure of the decision tree is represented in Figure 3.4. A single tree cannot fully grasp the complex nature of transistor data. That's why RFs with many DTs are used with two techniques: bagging (bootstrap aggregating) and random feature selection. Bagging is carried out by training every tree upon a random subset of training data with replacement. Random feature selection means that when each tree is splitting a node, it considers only a random subset of the available features. This decorrelation among trees in the forest results in a reliable and accurate model that can capture complex non-linear relationships. For a regression task like ours, the prediction of the RF is the average of the predictions from the individual trees.

Extremely Randomized Forests

Extremely Randomized Forests or Extra Trees (ETs) model is a modified version of RFs. ETs differ in how the individual trees are constructed. While RFs bootstrap-sample from the data and take a random subset of features to split at every node, ETs take this one step further by also randomly choosing split points for every feature in addition to choosing a random subset of features. This extra randomization often leads to improved generalization in certain situations and results in faster computation than RFs. Figure 3.5 shows the simplified diagram of extra trees and random forests.

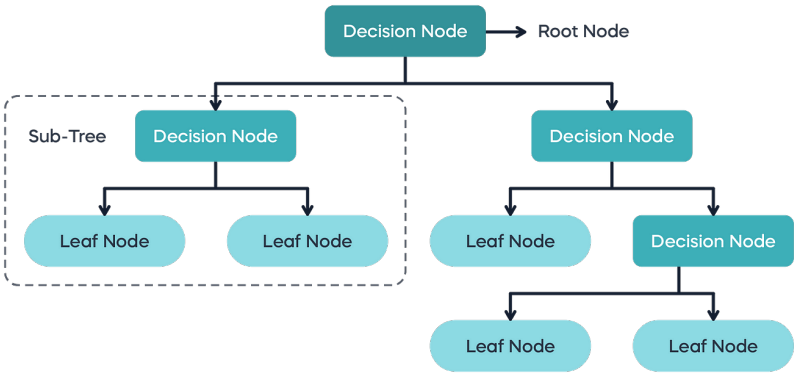


Figure 3.4: Decision tree structure [35]

3.4 Hyperparameter Optimization

Hyperparameter Space for Tree-Based Models

Tree-based models such as decision trees, random forests, and extra trees share several specific hyperparameters that determine their structure and influence the learning process. For example, "number of estimators" parameter controls the number of trees in the ensemble. "Maximum depth" parameter limits how deep each tree can grow. The detailed explanation of each hyperparameter can be found in this paper [36].

These configuration settings are not learned during training but are set before training begins. Therefore, tuning them requires background knowledge and a trial-and-error approach. Nevertheless, hyperparameters can be optimized using various techniques. First, we define a possible range, set of values, or categories for each parameter, thereby defining the hyperparameter space. Each hyperparameter represents a specific dimension in this space. Second, we explore this multidimensional space with selected algorithms to find the best combination of hyperparameters that results in the best model performance. These algorithms are commonly referred to as optimizers.

The hyperparameter space used in the study is as follows:

- n_estimators: 100 to 1000
- max_depth: 5 to 30
- min_samples_split: 2 to 20
- min_samples_leaf: 1 to 10

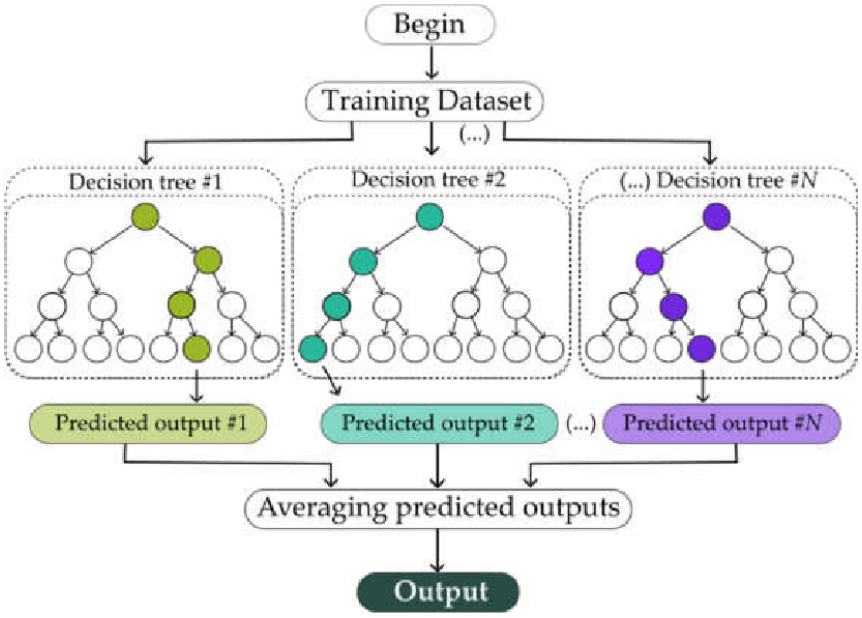


Figure 3.5: ET and RF structure [FigETrees]

- max_features: None, "sqrt", or "log2"
- bootstrap: True or False

The five different optimizers were implemented to fine-tune the tree-based models: random search, grid search, bayes search, optuna, and genetic algorithm.

Optimizers

The number of iterations for all optimizers, except Grid Search, was set to 100 based on a trial-and-error approach. Using more than 100 iterations did not produce observable improvements.

a. Random Search (RS) Random Search is the most straightforward type of hyperparameter optimization. It chooses parameter values at random within specified ranges or distributions [37]. After selecting a random configuration, the validation error (i.e., the error rate between actual and predicted values) is computed. This process is repeated 100 times, and the configuration that yields the best result is returned. Although simple, RS occasionally is capable of finding nearly optimal solutions in little time, particularly in high-dimensional spaces where exhaustive exploration becomes infeasible.

b. Grid Search (GS) Unlike other optimizers in this list, Grid Search works only with discrete values; hence it's hyperparameter space is adjusted to form a "grid" as below:

- n_estimators: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000.
- max_depth: 5, 10, 15, 20, 25, 30.
- min_samples_split: 2, 5, 10, 15, 20.
- min_samples_leaf: 1, 2, 5, 10.
- max_features: None, "sqrt", "log2".
- bootstrap: True or False.

Grid Search does not explore the hyperparameter space in the same iterative or probabilistic manner as other optimizers. Instead, it exhaustively evaluates all possible combinations defined in the grid, ensuring that every discrete configuration is tested. The total number of fits for an 80/20 validation split is calculated by multiplying the number of options for each parameter: 10 * 6 * 5 * 4 * 3 * 2 = 7,200. For 5-fold cross-validation, this number is multiplied by five, resulting in a total of 36,000 fits or iterations. This process is computationally expensive and time-consuming.

c. Bayes Search (BS) Bayesian Search is a Bayesian optimization technique that employs a Gaussian Process (GP) as a proxy model for approximating the objective function across hyperparameter space [38]. Rather than blindly drawing configurations, this optimizer iteratively chooses promising hyperparameter sets by balancing the trade-off between exploration (sampling in high-uncertainty regions) and exploitation (sampling from regions with good performance). At each iteration, the GP model is updated with new observations of the validation error, as well as having a utility function, which is maximized to determine the next candidate configuration. Whilst more computationally expensive to compute compared to Random Search, BS needs much fewer calls compared with an exhaustive Grid Search. It is especially useful in high-dimensional or complicated spaces.

d. Bayes Search with Tree-based Parzen Estimator (Optuna) Bayes Search with the Tree-based Parzen Estimator, as implemented in Optuna, is a Bayesian optimization method that relies on non-parametric density estimation rather than a Gaussian process. It builds separate density estimators for hyperparameter configurations that result in low (good) and high (bad) validation errors. It compares these distributions and selects new candidates that maximize the expected

improvement of the objective function. Optuna works particularly well in high-dimensional or complex search spaces since it models non-linear relationships in a flexible manner without specifying a parametric form. Like other Bayesian methods, it directs the search in an intelligent way by balancing exploration and exploitation, typically converging to nearly optimal hyperparameter configurations in fewer trials than Bayes Search with GP in certain cases.

e. Genetic Algorithm (GA) Genetic Algorithm is an evolutionary optimization technique based on natural selection concepts and genetic principles. It starts by producing an initial population of candidate hyperparameter settings, which are drawn at random from a predetermined search region. Each candidate is evaluated using a fitness function—in our case, based on validation performance such as reduced error—after which the algorithm applies genetic operators, including selection, crossover (recombination), and mutation, to evolve the population over successive generations [39].

In each generation, the best-performing candidates are more likely to be chosen as parents for the next generation, while random mutations add variety to explore more of the hyperparameter space. This tends to help the GA avoid local optima in which other optimizers might get stuck.

