Small-Signal Modelling of GaN HEMT using ANN, SVR, GPR and Tree-Based Supervised Models

Abishkozha Amangeldin, BTech in Electrical and Computer Engineering

Submitted in fulfilment of the requirements
for the degree of Master of Science
in Electrical and Computer Engineering

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

53 Kabanbay Batyr Avenue,
Nur-Sultan, Kazakhstan, 010000

Supervisors: Dr. Mohammad Hashmi and Dr. Galymzhan Nauryzbayev

April 2022

DECLARATION

I hereby, declare that this manuscript, entitled "*Small-Signal Modelling of GaN HEMT using ANN, SVR, GPR and Tree-Based Supervised Models*", is the results of my own work except for quotations and citations which have been duly acknowledged.
I also declare that, to the best of my knowledge and belief, it has not been previously or concurrently submitted, in whole or in part, for any other degree or diploma at Nazarbayev University or any other national or international institution.

(signature of the author)

Abishkozha Amangeldin
April 2022

Abstract

In this work, our objective is to devise an effective and an accurate small-signal model to elucidate the behavior of Gallium Nitride (GaN) High Electron Mobility Transistor (HEMT). At first, Support Vector Regression (SVR), Artificial Neural Network (ANN), Gaussian Process Regression (GPR) and Tree-based algorithms such as Decision Tree (DT), AdaBoost, Random Forests (RFs) are developed to exploit the bias, frequency, and temperature dependence. Thereafter, a reasonable comparison is carried out to find out the model that optimally describes the behavior of 10x200 μm GaN HEMT grown on Silicon. Furthermore, three different libraries are utilized: Keras, Scikit-learn and PyTorch to develop the ANN based models. All the developed models are evaluated in terms of mean squared error (MSE), mean absolute error (MAE) and coefficient of determination. It is observed that ANN based models guarantees the optimal prediction. Of the three models of ANN, Keras library based is the more efficient. Tree-based models are also shown to be a great alternative for the tabular based problems. GPR and SVR are the most expensive algorithms in terms of computational efficiency.

Acknowledgements

In the name of Allah, the Beneficent, the Merciful

Firstly, I would like to express my uttermost gratitude towards my supervisors **Dr. Mohammad Hashmi and Dr. Galymzhan Nauryzbayev**. It has been their supervision and direction throughout the duration of my studies which has allowed me to successfully complete this master's program. I am appreciative for all the hours of discussion they have offered me, especially in the areas of communications and embedded systems.

Secondly, I would like to show my indebtedness to the admission staff at Nazarbayev University whose efforts a lot of time go unnoticed.

Thirdly, I would like to express my profound gratitude to the PhD student, **Saddam Husain**, Department of Electrical Engineering, Nazarbayev University, for his constant support during the entire tenure of my studies.

Last but not the least, I would like to thank my family especially my wife and my friends for their constant support and encouragement throughout the program.

Table of Contents

List of Abbreviations & Symbols

1. PA	Power Amplifier
2. GaN HEMTs	Gallium Nitride High Electron Mobility Transistors
3. SSM	Small-Signal Model
4. LSM	Large-Signal Model
5. EPC	Efficient Power Conversion
6. International Rectifier	IR
7. FOM	Figure of Merit
8. LIDAR	Light Detection and Ranging
9. ML	Machine Learning
10. AI	Artificial Intelligence
11. CAD	Computer-Aided Design
12. ANN	Artificial Neural Network
13. SVR	Support Vector Regression
14. DT	Decision Tree
15. GPR	Gaussian Process Regression
16. RFs	Random Forests
17. SVM	Support Vector Machines
18. RVFFNN	Real-Valued Feedforward Neural Network
19. SMO	Sequential Minimal Optimization
20. SMA	Sequential Minimization Algorithm
21. MLP	Multilayer Perceptron
22. MSE	Mean Square Error
23. MRE	Mean Relative Error
24. NARX	Nonlinear Autoregressive with Exogenous Inputs
25. LM	Levenburg-Marquardt
26. BR	Bayesian Regularization
27. SCG	Steepest Conjugate Gradient
28. GA	Genetic Algorithm
29. V_{DS}	Drain, Source Voltage
30. V_{GS}	Gate, Source Voltage
31. T	Temperature
32. F	Frequency
33. W	Weight

34. ReLu	Rectified Linear Unit
35. GP	Gaussian Processes
36. CART	Classification and Regression Tree
37. MAE	Mean Absolute Error
38. R^2	Coefficient of determination
39. HO	Hyperparameter-Optimization
40. 2DEG	Two-Dimensional Electron Gas

List of Tables

List of Figures

Chapter 1- Introduction

Recently, there has been prolific interest in high-speed communication circuits for IoT, satellite, and 5G applications [1]-[3]. Due to this, the devices which could be operated over high frequencies and high temperatures are skyrocketing in demand [4]-[6]. GaN accommodates all the necessary prerequisites such as high electron saturation velocity, electron mobility, breakdown voltage and operating temperature for the effective design of power amplifiers (PAs) [4]. Furthermore, GaN HEMTs being a wide-bandgap semiconductor can be used for millimeter wave frequency applications. Several other GaN based devices are proposed in the literature such as GaN Arsenide. However, due to the strong polar nature of Nitride augmented with heterojunction formations of this device makes it the optimal choice [7].

The understanding of the behavior of the device is very crucial. GaN HEMT has a strong linear, non-linear, dynamic and electrothermal behavior. On top of this the behavior of the device can get corrupted because of self-heating and trapping effects induced errors. Researcher have tried different methods such as parameter extraction, curve-fitting to extrapolate the linear and non-linear behavior of the device. But this can be quite complex and time-consuming because of complicated dynamic behavior of the device. Furthermore, small-signal modelling works as a foundational basis of the large-signal modelling [8].

With the advent of learning-based approaches, the description of the device could be analyzed easily. The ML based methods can depict of the behavior of the device without need to worry about the deep physics and non-linear behavior of the devices. This facilitates the easy implementation of the device in circuit design tools and the PAs can be produced on mass-scale easily. Recently various researcher proposed ML based solutions to model the small-signal behavior of GaN HEMT [9]-[17].

1.1. Aims & Objectives

Learning based approaches have huge potential to model these types of devices. In this context, this study focuses on the most used ML algorithms. This work explores, investigates, and develops Support Vector Regression (SVR), Artificial Neural network (ANN), Gaussian Process Regression (GPR) and Tree-based algorithms such as Decision Tree (DT), AdaBoost and Random Forests (RFs) based small-signal models to exploit the biasing, frequencies and temperature-based dependence on s-parameters based outputs. A through comparison is conducted based on mean absolute error (MAE), mean squared error (MSE) and coefficient of determination (R^2). To further validate the robustness of the model's simulation curves also presented. In addition to this, three most comprehensive and successful libraries—Keras, Scikit-learn and pyTorch are utilized to developed ANN models. In brief, the objectives are as follows:

- To explore ML algorithms for the development of Small-signal modelling
- To develop ANN, SVR, GPR, DT, RFs and Boosting based models
- To develop ANN models based on Keras, Scikit-learn and PyTorch libraries
- To find the optimal architecture of all the developed models
- To compare the models based on MSE, MAE, R^2 and simulation curves
- Finally, elucidate and discuss the overall comparison

1.2. Literature review

As explained earlier, small-signal modelling of GaN HEMT is the foundational basis for the large-signal modelling and eventually for the design of the PAs. Anwar and Ahmed (2018) developed a hybrid small-signal model parameter extraction of GaN HEMT on Si and SiC substrate exploiting global optimization methods (see Appendix A). The study was motivated to develop an effective small-signal modelling which can lead to a better design of low-noise

amplifier and large-signal modelling. The quality of parameter extractions and their extracted values pave way to find the better small-signal modelling. They pointed that even the cost effectiveness of the fabricated device plays a major role while choosing the different substrates. Si has been abundantly available and has a well-established fabrication process. That drives the authors to look for the comparative study of Si and SiC based GaN HEMTs. Generally, the hybrid methods work in two-different ways. One way is to directly extrapolate the extrinsic elements of the models utilizing either dummy test structure or device itself. However, this technique, may result in impracticality of the model since the optimization functions have to be running for each bias conditions. But second procedure, extrapolating the values of intrinsic elements directly and optimize the extrinsic parameter, is more effective extracting technique. The paper utilized the efficient Particle Swarm Optimization (PSO) global techniques for tuning of the parameters. The authors shown that this method is effective and validates the results by plotting S-parameters fitting.

Marinkovic et al. (2017) presented a comparative study of the Neural Network based modelling and parameter extraction methods (see Appendix B). ANN has been inspired by human brain and is very effective in generalizing capability. Having in the mind the less-time taken by the ANN models, fast and real-time processing, and the ability to learn any type of dynamic non-linear relationship between the data, prompted the authors to conduct the study. The study tried to give insights in terms of accuracy, model-complexity, computational efficiency, and utility of the models. The ANN based model is developed utilizing temperature (20^{o}C to 80^{o}C), gate to source voltage (VGS), drain to source voltage (VDS) and frequency as inputs and Scattering (s-parameters) as outputs. The study concluded that the both the models were useful in terms of reflecting the s-parameters. However, ANN based modelling reproduces better efficiency since while simulating the outputs it considered the deep intrinsic phenomenon

associated with the device. It was also observed that the ANN has better generalization capability with constructive generalizing capability.

Giovanni Crupi et el. (2018) went further investigating about High-periphery modelling of GaN HEMT for a broad range up to 26 GHz and 200 ^{o}C temperature. They try to take extreme operating conditions of the GaN HEMT device to thoroughly investigate the behavior of the device in different region of operations. Like [18], they included a small-signal modelling to completely define the parameters and facilitations of the study. The temperature has significant effects on the performance of any transistors due to expense of more numbers of electron-hole pair generations. They found that, upon increasing the temperature, the device's performance is slowly deteriorating. Moreover, they also looked at the effects of the thermal based parameters by mathematical and systematic approaches.

After analyzing the results, Marinkovic et al. (2019) went further and conducted a comprehensive study of microwave FETs including GaN HEMT utilizing the ANN based learning approach. This paper also considered the same inputs as listed above but they also included the geometry of the device as inputs. They developed different models for each s-parameter. Generally, s-parameters can be represented into two ways: magnitude and phase representation or real and imaginary representation. Real and imaginary representation of the s-parameters share an advantage of being within the range of 0 to 1 except for S22. This helps the training algorithms to perform well on finding the optimal steepest path while searching for minimums. The one exception here is S22, which is a magnitude parameter and go a bit further than 0 to 1 range. The paper proposed, a preprocessing, logarithmic based to deal with this problem (see Appendix C). They concluded that this type of modelling can even simulate the problematic kink effect of the s-parameters.

Ahmad, Saddam et al. (2019 [12-13]) dig further on small-signal modelling of GaN

HEMT using ANN based modeling. Previous papers considered the most common MLP architecture. In order to provide a comparative study of MLP and other ANN's architectures, the authors looked in details. The authors for the first introduced Nonlinear Autoregressive with exogenous inputs (NARX) based architecture in series-parallel and parallel configurations [12]. From the theoretical point of view, series-parallel architecture has an upper age over only parallel configuration since it feedbacks the original outputs to the input end as opposed to estimated output in case of parallel architecture (see Appendix D). The study confirms this theoretical finding by proposing two different architectures showing though Mean squared error (MSE) analysis and parameter fitting techniques. They concluded that NARX in series-parallel configuration can really enhance the performance in terms of generalization and simulating values. However, one of the main disadvantages of this approach is the complexity of the model. Due to a greater number of hardware such as delay elements the architecture becomes less computational efficient.

That disadvantage reported in [12] prompted the authors to look for other architectures. In pursuit of this the authors presented an alternative architecture of Cascade MLP, for the first time small-signal of GaN HEMT to effectively simulate the s-parameters of GaN HEMT device [13]. The paper explored, different preprocessing techniques, different initialization methods, different training algorithms alongside another architecture. The compared the performance of MLP and Cascade MLP in terms of MSE, percentage mean relative error (% MRE) and fitting curves for each s-parameter. They conducted a thorough frequency and bias dependencies of the s-parameter output for the GaN HEMT device. They concluded that Cascade MLP has a little edge over MLP architecture due to more connections involved (see Appendix E).

Marinkovic et al. (2019) advanced their study of small-signal modelling. In this paper they considered a bigger range of temperature. Instead of 40^{o}C to 80^{o}C, temperature range

reported in [11]. The authors considered a broad range of 35ºC to 200ºC temperature. This facilitated the robust study of the device for higher temperature. They build their model with the same ANN-MLP architecture (see Appendix F). They concluded that the ANN could simulate the device operation for broad range of frequency and temperature.

Jarndal et al. (2019 [15-16]) explored further and developed an ANN based learning approach for the small-signal and large-signal modelling of active devices [15]. The authors detailed the frequency, temperature, and bias dependencies of the devices. The paper proposed another feedback-based ANN architecture. The basic idea is to flood the neurons with more information to effectively recognize the relationship between the datasets. Instead of real and imaginary the paper took s-parameters in the magnitude and phase representations. The authors presented a comprehensive study of fitting parameters and plotted the s-parameters at different temperature and different bias conditions. Lastly, the author validated his results into CAD tool (see Appendix G and H). This work was extended in [16].

In [16] presented an electrothermal modelling approach using ANN techniques. As the previous papers pointed out the deterioration of the performance at higher temperatures. There are two main reasons for this, self-heating and trapping effects. As it turn out, on higher temperature, self-heating occurs due to the power dissipation. This effect directly influences the electron speed in turn mobility that results in lowering speed of the drain current. This reduces the device output power, efficiency, and gain. Another effect, trapping effects seriously upsets the performance of the device. It is of two-type surface trapping and buffer trapping of them surface trapping is of most concerned. The authors developed different models to simulate different effects by first decomposing the device nonlinearities into partial nonlinearities. First the author proposes a solution for the isothermal case in which the device was operated for the unbiased conditions ensuring no trapping. After, considering the effect of the temperature the author

proposes a genetic algorithm augmented ANN model to simulate the temperature dependencies. Lastly, a CAD based model was implemented.

All the above ANN based models utilized mainly BP-based algorithms, which suffers from the effectiveness and uniqueness of the final solutions as each and every time it starts the updating process of weights and biases by randomly generating initial weights and biases. The solution of this problem could come in many folds. One of the solutions could be the augmentation of the global optimization techniques in the standard BP based algorithms. Jarndal et al. (2019 [19]) studied the different globalization techniques such as GA, PSO and grey wolf (GW) with ANN. They studied the performance of individual models. GA, in principle, offers better initials since it considers a more diverse search space. PSO, on the other hand, offers better computational efficiency. A recently proposed algorithm GW has proven to provide the overall performance almost similar the combination of both techniques. The authors played and tuned many parameters of ANN alongside parameters of the global methods in order to find the best performing models. They concluded the study by outlining the fitting results and better initials producing by each model.

Silicon is the most auspicious material for the fabrication of the power amplifiers due to its well-established low-cost fabrication technology. The researchers are looking for the better substrate for the fabrication of GaN HEMT devices. Jarndal et al (2020) have taken three devices GaN HEMT on Si, GaN HEMT on SiC and GaN HEMT on Diamond in their study (see Appendix I). They investigated threading dislocations due to different substrate than the original materials and simultaneously went further to study the overall change in the electrical and thermal performance of the devices. They developed TCAD based models for the effective implementation of these devices from the design point of view. Diamond exhibits better thermal conductivity as opposed to the other devices reported in this paper. This can really help in

reducing the self-heating caused by the power dissipations at higher temperature. They concluded that the current collapse is minimum in case of GaN on SiC as compared to others. But they found that trapping affects can significantly affect the electrothermal behavior of these devices. For this reason, they proposed that optimization of GaN on Diamond is necessary to reduce the dislocation of elements which can further improve the performance of the device.

Most of the papers reported in this literature review utilized the gain of ANN based approaches. It was noticed that the ANN type modelling suffers from the converging at the local minimums problem. Some paper try to provide a global optimized augmented solutions, some went by changing the activation functions and preprocessing steps etc. Ahmad, Saddam et al. (2020) proposed an alternative learning-based approach to deal with the same problem. They introduced SVR based small-signal model for the GaN HEMT device on SiC substrate. As it is well known that the SVR can drive the solutions towards global minimums because it has an inbuilt SRM principle as opposed to the traditional SRM techniques in the ANN based approaches. A SVR is a supervised learning-based approach which tries to find a better decision boundary to effectively simulate the outputs (see Appendix J). The paper considered temperature, frequency, VDS, VGS and geometry of the device as inputs and S-parameters in terms magnitude and phase as outputs. They considered a highly efficient outlier detection based preprocessing steps before training the model. In addition to this they provided a comprehensive analysis of the noise detection in this paper. It was found that the model parameters are required to be tuned for improved performance. So, the authors used PSO global technique to tune the hyperparameters. Lastly, they deployed their model into CAD tool to produce a prototype of power amplifier. The disadvantage of this type of modelling is to with the computational efficiency and size of the dataset. If the dataset is bigger, it may not perform well or take a longer time to train the model. This can be further explored by using non-parametric approaches.

