Received 26 July 2023; revised 19 September 2023; accepted 9 October 2023. Date of publication 12 October 2023; date of current version 26 October 2023.
The review of this article was arranged by Editor J. Wang.

Digital Object Identifier 10.1109/JEDS.2023.3324084

Comprehensive Investigation of ANN Algorithms Implemented in MATLAB, Python, and R for Small-Signal Behavioral Modeling of GaN HEMTs

SADDAM HUSAIN [1], BAGYLAN KADIRBAY [1], ANWAR JARNDAL [2] (Senior Member, IEEE), AND MOHAMMAD HASHMI [1] (Senior Member, IEEE)

1 School of Engineering and Digital Sciences, Nazarbayev University, 010000 Astana, Kazakhstan
2 Department of Electrical Engineering, University of Sharjah, Sharjah, UAE

CORRESPONDING AUTHOR: M. HASHMI (e-mail: mohammad.hashmi@nu.edu.kz)

This work was supported by Nazarbayev University through CRP under Grant 021220CRP0222 and through FDRG under Grant 20122022FD4113.

ABSTRACT Artificial Neural Network (ANN) is frequently utilized for the development of behavioral models of Gallium Nitride (GaN) High Electron Mobility Transistors (HEMTs). However, exhaustive investigation concerning the ANN algorithms implemented in major programming platforms for small-signal behavioral models of GaN HEMTs is generally not available. To fill this void, this paper carefully examines and evaluates ANN algorithms implemented in MATLAB, Python and R software environments for the development of accurate and efficient GaN HEMTs modelling. At first, the ANN based models are developed using MATLAB, Python's major frameworks namely Keras, PyTorch and Scikit-learn, and R's ANN framework namely H2O to model the GaN devices. Thereafter, an in-depth analysis is carried out to comprehend the usefulness of each framework in different application scenarios. At last, a detailed evaluation of the developed models in terms of generalization capability, training and prediction speed, seamless integration with the standard circuit design tool advanced design system, and of the development environments in respect of support and documentation, user-friendly interface, ease of model development, open-access and cost is carried out.

INDEX TERMS ANN, device modelling, GaN HEMTs, MATLAB, Python and R.

I. INTRODUCTION

Gallium Nitride (GaN) High Electronc Mobility Transistor (HEMT) has emerged as the main transistor device for the design of high frequency Power Amplifiers (PAs) [1], [2], [3], [4]. This is due to the exciting features such as high bandwidth, high power density, high breakdown electric field, and high electron saturation carrier velocity of GaN devices [5]. Furthermore, design of PAs require availability of design flow and accurate GaN HEMT models [6], [7]. Importantly, models which are comfortably incorporated into Computer-Aided Design (CAD) tools are vital [8]. The Equivalent Circuit (EC) and the behavioral modeling techniques are the most frequently used for the development of device models [9], [10], [11]. The EC methods extract the parameters based on the measurement set, nevertheless, keeping the correspondence with the physics of the device alive [12]. The EC techniques are faster in simulation—establishing them as the standard choice of academic and industrial research [13]. However, the accuracy of EC models significantly diminishes with the increase in frequency as the influence of the parasitic elements become noticeable. Therefore, they require more model elements in order to accurately mimic the behavior, that in turn makes the procedure inefficient and computationally expensive. The alternative, behavioral modeling overcomes most of the issues encountered in the EC based modeling and is therefore gaining acceptance from both the researchers and circuit designers [14].

In the last few years, some interesting work on small- and large-signal behavioral modeling, using Machine Learning (ML) techniques, of GaN HEMTs have been reported [15], [16], [17], [18]. These techniques have shown huge potential since they can explain linear, nonlinear and

dynamic behaviors if they are provided with adequate data. Furthermore, ML based models can be computationally inexpensive, scalable (if the input feature set also includes relevant scalable parameters such as transistor dimensions etc.), and exhibit extrapolative capabilities [19]. In spite of that, many features of ML algorithms are unexplored and require investigation. In fact, the selection of ML algorithms depend on a particular problem, complexity and distribution of the data, number of data samples, applied pre-processing techniques, class of ML algorithms, final design application, metrics to validate the models etc. [20]. Nevertheless, Artificial Neural Network (ANN) is extensively employed for the modelling of devices, operating at high frequency, to produce highly efficient behavioral models which manifest better interpolation and extrapolation abilities [19], [20], [21], [22], [23]. MATLAB, Python and R are frequently employed for implementation of ANN algorithms. However, they provide distinct implementation of ANN algorithms, unique training algorithms and optimizers and their corresponding parameters, unique way to process the data, and functional memory to run the algorithm. However, rarely, a generalized study is conducted to investigate the applicability, usefulness, shortcomings, complexity and simulation time of ANN algorithms implemented in MATLAB, Python and R for small-signal behavioral modelling of GaN HEMTs. This often cause confusion and unfamiliarity regarding the pros and cons of the available ANN algorithms in different software environments.

It is in this context that this paper provides a detailed investigation of the ANN algorithms implemented in MATLAB, Python, and R for small-signal behavioral modelling of GaN HEMTs. Firstly, it envisages to serve as a comprehensive resource for small-signal behavioral modelling of GaN HEMTs using ANN algorithms. Secondly, it paves the way for the researchers to make use of the most appropriate programming language for their application specific tasks. Thirdly, it stimulates conducting detailed investigation of other ML algorithms offered by these programming languages that in turn has the potential to advance the state-of-the-art modelling techniques. To achieve these stated goals, this paper systematically investigates ANN algorithms implemented in *MATLAB*, *Python* and *R* by developing Small-Signal Models (SSMs) of GaN HEMTs. Initially, MATLAB, Python's most commonly used ANN frameworks namely Keras, PyTorch, Scikit-learn, and R's ANN package H2O are explored for the development of SSMs of GaN HEMTs. Then, the developed models are evaluated for Average Mean Squared Error (MSE), Average Mean Absolution Error (MAE), and coefficient of determination (R^2) for both the training and testing sets. Finally, the developed models are examined for CAD compatibility, generalization capability and training and prediction speed, whilst the software environments are surveyed for support and documentation, user-friendly interface, ease of model development, open-access, and cost effectiveness.