It is worth noting that the GA itself has various parameters to be configured based on a trial-and-error process. For instance, in our implementation, we have configured:

- Population Size: 20
- Crossover Rate: 0.7
- Mutation Rate: 0.2
- Elitism: 3 (the number of top-performing candidates preserved in each generation)
- Tournament Size: 5
- Max Fidelity: 1 (a parameter placeholder, as fidelity was not utilized in this study)

This is derived after extensive experimentation and tuning to produce high-performance results in exploring the intricate, multi-modal hyperparameter space. Finally, after a fixed number of generations, the Genetic Algorithm produces the hyperparameter setting yielding highest validation performance as an efficient solution even in constrained search landscapes.

Chapter 4

Results and Discussions

4.1 Effect of Validation Techniques

The objective of this section is to compare model performance attained by employing two different validation methods: 80/20 training-validation split and 5-fold cross-validation (CV = 5). As discussed earlier, these methods are important for both hyperparameter tuning and model evaluation. The 80/20 split provides a computationally inexpensive method to estimate model generalization over a held-out set of the data, whereas 5-fold cross-validation gives a more accurate estimate of performance by taking averages over several validation folds, hopefully, minimizing the variance of a single split. In the following subsections of this section, the performance of all the models under study, estimated by both the 80/20 split and 5-fold cross-validation, is shown to analyze the effect of these various validation methods on the ultimate model performances.

4.1.1 ANN with Optuna

Table 4.1: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
80/20 Training-Validation Split									
MAE	3.57E-03	3.25E-03	2.90E-02	2.59E-02	1.52E-03	1.36E-03	4.39E-03	4.35E-03	9.17E-03
MSE	3.38E-05	2.31E-05	3.50E-03	1.69E-03	4.96E-06	3.84E-06	4.76E-05	4.61E-05	6.69E-04
R^2	99.97%	99.99%	99.88%	99.93%	99.85%	99.87%	99.97%	99.94%	99.93%
5-Fold Cross-Validation									
MAE	8.35E-03	5.68E-03	1.90E-02	1.11E-02	2.67E-03	1.51E-03	8.97E-03	1.02E-02	8.44E-03
MSE	2.86E-04	1.22E-04	1.53E-03	3.91E-04	1.11E-05	4.17E-06	2.59E-04	2.84E-04	3.61E-04
R^2	99.77%	99.93%	99.95%	99.98%	99.66%	99.85%	99.84%	99.64%	99.83%

Table 4.2: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
80/20 Training-Validation Split									
MAE	3.85E-03	3.41E-03	3.29E-02	2.85E-02	1.47E-03	1.35E-03	4.73E-03	5.41E-03	1.02E-02
MSE	3.51E-05	2.12E-05	4.00E-03	2.05E-03	4.18E-06	3.41E-06	6.15E-05	6.88E-05	7.81E-04
R^2	99.97%	99.98%	99.90%	99.94%	99.76%	99.77%	99.97%	99.85%	99.89%
5-Fold Cross-Validation									
MAE	6.65E-03	6.60E-03	2.11E-02	1.47E-02	2.46E-03	1.45E-03	1.01E-02	1.08E-02	9.23E-03
MSE	1.85E-04	1.41E-04	1.66E-03	6.28E-04	9.48E-06	3.95E-06	1.83E-04	2.65E-04	3.84E-04
R^2	99.86%	99.92%	99.96%	99.98%	99.45%	99.74%	99.91%	99.40%	99.78%

Table 4.1 and Table 4.2 show that the 80/20 split methodology had better results than 5-fold cross-validation for the ANN model with Optuna. The 80/20 split had lower MAE and MSE, and higher R^2 values for the majority of S-parameters. Exceptions were the real and imaginary parts of the S12 parameter, in which CV = 5 had a marginally better performance. In addition, hyperparameter tuning using 5-fold cross-validation took a considerably longer time, 14643.622 seconds, as opposed to 2421.8875 seconds when using the 80/20 split, thereby making the 80/20 split about 6 times faster with regard to tuning time.

4.1.2 Extra Trees with Random Search

Table 4.3: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	3.35E-03	3.60E-03	6.48E-03	6.23E-03	4.86E-04	3.92E-04	1.49E-03	8.76E-04	2.86E-03
MSE	1.37E-04	5.37E-05	8.61E-04	6.87E-04	6.40E-07	3.91E-07	1.13E-05	5.95E-06	2.20E-04
R^2	99.89%	99.97%	99.97%	99.97%	99.98%	99.99%	99.99%	99.99%	99.97%
80/20 Training-Validation Split									
MAE	2.74E-03	1.54E-03	6.48E-03	6.23E-03	3.49E-04	3.37E-04	9.26E-04	8.76E-04	2.43E-03
MSE	5.43E-05	1.40E-05	8.61E-04	6.87E-04	2.88E-07	2.82E-07	5.09E-06	5.95E-06	2.03E-04
R^2	99.96%	99.99%	99.97%	99.97%	99.99%	99.99%	100.00%	99.99%	99.98%

The 80/20 split generally gave slightly better results than 5-fold cross-validation for this model in both training and testing. Importantly, the 80/20 split was much faster, taking only 45.42 seconds on average compared to 154.0175 seconds for 5-fold CV. This indicates that for this case, the 80/20 split was a more efficient validation method without sacrificing performance.

Table 4.4: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	3.66E-03	4.40E-03	1.37E-02	1.35E-02	1.23E-03	1.18E-03	6.58E-03	6.67E-03	6.37E-03
MSE	7.68E-05	4.71E-05	1.92E-03	1.46E-03	5.11E-06	4.51E-06	1.50E-04	1.44E-04	4.76E-04
R^2	99.94%	99.97%	99.95%	99.96%	99.70%	99.70%	99.92%	99.68%	99.85%
80/20 Training-Validation Split									
MAE	2.80E-03	2.53E-03	1.37E-02	1.35E-02	1.14E-03	1.15E-03	5.54E-03	6.67E-03	5.88E-03
MSE	4.14E-05	2.09E-05	1.92E-03	1.46E-03	4.96E-06	4.45E-06	1.36E-04	1.44E-04	4.66E-04
R^2	99.97%	99.99%	99.95%	99.96%	99.71%	99.70%	99.93%	99.68%	99.86%

4.1.3 Extra Trees with Grid Search

Table 4.5: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.07E-03	2.87E-03	1.23E-02	1.41E-02	2.57E-04	3.61E-04	2.53E-03	3.31E-04	4.23E-03
MSE	1.32E-05	3.45E-05	2.46E-03	1.57E-03	1.43E-07	3.60E-07	1.78E-05	8.09E-07	5.12E-04
R^2	99.99%	99.98%	99.91%	99.94%	100.00%	99.99%	99.99%	100.00%	99.97%
80/20 Training-Validation Split									
MAE	2.04E-03	8.77E-04	4.68E-03	3.78E-03	2.62E-04	2.49E-04	6.46E-04	6.24E-04	1.64E-03
MSE	2.76E-05	3.82E-06	4.44E-04	2.64E-04	1.46E-07	1.44E-07	2.59E-06	2.02E-06	9.30E-05
R^2	99.98%	100.00%	99.98%	99.99%	100.00%	99.99%	100.00%	100.00%	99.99%

Table 4.6: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.94E-03	3.60E-03	1.91E-02	2.12E-02	1.16E-03	1.23E-03	6.99E-03	6.83E-03	7.76E-03
MSE	2.04E-05	3.38E-05	3.78E-03	2.76E-03	5.02E-06	4.71E-06	1.56E-04	1.42E-04	8.63E-04
R^2	99.98%	99.98%	99.91%	99.92%	99.71%	99.69%	99.92%	99.68%	99.85%
80/20 Training-Validation Split									
MAE	2.73E-03	1.95E-03	1.14E-02	1.13E-02	1.07E-03	1.08E-03	5.40E-03	6.55E-03	5.19E-03
MSE	4.05E-05	1.48E-05	1.18E-03	9.08E-04	4.87E-06	4.35E-06	1.34E-04	1.41E-04	3.03E-04
R^2	99.97%	99.99%	99.97%	99.97%	99.72%	99.71%	99.93%	99.68%	99.87%

Based on table data, 80/20 training-validation split outperformed 5-fold cross-validation consistently for the model in both training and the test phase. 80/20 training-validation split had lower average MAE and MSE, but higher average R^2 in both the datasets. Additionally, the 80/20 split was also greatly faster, taking

an average of 2824.68 seconds compared to the very long 13831.28 seconds for the 5-fold cross-validation. However, the fact that the 80/20 split outperformed 5-fold cross-validation in the case of predictive performance for the given model suggests that the 80/20 split, apart from being more computationally faster, also had better predictive performance.