Chapter 2- Model Development

2.1 Programming language

Python is an open programming language with dynamically interpreted connotations [22]. It is high-level and binding, making it especially suitable as a glue language for connecting components. Python encourages program flexibility and code reuse by supporting packages. Python comes with various tools and frameworks that make coding a breeze. NumPy is utilized for numerical computations, SciPy is used for more complex calculations, and scikit, used for training data mining and analysis processing, is the most popular library. These libraries are used with sophisticated frameworks like TensorFlow, PyTorch, and Scikit-learn. These tools and frameworks are critical for machine and deep learning applications. Python code is compact and legible, even for inexperienced developers, advantageous for device and deep learning applications. For this work, three popular libraries were used.

2.1.1 Keras library

Keras is a python API that works in the TensorFlow ML framework [23]. It was designed to enable fast testing. Getting from concept to outcome as rapidly as achievable is crucial to completing successful research.

Characteristics of the Keras library:

- Simple, but not overly so.
- Decreases developer cognitive strain, focusing on the essential portions of the problem.
- It follows the notion of forward-looking disclosure of complexity, which means that simple processes should be fast and straightforward. Still, arbitrarily complex workflows should be avoided via an explicit route that builds on what is already learned.

- Robust platform that delivers industry-leading performance and scalability: it is utilized by organizations and companies like NASA, YouTube, and Waymo.

2.1.2 Scikit-learn library

Scikit-learn is a library that includes a variety of supervised and unsupervised learning techniques. It is based on technology that may already be abreast with, such as pandas, Matplotlib, and NumPy [24].

Scikit-learn supports the following features:

- Regression
- Classification
- Clustering
- Model selection
- Preprocessing

2.1.3 PyTorch library

PyTorch is an ML library based on the Python programming language [25]. It is used to solve diverse tasks.PyTorch was developed by the Facebook Artificial Intelligence Group.

PyTorch allows two models:

- Tensor computing
- The Autodiff system

2.2 Preprocessing

Preprocessing is an essential part of every data science effort, and It is one of the most often used predictors of model success [26]. When data is preprocessed correctly, and features are precisely engineered, a model will likely generate substantially better results than when information is not adequately preprocessed. Our data includes 11 Columns, temperature, the

Vds, Vgs, frequency and real-imaginary parts of S-parameters. The temperature range is from 25 to 175 with a step size of 25. Moreover, the frequency range is 0.1 GHz -26 GHz.

For data preprocessing, there are four main steps:

- Splitting the dataset into training and testing dataset
- Checking of Missing data
- Feature scaling

2.2.1 Splitting the dataset

Train Test Split is a critical stage in ML [27]. This is critical since the model must be evaluated before being transmitted. Furthermore, because approaching information is sent silently, this judgement should be based on ambiguous information.

It was decided that the split would be done manually since a random split would not give a good result for these data. Before splitting data into test and training sets, first, we should split data into Inputs and Outputs. As mentioned before, Temperature (T), Vds, Vgs and Frequency (F) will be input, and S-parameters will be outputs. Since all combinations tested for a better forecast, temperatures of 100 and 175 were chosen for testing and 25, 50, 75, 125, and 150 for training.

2.2.2 Checking of Missing data

If the data set contains a lot of NaNs and junk values, the model will almost certainly perform trash. As a result, taking care of such missing data is critical. The simple python command checked the information for NaNs, and it gave negative consequences.

2.2.3 Feature scaling

Feature scaling is a way of standardizing a group of independent different data elements [28]. This is also known as data normalization during processing data and is frequently performed during the data preparation stage.

The core principle of standardization/normalization is consistently the same. Variables assessed at various scales may not make an equivalent contribution to the learnt function and model fitting and may result in bias. Min-Max Scaling normalization is used to solve problems in the proposed model.

The Min-Max scaler may use to standardize input features [29]. As a result, elements will be translated into the range [0,1], which means that the lowest and maximum values of the variable will be 0 and 1.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.1)$$

2.3 Artificial Neural Network

AI strategies encompass computational methodologies and tools that are data-driven and knowledge-driven for solving complicated problems in various study fields [30]. The first form of AI technology is an expert system, which permits a specialist to tackle an issue and draw results from rules using thought processes. ANN modelling aims to replicate brain activity in learning tasks or procedures to address issues in various sectors, including process engineering, agricultural engineering, electrical engineering, and many others [43].

ANN modelling involves systems that simulate how the brain generates information. Instead of programming procedures, ANN modelling gathers knowledge by describing patterns [31]. The ANN's self-adaptive and nonlinearity approaches make it indispensable for modelling,

approximation, classification, predicting complex data connections, noise reduction, and pattern recognition [32]. Data collecting and knowledge processing are accomplished by establishing relationships and formulae in the dataset for grouping, simulation, and classification.

ANN models could adapt solutions over time and analyze data fast. ANN models are rapidly being employed in several study domains because of their efficiency and attractiveness and their exceptional effectiveness in pattern identification, even in complicated problems [33].

2.3.1 Basic Structure of ANN

Neurons, an input layer, an output layer, hidden layer(s), weights, activation functions, and biases comprise the construction of an ANN, as shown in Fig. 2.1. Each component of the ANN models and how they function is briefly covered ahead of time.

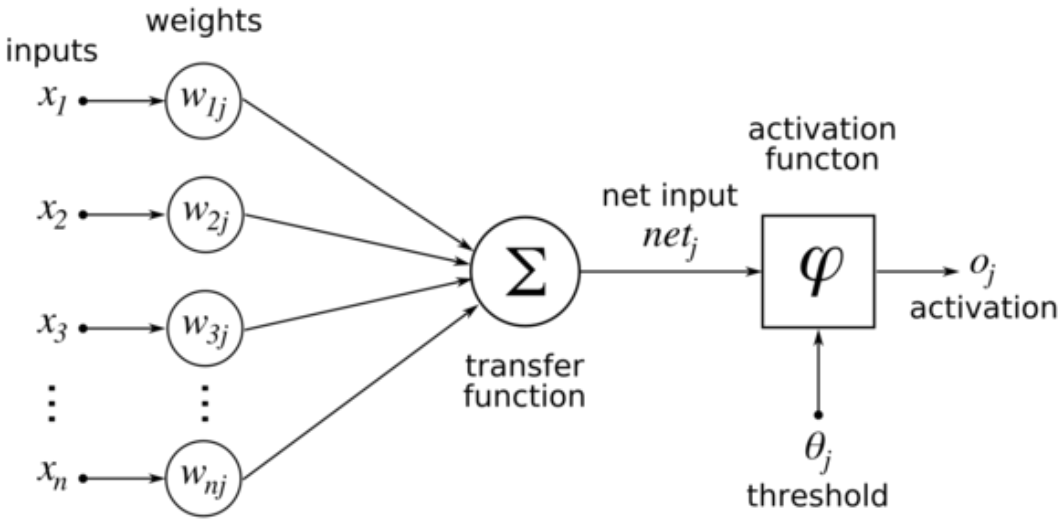

Figure 2.1: Structure of ANN model.

In following Fig. 2.2 present Proposed ANN model of GaN HEMT.

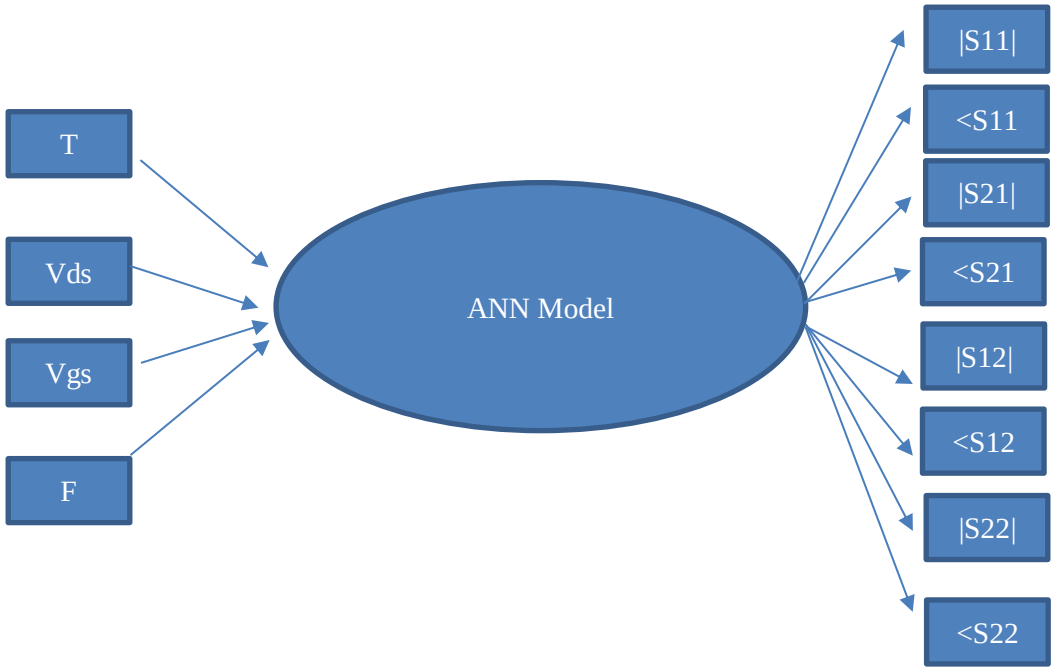

Figure 2.2: Proposed ANN model.

2.3.2 Model equation

The essential component of the proposed model that simulates the activity of biological neurons is ANN. Each neuron is a unit that combines inputs (X) and compares them to a threshold (q) to determine suitable output (Y) [34]. Weight (W) is employed in a weight matrix to assess the relevance of each input, and the consequences demonstrate the usefulness of each input data. The highest input weight value has the most influence on the neural network. The linear function's intercept is indicated as bias (b), introduced to the process to change the outputs together with the total of burdened inputs to the neurons. The tendency is employed to optimize the ANN model's fit to the provided input data to express an ANN model with a mathematical connection between several components, along-with output and input data:

$$\boldsymbol{Y = b + \sum_{n=1}^{m} W_n x_n} \qquad (2.1)$$

where X is the input, W is the weight, b is the bias, m is the amount of information, and

Y is the output. The activation functions expressed by many mathematical equations are in the following section.

2.3.3 Activation function

The activation function in ANN decides this node's output by taking into consideration the input data or a combination of input data. A standard circuit may be conceived of as a network connection of activation functions that can be "ON" (1) or "OFF" (0) depending on the input. In neural networks, this is comparable to a linear perceptron. Most nonlinear activation functions are known as nonlinearities [35]. The transfer function in Fig. 2.2 describes oscillations in a neuron's firing rate as a function of input data. Rectified Linear Unit (ReLU) activation function was utilized for this investigation.

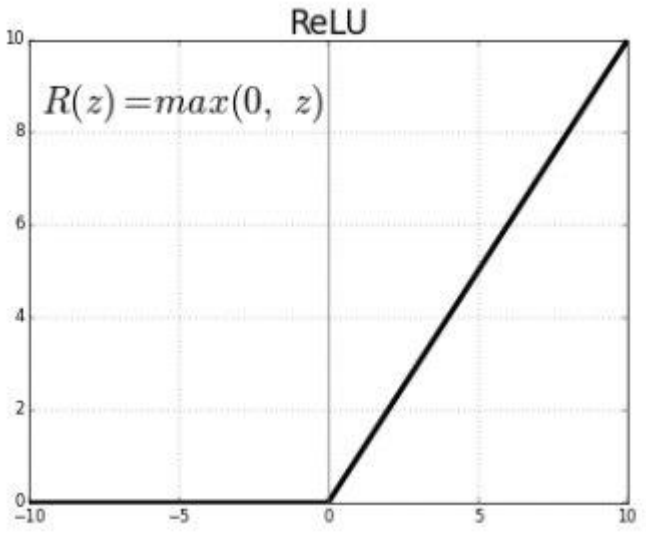

Figure 2.3: Graph of ReLU.

The following equation shows a mathematical expression of the ReLU activation function:

$$f_{(x)} = \begin{cases} 0 \ for & x < 0 \\ x \ for & x \geq 0 \end{cases} \qquad (2.2)$$

2.3.5 Architecture of neurons

The ANN structure is made up of three major components.:

- The first input layer has agreed on input nodes,
- Hidden nodes or neurons,
- The final output layer contains agreed output nodes [36].

All the input values required by the constructed neural network are contained in the input nodes. The importance of the functions of each node and its predecessors are added and sent to the next node in the network—parts of transfer [37]. In following Fig. 2.3 present Proposed ANN model architecture of GaN HEMT. The extended searching model with two hidden layers, each of 10 features, gave the best results.

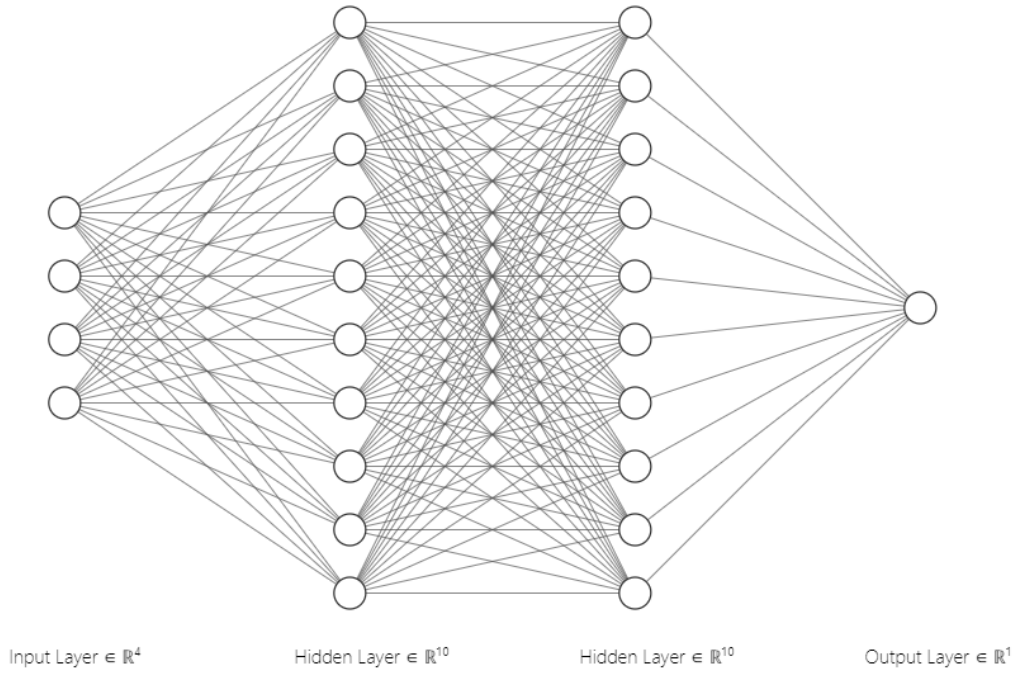

Figure 2.4: Proposed ANN model architecture.

2.3.6 Training processes

After creating the ANN topology for a specific application, it uses observed data. The

features that must be established before employing ANN modelling are weight and bias values. The starting values for these features are picked at random and then fixed. There are two types of training methodologies used in ANN modelling: (i) unsupervised methods and (ii) supervised methods [38]. The supervised practise was employed for this work.

It is impossible to assume that supervised training techniques will produce proper real-time reaction solutions to issues in the future. The supervised training techniques rely primarily on analyzing historical data, and the environment's fast-changing nature strongly impacts the accuracy of the simulation outputs. Furthermore, the supervised methods are based on the input and goal or define the differences cooperating to allow model training. The availability of goal vectors is the primary drawback of supervised training techniques, as they are often available retroactively. [53].