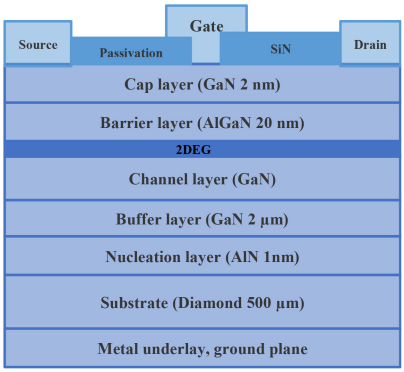

FIGURE 1. General epitaxial structure of GaN-on-Diamond HEMT [24].

In summary, the main contributions of this paper are: (i) a thorough and systematic approach to develop accurate and efficient SSMs using ANN for GaN HEMTs by making use of MATLAB, Keras, PyTorch, Scikit-learn, and H2O programming frameworks; (ii) determination of the respective topology using the standard approach; (iii) a thorough assessment of the developed SSMs on the grounds of MSE, MAE, R^2, CAD compatibility, generalization capability, and training and prediction speed and (iv) survey of the software environments with respect to support and documentation, user-friendly interface, ease in model development, open-access and cost. The next section includes the description of the GaN HEMTs and data processing procedures. Section III elucidates the model development framework. Section IV details the model validation. Finally, Section V provides a comprehensive discussion on the results followed by conclusion in Section VI.

II. DEVICES AND DATA PROCESSING PROCEDURE

This section elaborates on the devices, their characterization, and the frameworks of data preprocessing methodology.

A. DEVICE PHYSICS AND CHARACTERIZATION

A.1. GAN-ON-DIAMOND HEMT

The conventional epitaxial structure of GaN-on-Diamond HEMT, an active device is illustrated in Fig. 1. It is fabricated on 500 μm diamond substrate. Scanning from the bottom to the top, the epitaxial layers are constructed by making use of 1 nm AlN nucleation layer, 2 μm Fe-doped GaN buffer, 20 nm AlGaN barrier layer and 2 nm GaN cap layer. AlGaN barrier layer includes 0.3 nm Al-mole. The gate of this device is designed by E-beam lithography, and the gate-drain spacing (L_{gd}) and the respective gate-source spacing (L_{gs}) of this device are 2 μm and 1 μm. Furthermore, the gate-finger width (W_g), gate-length (L) and gate-number (N_g) are 125 μm, 0.25 μm and 4. Initially, this device was fabricated on epitaxial substrate (SiC) prior to the removal from original substrate and bonded onto a high thermal-conductivity chemical vapor deposition (CVD) polycrystalline diamond substrate [25]. The comprehensive details of

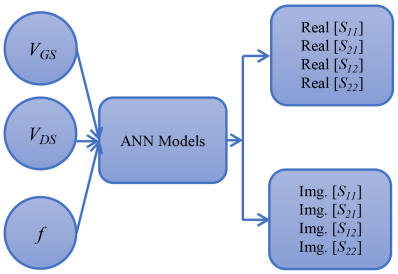

FIGURE 2. An inclusive layout of ANN based SSMs for GaN-on-Diamond HEMT.

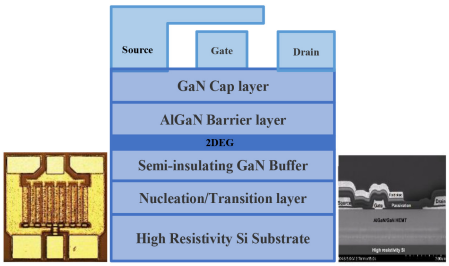

FIGURE 3. (Left to right) Photograph of studied 2-mm (10x200 μm) GaN-on-Si HEMT, corresponding general epitaxial structure and cross section SEM image of the implanted source field plate technology [19].

the physics and characterization of GaN-on-Diamond HEMT are reported in [24], [25], [26], [27].

The device is characterized using a N5245 Vector Network Analyzer (VNA) and outputs are recorded as real and imaginary S-parameters. The S-parameters measurement set-up includes two bias tees (one each at the input and the output), bias voltage source, RF wafer probe station, and a PC. Two-port calibration of the VNA is based on line-reflect-match technique using the calibration standard 104-783 from Cascade Microtech [27]. Overall, the measurement set consists of 36400 samples and a generic layout of the proposed modelling scheme for this device is depicted in Fig. 2. The theoretical and mathematical construct of ANN algorithm for device modeling purposes are explained in a number of earlier papers [19], [20], [21]. Here, the inputs to the device are gate to source voltage (V_{GS}), drain to source voltage (V_{DS}), and frequency (f). During characterization, V_{GS} is varied from –3 V to 0 V with a step size of 0.5 V, V_{DS} is swept from 0 V to 30 V with steps of 2.5 V, and frequency takes a wide range of 0.1 GHz to 40 GHz with a step size of 0.1 GHz.

A.2. GAN-ON-SI HEMT

In this work, GaN HEMT, 10 fingers of 200 μm each, grown on silicon (Si) substrate of 2 mm gate width is utilized. Fig. 3 shows the photograph of the coplanar waveguide (CPW) on-wafer device alongside its general epitaxial structure. It has been fabricated by Nitronex Corporation exploiting 0.5 μm NRF1 process and grown on Si [28]. The advance layer structure of this device includes a nucleation layer which diminishes the lattice-mismatch that results into low buffer trapping. An improved passivation procedure is implemented to attenuate the surface trapping effects and to enhance the RF characteristics. Furthermore, this device accommodates source field plate technology.

The measurement set-up to characterize this device is composed of N5242A VNA, dc power supplies, bias tees, and probe station. Before the measurement, a 2-port calibration of the VNA and the associated cables as a unit is carried out using the standard short-open-load-thru (SOLT) technique to get rid of the systematic errors and finally, outputs are collected in terms of the real and imaginary S-parameters [29].