4.1.4 Extra Trees with Bayes Search

Table 4.7: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.18E-03	2.74E-03	6.71E-03	3.82E-03	2.76E-04	4.95E-04	1.96E-03	1.45E-03	2.33E-03
MSE	1.20E-05	2.34E-05	9.15E-04	2.56E-04	1.66E-07	6.75E-07	1.72E-05	1.74E-05	1.55E-04
R^2	99.99%	99.99%	99.97%	99.99%	99.99%	99.98%	99.99%	99.98%	99.98%
80/20 Training-Validation Split									
MAE	2.01E-03	8.72E-04	4.65E-03	4.21E-03	3.46E-04	2.48E-04	6.42E-04	5.62E-04	1.69E-03
MSE	2.76E-05	3.82E-06	4.44E-04	3.08E-04	2.82E-07	1.42E-07	2.60E-06	1.78E-06	9.85E-05
R^2	99.98%	100.00%	99.98%	99.99%	99.99%	100.00%	100.00%	100.00%	99.99%

Table 4.8: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.86E-03	3.25E-03	1.37E-02	1.13E-02	1.20E-03	1.26E-03	6.53E-03	7.11E-03	5.78E-03
MSE	1.78E-05	2.55E-05	1.99E-03	8.97E-04	5.14E-06	4.79E-06	1.53E-04	1.55E-04	4.06E-04
R^2	99.99%	99.99%	99.95%	99.97%	99.70%	99.68%	99.92%	99.65%	99.86%
80/20 Training-Validation Split									
MAE	2.81E-03	1.93E-03	1.13E-02	1.20E-02	1.13E-03	1.08E-03	5.34E-03	6.34E-03	5.24E-03
MSE	4.34E-05	1.47E-05	1.19E-03	9.39E-04	4.90E-06	4.35E-06	1.33E-04	1.40E-04	3.09E-04
R^2	99.97%	99.99%	99.97%	99.97%	99.71%	99.71%	99.93%	99.69%	99.87%

For the Extra Trees model with Bayes Search, the 80/20 Training-Validation Split often produced better performance compared to 5-fold cross-validation. Both training and testing results for the 80/20 split had lower average MAE and MSE, as well as a moderately higher average R^2, showing greater predictive accuracy. Additionally, the 80/20 split proved to be substantially more computationally faster, taking an average of 192.22 seconds compared to the much higher 392.34 seconds required by 5-fold cross-validation. Hence, for the given model, the 80/20 split, in addition to its better generalization performance, achieved so in a very shorter duration of time.

4.1.5 Extra Trees with Optuna

Table 4.9: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.39E-03	1.24E-03	6.12E-03	4.35E-03	3.48E-04	2.95E-04	8.88E-04	1.07E-03	1.96E-03
MSE	2.46E-05	7.62E-06	7.63E-04	2.24E-04	2.86E-07	2.22E-07	4.59E-06	6.12E-06	1.29E-04
R^2	99.98%	100.00%	99.97%	99.99%	99.99%	99.99%	100.00%	99.99%	99.99%
Training-Validation Split									
MAE	2.01E-03	8.72E-04	4.65E-03	4.21E-03	3.46E-04	2.48E-04	6.42E-04	5.62E-04	1.69E-03
MSE	2.76E-05	3.82E-06	4.44E-04	3.08E-04	2.82E-07	1.42E-07	2.60E-06	1.78E-06	9.85E-05
R^2	99.98%	100.00%	99.98%	99.99%	99.99%	100.00%	100.00%	100.00%	99.99%

Table 4.10: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	2.13E-03	2.21E-03	1.31E-02	1.25E-02	1.14E-03	1.10E-03	5.52E-03	6.62E-03	5.54E-03
MSE	2.57E-05	1.69E-05	1.66E-03	1.07E-03	4.93E-06	4.40E-06	1.35E-04	1.43E-04	3.82E-04
R^2	99.98%	99.99%	99.96%	99.97%	99.71%	99.71%	99.93%	99.68%	99.87%
80/20 Training-Validation Split									
MAE	2.81E-03	1.93E-03	1.13E-02	1.20E-02	1.13E-03	1.08E-03	5.34E-03	6.34E-03	5.24E-03
MSE	4.34E-05	1.47E-05	1.19E-03	9.39E-04	4.90E-06	4.35E-06	1.33E-04	1.40E-04	3.09E-04
R^2	99.97%	99.99%	99.97%	99.97%	99.71%	99.71%	99.93%	99.69%	99.87%

In the case of the Extra Trees model using Optuna, the 80/20 Training-Validation Split tended to be better performing compared to 5-fold cross-validation. Of the training set, the 80/20 split displayed an evident outperformance in average MAE (1.69E-03 compared to 1.96E-03) and average MSE (9.85E-05 compared to 1.29E-04), with average R^2 being nearly identical (99.99%). Equally so, in the test set, the 80/20 split displayed a minor outperformance in average MAE (5.24E-03 compared to 5.54E-03) and a noticeable one in average MSE (3.09E-04 compared to 3.82E-04) with similar average R^2 values (99.87%). In addition to that, the 80/20 Training-Validation Split took a substantially shorter time to run, taking around 192 seconds compared to the 268 seconds of 5-fold cross-validation, saving around 76 seconds of time.

4.1.6 Extra Trees with Genetic Algorithm

For GA based ET model, 5-fold cross-validation consistently outperformed the 80/20 split of the training data into training and validation sets. On the training

set, 5-fold CV produced a lower average MAE (1.54E-03 against 2.07E-03) and a lower average MSE (1.17E-04 against 1.94E-04), while both methods had the same average R^2 value (99.99%). Also in the test set, 5-fold CV had a lower average MAE (5.25E-03 against 5.65E-03) and a lower average MSE (3.61E-04 against 4.63E-04) but a very marginally higher average R^2 value (99.87% against 99.86%). This indicates that in case of this model and data, 5-fold cross-validation had a better generalization performance compared to the 80/20 split.

Table 4.11: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	7.53E-04	6.57E-04	6.47E-03	1.94E-03	4.46E-04	4.16E-04	9.20E-04	7.44E-04	1.54E-03
MSE	4.96E-06	2.06E-06	8.64E-04	5.54E-05	5.21E-07	4.58E-07	5.06E-06	3.00E-06	1.17E-04
R^2	100.00%	100.00%	99.97%	100.00%	99.98%	99.98%	100.00%	100.00%	99.99%
80/20 Training-Validation Split									
MAE	7.70E-04	6.57E-04	6.47E-03	6.19E-03	4.46E-04	4.37E-04	8.76E-04	7.41E-04	2.07E-03
MSE	5.15E-06	2.06E-06	8.64E-04	6.74E-04	5.21E-07	5.11E-07	4.49E-06	2.95E-06	1.94E-04
R^2	100.00%	100.00%	99.97%	99.97%	99.98%	99.98%	100.00%	100.00%	99.99%

Table 4.12: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.58E-03	1.79E-03	1.39E-02	1.02E-02	1.17E-03	1.19E-03	5.52E-03	6.65E-03	5.25E-03
MSE	1.34E-05	1.35E-05	1.93E-03	6.41E-04	5.00E-06	4.55E-06	1.36E-04	1.42E-04	3.61E-04
R^2	99.99%	99.99%	99.95%	99.98%	99.71%	99.70%	99.93%	99.68%	99.87%
80/20 Training-Validation Split									
MAE	1.59E-03	1.79E-03	1.39E-02	1.34E-02	1.17E-03	1.21E-03	5.55E-03	6.56E-03	5.65E-03
MSE	1.35E-05	1.35E-05	1.93E-03	1.46E-03	5.00E-06	4.60E-06	1.35E-04	1.41E-04	4.63E-04
R^2	99.99%	99.99%	99.95%	99.96%	99.71%	99.69%	99.93%	99.68%	99.86%

4.1.7 Random Forest with Random Search

According to the table data, the 80/20 training-validation split consistently outperformed 5-fold cross-validation for the RF Random Search model. In the training data, the 80/20 split had a large advantage in average MAE (2.45E-03 compared to 4.30E-03) and average MSE (2.43E-04 compared to 5.74E-04) and a marginally higher average R^2 (99.98% compared to 99.96%). In the test data, the 80/20 split had a marginally better average MAE (1.33E-02 compared to 1.42E-02) and a better average MSE (8.63E-04 compared to 1.27E-03) with a marginally higher average R^2 (99.52% compared to 99.50%).