The generalized delta rules-based backpropagation algorithm (BPA) is generally used in ANN modelling as a training approach for setting weight and bias values [54]. The simplicity of understanding and execution is a fundamental reason for the BPA's success in training ANN models. The first step in using this approach is estimating the starting values of the weights and biases. Iteratively updating the starting values of the importance and prejudice is required. The recurrent training procedure is copied until the threshold conditions. The delta rules based BPA is defined in (2.3), (2.4).

$$E = (y - y')^2 \rightarrow \frac{\partial E}{\partial y_j^{(l)}} = 2 \text{ x } (\mathrm{y_j} - \mathrm{y_j'}) \qquad (2.3)$$

$$\omega_{ij}^{ne\omega(l)} = \omega_{ij}^{old(l)} + r\left(-\frac{\partial E}{\partial \omega_{ij}^{(l)}}\right) \qquad (2.4)$$

where E is the AER, y and y' are the goals and simulated output, $\omega_{ij}^{(l)}$ is the weight for connecting the i^{th} to the j^{th} in the first layer, with a learning rate that ranges between 0 and 1[55].

In every iteration, weights are determined by subtracting the negative gradient of the performance function from the steepest descent direction. The BPA method's delta rule calculates the rise of the error function using the mathematical chain rule of differentiation [56].

MSE and model complexity linked to process runtime and BPA iteration numbers are the threshold conditions for stopping the training procedures. Thus, the discontinued training is connected to the model's complexity and is characterized by the two primary difficulties in ANN simulation models [57]. The elements that cause the ANN model to be underfitting and overfit tend to limit the ANN models' performance capabilities. As a result, bias is minimized, but variance increases in response to model complexity [58]. The model's low complexity enhances its tendency, resulting in poor generalization and underfitting.

Data hard fitting reduces training errors, increasing testing errors, and vice versa [58]. On the other side, as model depth rises, the data becomes more challenging to fit, and the training error decreases. As a result, model overfitting results in raises a testing error and poor generalization. Typically, the complexity level determines by considering the bias-variance trade-off. Figure 2.4 displays typical testing and training error behaviour as a function of a large number of variables.

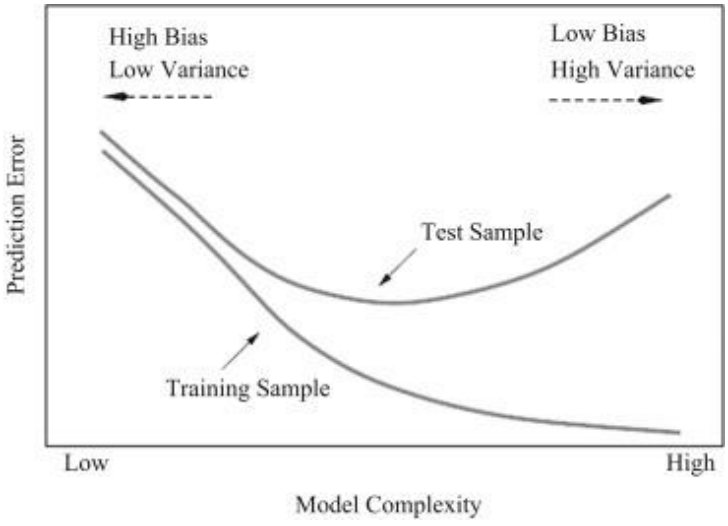

Figure 2.5: Bias variance trade-off.

2.4 Support Vector Regression

The regression problem extends the classification problem by requiring the model to deliver a continuous-valued result rather than a finite-valued result.

SVMs, address binary classification problems by transforming them into constructive optimization problems [59]. The optimization issue entails establishing a reasonable border between the accurately and hyperplane classifying. SVM's use support vectors to represent this ideal hyperplane. The SVM's sparse solution and high generalization lend themselves well to adaption to regression issues. SVM is generalized to SVR by incorporating an insensitive zone known as the tube around the function. T While optimizing the complexity of the model and prediction error, this tube reformulates to find the line that best predicts the continuous-valued function. The optimization problem was specified by first developing a convex-insensitive loss function to be lowered and then selecting the flattest tube that encompasses the majority of the training instances. SVM, which has been training samples that lie beyond the tube's perimeter, is used to represent the hyperplane. The geometrical parameters of the box and loss function are combined to form a multi-objective part. Then, using accurate mathematical optimization techniques, convex optimization is solved, which has a unique solution.

SVR is derived from the SVM classification system. The purpose of introducing SVR is to predict continuous values [60]. SVR's applicability for linear and nonlinear datasets is one of its benefits. SVR incorporates certain SVM and linear regression techniques [43]. This regression approach provides for error minimization within a certain threshold, and the overall operation of SVR is depicted in Fig. 2.5.

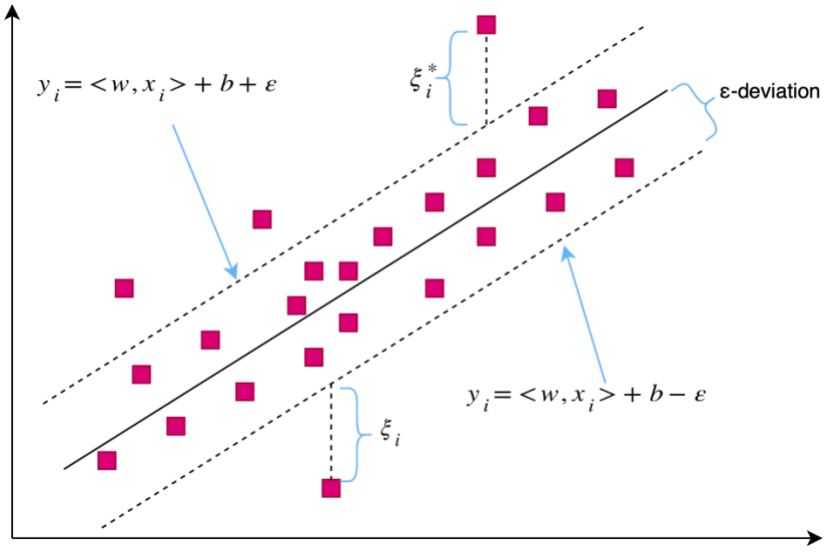

Figure 2.6: SVR concept.

The solid line in the scheme's centre is a hyperplane function that divides data points of distinct classes and is denoted as:

$$\boldsymbol{W}x + b = 0 \qquad (2.5)$$

Two black dots along the hyperplane represent boundary lines, which are represented as follows:

$$\boldsymbol{W}x + b = +\varepsilon \text{ and } \boldsymbol{W}x + b = -\varepsilon \qquad (3.4)$$

The support vectors formula is put within limits, yielding the following new form:

$$-\varepsilon \leq \boldsymbol{W}x + b \leq +\varepsilon \qquad (2.6)$$

The SVR aims to build a boundary that causes most data points to fall within the hyperplane limits. The distance between the hyperplane and the borders remains constant, with the value of SVR considering the places outside the dashed lines.

2.5 Gaussian Process Regression Model

2.5.1 Gaussian processes

Gaussian processes (GPs) are infinite-dimensional extensions of complex Gaussian distributions. Correctly, a Gaussian process creates data scattered throughout a domain so that every bounded subset of the field follows a Gaussian probability. As a result, GPs are as ubiquitous as they are bare. After some consideration, the observations in every data, y = {y_1, y_n}, may always be regarded from any multivariate Gaussian distribution. This data set may therefore be paired with a GP. The mean of this partner GP is frequently believed to be 0 everywhere. In such instances, the covariance function connects $k(x, x')$ One observation to another. The following equation shows the squared exponential equation.

$$k(x, x') = \sigma^2 \exp\left[\frac{-(x-x')^2}{2l^2}\right] \qquad (2.8)$$

where the maximum permissible covariance is specified as σ^2 — this should be a considerable value for functions covering a wide range on the y axis. If $x = x'$, $k(x, x')$ Approaches this maximum, indicating that f(x) is virtually completely correlated with f(x's). It is a good thing: our neighbours must be similar for our function to seem smooth. It is not close to x'. We get $k(x, x')$The amount of separation depends on the length parameter, l. = 0, indicating that the two locations cannot 'see' each other. As a result, distant observations will not influence interpolation at new values. Thus, there is a lot of leeways built-in (2.10). However, there is little flexibility: the data is frequently noisy due to measurement mistakes and other factors. Through a Gaussian noise model, each observation y may be conceived of as being connected to an underlying function f(x):

$$y = f(x) + N(0, \sigma^2) \qquad (2.9)$$

Something that should be recognizable to anyone who has done regression previously.

The search for f(x) is referred to as regression. For the sake of clarity on the following page, we employ the new technique of folding the noise into $k(x, x')$ by writing:

$$k(x, x') = \sigma^2 \exp\left[\frac{-(x-x')^2}{2l^2}\right] + \sigma^2 \delta(x, x') \qquad (2.10)$$

$\delta(x, x')$ Is Kronecker delta function.

2.5.2 Gaussian Process Regression

GPR is a multivariate, Bayesian method gaining popularity in ML. GPR has several advantages, including the ability to perform successfully on datasets and provide confidence metrics on predictions [44]. Unlike many traditional supervised learning approaches, which learn specific values for each variable, The Bayesian method entails a probabilistic model that encompasses all possible values. The Bayesian method operates by first determining a posterior probability, p(w), on the parameter, w, and then redistributing probability.

Bayes' Rule:

$$p(w|y, X) = \frac{p(y|X, w)p(w)}{p(y|X)} \qquad (2.11)$$

$$posterior = \frac{likelihood\ x\ prior}{marginal\ likelihood} \qquad (2.12)$$

Assume y=wx+ ϵ is a linear function. Therefore, the posterior distribution is the updated distribution $p(w|y, X)$. To get predictions at previously unobserved places of interest can be computed by weighing all potential forecasts by their determined posterior distribution:

$$p(f * |x *, y, X) = \int_w p\,(f * |x *, w)p(w|y, X)dw \qquad (2.13)$$

For integral to be tractable, both prior and probability are commonly considered Gaussian. Using this assumption, it generates a Gaussian distribution from which it can extract a point forecast using the mean and quantify.

Because the GPR is non - parametric, it evaluates the probability circulation of overall

feasible features that correlate to the Rather than the probability distribution, consider the facts of the parameters of a specific function. Nonetheless, we create a prior, calculate the future using the data for training, and compute the anticipated posterior distribution on our points of interest, as previously discussed.

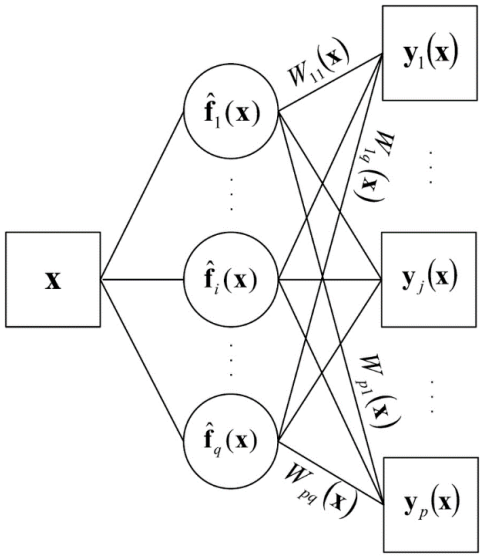

Figure 2.7: Structure of Gaussian process regression.

2.5.3 Model Selection

During version selection, the form of implicit functionality and covariance kernel functionality in the GP is decided and changed. Typically, the implicit function is constant, either 0 or implicit, of the instruction data set. Commonly used kernel features consist of constant exponential, linear, and rectangular Matern kernels and an aggregate of many kernels. The functionality of the covariance kernel has many options: it could take much bureaucracy as long because it adheres to the characteristics of a kernel [45].

This kernel contains 2 hyperparameters: l for length and 2 for signal variance. We may choose from some kernels in scikit-learn and provide their hyperparameters' initial value and constraints. Following the specification of the kernel, it can now define further options for the

GP framework. For instance, alpha is the dispersion of the noise on the labels. Normalized y denotes the constant function. Increasing the log margin probability of the training set is a typical method for tuning the hyperparameters.

2.6. Decision Trees

DT is a modelling technique used in statistics, data mining, and ML. It uses a DT to make inferences about an object's goal value based on its observations. In classification trees, objective change can take a separate set; leaves in these tree structures represent branches represent feature conjunctions, and class labels that lead to brands. Regression trees are a kind of DT with a parameter that can have a range of data. DT are among the most widely used ML approaches due to their conciseness and simplicity. A DT can be used in decision analysis to describe decision making graphically and explicitly. A DT is used in data mining to explain data and is a data mining approach that is widely utilized [46].

A DT is an honest way to classify cases. Suppose that each piece of information entered has discrete finite domain names, and there may be a proper function called "classification" for this section. Each detail of the categorization area is called a class. Arcs emanating from a node simply labelled with a distribution function are labelled with the skill values of each target function. The attitude goes to a subsidiary selection node on a separate distribution function. Therefore, each node in the tree is diagnosed with a category or distribution function, indicating that the tree classifies records defined in a specific model or random distribution. A tree is built using a means of dividing the provisioning set, which serves as the tree's root node, into subsets, which function as the successor children of the tree. This recursive approach to each derived subset is called recursive partitioning. It is based mainly on a classification chain based on complete cleavage criteria [47].

DT in record mining will also be described as a set of mathematical and computational tactics that help describe, classify, and generalize a given data set. Data is presented in the form of entries such as:

$$(x, Y) = (x1, x2, x3, \dots, xk, Y) \quad (2.15)$$

The dependent variable Y is a target variable that we attempt to comprehend, categorize, or generalize. Vector x is made from things that are used in this activity.

Data mining uses two types of DT. When doing a classification tree analysis, the projected output is the class (discrete) to which the data belongs. When the anticipated outcome is a regression tree analysis is employed natural. The phrase classification and regression tree (CART) analysis refers to either of the abovementioned approaches initially described [48]. There are some parallels between regression trees and classification trees, but there are also significant distinctions, such as the process used to decide where to divide [48].

DT constructs the model based on regression in a tree structure, as illustrated in Fig. 2.8. The trees are divided into smaller subsets. An accompanying DT is created at the same time as a decision node, at least with two components, reflecting one of the tested predictor's values. The choices in the tree start at the root node and end at the leaf node, with the leaf node storing the replies. The leaf node reflects the target's final answer. The root of a tree's tree is the highest decision node that significantly contributes.

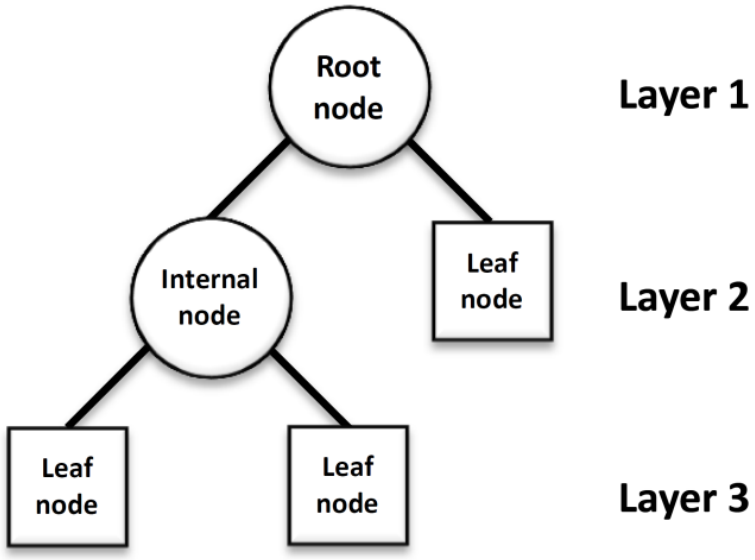

Figure 2.8: Basic Structure of DT.

2.7 Random Forest

Rf is a supervised ML algorithm regularly applied in regression and classification applications [49]. He builds woods chosen from many specimens and uses their majority vote for regression and classification. The essential factor of the RF algorithm is that it can deal with unit facts with every non-stop variable and express variable. It outperforms other algorithms in categorization tasks.

2.7.1 The Random Forest Algorithm in Action

Before we can grasp how the random forest works, we must first understand the ensemble approach. The concept "ensemble" relates to a grouping of numerous models. As a result, a group of models is utilized to create predictions rather than a single model.

Ensemble employs two types:

1. Bagging generates a distinct training subset from the sample training set with replacement, and a majority vote determines the final output. As an example, consider Random Forest.