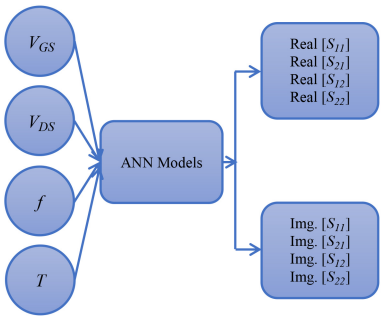

FIGURE 4. Layout of ANN based SSMs for GaN-on-Si HEMT.

TABLE 1. Ranges of V_{GS} and V_{DS} with the step size for GaN-on-Si HEMT.

V_{GS}(V)		V_{DS}(V)
Range	Step size	
-2 to 2	0.2	7
-2 to -0.4	0.1	28
-2 to -0.8	0.1	48

The measurement is performed over a wide range of temperatures from 25°C to 175°C with a step size of 25°C after mounting the device on temperature controlled thermal chuck. The device's internal temperature is calculated as the aggregate of the device's self heating and temperature defined by the thermal chuck. In addition, the measurement was carried out over the frequency range of 0.1 GHz to 26 GHz.

The measurement data for this device comprises of 103057 samples and a generic layout of the proposed modelling scheme for this device is depicted in Fig. 4. The inputs are V_{GS}, V_{DS}, f, and temperature (T). The outputs are real and imaginary S-parameters. The distribution of the biasing applied to this device is given in Table 1.

B. DATA PROCESSING METHODOLOGY

This section discusses the frameworks for data processing methodology used in this paper. A simple flow chart to develop and test the models are depicted in Fig. 5. As already mentioned, here we develop the ANN based models using MATLAB, Python, and R software and evaluate them on various metrics. All steps including data preparation, training and testing sets, preprocessing steps, metrics to validate the models etc. are same except the models are defined

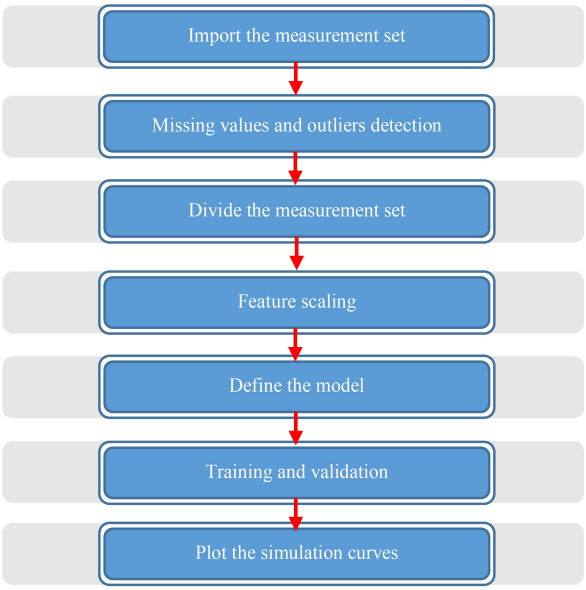

FIGURE 5. Data processing methodology utilized in this work.

and trained according to the respective software or library package. All the models are developed according to the same procedure outlined below:

- Firstly, the measurement set is imported
- Thereafter, a program is developed to clearly search for missing values and outliers
- The full measurement set is divided into training and testing sets
- To drive the ranges between the particular values, feature scaling is performed
- The models are defined according to the software or library protocol
- The models are trained on the training set and validated over the testing set
- Lastly, the simulation plots are obtained

Once the measurement set is imported it is pertinent to understand the overall distribution of the outputs. This analysis assists to comprehend the differences in ranges of the parameters and gives idea regarding any measurement anomaly. The distributions of the outputs are plotted for all the devices to look for the same (not shown here). We observed that the ranges for the parameters are different. Having different ranges for the parameters can be detrimental for the algorithm as orientation of the contour of its error function may forms a skewed or distorted elliptical shape. This drives the algorithm to converge at local minimums as opposed to global minimums. Similarly, the outliers or irregular peaks can affect the placement of the decision boundary of an algorithm and force the algorithm to set up a decision boundary, which is not the optimal. Because of above-mentioned reasons the removal of outliers and feature scaling are indispensable. Another aspect of the preprocessing stage is the division of the data. It is essential to find out the optimal division. It is achieved by trial-and-error analysis.

Thereafter, the models are defined, trained and validated. The validation and evaluation of the developed models are carried out in terms of MSE (1), MAE (2), R^2 (3) for both training and testing sets, time required in training and testing, overfitting, and simulation plots.

$$\text{MSE} = \frac{1}{n_{samples}} \sum_{k=1}^{n_{samples}} \left(y_k^{meas} - y_k^{pred} \right)^2 \tag{1}$$

$$\text{MAE} = \frac{1}{n_{samples}} \sum_{k=1}^{n_{samples}} |y_k^{meas} - y_k^{pred}| \tag{2}$$

$$R^2 = 1 - \frac{\sum_{k=1}^{n_{samples}} \left(y_k^{meas} - y_k^{pred} \right)^2}{\sum_{k=1}^{n_{samples}} \left(y_k^{meas} - \bar{y} \right)^2} \tag{3}$$

where $\bar{y} = \frac{1}{n_{samples}} \sum_{k=1}^{n_{samples}} y_k^{meas}$

B.1. DATA PROCESSING FOR GAN-ON-DIAMOND HEMT

A program is developed to actively search for missing values, outliers and uneven peaks. The outliers and uneven peaks are searched with respect to the mean. We observed that there are no missing values and very minimal outliers due to measurement anomalies. The dataset is divided according to V_{DS} as the following: the conditions where $V_{DS} = 0$, 2.5, 7.5, 10, 12.5, 17.5, 20, 22.5, 27.5 and 30V are met—those samples are included into the training set and remaining samples are incorporated into the testing set. Post division, the training and testing sets comprise of 28000 and 8400 samples, respectively. Feature scaling is done in such a way that all the parameters are brought between –1 to 1. Then the models are defined and trained in accordance with the software or the chosen library.