Table 4.13: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	2.03E-03	2.55E-03	1.11E-02	1.39E-02	4.87E-04	4.71E-04	2.24E-03	1.63E-03	4.30E-03
MSE	4.26E-05	1.70E-05	2.81E-03	1.69E-03	8.60E-07	6.70E-07	2.05E-05	1.20E-05	5.74E-04
R^2	99.97%	99.99%	99.90%	99.93%	99.97%	99.98%	99.99%	99.99%	99.96%
80/20 Training-Validation Split									
MAE	1.19E-03	1.46E-03	7.38E-03	7.06E-03	2.73E-04	3.64E-04	1.04E-03	8.48E-04	2.45E-03
MSE	1.68E-05	1.07E-05	1.32E-03	5.88E-04	2.87E-07	4.31E-07	5.52E-06	3.47E-06	2.43E-04
R^2	99.99%	99.99%	99.95%	99.98%	99.99%	99.98%	100.00%	100.00%	99.98%

Table 4.14: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	4.36E-03	5.26E-03	2.79E-02	3.12E-02	3.28E-03	3.33E-03	1.70E-02	2.09E-02	1.42E-02
MSE	8.13E-05	6.66E-05	4.62E-03	4.32E-03	1.92E-05	1.68E-05	5.11E-04	5.20E-04	1.27E-03
R^2	99.94%	99.96%	99.89%	99.87%	98.88%	98.88%	99.74%	98.83%	99.50%
80/20 Training-Validation Split									
MAE	4.08E-03	5.07E-03	2.55E-02	2.72E-02	3.25E-03	3.33E-03	1.68E-02	2.09E-02	1.33E-02
MSE	6.44E-05	6.12E-05	3.11E-03	2.63E-03	1.85E-05	1.66E-05	4.89E-04	5.15E-04	8.63E-04
R^2	99.95%	99.97%	99.92%	99.92%	98.92%	98.90%	99.75%	98.84%	99.52%

4.1.8 Random Forest with Grid Search

Table 4.15: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	5.88E-04	6.82E-04	3.25E-03	3.69E-03	1.62E-04	1.54E-04	6.53E-04	4.77E-04	1.21E-03
MSE	3.38E-06	1.60E-06	2.42E-04	1.35E-04	8.37E-08	6.64E-08	1.85E-06	9.03E-07	4.81E-05
R^2	100.00%	100.00%	99.99%	99.99%	100.00%	100.00%	100.00%	100.00%	100.00%
80/20 Training-Validation Split									
MAE	7.15E-04	1.33E-03	4.21E-03	4.39E-03	2.64E-04	2.55E-04	6.39E-04	7.97E-04	1.58E-03
MSE	5.30E-06	6.47E-06	4.21E-04	2.24E-04	2.14E-07	1.95E-07	1.93E-06	2.97E-06	8.28E-05
R^2	100.00%	100.00%	99.99%	99.99%	99.99%	99.99%	100.00%	100.00%	99.99%

In case of the Grid Search optimized RF model, 5-fold CV performed a little better compared to the 80/20 split. On the training set, 5-fold CV had a lower average MAE (1.21E-03 vs 1.58E-03) and lower average MSE (4.81E-05 vs 8.28E-05) along with a higher average R^2 (100.00% vs 99.99%) by a margin. In the test set, the average MAE (1.27E-02 vs 1.28E-02) and average MSE (6.59E-04 vs 6.95E-

04) were a little better in the case of 5-fold CV, while the average R^2 (99.53%) of both approaches remained the same. Hence, in the case of the RF Grid Search model, 5-fold CV appears to possess a better generalization performance compared to the 80/20 split based. Notably, however, 5-fold CV took significantly longer, averaging approximately 36245 seconds compared to the much faster 80/20 split which averaged around 2440 seconds.

Table 4.16: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	3.92E-03	4.69E-03	2.34E-02	2.56E-02	3.25E-03	3.28E-03	1.67E-02	2.08E-02	1.27E-02
MSE	5.43E-05	5.16E-05	2.07E-03	2.07E-03	1.85E-05	1.62E-05	4.83E-04	5.10E-04	6.59E-04
R^2	99.96%	99.97%	99.95%	99.94%	98.92%	98.92%	99.76%	98.85%	99.53%
80/20 Training-Validation Split									
MAE	3.97E-03	4.95E-03	2.38E-02	2.60E-02	3.26E-03	3.30E-03	1.67E-02	2.08E-02	1.28E-02
MSE	5.55E-05	5.91E-05	2.25E-03	2.16E-03	1.85E-05	1.63E-05	4.85E-04	5.12E-04	6.95E-04
R^2	99.96%	99.97%	99.94%	99.94%	98.92%	98.91%	99.75%	98.85%	99.53%

4.1.9 Random Forest with Bayes Search

Table 4.17: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	5.88E-04	1.32E-03	3.23E-03	3.63E-03	2.03E-04	2.69E-04	6.61E-04	8.82E-04	1.35E-03
MSE	3.39E-06	6.38E-06	2.37E-04	1.28E-04	1.43E-07	2.21E-07	1.85E-06	3.34E-06	4.75E-05
R^2	100.00%	100.00%	99.99%	99.99%	100.00%	99.99%	100.00%	100.00%	100.00%
80/20 Training-Validation Split									
MAE	7.32E-04	1.35E-03	4.17E-03	4.35E-03	2.59E-04	2.42E-04	6.24E-04	7.91E-04	1.56E-03
MSE	5.26E-06	6.59E-06	4.17E-04	2.21E-04	2.13E-07	1.86E-07	1.85E-06	3.01E-06	8.19E-05
R^2	100.00%	100.00%	99.99%	99.99%	99.99%	99.99%	100.00%	100.00%	99.99%

In general, 5-fold CV had a marginally better performance relative to the 80/20 split in regards to predictive accuracy. In the training set, 5-fold CV had a marginally lower average MAE (1.35E-03 vs. 1.56E-03) and a considerably lower average MSE (4.75E-05 vs. 8.19E-05), and a marginally higher average R^2 (100.00% compared to 99.99%). In the test set, the average MAE (1.27E-02 compared to 1.28E-02) and average MSE (6.57E-04 compared to 6.93E-04) were marginally better in 5-fold CV, while both methods yielded the same average R^2 (99.53%). The 80/20 Training-Validation Split, however, was appreciably faster, taking an average of 261.32 seconds compared to the very much longer 713.04 seconds that 5-fold

Table 4.18: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	3.92E-03	4.95E-03	2.34E-02	2.56E-02	3.25E-03	3.29E-03	1.67E-02	2.08E-02	1.27E-02
MSE	5.43E-05	5.89E-05	2.06E-03	2.05E-03	1.85E-05	1.64E-05	4.83E-04	5.13E-04	6.57E-04
R^2	99.96%	99.97%	99.95%	99.94%	98.92%	98.91%	99.76%	98.85%	99.53%
80/20 Training-Validation Split									
MAE	3.97E-03	4.96E-03	2.38E-02	2.60E-02	3.26E-03	3.30E-03	1.67E-02	2.08E-02	1.28E-02
MSE	5.55E-05	5.93E-05	2.24E-03	2.16E-03	1.85E-05	1.63E-05	4.84E-04	5.13E-04	6.93E-04
R^2	99.96%	99.97%	99.94%	99.94%	98.92%	98.91%	99.75%	98.85%	99.53%

4.1.10 Random Forest with Optuna

Table 4.19: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	7.02E-04	8.70E-04	1.14E-02	7.17E-03	2.87E-04	2.44E-04	1.02E-03	5.13E-04	2.78E-03
MSE	5.19E-06	3.65E-06	1.42E-03	5.94E-04	2.33E-07	1.88E-07	5.33E-06	1.32E-06	2.54E-04
R^2	100.00%	100.00%	99.95%	99.98%	99.99%	99.99%	100.00%	100.00%	99.99%
80/20 Training-Validation Split									
MAE	9.68E-04	1.33E-03	6.40E-03	4.28E-03	2.65E-04	1.78E-04	1.04E-03	9.52E-04	1.93E-03
MSE	9.78E-06	6.46E-06	9.74E-04	2.17E-04	2.16E-07	1.01E-07	5.51E-06	3.97E-06	1.52E-04
R^2	99.99%	100.00%	99.97%	99.99%	99.99%	100.00%	100.00%	99.99%	99.99%

Table 4.20: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	3.97E-03	4.84E-03	2.69E-02	2.73E-02	3.26E-03	3.30E-03	1.68E-02	2.08E-02	1.34E-02
MSE	5.55E-05	5.39E-05	3.29E-03	2.66E-03	1.85E-05	1.63E-05	4.89E-04	5.12E-04	8.87E-04
R^2	99.96%	99.97%	99.92%	99.92%	98.92%	98.91%	99.75%	98.85%	99.53%
80/20 Training-Validation Split									
MAE	4.04E-03	4.95E-03	2.50E-02	2.60E-02	3.26E-03	3.28E-03	1.68E-02	2.09E-02	1.30E-02
MSE	6.15E-05	5.90E-05	2.90E-03	2.15E-03	1.85E-05	1.62E-05	4.89E-04	5.15E-04	7.76E-04
R^2	99.95%	99.97%	99.93%	99.94%	98.92%	98.92%	99.75%	98.84%	99.53%

For RF Optuna model, the 80/20 Training-Validation Split consistently outperformed 5-fold cross-validation. On the training set, the 80/20 split had a lower average MAE (1.93E-03 to 2.78E-03) as well as a lower average MSE (1.52E-04 to 2.54E-04), both measures showing an average R^2 value of 99.99%. Equally, in the

test set, the 80/20 split had a marginally lower average MAE (1.30E-02 to 1.34E-02) and a lower average MSE (7.76E-04 to 8.87E-04) yet the same average R^2 of 99.53%. Additionally, the 80/20 Training-Validation Split took a much shorter time, an average of 106.10 seconds, compared to the 523.42 seconds that 5-fold cross-validation took.