2. Boosting– It merges weak learners with solid learners through producing models with the best accuracy.

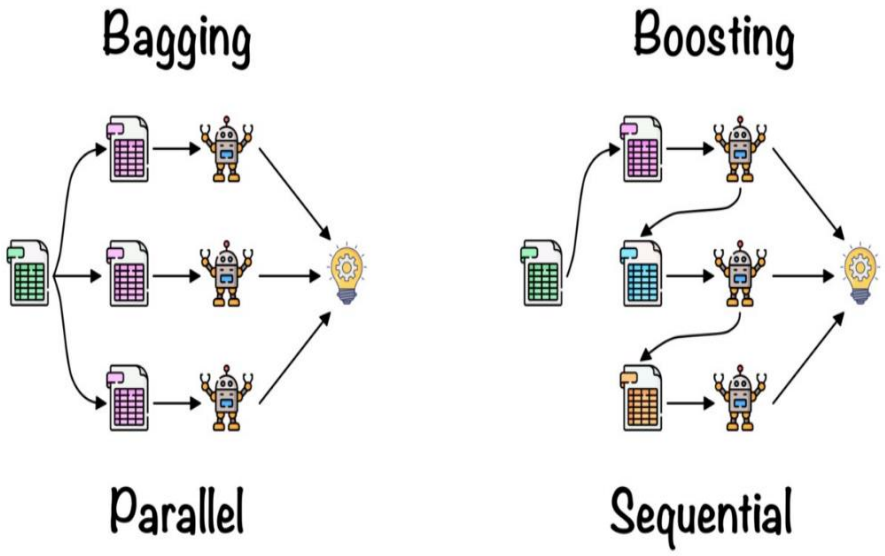

Figure 2.9: Ensemble methods.

2.7.2 Bagging

Now, let's look at roughly everything that should be done in bagging. RF use the bagging technique known as bootstrap aggregation [50]. RF, as previously said, is essentially based on the notion of bagging, and it chooses a random pattern from a set of statistics. As a result, each version is constructed utilizing the drawings supplied by the original data via row sampling. The bootstrap approach is used to choose the line's degree of replacement. Each version is now separately trained, and effects are created. After combining the models' findings, the outcomes mostly depend on the collective vote. All outcomes and emerging outcomes are generally based on majority voting.

2.8 Hyperparameter Optimization

ML models contain hyperparameters that must be set to optimize the model to the dataset. However, the broad impacts of hyperparameters on a framework are commonly recognized. Understanding how to properly configure a training set and a combination of for a given dataset, interacting hyperparameters [51]. For setting hyperparameters, there are frequently general multi-objective rules of thumb. Hyperparameter Optimization (HO) or hyperparameter tweaking, and the Python ML toolkit supports it. A HO produces a single set of high-achieving hyperparameters that may be used to design the model.

2.8.1 Types of Hyperparameter Optimization

Hyperparameters are choice or customization points in an ML model that allows it to be adjusted to a specific job or dataset. The developer offers a model configuration parameter to steer the learning process for a particular dataset. The internal coefficients of ML models are generated by training or tweaking the network using a training dataset. Variables and hyperparameters are still not synonymous. Parameters are learned automatically. However, hyperparameters must be manually provided to facilitate learning. A hyperparameter has a recognized impact on a model; however, it is unknown how to optimize the configuration of a hyperparameter for a given dataset. Furthermore, many ML models include several hyperparameters that may associate nonlinearly.

It is necessary to look for a collection of hyperparameters. Consequently, the model performs exceptionally well in the dataset. It is referred to as HO, tweaking, or hyperparameter search. The first thing in an optimization technique is to define a search space. Every hyperparameter is a separate dimension, and the size scale represents the range of values that the hyperparameter can take.

The volume will be explored, with each dimension being a hyperparameter and each point representing a model option. A vector with a distinct value for each hyperparameter value is a point in the search space. The optimization technique seeks to find a vector that produces the best performance of the model after learning, such as high precision or the smallest error. Various optimization techniques are utilized, but the simplest and most frequent are random search and grid search:

- Random search. Create a search region as a limited range of hyperparameter values and measurements made at random from that domain.
- Grid Search. Make a search space by describing it as a framework of hyperparameter values and assessing each point inside it.

2.8.2 Random search

Random search is comparable to grid search in that it checks a randomly selected portion of the grid's points rather than all of them. The smaller this selection, the faster the optimization, but the less accurate it is. The richer the dataset, the more precise the optimization, but the closer it comes to a grid search. Random search is a great solution when data have numerous hyperparameters with a fine-grained grid of values. We can generate a relatively excellent set of hyperparameter values by using a subset of 5-100 randomly picked points. It is improbable to be the optimal point, but it may be a decent group of values that leads to a good model. In this work, random search methods were used. It has more powerful benefits than grid search, especially for the dataset used. In this work, random search methods were used. It has more powerful benefits than grid search, especially for the dataset used. Except for the ANN model, random search hyperparameter tuning is used in all other models. Each method has its hyperparameters. So, by using the scikit-learn library, we can easily find hyperparameters for each model.

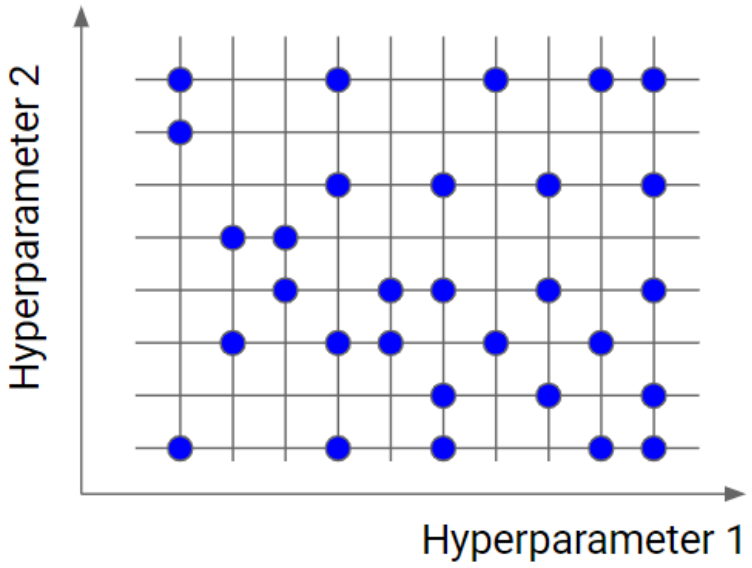

Figure 2.10: Example of random search.

2.9 Loss function

A loss function in ML measures how well the ML model predicts the predicted outcome, often known as the ground truth [52]. The loss function accepts our model's output value and the ground truth predicted value. The loss function's output is known as the loss and measuring how good our model predicted the event. A significant loss number indicates that our model performed exceptionally poorly. A low loss number indicates that our model worked admirably. The correct loss function must be used to train an accurate model. Certain loss functions will have specific qualities that will assist the model in learning in a particular way. This part will look at the two most frequent loss functions used in ML Regression. Bellow will describe how they function, their advantages and disadvantages, and how to use them when training regression models effectively.

2.9.1 Mean Squared Error

The MSE is likely the most basic and widely used loss function, and it is frequently taught in introductory ML classes [53]. To compute the MSE, divide the difference between the

ground truth and the model's predictions by two, square it, and average it across the whole data. The MSE equations are presented follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (2.16)$$

Where N is the number of samples being tested, the following figure 2.11 shows the function of MSE.

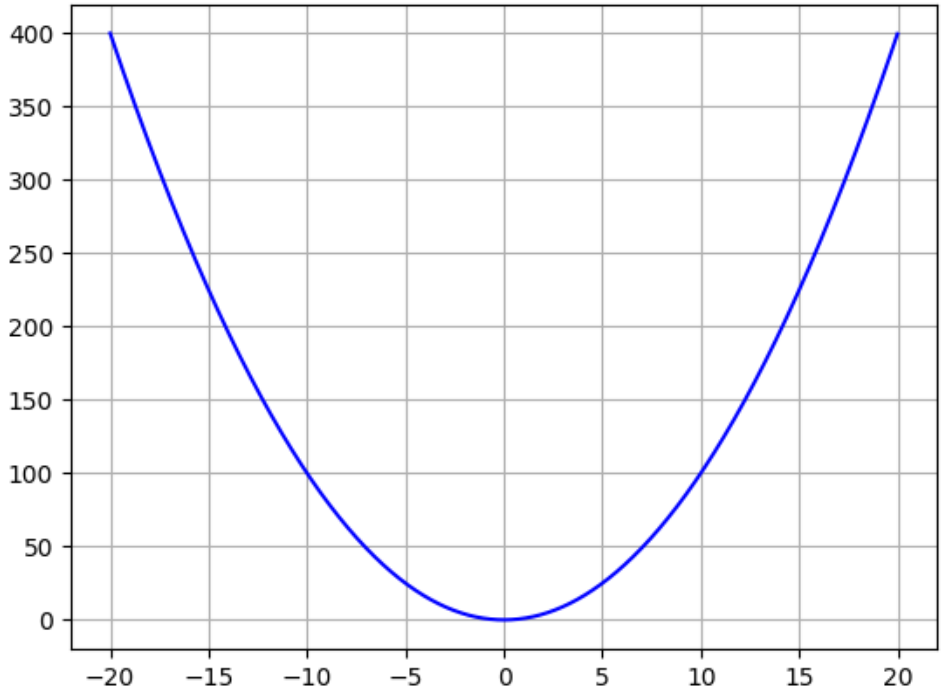

Figure 2.11: MSE Loss Function.

Because the squaring part of the function gives these errors more weight, the MSE ensures that our training set does not have any outlier predictions with errors. If our model makes a single, extremely inaccurate forecast, the squaring section of the process multiplies the mistake. In many practical situations, though, we don't worry about these outliers and instead seek a well-rounded model that performs well enough on the majority.

2.9.2 Mean Absolute Error

The Mean Absolute Error (MAE) is like the MSE, but it has the opposite features

virtually! To calculate the MAE, divide the differential between the model's predictions and the classification algorithm by the absolute value and average it across the whole dataset. The MAE, like the MSE, will never be negative because we are always taking the total value of the errors in this case. The following equation officially defines the MAE:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \qquad (2.17)$$

The advantage of the MAE is that it immediately offsets the disadvantage of the MSE. Because it uses the absolute value, all errors are evaluated on a similar linear scale. [53]. As a result, unlike the MSE, we will not emphasize our outliers, and our loss function gives a general and consistent assessment of how well our model is working.

The MAE's disadvantages are that if we care about our model's outlier predictions, the MAE will be less successful. The big mistakes caused by outliers are weighted the same as the more minor errors. As a result, our model may be excellent most of the time while producing a few terrible forecasts now and again. Fig. 2.12 shows the comparison of MSE and MAE loss functions.

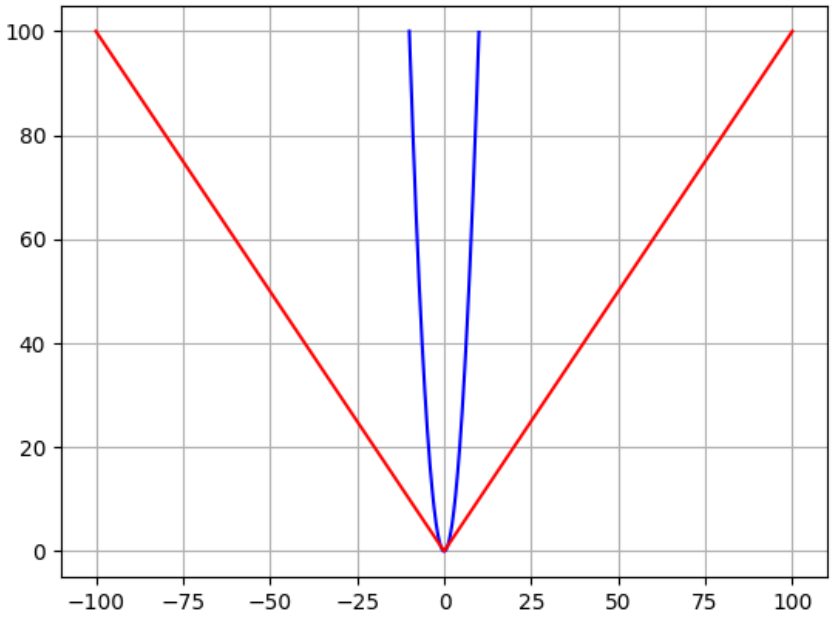

Figure 2.12: MAE (red) and MSE (blue) loss functions.

2.9.3 Coefficient of determination

Coefficient of determination (R^2) is a statistics metric in a regression model which establishes how much the independent variable can account for the dependent variable [54]. R2 indicates how well the data fitted the regression model in those other worlds. R^2 can have any value from 0 to 1. Although the statistical measure gives some helpful insights into the regression model, the user should not depend only on it when evaluating a statistical model. The graph reveals nothing about the causal link between the variables.

Furthermore, it does not show the accuracy of the regression model. As a result, when studying R^2 in conjunction with the other variables, the user should always conclude the model when using a statistical model. The most typical use of analyzer-squared is to discover how well the regression model fits the observational data. In general, a higher R2 indicates a better model fit. However, a high R2 is not always advantageous to the regression model.

The type of variables used in the model, the methods of the relationship between variables, and the data transformation performed all impact the statistical measure's quality. A high R2 may occasionally indicate problems with the regression model.

A low R2 score for prediction models is a bad sign. On the other hand, a good model may have a little markdown. There are no hard and fast rules for incorporating a statistical metric into a model evaluation [55]. The experiment's context or prediction is crucial, and the statistics' insights may differ based on the circumstances.

The formula of R^2:

$$\boldsymbol{R^2 = 1 - \frac{SS_{regression}}{SS_{total}}},$$

$$\boldsymbol{= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}} \qquad (2.18)$$

SS_{total} is the total sum of squares, and $SS_{regression}$ is the regression sum of squares.

Chapter 3- Result Section

Results & Discussions chapters consist of four sections: regression model outputs, library comparison and model comparison to performall computations Python libraries: *NumPy, Pandas, Keras, Scikit-Learn, PyTorch, Matplotlib, SciPy* was employed.

3.1 Regression model outputs

This section contains the outcomes of all models trained many times, with the best results being selected. As a result, we consider Training Loss, MSE, MAE, R^2.

3.1.1 Keras ANN model

ANN was developed using Python language. *Keras* Python library was used to train the ANN model. The MAE, MSE and R2 error estimations were added to the code to check the model's performance. As a result of training, all the parameters prediction was perfect. R^2 coefficient for all parameters was more than 0.99%. In Fig. 3.1, we can see the decrease of training loss by epochs.

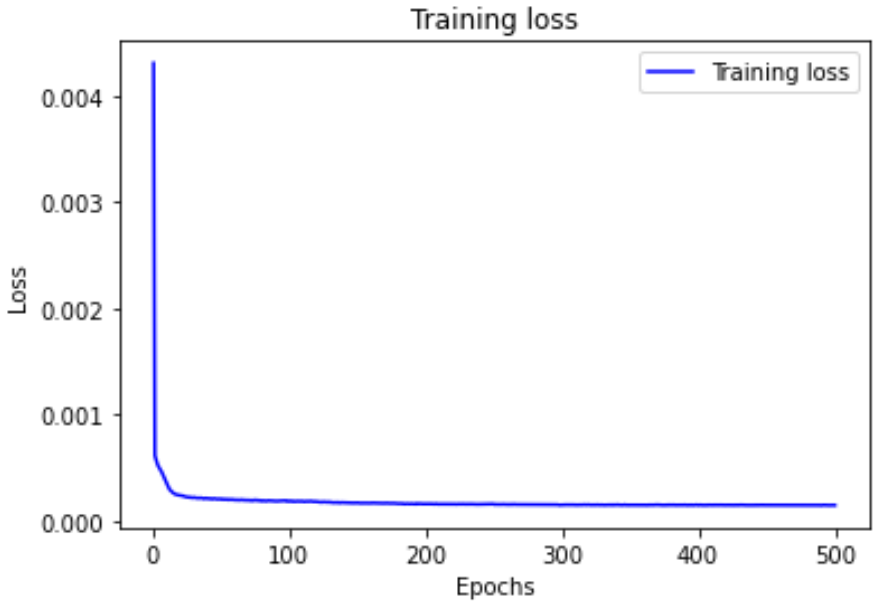

Figure 13.1: Decreasing Training Loss by Epochs.

The proposed model uses ANN Architecture with two hidden layers, each with ten features and a ReLu activation function. The predicted MSE, MAE, and R2 are calculated for the training and testing set presented in Tables 3.1 and 3.2.