B.2. DATA PROCESSING FOR GAN-ON-SI HEMT

The same program is used to check the missing values and outliers. We observed even for this device there are no missing values and outliers can be ignored. For GaN-on-Si, the dataset is split using temperature as a reference parameter. The data samples which correspond to the temperature values 25, 50, 75, 125 and 150°C are added into the training set and remaining samples form the testing set. Finally, the training set contains 78997 training examples and testing set has 24060 samples. The range of the parameters are changed to –1 to 1. The same computation metrics (as explained before) are used to test the models and compute the generalization (accuracy on the testing set) of the developed models.

III. MODEL DEVELOPMENT FRAMEWORK

In this work, we developed ANN based models using deeper-layer topology. The rationale of using more hidden layers as opposed to one hidden layer, even-though one hidden layer with sufficient neurons can approximate any relationships, originates from the structure of the neurons and the way they learn the information. Furthermore, use of deeper neural networks minimize the adverse effects of feature extraction

and enable the neurons to fetch intrinsic nonlinear relationships. The downside of this approach is overfitting that needs to be tackled. The theoretical and mathematical construct of ANN are well studied and well established [19], [20], [21]. Therefore, this paper focuses on the practical implementation of ANN for modeling purposes. Nonetheless, (4)–(7), illustrates the analytical blueprints (for GaN-on-Si HEMT, however, same conditioning can be normalized for lesser or more inputs) of the outputs at different layers. The notations symbolize the following: k, m and n are the presumed number of nodes in first, second and third hidden layers, respectively; ActFcn indicates the activation function at each layer; $hl_i^{(2)}$, $hl_j^{(3)}$, $hl_l^{(4)}$ and S-parameters represent the outputs at each layer; IP stands for inputs; w_{i1}^1 to w_{i6}^1 are the weights joining the inputs to the first hidden layer; $w_{ji}^{(2)}$, $w_{lj}^{(3)}$ and $w_l^{(4)}$ denote the weights connecting other hidden layers; $b_i^{(2)}$, $b_j^{(3)}$, $b_l^{(4)}$ and $b_y^{(5)}$ are the biases at each layers.

$$hl_i^{(2)} = \sum_{i=1}^{k} \text{ActFcn}\Big(b_i^{(2)} + w_{i1}^{(1)} \times IP_1 + w_{i2}^{(1)} \times IP_2 + w_{i3}^{(1)} \times IP_3 + w_{i4}^{(1)} \times IP_4 \tag{4}$$

$$hl_j^{(3)} = \sum_{j=1}^{m} \text{ActFcn}\left(b_j^{(3)} + \sum_{i=1}^{k} w_{ji}^{(2)} \times hl_i^{(2)}\right) \tag{5}$$

$$hl_l^{(4)} = \sum_{l=1}^{n} \text{ActFcn}\left(b_l^{(4)} + \sum_{j=1}^{m} w_{lj}^{(3)} \times hl_j^{(3)}\right) \tag{6}$$

$$\text{S-parameters} = b_y^{(5)} + \sum_{l=1}^{n} w_l^{(4)} \times hl_l^{(4)} \tag{7}$$

A. COMPOSITION OF ANN BASED MODELS DEVELOPED IN MATLAB

One of the key applications of MATLAB is found in ML. It embodies tool-boxes for the real world ML problems which makes the analysis straightforward. It simplifies the entire procedure of importing the data, analysis of the data, development of the algorithms and models, and deployment of the models to the real world applications.

A.1. MODELS FOR GAN-ON-DIAMOND HEMT

Preprocessed data are exploited to obtain the topology of the models. In order to determine the topology, we exploited the standard trial-and-error method for each S-parameter separately. Initially, we built a small size model and examined the error behaviors (by analysing the error behaviors for the training and testing sets). Then, we increased the size of the model (i.e., by increasing the number of hidden layers and related neurons etc.) and again examined the error behaviors. This process in repeated iteratively until the model overfitted (here, model is set to be overfitted when the training behavior is very good, however, testing behavior is very poor). Once we acquired the topology that overfits, we gradually decreased the size to achieve the best fitting topology for the studied device. In parallel, we also analyzed distinct combinations of activation functions and other relevant parameters. After the analysis, the optimized number of hidden layers and nodes are given below. Furthermore, we observed, *tan-sigmoid* activation functions at the junctions of hidden layers and *pure-linear* at the output layer rendered the most satisfactory results.

- Model's architecture for real [S_{11}]: 3-6-6-6-1 [Number of input nodes (V_{GS}, V_{DS} and f)-neurons in the first hidden layer-neurons in the second hidden layer- neurons in the third hidden layer-number of output nodes]
- Model's architecture for imaginary [S_{11}]: 3-6-6-6-1
- Model's architecture for real [S_{21}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{21}]: 3-6-6-6-1
- Model's architecture for real [S_{12}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{12}]: 3-6-6-6-1
- Model's architecture for real [S_{22}]: 3-8-8-8-1
- Model's architecture for imaginary [S_{22}]: 3-8-8-8-1

A.2. MODELS FOR GAN-ON-SI HEMT

Once again, the similar approach as explained above (see Section III-A1) is adopted to get the best fitting topology of the ANN based models for the GaN-on-Si HEMT device. Furthermore, we observed that the *tan-sigmoid* at hidden layers and *pure-linear* activation functions are perfect choice for the modelling of this device. The implemented model's architecture is given below:

- Model's architecture for real [S_{11}]: 4-8-8-8-1 (input nodes are: (V_{GS}, V_{DS}, f and T)
- Model's architecture for imaginary [S_{11}]: 4-8-8-8-1
- Model's architecture for real [S_{21}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{21}]: 4-8-8-8-1
- Model's architecture for real [S_{12}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{12}]: 4-8-8-8-1
- Model's architecture for real [S_{22}]: 4-10-10-10-1
- Model's architecture for imaginary [S_{22}]: 4-10-10-10-1

B. COMPOSITION OF ANN BASED MODELS DEVELOPED IN PYTHON

Python is a high-level, object-oriented and user-friendly programming language. It embodies a plethora of free, open-source, readable, easily implementable, and easily integrable libraries. Furthermore, the clarity and user-friendlier nature of the syntax makes Python one of the most popular choices for the implementation of ANN algorithms. Out of the many Python libraries, we have used three of the most robust and popular choices namely Keras, PyTorch and Scikit-Learn for the modelling of the devices.