4.1.11 Random Forest with Genetic Algorithm

Table 4.21: Training Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	1.45E-03	1.89E-03	7.53E-03	3.54E-03	2.85E-04	3.36E-04	1.04E-03	1.05E-03	2.14E-03
MSE	2.23E-05	1.35E-05	1.33E-03	1.15E-04	2.59E-07	4.06E-07	5.55E-06	5.27E-06	1.87E-04
R^2	99.98%	99.99%	99.95%	100.00%	99.99%	99.99%	100.00%	99.99%	99.99%
80/20 Training-Validation Split									
MAE	1.45E-03	1.89E-03	8.77E-03	3.54E-03	2.63E-04	3.36E-04	1.04E-03	8.26E-04	2.26E-03
MSE	2.23E-05	1.35E-05	1.67E-03	1.15E-04	2.16E-07	4.06E-07	5.59E-06	3.11E-06	2.29E-04
R^2	99.98%	99.99%	99.94%	100.00%	99.99%	99.99%	100.00%	100.00%	99.99%

Table 4.22: Testing Set Results

Metric	S11		S12		S21		S22		Average
	Real	Imag	Real	Imag	Real	Imag	Real	Imag	
5-Fold Cross-Validation									
MAE	4.27E-03	5.28E-03	2.55E-02	2.53E-02	3.26E-03	3.30E-03	1.68E-02	2.09E-02	1.31E-02
MSE	7.19E-05	6.74E-05	3.09E-03	1.94E-03	1.85E-05	1.65E-05	4.89E-04	5.18E-04	7.76E-04
R^2	99.95%	99.96%	99.92%	99.94%	98.92%	98.90%	99.75%	98.84%	99.52%
80/20 Training-Validation Split									
MAE	4.27E-03	5.28E-03	2.65E-02	2.53E-02	3.26E-03	3.30E-03	1.68E-02	2.08E-02	1.32E-02
MSE	7.19E-05	6.74E-05	3.55E-03	1.94E-03	1.85E-05	1.65E-05	4.89E-04	5.13E-04	8.33E-04
R^2	99.95%	99.96%	99.91%	99.94%	98.92%	98.90%	99.75%	98.85%	99.52%

Based on table data, 5-fold CV had a better performance compared to the 80/20 split in the case of this model. 5-fold CV attained a lower average MAE (2.14E-03 vs 2.26E-03) and lower average MSE (1.87E-04 vs 2.29E-04) in the training set, whereas both had the same high average R^2 of 99.99%. Likewise, in the testing set, 5-fold CV had a marginally lower average MAE (1.31E-02 vs 1.32E-02) and lower average MSE (7.76E-04 vs 8.33E-04) while both had an average R^2 of 99.52%. This implies that in the case of this model and data, 5-fold cross-validation had marginally better generalization performance when compared to the 80/20 split using these average measures.

4.1.12 Performance and Time Comparison

Table 4.23: Time Comparison (in seconds)

Model	CV = 5			80/20 split		
	Tuning	Training	Testing	Tuning	Training	Testing
ET RS	154.02	0.59	0.06	45.42	0.97	0.09
ET Optuna	268.31	1.27	0.12	192.22	1.87	0.15
ET GA	310.22	0.98	0.17	168.33	0.99	0.10
ET BS	392.34	1.04	0.11	192.22	1.87	0.15
RF GA	394.91	1.80	0.11	81.37	1.77	0.11
RF RS	448.72	1.97	0.26	92.47	2.34	0.13
RF Optuna	523.42	1.87	0.11	106.10	1.48	0.09
RF BS	713.04	3.89	0.18	261.32	3.26	0.19
ET GS	13831.28	53.88	0.31	2824.68	1.20	0.10
ANN Optuna	14643.62	106.59	1.95	2421.89	70.17	2.20
RF GS	36245.20	30.05	1.11	6101.17	6.49	0.34

In this section, we tested the effect of using 5-fold cross-validation against an 80/20 training-validation split in a range of machine learning models. Analysis of performance indicated that the advantages of CV=5 in predictive accuracy were not uniform among all the models. Some models had minor improvements (RF with GS, RF with GA), while a few had no improvements at all (RF with Optuna, ET with Optuna, RF with GA), and in many cases, the 80/20 split yielded in a slightly better performance (ET with RS, RF with RS, ET with GS, ET with BS).

Unlike in tree-based models, a noticeable difference in R^2 was observed in the ANN model tuned with Optuna. The 5-fold CV test results (averaged across S-parameters) showed an R^2 of 99.78%, whereas the 80/20 split model achieved 99.89%. This tendency is non-intuitive in case of ANN models and might be due to inconsistencies in model development. The reason behind this discrepancy should be investigated further. However, for tree-based models trained with GaN HEMT data, the benefits of cross-validation are not apparent.

The computational expense of the validation strategies was also investigated by comparing the average training, tuning, and testing times of each model for both CV=5 and the 80/20 split. Table 4.23 presents the averaged time across S-parameters for all models. These results point to a large disparity in processing time. On average, for all the tested models, the 80/20 split was roughly 9 times faster than CV=5. The large disparity in computational performance suggests an important trade-off to be made between potential gains in model performance and the greater resource requirements of CV=5.

4.2 Model Performance Comparisons

4.2.1 Averaged Train-Test Results

From the previous section, it was clear that the 80/20 validation-test split is computationally efficient and, in many cases, as accurate as 5-fold cross-validation. For the ANN model, the former produced significantly better results for three S-parameters, while the latter performed better only at S12 real and imaginary targets. For the tree-based models, the performance gap was not as drastic. However, computational time was obviously far less for the 80/20 split method. Therefore, for further analysis, we consider only models with the 80/20 split.

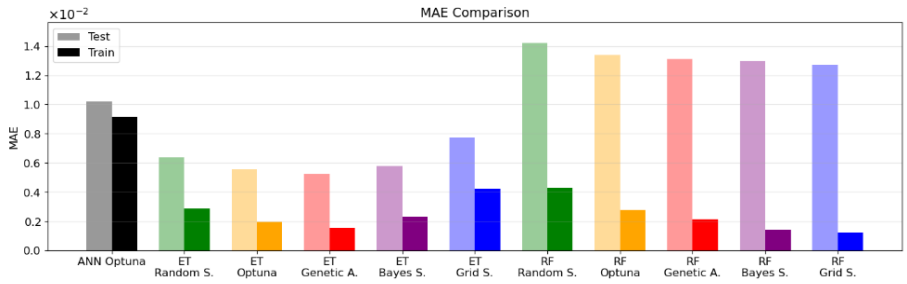


Figure 4.1: Comparison of test-train MAE values for all models

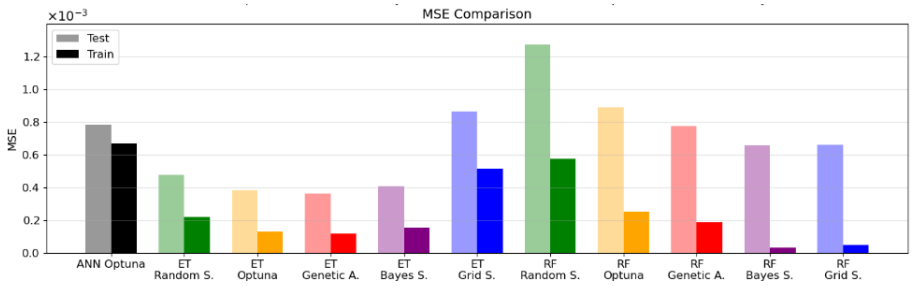


Figure 4.2: Comparison of test-train MSE values for all models

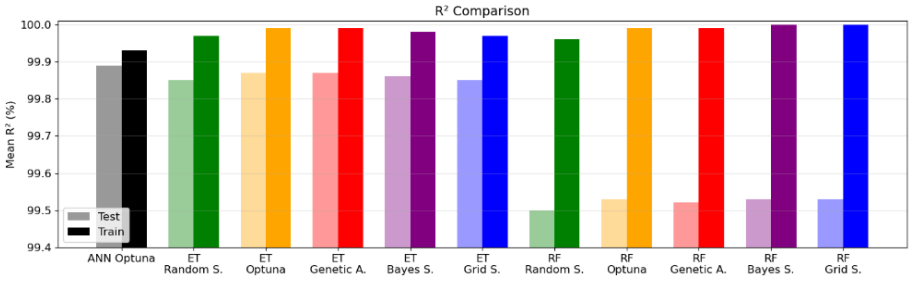


Figure 4.3: Comparison of test-train R2 values for all models

Figures 4.1, 4.2, and 4.3 show the MAE, MSE, and R^2 values for all S-parameters of each model evaluated. As can be seen from the paired bars for training and

test results, RF models tended to have higher overfitting, as shown by a larger difference between their training and test performance, as compared with other model types. This is also confirmed by their test metrics, where RF models had lower R^2 values, as well as significantly higher MSE and MAE from the test set compared with Extra Trees (ET) as well as with the ANN model. In turn, the ANN model looked least prone to overfitting, as it had the best test R^2. ET models optimized with GA, Bayes Search, and Optuna closely trailed the ANN in terms of R^2. In the MAE and MSE comparisons, all ET models, with the exception of the ET model optimized with Grid Search, surpassed the performance of the ANN. Within the RF family, only the models optimized by GA, Bayes Search, and Grid Search showed better MSE results compared to the ANN and the Grid Search-optimized ET models; however, no RF model managed to outperform the ET and ANN models in both MAE and R^2 metrics.