Table 1: Keras ANN Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	2.5171e-05	0.0026	0.9984
ImS11	6.1956e-03	0.0034	0.9967
ReS21	9.9812e-05	0.0046	0.9943
ImS21	8.9148 e-04	0.0038	0.9958
ReS12	5.1546e-05	0.0030	0.9974
ImS12	4.8494e-06	0.0025	0.9969
ReS22	8.9849e-06	0.0034	0.9952
ImS22	3.8945e-06	0.0024	0.9979

Table 1: Keras ANN Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	2.6516e-05	0.0029	0.9978
ImS11	6.9419e-03	0.0041	0.9968
ReS21	1.6516e-04	0.0051	0.9945
ImS21	9.1848e-04	0.0046	0.9936
ReS12	5.5785e-05	0.0038	0.9978
ImS12	5.0125e-06	0.0031	0.9942
ReS22	9.5171e-05	0.0041	0.9971
ImS22	4.5291e-05	0.0045	0.9950

For this model, the V_{GS}=-1.600V V_{DS}=48.00V bias condition gave the best prediction. Fig. 3.2 demonstrate a comparison of simulated and measured data of ReS11 parameters.

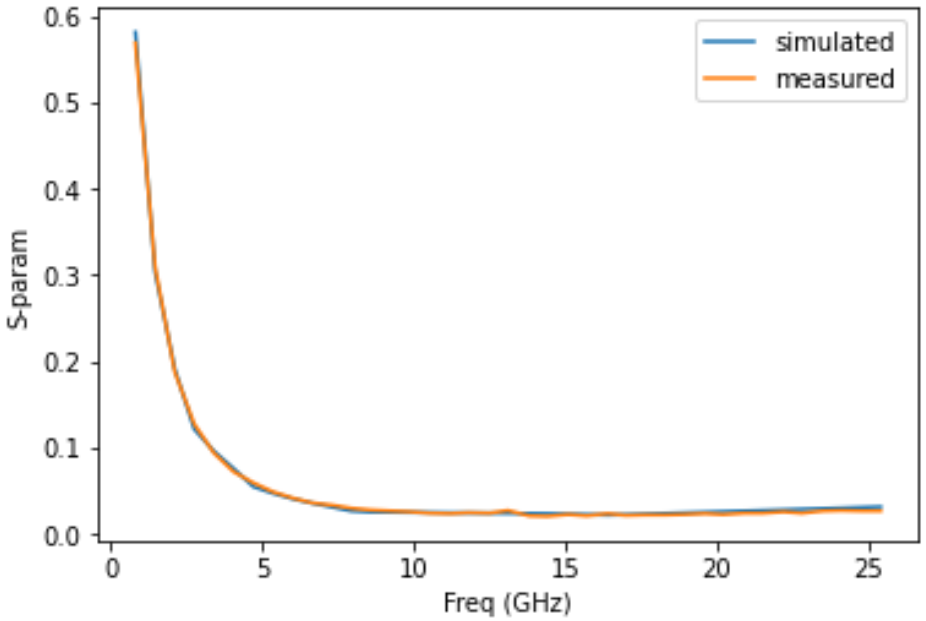

Figure 3.2: ReS11 parameter.

3.1.2 Scikit-learn ANN model

Scikit-Learn ANN model uses using ReLu activation function as Keras. In this case, though, we have the Stochastic Gradient Descent method. (SGD). Tables 3.3 and 3.4 present model metrics results.

Table 2 Scikit-learn ANN Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	3.4963e-05	0.0498	0.9929
ImS11	6.7820e-03	0.0145	0.9910
ReS21	3.4318e-05	0.0498	0.9912
ImS21	4.7463 e-04	0.0984	0.9818
ReS12	5.8531e-05	0.0492	0.9946
ImS12	4.7949e-06	0.0796	0.9749
ReS22	1.7462e-05	0.0496	0.9938
ImS22	2.4762e-05	0.1268	0.9922

Table 3 Scikit-learn ANN Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	4.8561e-03	0.0549	0.9918
ImS11	7.5614e-03	0.0167	0.9904
ReS21	3.8752e-02	0.0579	0.9913
ImS21	5.6846e-03	0.1358	0.9805
ReS12	4.6487e-03	0.0456	0.9935
ImS12	5.9845e-02	0.0845	0.9714
ReS22	2.3516e-03	0.0641	0.9945
ImS22	9.1965e-02	0.1469	0.9913

In this model, the best-predicted bias condition is V_{GS}=-2.000V, V_{DS}=28.00V. Fig. 3.2 compares simulated and measured data of ImS21 parameters.

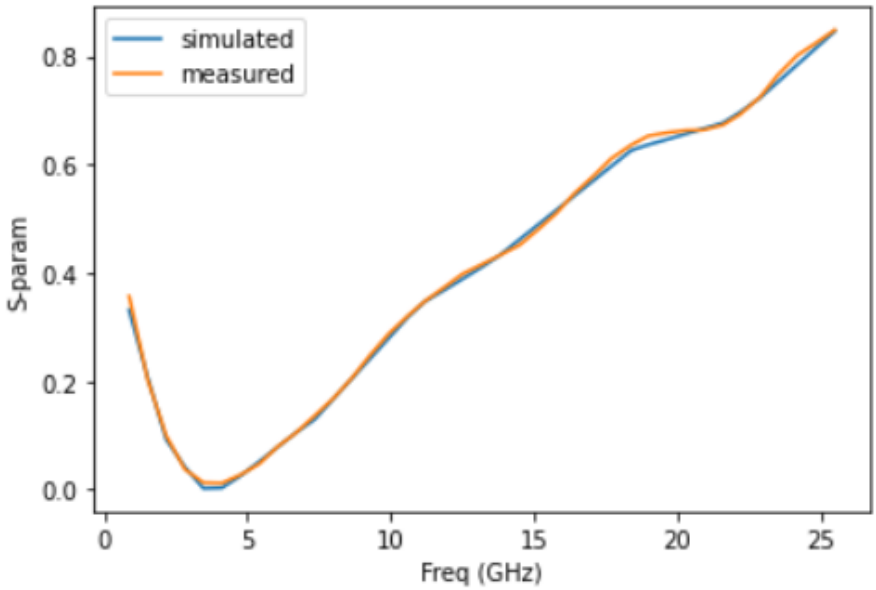

Figure 3.3 ImS22 parameter.

3.1.3 PyTorch ANN model

We changed the data to the torch in this model, and the Adam optimizer was used. Tables 3.5 and 3.6 present model metrics results.

Table 4PyTorch ANN Model Error Estimators, Testing set

Training	MSE	MAE	R^2, %
ReS11	4.9849e-03	0.0548	0.9918
ImS11	6.9482e-03	0.0657	0.9904
ReS21	1.4766e-02	0.0413	0.9913
ImS21	3.6354e-03	0.0432	0.9805
ReS12	3.7498e-03	0.0447	0.9935
ImS12	5.3268e-02	0.0094	0.9714
ReS22	6.9871e-03	0.0349	0.9945
ImS22	4.6846e-02	0.0665	0.9913

Table 5 PyTorch ANN Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	6.6841e-03	0.0629	0.9918
ImS11	7.6324e-03	0.0746	0.9904
ReS21	3.8496e-02	0.0483	0.9913
ImS21	4.8217e-03	0.0496	0.9805
ReS12	5.6324e-03	0.0145	0.9935
ImS12	7.4485e-02	0.0674	0.9714
ReS22	8.7436e-03	0.0465	0.9945
ImS22	6.008e-02	0.0784	0.9913

For this model, the V_{GS}=-1.300V V_{DS}=48.00V bias condition gave the best prediction.

Fig. 3.3 demonstrate a comparison of simulated and measured data of ImS11 parameters.

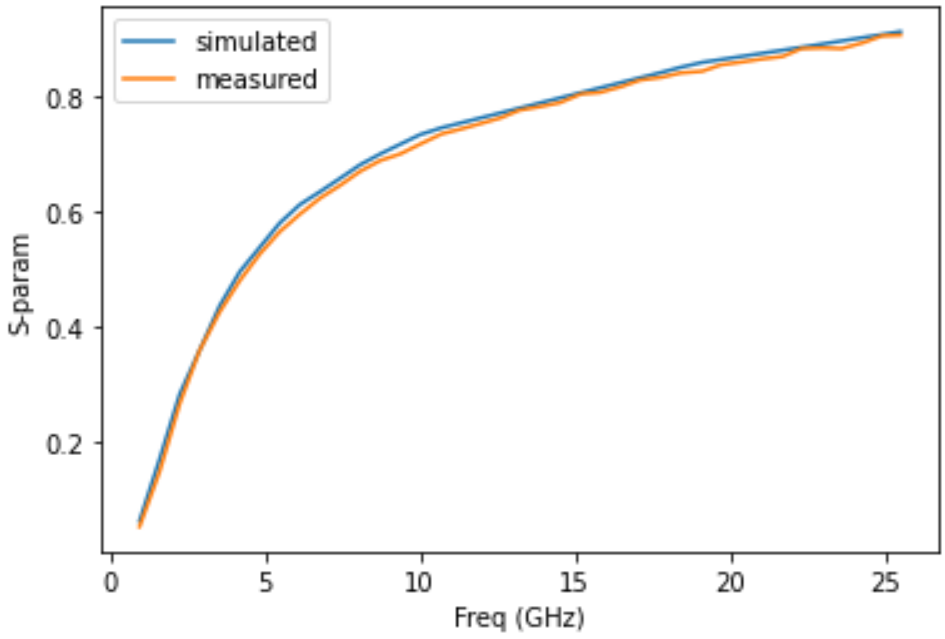

Figure 3.4 ImS11 parameter.

3.1.4 SVR model

An SVR model rbk kernel was used. By hyperparameter tuning, the best hypermeters were found in the following Table 3.7 and 3.8 present model metrics results.

Table 6SVR Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	0.0017	0.0007	0.9966
ImS11	0.0161	0.0093	0.9256
ReS21	0.0063	0.0029	0.9403
ImS21	0.0088	0.0037	0.9115
ReS12	0.0084	0.0053	0.9072
ImS12	0.0091	0.0061	0.9413
ReS22	0.0124	0.0082	0.9917
ImS22	0.0212	0.0145	0.9876

Table 7SVR Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	0.0023	0.0015	0.9907
ImS11	0.0168	0.0110	0.9157
ReS21	0.0054	0.0026	0.9548
ImS21	0.0067	0.0031	0.9059
ReS12	0.0130	0.0085	0.9261
ImS12	0.0128	0.0080	0.9345
ReS22	0.0130	0.0199	0.9790
ImS22	0.0258	0.0205	0.9782

The VGS=-1.500V VDS=48.00V bias condition for the SVR model gave the best prediction. Fig. 3.4 demonstrate a comparison of simulated and measured data of ReS_{22} parameters.

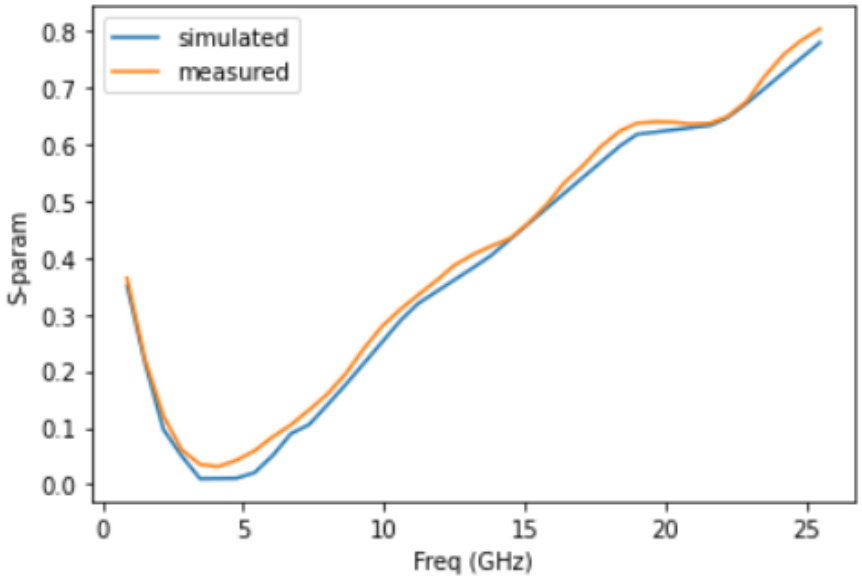

Figure 3.5 ReS22 parameter.

3.1.5 GPR model

GPR model gave less prediction than previous models, and this model took a long time to train, about 65 minutes for one training. The best hypermeters found in Tables 3.9 and 3.10 present model metrics results by hyperparameter tuning.

Table 8GPR Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	0.0119	0.0334	0.9874
ImS11	0.0142	0.0630	0.9841
ReS21	0.0283	0.0512	0.9702
ImS21	0.0214	0.0562	0.9882
ReS12	0.0227	0.0529	0.9755
ImS12	0.0235	0.0842	0.9746
ReS22	0.0077	0.0411	0.9918
ImS22	0.0124	0.0822	0.9875

Table 9GPR Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	0.0142	0.0475	0.9867
ImS11	0.0156	0.0674	0.9841
ReS21	0.0247	0.0504	0.9852
ImS21	0.0263	0.0582	0.9782
ReS12	0.0337	0.0699	0.9701
ImS12	0.0316	0.0961	0.9718
ReS22	0.0094	0.0518	0.9910
ImS22	0.0157	0.0989	0.9807

For this model, the V_{GS}=-1.300V V_{DS}=28.00V bias condition gave the best prediction.

Fig. 3.5 demonstrate a comparison of simulated and measured data of the ReS21 parameter.

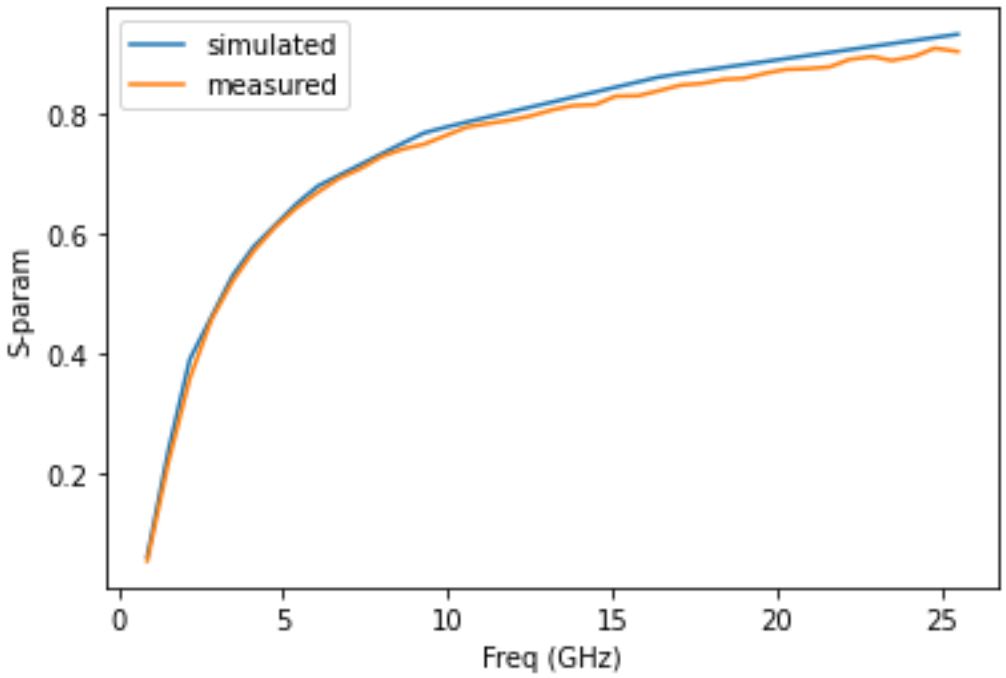

Figure 3.6 14ReS21 parameter.

3.1.6 DT model

DT model gave good prediction, and training of this model was fast, about 1 minute for one workout. The best hypermeters found in Tables 3.11 and 3.12 present model metrics results by hyperparameter tuning.

Table 10DT Model Error Estimators, Training set

Training set	MSE	MAE	R^2, %
ReS11	0.0065	0.0259	0.9932
ImS11	0.0079	0.0469	0.9919
ReS21	0.0604	0.0567	0.9173
ImS21	0.0315	0.0426	0.9699
ReS12	0.0374	0.0835	0.9590
ImS12	0.0296	0.0924	0.9679
ReS22	0.0187	0.0656	0.9801
ImS22	0.0030	0.0378	0.9969

Table 11DT Model Error Estimators, Testing set

Testing set	MSE	MAE	R^2, %
ReS11	0.0071	0.0284	0.9924
ImS11	0.0090	0.0501	0.9909
ReS21	0.0635	0.0572	0.9259
ImS21	0.0319	0.0498	0.9634
ReS12	0.0397	0.0912	0.9547
ImS12	0.0314	0.0954	0.9642
ReS22	0.0191	0.0668	0.9777
ImS22	0.0036	0.0385	0.9923

For this model (V_{GS}=-2.000V V_{DS}=7.00V) bias condition gave the best prediction. Fig. 3.6 demonstrate a comparison of simulated and measured data of the ImS21 parameter.