Keras, running on top of TensorFlow, is reckoned as one of the most powerful deep learning Application Programming Interface (API). It is utilized to work out a large pool of problems including regression-based and classification-based problems. Keras opens the horizons for finding the equilibrium of building simple, flexible and powerful ANN models. More details about Keras library can be found here [30].

PyTorch is an open-source ML library, primarily developed and managed by Facebook's AI research team. It is mainly used for the development of ANN models for the applications like image processing, regression problems, classifications problems, language processing etc. It is favoured and widely used by AI researchers and ML communities because of its ease-of-use, flexibility and compatibility with Python. It is comparatively fast since it supports CPU, GPU, and parallel processing. Further details about the library can be accessed through [31], [32].

Scikit-learn is known as one of the most comprehensive and robust library for ML problems. It comprises a range of supervised and unsupervised algorithms. It supports powerful third-party libraries like Numerical Python (NumPy), Scientific Python (SciPy), pandas, matplotlib etc. It is principally written in Python programming language. The syntax used to define the models are simple and easy to develop. Details can be found here [33].

B.1. MODELS FOR GAN-ON-DIAMOND HEMT

We used the same topology and other relevant parameters (such as hidden layers, nodes, activation functions etc.) to develop the ANN based models in Keras, PyTorch and Scikit-learn as of models developed in MATLAB. The utilized architecture is given below:

- Model's architecture for real [S_{11}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{11}]: 3-6-6-6-1
- Model's architecture for real [S_{21}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{21}]: 3-6-6-6-1
- Model's architecture for real [S_{12}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{12}]: 3-6-6-6-1
- Model's architecture for real [S_{22}]: 3-8-8-8-1
- Model's architecture for imaginary [S_{22}]: 3-8-8-8-1

B.2. MODELS FOR GAN-ON-SI HEMT

Models are developed in Keras, PyTorch and Scikit-learn using the same topology and parameters equivalent to models developed in MATLAB.

- Model's architecture for real [S_{11}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{11}]: 4-8-8-8-1
- Model's architecture for real [S_{21}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{21}]: 4-8-8-8-1
- Model's architecture for real [S_{12}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{12}]: 4-8-8-8-1
- Model's architecture for real [S_{22}]: 4-10-10-10-1
- Model's architecture for imaginary [S_{22}]: 4-10-10-10-1

C. COMPOSITION OF ANN BASED MODELS DEVELOPED IN R

R is an open-source programming language which can be applied for a wide-variety of statistical, data analysis and ML-based problems. R is supported and obtainable on Windows, Linux and macOS operating systems. R environment renders a conclusive data handling and storage amenity. R comprises of well-organised and well-supported packages which can be exploited to process daunting ML tasks. The packages are easily integrable and effortless to use. For this work, we are using R for ANN based behavioral modelling purposes. ANN, in this work, is implemented in R through H2O package for the modelling of the GaN devices. H2O is a comprehensive, scalable and open-source ML platform. It is used to build tree-based algorithms (such as random forests, gradient boosting machines etc.), generalized addictive models, k-means, ensemble methods, Naive Bayes, ANNs and many more. Furthermore, it is an in-memory platform, meaning the data to be used are loaded into the main memory (RAM) that makes the training and other operations faster. Furthermore, H2O grants the flexibility to define multiple hidden layers, with many nodes, activation functions at each layer and bunch of the training algorithms. But, the backend H2O Java engine is required for launching H2O it in R. A comprehensive and detailed knowledge about H2O package can be found in [34], [35].

C.1. MODELS FOR GAN-ON-DIAMOND HEMT

Akin to models developed in MATLAB and Python, the same topology and relevant parameters are utilized to develop ANN based SSM in R-H2O as shown below:

- Model's architecture for real [S_{11}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{11}]: 3-6-6-6-1
- Model's architecture for real [S_{21}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{21}]: 3-6-6-6-1
- Model's architecture for real [S_{12}]: 3-6-6-6-1
- Model's architecture for imaginary [S_{12}]: 3-6-6-6-1
- Model's architecture for real [S_{22}]: 3-8-8-8-1
- Model's architecture for imaginary [S_{22}]: 3-8-8-8-1

C.2. MODELS FOR GAN-ON-SI HEMT

Models are developed in R-H2O using the same topology and parameters equivalent to models developed in MATLAB and Python.

- Model's architecture for real [S_{11}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{11}]: 4-8-8-8-1
- Model's architecture for real [S_{21}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{21}]: 4-8-8-8-1
- Model's architecture for real [S_{12}]: 4-8-8-8-1
- Model's architecture for imaginary [S_{12}]: 4-8-8-8-1
- Model's architecture for real [S_{22}]: 4-10-10-10-1
- Model's architecture for imaginary [S_{22}]: 4-10-10-10-1

IV. MODEL VALIDATION

This section demonstrates the models' training strategies, prediction abilities, simulation plots at distinct inputs, and the time required for training and prediction.