4.2.2 Model Comparison Across S-Parameters

For detailed analysis, the two optimizers (Genetic Algorithm and Bayes Search) with the best trade-off between time efficiency and accuracy for both Extra Trees (ET) and Random Forest (RF) regressors were chosen. Figures 4.4, 4.5, and 4.6 present the test metrics—MAE, MSE, and R^2—of each S-parameter for five models, including the ANN. The chosen optimizers, GA and Bayes Search, have very analogous patterns of performance for both RF as well as ET models, with very slight differences between them. However, specifically for the ET regressor, the Genetic Algorithm performs better than Bayes Search by a slight margin. Across this comparison, the RF models consistently deliver the poorest performance. The ANN model's performance is less consistent, varying depending on the specific S-parameter, at times surpassing the tree-based models and at other times performing worse than all of them.

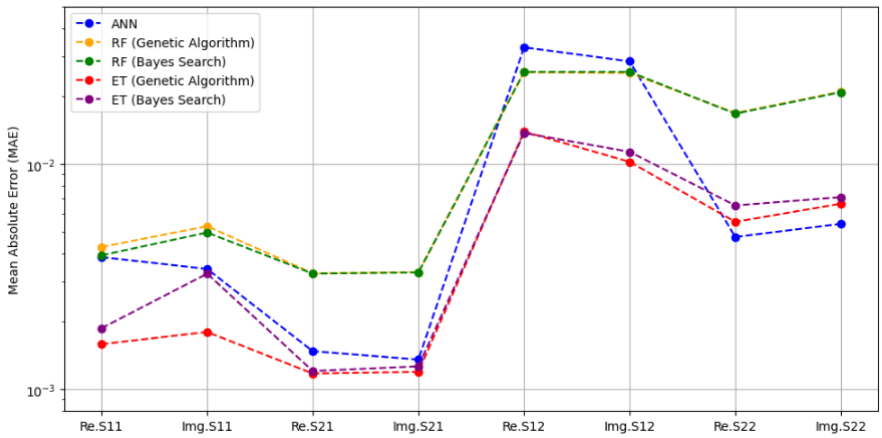


Figure 4.4: Comparison of test-train R2 values for all models

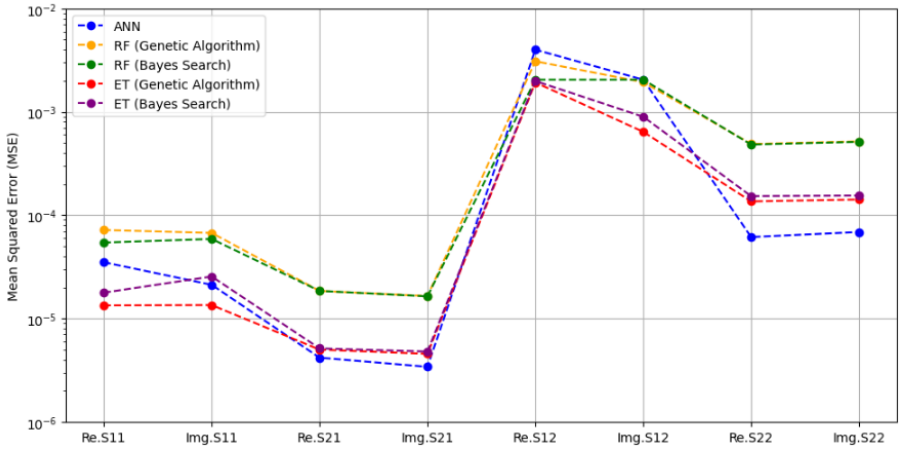


Figure 4.5: Comparison of test-train R2 values for all models

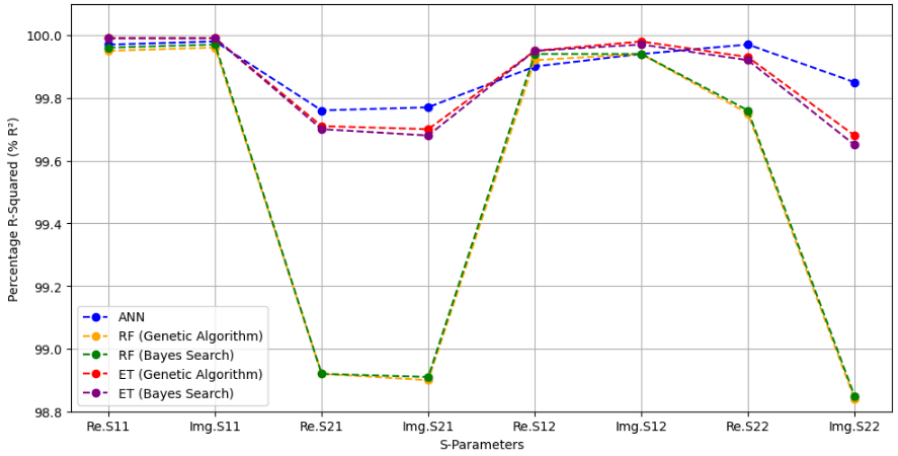


Figure 4.6: Comparison of test-train R2 values for all models

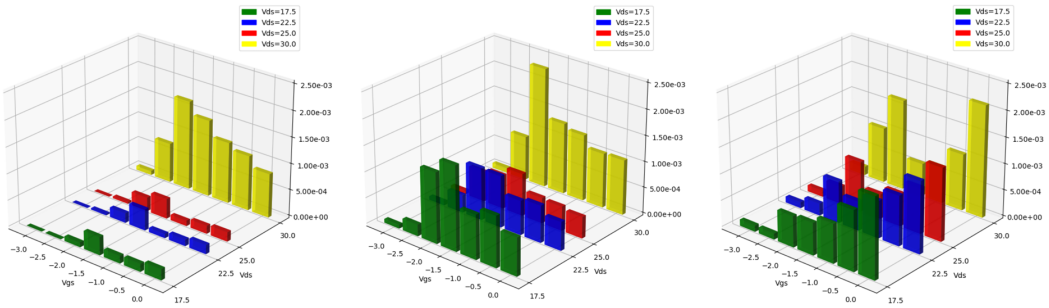
4.2.3 Performance Across Bias Conditions

Figure 4.7 presents three-dimensional visualizations of the mean squared error (MSE) across varying gate-to-source (V_{GS}) and test drain-to-source (V_{DS}) voltages. To simplify comparisons and conserve space, the MSE for each bias condition is averaged over all S-parameters. Figure 4.7(a) illustrates the performance of the GA-optimized Extra Trees (ET) model, Figure 4.7(b) shows the GA-optimized Random Forest (RF) model, and Figure 4.7(c) depicts the Optuna-optimized ANN model. Only one of the optimizers was chosen for the tree models because the initial comparisons revealed equivalent results from Genetic Algorithm and Bayes Search. In the above 3D plots, the higher the bar, the larger the difference between predicted and actual values, meaning poorer modeling precision, whereas lower bars represent higher precision.

In the interpolation range (at V_{DS} values of 17.5 V, 22.5 V, and 25.0 V), the ET GA

model performs considerably better in terms of predictive accuracy, with typical values of MSE between 10^{-7} and 10^{-5}. The MSE for the RF GA model is higher, generally between 10^{-5} and 10^{-4}. The ANN model's MSE, for the same region, is of the order of 10^{-4}, indicating that, occasionally, ET GA and RF GA are able to interpolate points of higher accuracy than the ANN. However, the ANN model is seen to have a relatively uniform distribution of MSE values for different biasing within the range of interpolation, not suffering from massive jumps in error.

When the extrapolation limit is reached at V_{DS} = 30.0 V, all three models exhibit a rise in MSE. The ET GA model, otherwise performing well within the interpolation domain, exhibits a rather dramatic increase in MSE, whereas the ANN model often outperforms and produces steadier results compared to ET GA for the boundary value of extrapolation. The RF GA model consistently yields the highest value of MSE under most cases considered under evaluation. In short, the ET GA model is best for the problem of interpolation, whereas the ANN model provides a balance of performance with higher robustness for the problem of extrapolation. The RF GA model, by and large, lags behind the other two models for accuracy for most of the prediction problems.