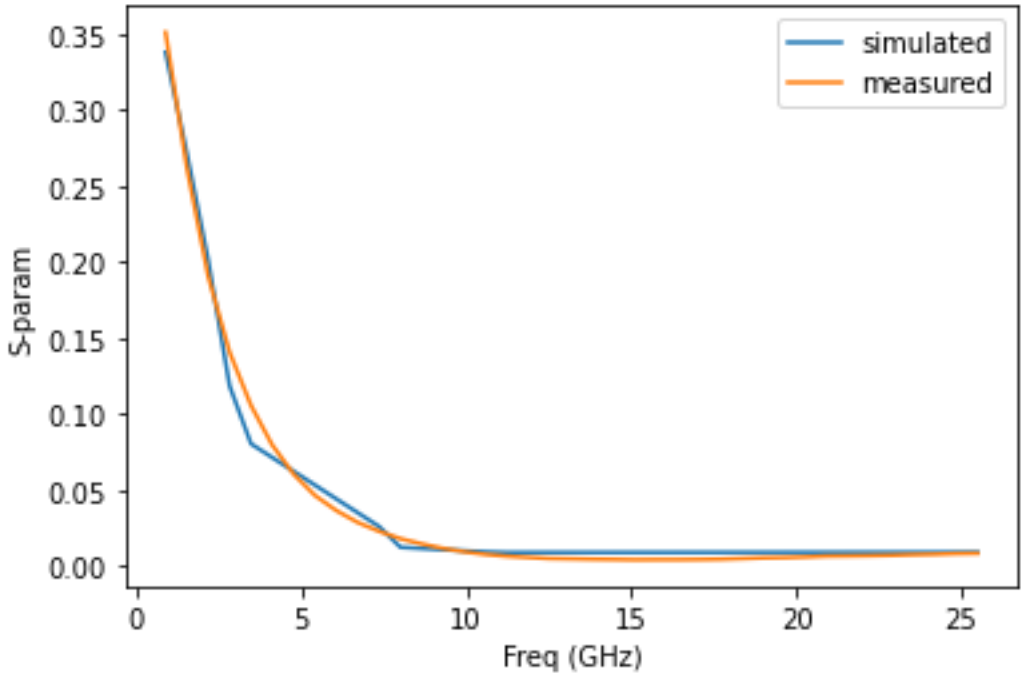

Figure 3.7 ImS21 parameter.

3.1.7 RF model

The proposed model uses hyperparameter tuning. The predicted MSE, MAE, and R2 are calculated for the training and testing set presented in Tables 3.13 and 3.14.

Table 12RF Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	0.0095	0.0308	0.9899
ImS11	0.0095	0.0492	0.9901
ReS21	0.0396	0.0411	0.9289
ImS21	0.0155	0.0414	0.9737
ReS12	0.0167	0.0503	0.9957
ImS12	0.0079	0.0433	0.9915
ReS22	0.0076	0.0381	0.9919
ImS22	0.0029	0.0323	0.9969

Table 13RF Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	0.0102	0.0338	0.9892
ImS11	0.0106	0.0531	0.9891
ReS21	0.0457	0.0417	0.9459
ImS21	0.0199	0.0414	0.9718
ReS12	0.0179	0.0593	0.9940
ImS12	0.0098	0.0503	0.9903
ReS22	0.0082	0.0413	0.9910
ImS22	0.0045	0.0384	0.9948

For this model, the V_{GS}=-2.000V V_{DS}=28.00V bias condition gave the best prediction.

Fig. 3.7 demonstrate a comparison of simulated and measured data of the ReS12 parameter.

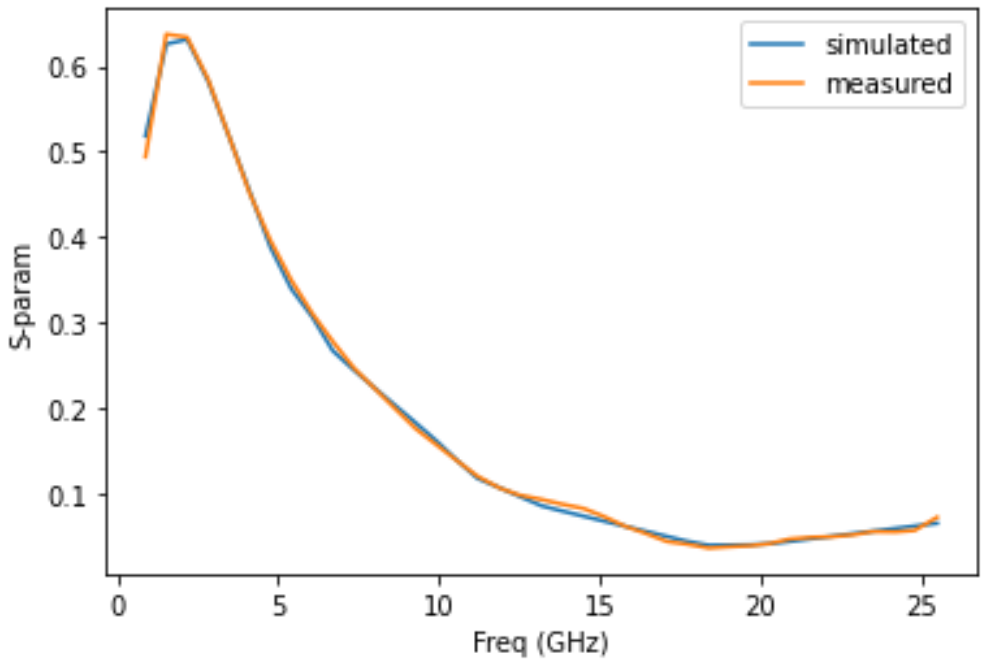

Figure 3.8 ReS12 parameter.

3.1.8 Boosting

The Boosting model uses hyperparameter tuning. The predicted MSE, MAE, and R2 are calculated for the training and testing set presented in Tables 3.15 and 3.16.

Table 14Boosting Model Error Estimators, Training set

Training	MSE	MAE	R^2, %
ReS11	0.0135	0.0054	0.9847
ImS11	0.0073	0.0099	0.9916
ReS21	0.0048	0.0016	0.9451
ImS21	0.0137	0.0030	0.9555
ReS12	0.0131	0.0058	0.9941
ImS12	0.0095	0.0067	0.9925
ReS22	0.0082	0.0061	0.9945
ImS22	0.0170	0.011	0.9980

Table 15Boosting Model Error Estimators, Testing set

Testing	MSE	MAE	R^2, %
ReS11	0.0193	0.0063	0.9823
ImS11	0.0084	0.0155	0.9887
ReS21	0.0062	0.0078	0.9546
ImS21	0.0194	0.0044	0.9936
ReS12	0.0153	0.0055	0.9945
ImS12	0.0087	0.0063	0.9932
ReS22	0.0089	0.0074	0.9912
ImS22	0.0184	0.0181	0.9952

For this model, the V_{GS}=-1.8.000V V_{DS}=7.00V bias condition gave the best prediction.

Fig. 3.8 demonstrates a comparison of the simulated and measured data of the ImS12 parameter.

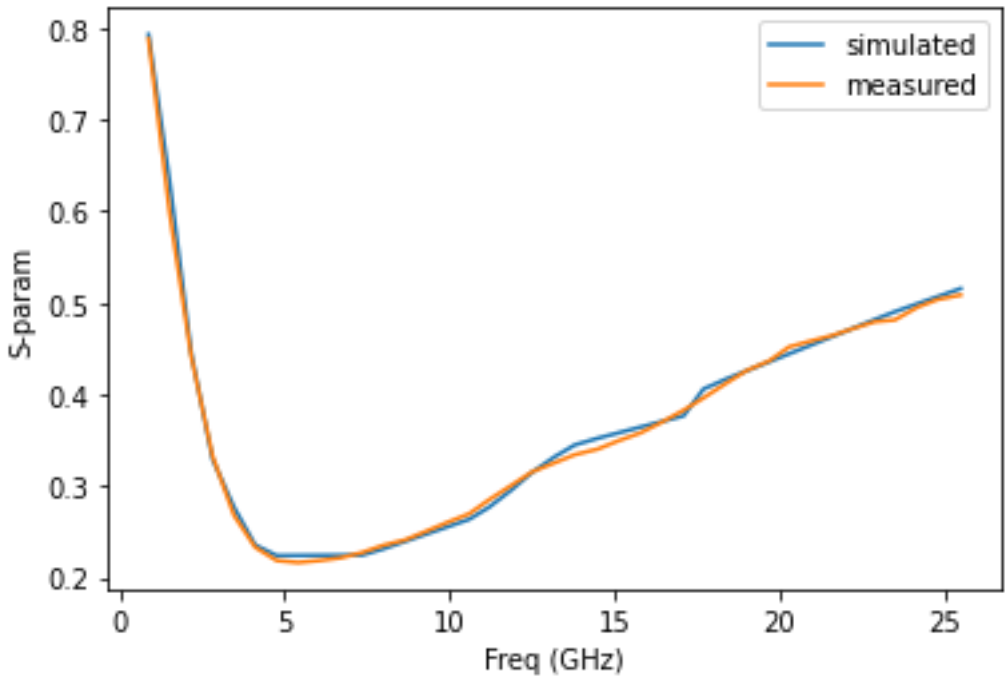

Figure 3.9 ImS12 parameter.

3.2 Library comparison

All the ANN models gave excellent results. However, the Keras library has the best performance than Scikit-Learn and PyTorch libraries. The time for training in Keras library is a high one. Comparing Scikit-Learn and PyTorch, the R2 metrics are generally the same. Nevertheless, by MSE metrics, Scikit-Learn is better. Moreover, using Scikit-Learn is easier than PyTorch. In the following Table: 3.17 and 3.18, MSE metrics of all parameters of three libraries are present.

Table 16ANN Models MSE, Testing set

Testing	Keras	Scikit-Learn	PyTorch
ReS11	2.5171e-05	3.4963e-05	4.9849e-03
ImS11	6.1956e-03	6.7820e-03	6.9482e-03
ReS12	9.9812e-05	3.4318e-05	1.4766e-02
ImS12	8.9148 e-04	4.7463 e-04	3.6354e-03
ReS21	5.1546e-05	5.8531e-05	3.7498e-03
ImS21	4.8494e-06	4.7949e-06	5.3268e-02
ReS22	8.9849e-06	1.7462e-05	6.9871e-03
ImS22	3.8945e-06	2.4762e-05	4.6846e-02

Table 17ANN Models MSE, Testing set

Testing	Keras	Scikit-Learn	PyTorch
ReS11	2.6516e-05	4.8561e-03	6.6841e-03
ImS11	6.9419e-03	7.5614e-03	7.6324e-03
ReS12	1.6516e-04	3.8752e-02	3.8496e-02
ImS12	9.1848e-04	5.6846e-03	4.8217e-03
ReS21	5.5785e-05	4.6487e-03	5.6324e-03
ImS21	5.0125e-06	5.9845e-02	7.4485e-02
ReS22	9.5171e-05	2.3516e-03	8.7436e-03
ImS22	4.5291e-05	9.1965e-02	6.0081e-02

As mentioned before following, Table 3.19 shows the time for one training.

Table 18ANN libraries time comparison

	Keras	Scikit-Learn	PyTorch
Training time	16 min	2 min	1 min

3.3 Model comparison

This section compares all the presented models by R^2 and MSE metrics; the best model is ANN. All S-parameters of the ANN model have more than 0.99% prediction. Then RF, the third possession is DT, then SVR and GPR. The faster is DT, then RF, Boostin, Keras ANN, SVR and GPR. Tables 3.20 and 3.21 present the R^2 of all parameters.

Table 19Models R2, Testing set

Training	ANN	SVR	DT	RF	Boosting	GPR
ReS11	0.9984	0.9966	0.9932	0.9899	0.9847	0.9874
ImS11	0.9967	0.9256	0.9919	0.9901	0.9916	0.9841
ReS12	0.9943	0.9403	0.9173	0.9289	0.9451	0.9702
ImS12	0.9958	0.9115	0.9699	0.9737	0.9555	0.9882
ReS21	0.9974	0.9072	0.9590	0.9817	0.9941	0.9755
ImS21	0.9969	0.9413	0.9679	0.9915	0.9925	0.9746
ReS22	0.9952	0.9917	0.9801	0.9919	0.9945	0.9918
ImS22	0.9979	0.9876	0.9969	0.9969	0.9980	0.9875

Table 20Models R2, Testing set

Testing	ANN	SVR	DT	RF	Boosting	GPR
ReS11	0.9978	0.9907	0.9924	0.9892	0.9823	0.9867
ImS11	0.9968	0.9157	0.9909	0.9891	0.9887	0.9841
ReS12	0.9945	0.9548	0.9259	0.9459	0.9546	0.9852
ImS12	0.9936	0.9059	0.9634	0.9718	0.9936	0.9782
ReS21	0.9978	0.9261	0.9547	0.9800	0.9945	0.9701
ImS21	0.9942	0.9345	0.9642	0.9903	0.9932	0.9718
ReS22	0.9971	0.9790	0.9777	0.9910	0.9912	0.9910
ImS22	0.9950	0.9782	0.9923	0.9948	0.9952	0.9807

Chapter 4- Conclusions

Exploring the application of ML techniques for GaN-on-Si HEMT device modelling is an essential and timely research area since ML techniques can help overcome difficulties involving complexity and parameter computation time.

The GaN-on-Si HEMT device was considered in this thesis. For next-generation power devices, the GaN HEMT device is a strong contender. Because of its higher breakdown strength, faster-switching speed, and improved thermal conductivity outperforms classic Si-based power devices. Eight machine learning models were created, as previously stated. SVR, DT, GPR, RF, and Boosting are three separate libraries used for ANN. As this thesis explores, it is possible to compare the models by the findings. All provided models are accurate and make reasonable expectations. The DT model is the fastest, followed by the RF, Boosting, Keras ANN, SVR, and GPR models. Keras's ANN model was shown to be the most efficient. The proposed models can be integrated into CAD software for analysis and simulation.

As a future project development, explore different libraries for other ML models. Furthermore, planning to do a combination of ML models, which can be used simultaneously with much better forecasts.

Bibliography

[1] T. Leng *et al.*, “Non-Volatile RF Reconfigurable Antenna on Flexible Substrate for Wireless IoT Applications,” *IEEE Access*, vol. 9, pp. 119395-119401, Aug. 2021.

[2] B. Frankston, “Consumer Technology Versus 5G,” *IEEE Consum. Electron Mag.*, vol. 10, no. 2, pp. 43-50, March 2021.

[3] N. F. M. Aun *et al.*, “Revolutionizing Wearables for 5G: 5G Technologies: Recent Developments and Future Perspectives for Wearable Devices and Antennas,” *IEEE Microw. Mag.*, vol. 18, no. 3, pp. 108-124, May 2017.

[4] F. Zeng *et al.*, “A Comprehensive Review of Recent Progress on GaN High Electron Mobility Transistors: Devices, Fabrication and Reliability,” *Electronics*, vol. 7, no. 12, p. 377, Dec. 2018.

[5] A. Jarndal *et al*., “Large-Signal Modeling of GaN HEMTs Using Hybrid GA-ANN, PSO-SVR, and GPR-Based Approaches,” in *IEEE J of the Electron Devices Soc.*, vol. 9, pp. 195-208, Nov. 2021.

[6] X. Du *et al.*, “ANN-Based Large-Signal Model of AlGaN/GaN HEMTs With Accurate Buffer-Related Trapping Effects Characterization,” *IEEE Trans. Microw. Theory Tech.*, vol. 68, no. 7, pp. 3090-3099, July 2020.

[7] U. K. Mishra, P. Parikh and Yi-Feng Wu, "AlGaN/GaN HEMTs-an overview of device operation and applications," in Proceedings of the IEEE, vol. 90, no. 6, pp. 1022-1031, June 2002.

[8] S. Colangeli, A. Bentini, W. Ciccognani, E. Limiti and A. Nanni: ‘GaN-Based Robust Low-Noise Amplifiers’, IEEE Transactions on Electron Devices, vol. 60, no. 10, Oct. 2013.