A. ANN BASED MODELS DEVELOPED IN MATLAB

The training of the models involve numerous parameters. But, once the network is configured the very next step

TABLE 2. Evaluation of ANN based models developed in MATLAB for GaN-on-Diamond HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	3.71e-6	4.85e-6	1.01e-4	1.21e-4	1.02e-6	1.12e-6	1.37e-5	1.22e-5
MAE	All bias	1.53e-3	1.72e-3	5.51e-3	6.82e-3	5.12e-4	8.21e-4	2.66e-3	1.15e-3
% R-Squared	All bias	99.99	99.99	99.98	99.98	99.97	99.97	99.99	99.99
Training time (s)		111.42	113.18	111.60	112.00	122.87	115.18	132.27	135.73

TABLE 3. Evaluation of ANN based models developed in MATLAB for GaN-on-Diamond HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	6.02e-6	8.54e-6	7.58e-4	2.73e-4	1.32e-6	1.24e-6	3.03e-5	3.55e-5
MAE	All bias	1.81e-3	2.12e-3	9.66e-3	8.34e-3	7.14e-4	8.55e-4	3.25e-3	4.12e-3
% R-Squared	All bias	99.98	99.98	99.96	99.97	99.96	99.96	99.98	99.98
Prediction time (s)		0.0138	0.0136	0.0134	0.0136	0.0139	0.0138	0.0155	0.0158

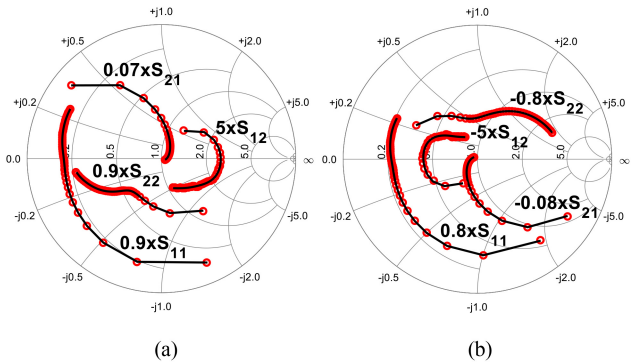

FIGURE 6. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Diamond HEMT at: (a) $V_{GS} = -1$ V and $V_{DS} = 12.5$ V and (b) $V_{GS} = 0$ V and $V_{DS} = 25$ V for the frequency range of 0.1 GHz to 40 GHz (ANN based models developed in MATLAB).

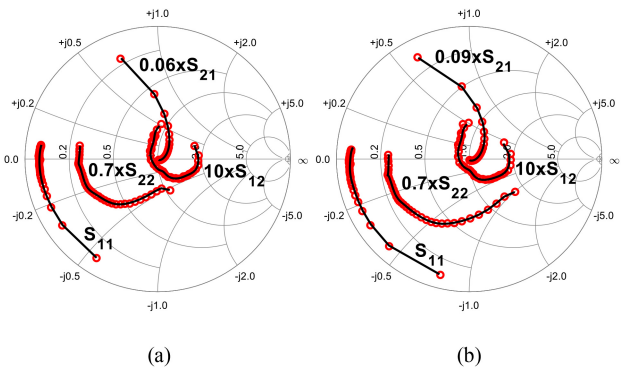

FIGURE 7. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Si HEMT at: (a) $V_{GS} = -0.7$ V, $V_{DS} = 28$ V and $T = 50°C$ and (b) $V_{GS} = -1.4$ V, $V_{DS} = 48$ V and $T = 100°C$ for the frequency range of 0.1 GHz to 26 GHz (ANN based models developed in MATLAB).

is the initialization of the weights and biases. The initial values of the weights are extremely crucial to determine the convergence of the final solutions and, therefore tuning for the right method of initialization is distinctly critical. In this context, the initial weights and biases are initialized between -1 to 1 for all the developed models in MATLAB. Moreover, Levenberg-Marquardt (LM) algorithm is utilized to train the models. Furthermore, as epoch is an important hyperparameter, which directly regulates the convergence behavior of the potential solutions, it is set to 2500 for the modelling of both devices. Similarly, maximum number of validation fails (max fails) is set to 2500, so that ANN based models could be trained for full training epochs.

As reported earlier, the models are examined in terms of average MSE, average MAE and R^2 (R^2 characterizes the regression analysis for each model) on the training and testing samples (generalization capability) separately. Furthermore, simulation curves are drawn for each device at distinct input conditions. The error profiles and simulation curves for GaN-on-Diamond HEMT and GaN-on-Si HEMT (for models developed in MATLAB) on the training and testing samples are presented through Tables 2-5 and Figs. 6-7. We can infer from the Tables and Figures that models developed in MATLAB efficiently and accurately simulate the behaviors of the GaN devices for the entire frequency of operation. The detailed discussion on the results are provided in Section V.

B. ANN BASED MODELS DEVELOPED IN PYTHON

B.1. GAN DEVICES: KERAS MODELS

The models in Keras are developed using Adam algorithm (ADAM) and the learning rate of ADAM is tuned between 1e-4 to 1. Similar to models developed in MATLAB, training epochs and loss function are set to 2500 and MSE, respectively for both devices. Likewise, the weights and biases are initialized between -1 to 1. In this work, the batch size is examined for the values of 16, 32, 48, 64 and 80, and it is found that 32 is the most efficient. Finally, the models developed in Keras are trained for both GaN devices and tested to probe the generalization capability of the models. Tables 6-9 and Figs. 8-9 present the results for both the devices. We can infer from the tabulated and plotted results that the models developed in Keras render excellent generalization capability for both GaN devices, and are able

TABLE 4. Evaluation of ANN based models developed in MATLAB for GaN-on-Si HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	1.48e-5	1.61e-5	1.85e-4	1.14e-4	1.01e-6	1.02e-6	1.15e-5	2.07e-5
MAE	All bias	2.84e-3	3.05e-3	6.05e-3	5.16e-3	5.74e-4	6.41e-4	2.63e-3	3.53e-3
% R-Squared	All bias	99.98	99.95	99.85	99.87	99.91	99.88	99.98	99.93
Training time (s)		416.60	406.13	398.26	403.62	405.85	404.45	544.85	566.00