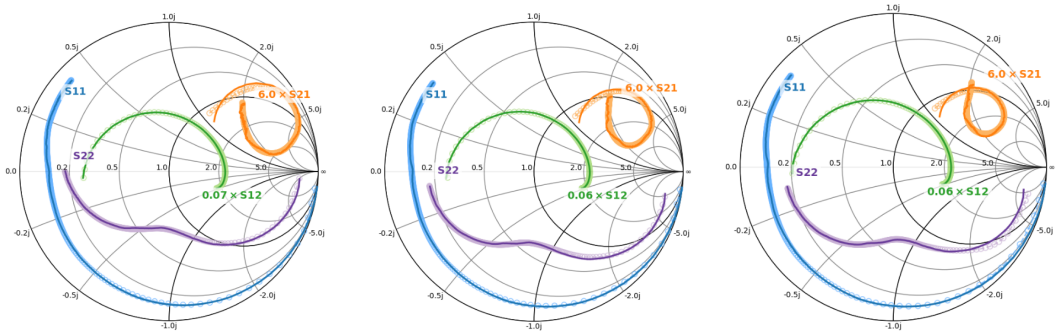


(a) GA-optimized ET (b) GA-optimized RF (c) Optuna-optimized ANN

Figure 4.7: 3D plots of MSE across VGS and VDS for selected models.

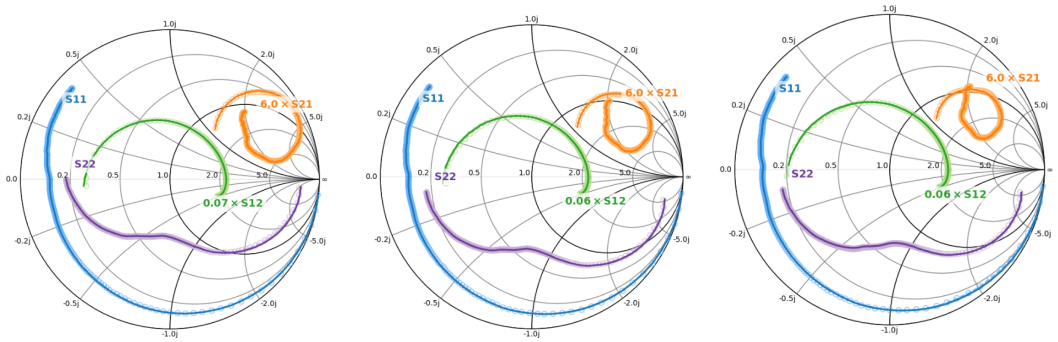
4.2.4 Smith Charts

These trends are seen consistently in Figures 4.8, 4.9, and 4.10, where Smith charts for different bias points are shown. The ANN as well as the ET models have similar levels of performance, reflecting very good agreement between their simulated S-parameters and measured S-parameters. Although the RF model is satisfactory, its accuracy is not as good as with the ANN and ET models.



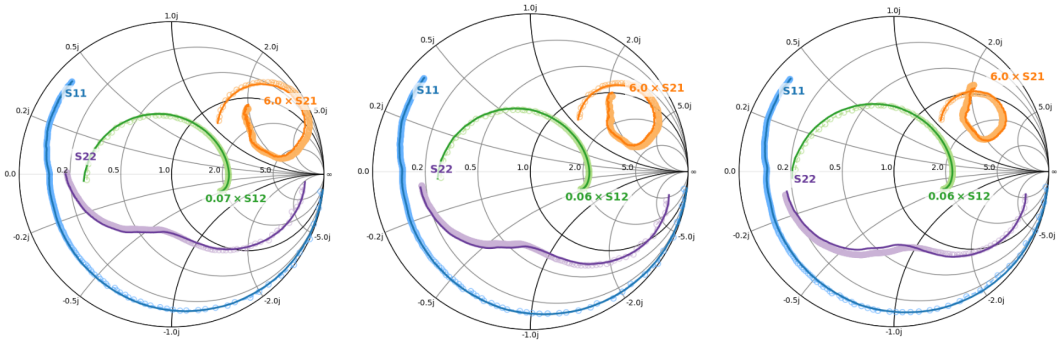
(a) V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C **(b)** V_{GS} = -2.0 V, V_{DS} = 17.5 V, T = 85°C **(c)** V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C

Figure 4.8: Measured (lines) and simulated (symbols) S parameters for GaN HEMT using ANN Optuna across frequencies 0.1–40 GHz.



(a) V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C **(b)** V_{GS} = -2.0 V, V_{DS} = 17.5 V, T = 85°C **(c)** V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C

Figure 4.9: Measured (lines) and simulated (symbols) S parameters for GaN HEMT using ET GA across frequencies 0.1–40 GHz.



(a) V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C **(b)** V_{GS} = -2.0 V, V_{DS} = 17.5 V, T = 85°C **(c)** V_{GS} = -1.5 V, V_{DS} = 25.0 V, T = 25°C

Figure 4.10: Measured (lines) and simulated (symbols) S parameters for GaN HEMT using RF GA across frequencies 0.1–40 GHz.

Chapter 5

Conclusion

Based on this study's findings, several key conclusions emerge regarding machine learning models for microwave transistor modeling, particularly concerning validation, optimization, and model comparisons.

Firstly, the 80/20 training-validation split outperformed 5-fold cross-validation (CV = 5) computationally, though frequently providing equivalent, or occasionally higher, predictive accuracy. Although there were inconsistent, occasionally marginal, gains for CV = 5, the 80/20 split was, as an average, around 9 times as fast. This trade-off for potentially small accuracy improvements over much higher computational expense is in favor of the 80/20 split.

Second, comparing optimizers for tree-based models yielded differing effectiveness and implementation simplicity. Extra randomness in Extra Trees models yields better results than random forests. For Extra Trees, GA, BS, and Optuna have very close results, with GA achieving slightly better results. We should keep in mind that GA (genetic algorithm) has its own parameters to be tuned; it will take extra time and a trial-and-error approach. Bayes Search and Optuna are easier to implement. They tend to overfit much, which is why after some regularization and pruning, they find an optimal bias-variance trade-off. Tuning times varied, with the 80/20 split consistently faster.

Thirdly, comparing tree-based models (ET GA, RF GA) with the ANN Optuna model showed distinct regional performance. When it comes to the ANN model, Optuna is utilized as a hyperparameter tuner, and the ADAM optimizer is used for internal weight updates. Its results are stable across different bias points, even in extrapolation regions, unlike tree-based models, which show higher extrapolation errors. For Extra Trees, the drop in performance in this edge region is the sharpest. Extra Trees performed better in interpolation than RF and ANN models but achieved similar results in extrapolation regions. Nevertheless, we cannot say that tree-based models with extra randomness are better than ANNs, as the ANN model in this study was not tuned and developed as thoroughly or with different

optimizers as tree-based models. However, tree-based models are definitely easier to develop and implement than neural networks due to their faster training time and more visually intuitive structure. The averaged time to tune the ANN model with an 80/20 validation-test split is 2421.89 seconds, as opposed to the tuning time of the GA-optimized Extra Trees model, with an average time of 168.33 seconds (approximately 14 times less). Despite taking the much less time, it beat the ANN model in interpolation.

In short, the 80/20 split provides provides a better trade-off between cost and performance compared to CV = 5. Among tree models, Extra Trees generally perform better. While ET GA excels in interpolation and ANN Optuna in extrapolation robustness, tree-based models are notably simpler and faster to implement.

Bibliography

[1] Jianjun Gao. *RF and microwave modeling and measurement techniques for compound field effect transistors*. Jan. 2010, pp. 1–340. ISBN: 9781891121890. DOI: 10.1049/SBEW027E.

[2] J.J. Liou and Frank Schwierz. "Evolution and recent advances in RF/microwave transistors". In: *Journal of Telecommunications and Information Technology* (Mar. 2004). DOI: 10.26636/jtit.2004.1.224.

[3] Wenrui Hu and Yong-Xin Guo. "An Evolutionary Multilayer Perceptron-Based Large-Signal Model of GaN HEMTs Including Self-Heating and Trapping Effects". In: *IEEE Transactions on Microwave Theory and Techniques* PP (Dec. 2021), pp. 1–1. DOI: 10.1109/TMTT.2021.3132892.

[4] Ting Leng et al. "Non-Volatile RF Reconfigurable Antenna On Flexible Substrate for Wireless IoT Applications". In: *IEEE Access* PP (Aug. 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3107486.

[5] Bob Frankston. "Consumer Technology vs. 5G". In: *IEEE Consumer Electronics Magazine* PP (Nov. 2020), pp. 1–1. DOI: 10.1109/MCE.2020.3037418.