[9] Z. Marinković, G. Crupi, A. Caddemi and V. Marković, "GaN HEMT small-signal modelling: Neural networks versus equivalent circuit," *2017 IEEE 30th International Conference on Microelectronics (MIEL)*, pp. 153-156, 2017.

[10] G. Crupi, A. Raffo, V. Vadalà, G. Vannini and A. Caddemi, "High-periphery GaN HEMT modeling up to 65 GHz and 200 °C", *Solid-State Electronics*, vol. 152, pp. 11-16, 2019.

[11] Z. Marinković, G. Crupi, A. Caddemi, V. Marković and D. Schreurs, "A review on the artificial neural network applications for small-signal modeling of microwave FETs", *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 33, no. 3, 2019.

[12] A. Khusro, S. Husain, M. S. Hashmi, M. Auyuneur and A. Q. Ansari, "A Reliable and Fast ANN Based Behavioral Modeling Approach for GaN HEMT," 2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 277-280, 2019.

[13] A. Khusro, S. Husain, M. Hashmi and A. Ansari, "Small signal behavioral modeling technique of GaN high electron mobility transistor using artificial neural network: An accurate, fast, and reliable approach", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 30, no. 4, 2019.

[14] Z. Marinković *et al*., "Temperature Dependent Small-Signal Neural Modeling of High-Periphery GaN HEMTs," *2019 14th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS)*, pp. 33-36, 2019.

[15] A. Jarndal, "On Neural Networks Based Electrothermal Modeling of GaN Devices", *IEEE Access*, vol. 7, pp. 94205-94214, 2019.

[16] A. Jarndal, "Neural network electrothermal modeling approach for microwave active devices", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 29, no. 9, 2019.

[17] A. Majumdar, S. Chatterjee, S. Chatterjee, S. Chaudhari and D. Poddar, "An ambient temperature dependent small signal model of GaN HEMTusing method of curve fitting", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 30, no. 12, 2020.

[18] [7]A. Jarndal and A. Hussein, "Hybrid small-signal model parameter extraction of GaN HEMTs on Si and SiC substrates based on global optimization", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 29, no. 10, 2018.

[19] A. Jarndal, S. Hamdan and M. Bettayeb, "On Neural Networks Modeling Based on GA, PSO and GW Optimization Techniques," *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pp. 1-5, 2019.

[20] A. Jarndal, L. Arivazhagan and D. Nirmal, "On the performance of GaN-on-Silicon, Silicon-Carbide, and Diamond substrates", *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 30, no. 6, 2020.

[21] A. Khusro, S. Husain, M. S. Hashmi, A. Q. Ansari, and S. Arzykulov, "A Generic and Efficient Globalized Kernel Mapping-Based Small-Signal Behavioral Modeling for GaN HEMT," IEEE Access, vol. 8, pp. 195046–195061, 2020.

[22] What is Python? Executive Summary", Python.org, 2022. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 19- Mar- 2022].

[23] F. Chollet, "Deep Learning Chatbot using Python", International Journal of Advanced Trends in Computer Science and Engineering, vol. 10, no. 3, pp. 2019-2023, 2021.

[24] "Machine Learning in Python", scikit-learn, 2017. [Online]. Available: http://scikit-Iearn.org/stable/.

[25] C. Fries, "Stochastic Automatic Differentiation: Automatic Differentiation for Monte-Carlo Simulations", SSRN Electronic Journal, 2017.

[26] C. Carpineto, Concept Data Analysis: Theory and Applications, 1st ed. Wiley, 2004.

[27] "Everything you need to know about Min-Max normalization in Python", Medium, 2022. [Online]. Available: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79. [Accessed: 19- Mar- 2022].

[28] Z.R. Yang, Z. Yang "SPRINGS: Prediction of Protein-Protein Interaction Sites Using Artificial Neural Networks", Journal of Proteomics & Computational Biology, vol. 1, no. 1, pp. 01-07, 2014.

[29] S. Qian, H. Liu, C. Liu, S. Wu and H. S. Wong, "Adaptive activation functions in convolutional neural networks", Neurocomouting, vol. 272, pp. 204-212, 2018.

[30] P. McAdam and P. McNelis, "Forecasting inflation with thick models and neural networks", Economic Modelling, vol. 22, no. 5, pp. 848-867, 2005.

[31] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research", Journal of Pharmaceutical and Biomedical Analysis, vol. 22, no. 5, pp. 717-727, 2000.

[32] K. LEE, D. BOOTH and P. ALAM, "A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean firms", Expert Systems with Applications, vol. 29, no. 1, pp. 1-16, 2005.

[33] F. Jiang et al., "Artificial intelligence in healthcare: past, present and future", Stroke and Vascular Neurology, vol. 2, no. 4, pp. 230-243, 2017.

[34] A. Hussain, P. Liatsis, M. Khalaf, H. Tawfik and H. Al-Asker, "A Dynamic Neural Network Architecture with Immunology Inspired Optimization for Weather Data Forecasting", Big Data Research, vol. 14, pp. 81-92, 2018.

[35] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification", Neurocomputing, vol. 50, pp. 223-235, 2003.

[36] N. Chitsaz, A. Azarnivand and S. Araghinejad, "Pre-processing of data-driven river flow forecasting models by singular value decomposition (SVD) technique", Hydrological Sciences Journal, vol. 61, no. 12, pp. 2164-2178, 2016.

[37] Z. Chan, H. Ngan, A. Rad, A. David and N. Kasabov, "Short-term ANN load forecasting from limited data using generalization learning strategies", Neurocomputing, vol. 70, no. 1-3, pp. 409-419, 2006.

[38] K. Nordhausen, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition by Trevor Hastie, Robert Tibshirani, Jerome Friedman", International Statistical Review, vol. 77, no. 3, pp. 482-482, 2009.

[39] I. E1-Naqa, Y. Yang, M. N. Wernick, N. P. Galasanos, and R. M. Nishikawa, "A support vector machine approach for detection of microcalcifications," IEEE Trans. Med. Imag., vol. 21, no. 12, Dec. 2002.

[40] A.S. Bozkir et al., "Geological Strength Index (GSI) Determination by Local Image Descriptors and Machine Learning Methods", in *Human-Computer Interaction*, T. Ozseven, Nova Science Publishers, 2020.

[41] D. Ellis, E. Sommerlade and I. Reid, "Modelling pedestrian trajectory patterns with Gaussian processes", *Proc. 11th IEEE Int. Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA '05)*, pp. 1533-2306, 2005.

[42] Luca Pasolli, Farid Melgani and Enrico Blanzieri, "Gaussian process regression for estimating chlorophyll concentration in subsurface waters from remote sensing data", *Geoscience and Remote Sensing Letters IEEE*, vol. 7, no. 3, pp. 464-468, 2010.

[43] K. Driessens, J. Ramon and T. Gärtner, "Graph kernels and Gaussian processes for relational reinforcement learning", *Machine Learning*, vol. 64, no. 1-3, pp. 91-119, 2006.

[44] A. Gordon, L. Breiman, J. Friedman, R. Olshen and C. Stone, "Classification and Regression Trees.", *Biometrics*, vol. 40, no. 3, p. 874, 1984.

[45] J. R. Quinlan, "Induction of decision trees", *Machine learning*, vol. 1, pp. 81-106, 1986.

[46] S. A. Mitrofanov and E. S. Semenkin, "Tree retraining in the decision tree learning algorithm", *IOP Conference Series: Materials Science and Engineering*, vol. 1047, pp. 012082, 2021.

[47] J. Quinlan, "Simplifying decision trees", *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 221-234, 1987.

[48] T. Kam, "The random subspace method for constructing decision forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832-844, 1998.

[49] Education, "What is Bagging?", *Ibm.com*, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/bagging#:~:text=Bagging%2C%20also%20known%20as%20bootstrap,be%20chosen%20more%20than%20once. [Accessed: 19- Mar- 2022].

[50] J. Rasley, Y. He, F. Yan, O. Ruwase and R. Fonseca, "HyperDrive: Exploring Hyperparameters with POP Scheduling", *Proceedings of Middleware '17*, 2017.

[51] S. Raschka, J. Patterson, and C. Nolet, "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence", *information*, vol. 11, no. 4, p. 193, 2020.

[52] H. Zhang and C. Huang, "A note on processes with random stationary increments", *Statistics & Probability Letters*, vol. 94, pp. 153-161, 2014.

[53] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance", *Climate Research*, vol. 30, pp. 79-82, 2005.

[54] F. David, R. Steel and J. Torrie, "Principles and Procedures of Statistics.", *Biometrika*, vol. 48, no. 12, p. 234, 1961.

[55] L. Magee, "R 2 Measures Based on Wald and Likelihood Ratio Joint Significance Tests", *The American Statistician*, vol. 44, no. 3, p. 250, 1990.

Appendix

A. Small-signal equivalent circuit model proposed in [18]

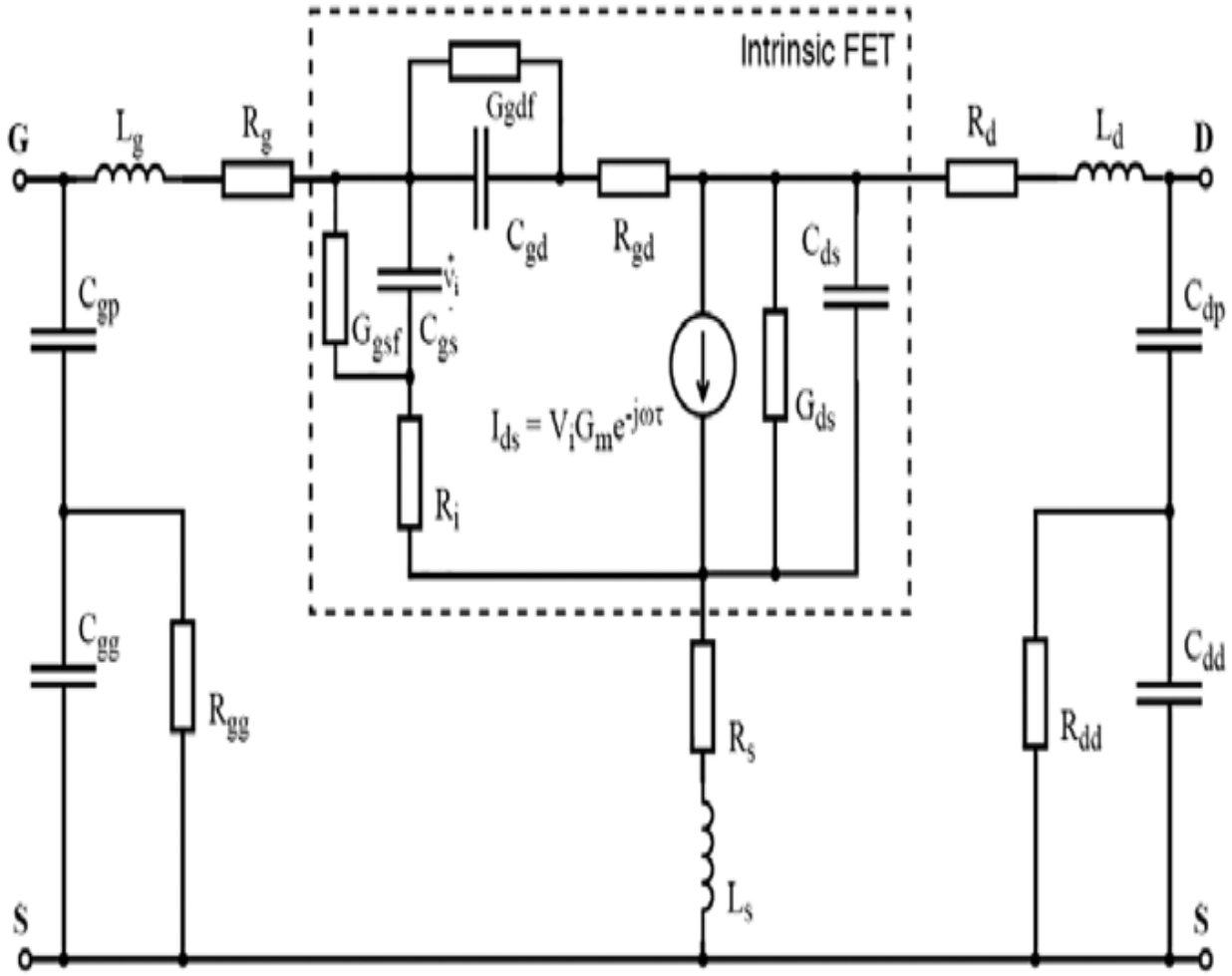

Fig. A1. Small-signal equivalent circuit model utilized to build the model in [18]

B. Proposed model in [9]

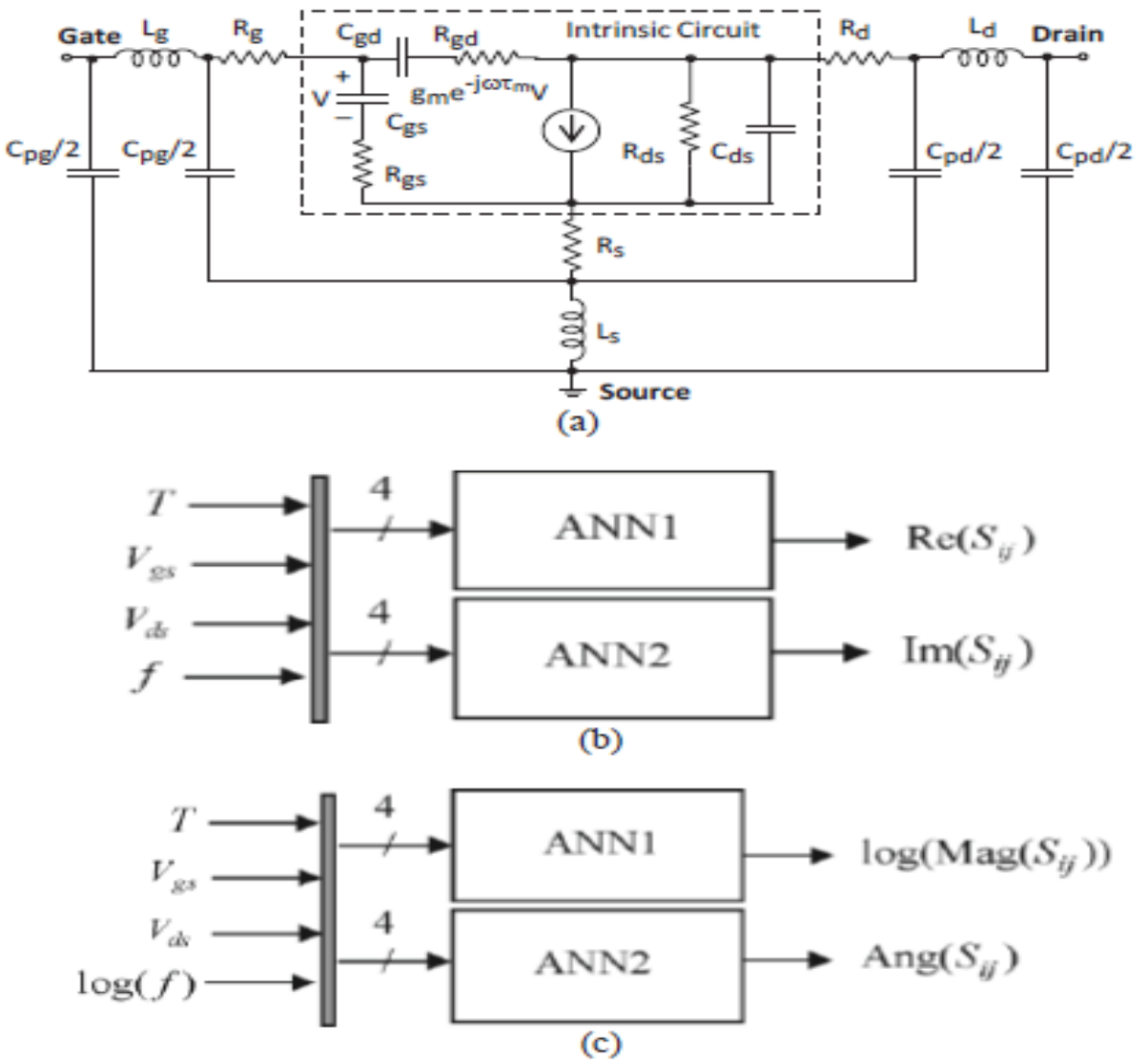

Fig. B1. Model studied for GaN HEMT (a) Equivalent circuit model (b) neural model for S_{11}, S_{12} and S_{22} (c) neural model for S_{21}