TABLE 5. Evaluation of ANN based models developed in MATLAB for GaN-on-Si HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	1.36e-4	1.07e-4	4.23e-4	5.47e-4	5.41e-6	5.77e-6	1.42e-4	1.88e-4
MAE	All bias	5.85e-3	8.61e-3	6.78e-3	6.39e-3	8.23e-4	9.51e-4	4.33e-3	8.12e-3
% R-Squared	All bias	99.91	99.72	99.72	99.71	99.55	99.35	99.91	99.79
Prediction time (s)		0.0179	0.0198	0.0185	0.0188	0.0183	0.0185	0.0243	0.0241

TABLE 6. Evaluation of ANN based models developed in Python-Keras for GaN-on-Diamond HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	7.88e-5	6.23e-5	8.56e-6	6.01e-5	7.12e-5	6.09e-5	5.83e-5	4.48e-5
MAE	All bias	5.23e-3	4.93e-3	3.21e-3	3.35e-3	4.54e-3	3.11e-3	5.56e-3	5.21e-3
% R-Squared	All bias	99.98	99.97	99.98	99.98	99.95	99.95	99.99	99.98
Training time (s)		2466.96	2471.57	2456.56	2473.73	2434.79	2443.95	2501.23	2517.90

TABLE 7. Evaluation of ANN based models developed in Python-Keras for GaN-on-Diamond HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	8.26e-5	7.55e-5	5.24e-5	6.46e-5	7.36e-5	6.25e-5	2.59e-4	3.68e-4
MAE	All bias	5.69e-3	5.12e-3	3.43e-3	3.77e-3	4.97e-3	3.25e-3	6.04e-3	6.61e-3
% R-Squared	All bias	99.97	99.96	99.94	99.97	99.94	99.94	99.94	99.93
Prediction time (s)		0.3516	0.3692	0.3625	0.3685	0.3524	0.3578	0.4102	0.4164

TABLE 8. Evaluation of ANN based models developed in Python-Keras for GaN-on-Si HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	3.22e-5	3.31e-5	6.43e-5	1.66e-5	3.16e-5	8.24e-5	3.43e-5	3.98e-5
MAE	All bias	3.12e-3	4.65e-3	1.14e-3	2.06e-3	3.53e-3	6.08e-3	4.87e-3	4.74e-3
% R-Squared	All bias	99.91	99.89	99.81	99.87	99.72	99.74	99.96	99.95
Training time (s)		7163.89	7169.77	7014.31	7048.69	7109.89	7137.66	7206.99	7218.91

TABLE 9. Evaluation of ANN based models developed in Python-Keras for GaN-on-Si HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	4.69e-5	5.52e-5	1.18e-4	4.57e-4	6.13e-5	1.01e-4	4.52e-4	1.35e-4
MAE	All bias	3.37e-3	6.16e-3	6.43e-3	5.27e-3	6.23e-3	8.82e-3	6.61e-3	7.48e-3
% R-Squared	All bias	99.89	99.79	99.58	99.61	99.51	99.29	99.89	99.81
Prediction time (s)		1.2775	1.2009	1.3178	1.3365	1.3089	1.3536	1.4059	1.4125

to accurately mimic the measurement data behavior (see Section V for more details).

B.2. GAN DEVICES: PYTORCH MODELS

The models are formulated using the standard protocols of PyTorch library. Prior to the training, the inputs and outputs are converted to tensors. The same optimizer (ADAM), loss function (MSE) and weights initialization method (-1 to 1) are used for the development of models in PyTorch. Similar to MATLAB and Keras, here as well training epochs is set to 2500. The results of the models developed in PyTorch for GaN devices are tabulated in Tables 10-13. From the Tables, we can notice that the models developed in PyTroch have shown competitive results as compared all other developed models for both devices (see Section V for more details).

TABLE 10. Evaluation of ANN based models developed in Python-PyTorch for GaN-on-Diamond HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	8.52e-5	1.06e-4	8.15e-6	4.92e-5	3.21e-5	5.33e-5	1.05e-4	3.55e-5
MAE	All bias	6.95e-3	6.01e-3	1.61e-3	4.18e-3	4.19e-3	5.45e-3	7.08e-3	4.14e-3
% R-Squared	All bias	99.93	99.93	99.93	99.91	99.83	99.91	99.93	99.95
Training time (s)		7.046	7.416	7.455	7.465	7.055	7.385	7.515	7.505

TABLE 11. Evaluation of ANN based models developed in Python-PyTorch for GaN-on-Diamond HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	8.63e-5	1.29e-4	2.85e-5	2.42e-4	3.69e-5	7.29e-5	2.13e-4	3.36e-4
MAE	All bias	7.13e-3	6.54e-3	3.15e-3	5.74e-3	4.48e-3	7.67e-3	8.42e-3	6.73e-3
% R-Squared	All bias	99.92	99.91	99.89	99.48	99.82	99.90	99.89	99.91
Prediction time (s)		0.0017	0.0017	0.0016	0.0016	0.0016	0.0016	0.0019	0.0019

TABLE 12. Evaluation of ANN based models developed in Python-PyTorch for GaN-on-Si HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	8.87e-5	8.39e-5	1.48e-5	4.29e-5	3.25e-5	1.89e-4	2.57e-4	1.21e-4
MAE	All bias	5.65e-3	5.23e-3	1.36e-3	2.26e-3	2.14e-3	9.99e-3	8.47e-3	8.49e-3
% R-Squared	All bias	99.88	99.88	99.62	99.66	99.67	99.45	99.86	99.77
Training time (s)		30.57	31.05	31.28	30.64	30.87	30.58	37.09	37.13

TABLE 13. Evaluation of ANN based models developed in Python-PyTorch for GaN-on-Si HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	4.24e-4	3.35e-4	9.91e-5	9.51e-5	9.46e-5	3.89e-4	3.74e-4	3.41e-4
MAE	All bias	7.24e-3	9.51e-3	7.54e-3	8.26e-3	5.03e-3	1.13e-2	9.29e-3	9.59e-3
% R-Squared	All bias	99.72	99.65	99.55	99.59	99.48	99.15	99.83	99.73
Prediction time (s)		0.0025	0.0026	0.0027	0.0025	0.0024	0.0026	0.0029	0.0029