[6] Wong Wai Kiat et al. "Compact Wearable Antenna for Millimeter-Wave (mm-Wave) Fifth Generation". In: *Proceedings of International Conference on Artificial Life and Robotics* 28 (Feb. 2023), pp. 615–620. DOI: 10.5954/ICAROB.2023.OS25-5.

[7] Shuman Mao et al. "Physics-based compact models of GaN HEMTs for high power RF applications: A review (Invited Paper)". In: *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* (2024). URL: https://api.semanticscholar.org/CorpusID:271911137.

[8] Antonio Raffo et al. "A New Modeling Technique for Microwave Multicell Transistors Based on EM Simulations". In: *IEEE Transactions on Microwave Theory and Techniques* 68 (2020), pp. 3100–3110. URL: https://api.semanticscholar.org/CorpusID:213909309.

[9] Cheng Ai-Qiang et al. "A large signal scaling model of high power GaN microwave device". In: *Acta Physica Sinica* (2023). URL: https://api.semanticscholar.org/CorpusID:258926887.

[10] Junjun Qi et al. "Small-signal modeling of microwave transistors using radial basis function artificial neural network-comparison of different methods for spread constant determined". In: *International Journal of RF and Microwave Computer-Aided Engineering* 32 (Mar. 2022). DOI: 10.1002/mmce.23145.

[11] Ahmad Khusro et al. "A new and Reliable Decision Tree Based Small-Signal Behavioral Modeling of GaN HEMT". In: Aug. 2019, pp. 303–306. DOI: 10.1109/MWSCAS.2019.8885334.

[12] Saddam Husain, Galymzhan Nauryzbayev, and Mohammad Hashmi. "GA Assisted ANN based GaN HEMT Model Development and Demonstration of its CAD Incorporation for Class-F Power Amplifier". In: *2023 7th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*. 2023, pp. 1–3. DOI: 10.1109/EDTM55494.2023.10102998.

[13] Nurullah Calik et al. "Deep-learning-based precise characterization of microwave transistors using fully-automated regression surrogates". In: *Scientific Reports* 13 (Jan. 2023), p. 1445. DOI: 10.1038/s41598-023-28639-4.

[14] Gaurav Bhargava et al. "Auto-encoder based hybrid machine learning model for microwave scaled GaAs pHEMT devices". In: *International Journal of RF and Microwave Computer-Aided Engineering* 32 (Aug. 2022). DOI: 10.1002/mmce.23339.

[15] Neda Ahmad and Vandana Nath. "Evaluating Nine Machine Learning Algorithms for GaN HEMT Small Signal Behavioral Modeling through K-fold Cross-Validation". In: *Engineering, Technology & Applied Science Research* (2024). URL: https://api.semanticscholar.org/CorpusID:271697275.

[16] Jialin Cai et al. "Machine learning technique based extremely broadband small-signal behavioral model for InP DHBTs". In: *International Journal of RF and Microwave Computer-Aided Engineering* 30 (Apr. 2020). DOI: 10.1002/mmce.22235.

[17] Lung-Hsing Hsu et al. "Development of GaN HEMTs Fabricated on Silicon, Silicon-on-Insulator, and Engineered Substrates and the Heterogeneous Integration". In: *Micromachines* 12.10 (2021). ISSN: 2072-666X. DOI: 10.3390/mi12101159. URL: https://www.mdpi.com/2072-666X/12/10/1159.

[18] Lin Cheng et al. "An Aging Small-signal Modeling Method of Microwave Transistors Using GA-ELM Neural Network". In: *2023 International Symposium of Electronics Design Automation (ISEDA)* (2023), pp. 385–389. URL: https://api.semanticscholar.org/CorpusID:261127803.

[19] Saddam Husain et al. “Comprehensive Investigation of ANN Algorithms Implemented in MATLAB, Python, and R for Small-Signal Behavioral Modeling of GaN HEMTs”. In: *IEEE Journal of the Electron Devices Society* 11 (2023), pp. 559–572. DOI: 10.1109/JEDS.2023.3324084.

[20] Filiz Güneş et al. “Cost-effective GRNN-based modeling of microwave transistors with a reduced number of measurements”. In: *International Journal of Numerical Modelling: Electronic Networks* 30 (2017). URL: https://api.semanticscholar.org/CorpusID:60459203.

[21] Qi jun Zhang et al. “Simulation and Automated Modeling of Microwave Circuits: State-of-the-Art and Emerging Trends”. In: *IEEE Journal of Microwaves* 1 (2021), pp. 494–507. URL: https://api.semanticscholar.org/CorpusID:231617073.

[22] Spyridon Pavlidis, Greg Medwig, and Michael Thomas. “Ultrawide-Bandgap Semiconductors for High-Frequency Devices”. In: *IEEE Microwave Magazine* 25 (2024), pp. 68–79. URL: https://api.semanticscholar.org/CorpusID:272705948.

[23] Andrei Grebennikov. “A high-efficiency transmission-line GaN HEMT Class E power amplifier”. In: *High Frequency Electronics* 8 (Dec. 2009). Figure 3: Test board of Class E GaN HEMT power amplifier. [Online]. Available: https://www.researchgate.net/figure/Test-board-of-Class-E-GaN-HEMT-power-amplifier_fig3_264237010, pp. 16–24.

[24] Frank Schwierz. *Microwave transistors—the last 20 years (invited)*. 2000.

[25] S. Ghosh et al. *GaN HEMT Modeling for Power and RF Applications using ASM-HEMT*. Available at: https://www.researchgate.net/publication/320652666_GaN_HEMT_modeling_for_power_and_RF_applications_using_ASM-HEMT. 2021.

[26] Y. S. Chauhan and et al. *GaN Transistor Modeling for RF and Power Electronics*. Elsevier, 2020.

[27] AMCAD Engineering Team. *GaN HEMT Transistor Compact Modeling based on Pre-RF Pulse I-V Measurements*. Available at: https://www.mos-ak.org/silicon_valley_2023/presentations/Li_MOS-AK_Silicon_Valley_2023.pdf. 2024.

[28] “GaN-on-Si Technology for High-power Transistors”. In: *NTT Review* (2014). URL: https://www.ntt-review.net/archive/ntttechnical.php?contents=ntr201405fa4.html.

[29] Francisco Pasadas Cantos. "Modelling of field-effect transistors based on 2D materials targeting high-frequency applications". PhD thesis. PhD thesis. Bellaterra, Spain: Universitat Autònoma de Barcelona, Departament d'Enginyeria Electrònica, May 2017, p. 164. ISBN: 9788449071287. URL: https://hdl.handle.net/10803/405314.

[30] Juan Pablo Duarte. *Mathematical Compact Models of Advanced Transistors for Numerical Simulation and Hardware Design*. Technical Report UCB/EECS-2018-24. EECS Department, University of California, Berkeley, May 2018. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-24.pdf.

[31] Yogesh Singh Chauhan. "Compact modeling of high voltage MOSFETs". PhD thesis no. 3915. PhD thesis. Lausanne, Switzerland: École Polytechnique Fédérale de Lausanne, Dec. 2007. DOI: 10.5075/epfl-thesis-3915. URL: https://infoscience.epfl.ch/record/109934/files/EPFL_TH3915.pdf.

[32] Vital Flux. "MSE vs RMSE vs MAE vs MAPE vs R-Squared: When to Use?" In: (2024). URL: https://vitalflux.com/mse-vs-rmse-vs-mae-vs-mape-vs-r-squared-when-to-use/.

[33] Plain English AI. "Understanding Common Regression Evaluation Metrics: MAE, MSE, RMSE, R2, and Adjusted R2". In: (2024). URL: https://ai.plainenglish.io/understanding-common-regression-evaluation-metrics-mae-mse-rmse-r2-and-adjusted-r2-6c5709e614c4?gi=b8c982ea5886.

[34] Gerhard Paaß. *Deep Learning: How do deep neural networks work?* Lamarr Institute for Machine Learning and Artificial Intelligence Blog. Blog post, 21 April 2021. Apr. 2021. URL: https://lamarr-institute.org/blog/deep-neural-networks/.

[35] The 365 Team. *Introduction to Decision Trees: Why Should You Use Them?* https://365datascience.com/tutorials/machine-learning-tutorials/decision-trees/. Blog post, 15 May 2024. May 2024.

[36] Rafael Gomes Mantovani et al. "Better Trees: An Empirical Study on Hyperparameter Tuning of Classification Decision Tree Induction Algorithms". In: *arXiv preprint arXiv:1812.02207* (2018). URL: https://arxiv.org/abs/1812.02207.

[37] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.10 (2012), pp. 281–305. URL: http://jmlr.org/papers/v13/bergstra12a.html.

[38] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012, pp. 2951–2959. URL: https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.

[39] Amine Sallah et al. "Machine learning for detecting fake accounts and genetic algorithm-based feature selection". In: *Data & Policy* 6 (2024), e15. DOI: 10.1017/dap.2023.46. URL: https://doi.org/10.1017/dap.2023.46.