C. Proposed ANN based model for GaN HEMT [11]

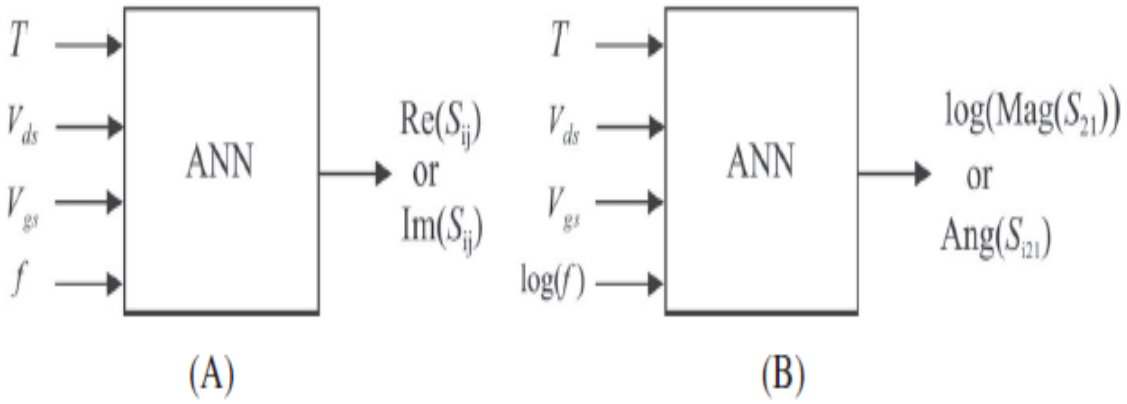

Fig. C1. Model studied for GaN HEMT (A) neural model for S_{11}, S_{12} and S_{22} (B) neural model for S_{21}

D. NARX architecture in [12]

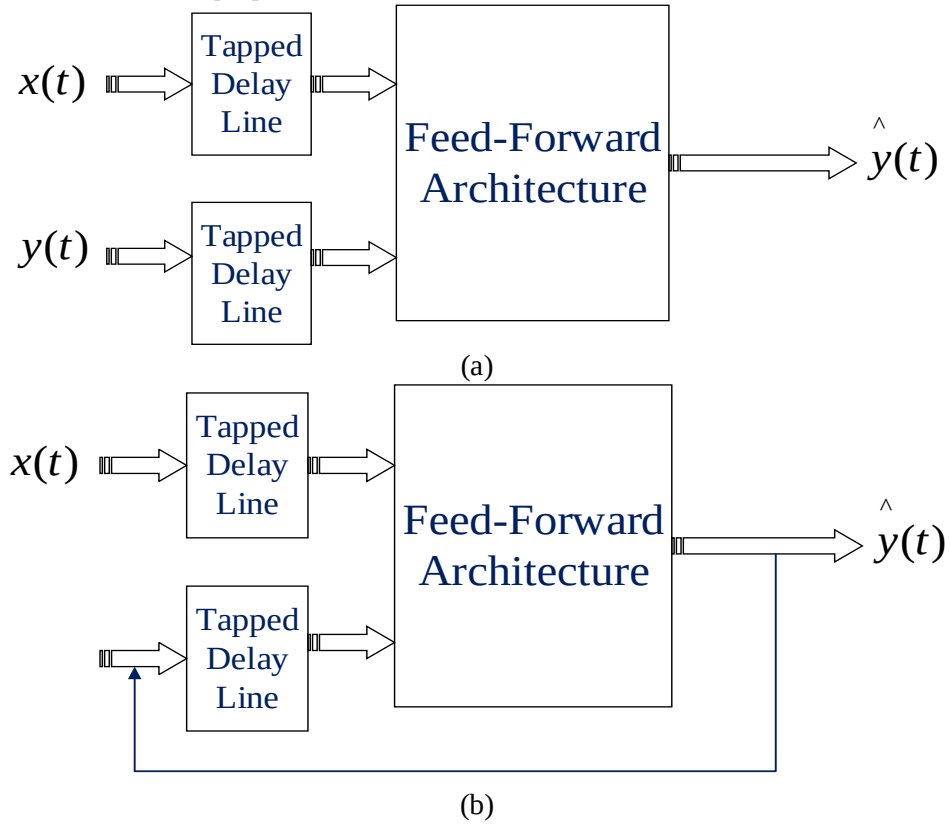

Fig. D1. NARX architecture (A) Series-parallel (B) Parallel

E. Proposed ANN-MLP architecture for small-signal modelling of GaN HEMT [13]

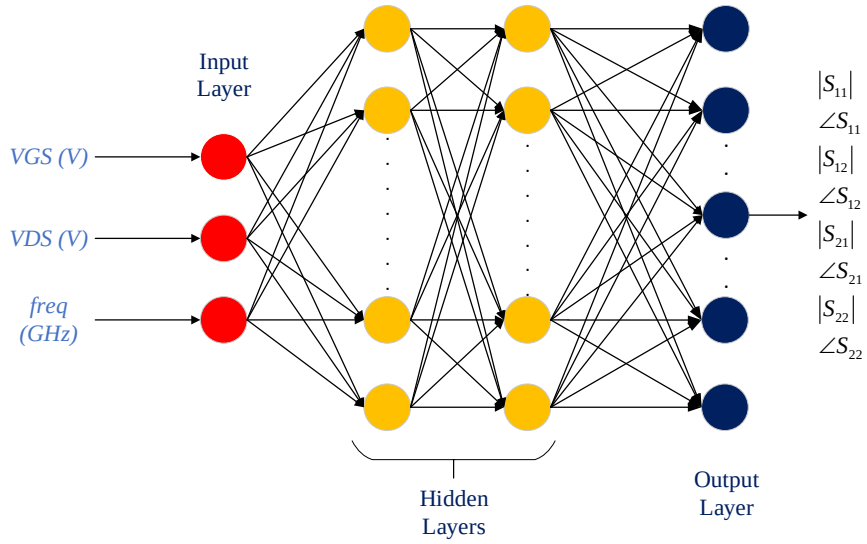

Fig. E1. Proposed MLP architecture

F. Results of proposed models in [14]

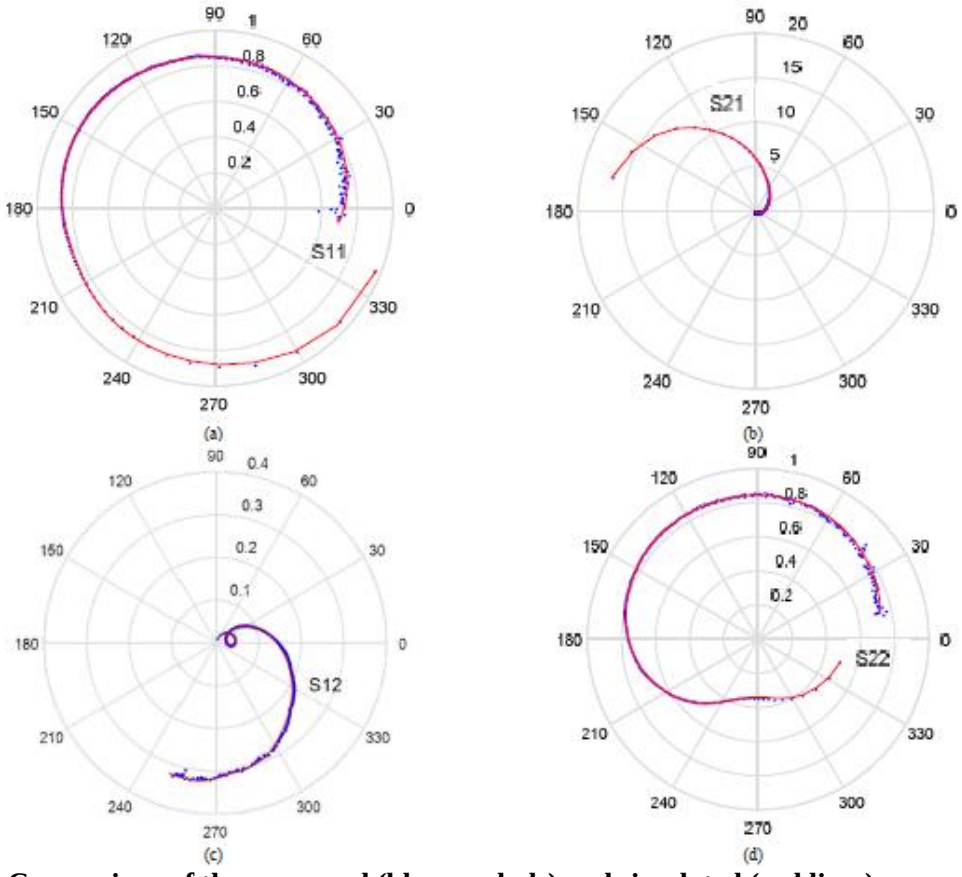

Fig. F1. Comparison of the measured (blue symbols) and simulated (red lines) s-parameters

G. Final implemented model in TCAD software [15]

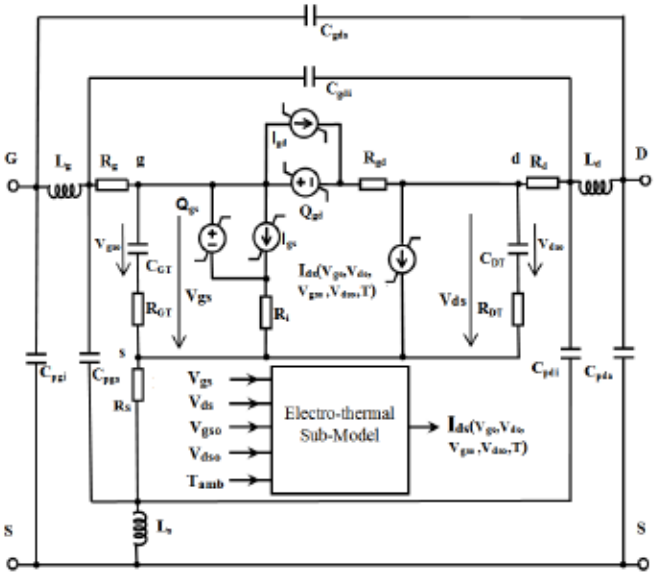

Fig. G1. Large-signal modelling of GaN HEMT

H. Final ADS implementation shown in [16]

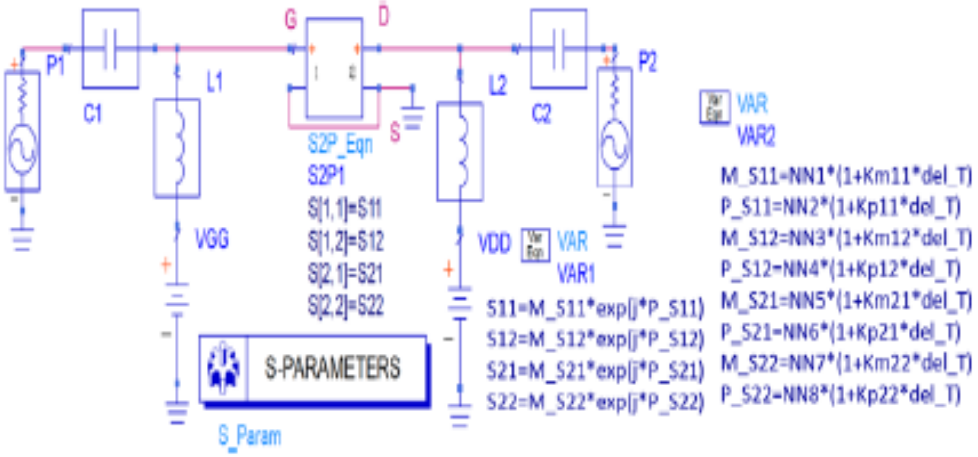

Fig. G1. Schematic of ADS implementation

Appendix I

The schematic of fabricated device [20]

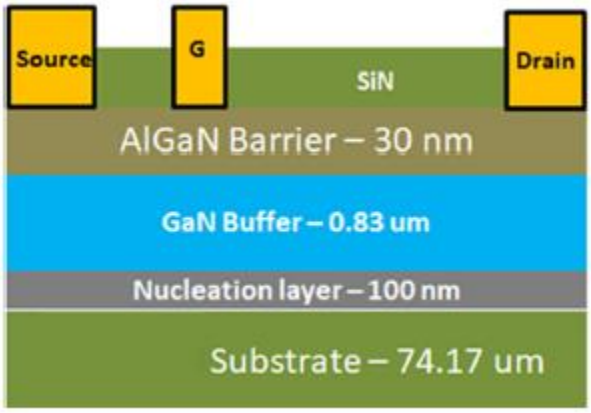

Fig. I1. Schematic of GaN HEMT

The mathematical derivation of Support Vector regression**.** Suppose the training examples is described as $\{\boldsymbol{x_i}, \boldsymbol{y_i}\}$, where *i* varies from $\mathbf{1}$ to $\boldsymbol{n}$, $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{d}}$ and $\boldsymbol{y_i} \in \mathbb{R}$. Then the fitting function is given in equation (3). Here, $\boldsymbol{f(x)}$ denotes the predicted value based on the optimal sets of weights and biases.

$$\boldsymbol{f(x) = \omega^T \varphi(x_i) + b} \qquad (1)$$

The input vectors x_i can be mapped into higher-dimensional space, $\varphi(x_i)$, where $\varphi(x_i)$ serves as a kernel trick to facilitate the easier solution of the non-linear problems by mapping them into higher dimensional space and performing the simple dot products. To address the training examples that lies beyond the ε-insestitive zone, slack variables ξ_i and ξ_i^* at each point can be introduced as the soft-margin regression. Therefore, error equation can be modified to equation (4).

$$\text{Minimize } \left(\frac{1}{2}\left|\left|\omega^2\right|\right| + C\sum_{i=1}^{n} L(\varepsilon)\right). \qquad (2)$$

$$L(\varepsilon) = \begin{cases} 0, & \text{if } |y - f(x)| < \varepsilon \\ |y - f(x)| - \varepsilon, & \text{otherwise.} \end{cases} \quad (3)$$

$$\text{Minimize } \left(\frac{1}{2} \left|| \omega^2 \right|| + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \right), \quad (4)$$

subject to the following constraints:

$$y_i - (\langle \omega, x_i \rangle + b) \leq \varepsilon + \xi_i$$

$$(\langle \omega, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i \geq 0$$

$$\xi_i^* \geq 0. \quad (5)$$

This primal optimization problem is solved by exploiting the Lagrange multipliers reproduced in equation (6)

$$L = \frac{1}{2} \left|| \omega^2 \right|| + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) - \sum_{m=1}^{n} (\eta_m \xi_i + \eta_m^* \xi_i^*)$$

$$+ \sum_{m=1}^{n} \alpha_m \left(f(x_m) - y_m - \varepsilon - \xi_m \right)$$

$$+ \sum_{m=1}^{n} \alpha_m^* (y_m - f(x_m) - \varepsilon - \xi_m^*) \quad (6)$$

where α_m, α_m^*, η_m and η_m^* are non-negative Lagrangian multipliers. Putting the partial derivatives of (8) with respect to primal variables (ω, b, ξ_i, ξ_i^*) to zero for optimality and substituting those values back in (7) yields the dual formula in (7). This method then needs to satisfy the complementarily Karush-Kuhn Tucker (KKT) conditions in equation (9). The optimal solution can then be found using Sequential Minimal Optimization (SMO) algorithm.

$$\text{Maximize } \left(\sum_{i=1}^{n} y_i (\alpha_i^* + \alpha_i) - \varepsilon(\alpha_i^* + \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i, x_j) \right)$$

(7)

So that

$$\sum_{i=1}^{n} y_i\,(\alpha_i - \alpha_i^*) = 0, \text{ where } \alpha_i,\ \alpha_i^* \in [0\ C]\ . \quad (8)$$

$$\alpha_i(\varepsilon + \xi_i - y_i + f(x_i)) = 0;\ \xi_i(C - \alpha_i) = 0 \quad (9)$$

$$\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + f(x_i)\right) = 0;\ \xi_i^*\ (C - \alpha_i^*) = 0;\ \forall i \quad (10)$$

$$\alpha_i\alpha_i^* = 0;\ \eta_i\eta_i^* = 0. \quad (11)$$

So, the predicted value can be expressed as:

$$f(x) = \sum_{j=1}^{n}(\alpha_j^* - \alpha_j)\ K(x, x_j) + b, \quad (12)$$

Back Cover