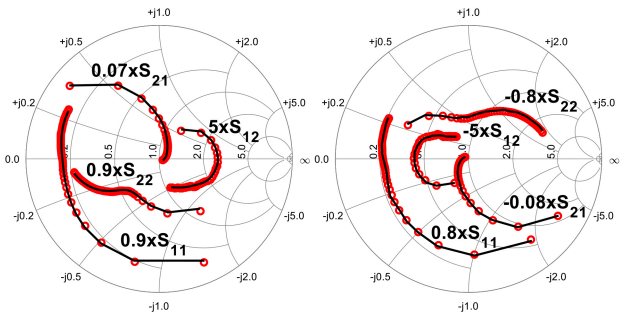

FIGURE 8. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Diamond HEMT at: (a) $V_{GS} = -1$ V and $V_{DS} = 12.5$ V and (b) $V_{GS} = 0$ V and $V_{DS} = 25$ V for the frequency range of 0.1 GHz to 40 GHz (ANN based models developed in Python-Keras).

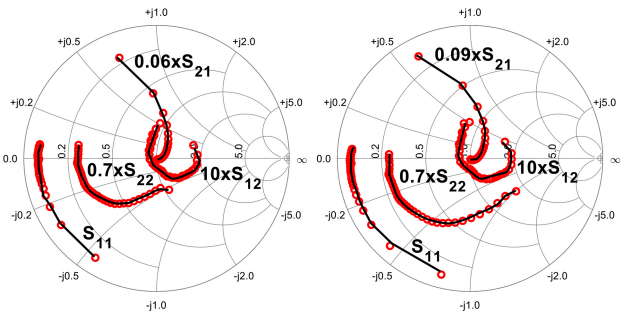

FIGURE 9. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Si HEMT at: (a) $V_{GS} = -0.7$ V, $V_{DS} = 28$ V and $T = 50°C$ and (b) $V_{GS} = -1.4$ V, $V_{DS} = 48$ V and $T = 100°C$ for the frequency range of 0.1 GHz to 26 GHz (ANN based models developed in Python-Keras).

B.3. GAN DEVICES: SCIKIT-LEARN MODELS

The models are developed as per the standard protocols laid down by the Scikit-learn library. Here, we used the same optimizer (ADAM) as of Keras and PyTorch. We exploited the Multi-Layer Perceptron Regressor (MLPR) class in Scikit-learn library to define ANN. Like the other developed models, epochs, max fails, and loss functions are set to 2500, 2500 and MSE, respectively. These setting are adopted for both GaN devices tested in this work. Once the models are trained and tested, the error tables are constructed for both the training and testing samples for both the devices. The results are given in Tables 14-17. We can deduce from the results that the performance of the models developed in Scikit-learn are almost analogous with

TABLE 14. Evaluation of ANN based models developed in Python-Scikit-learn for GaN-on-Diamond HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	8.15e-5	8.51e-5	4.11e-5	3.98e-5	6.21e-5	6.43e-5	8.24e-5	8.19e-5
MAE	All bias	5.42e-3	7.39e-3	2.93e-3	4.47e-3	5.46e-3	4.37e-3	6.68e-3	3.64e-3
% R-Squared	All bias	99.94	99.95	99.86	99.94	99.83	99.89	99.98	99.96
Training time (s)		138.04	135.53	129.66	135.24	132.83	127.62	148.90	149.52

TABLE 15. Evaluation of ANN based models developed in Python-Scikit-learn for GaN-on-Diamond HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	9.54e-5	8.91e-5	5.55e-5	4.38e-5	6.55e-5	7.72e-5	3.04e-4	4.58e-4
MAE	All bias	6.24e-3	7.45e-3	4.85e-3	4.81e-3	5.89e-3	5.13e-3	3.93e-3	4.86e-3
% R-Squared	All bias	99.92	99.95	99.79	99.93	99.81	99.86	99.90	99.91
Prediction time (s)		0.0059	0.0061	0.0060	0.0063	0.0059	0.0059	0.0074	0.0076

TABLE 16. Evaluation of ANN based models developed in Python-Scikit-learn for GaN-on-Si HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	7.84e-5	8.45e-5	9.26e-5	8.48e-5	4.41e-5	1.13e-4	6.27e-5	8.47e-5
MAE	All bias	4.93e-3	5.67e-3	5.81e-3	5.55e-3	3.66e-3	7.05e-3	5.96e-3	6.53e-3
% R-Squared	All bias	99.87	99.87	99.60	99.65	99.63	99.65	99.95	99.91
Training time (s)		408.41	413.38	410.50	412.07	408.96	411.17	448.24	449.34

TABLE 17. Evaluation of ANN based models developed in Python-Scikit-learn for GaN-on-Si HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	2.27e-4	2.55e-4	6.51e-4	4.98e-4	7.51e-5	3.31e-4	2.23e-4	3.25e-4
MAE	All bias	6.97e-3	9.83e-3	8.58e-3	7.68e-3	5.59e-3	8.84e-3	8.01e-3	9.55e-3
% R-Squared	All bias	99.71	99.67	99.53	99.59	99.47	99.19	99.85	99.72
Prediction time (s)		0.0183	0.0186	0.0183	0.0185	0.0191	0.0189	0.0215	0.0219

the performance of the models developed PyTorch for both GaN devices. A thorough discussions is provided in the next section.

C. ANN BASED MODELS DEVELOPED IN R

The H2O library is very promising for ML based modeling tasks as it provides computational efficiency since the data processing is done through RAM. Prior to the training, the predictors and predicted variables are transferred from R to H2O instance for superior speed. To develop the ANN based models for both GaN devices, here, the same ADAM optimizer, MSE loss function and weight initialization method (initial weights and biases are initialized between –1 to 1) are used. Moreover, training epochs are set to 2500 for both GaN devices. Finally, the training efficiency and generalization capabilities of the tested devices are recorded and presented in Tables 18-21. Furthermore, simulation plots are depicted in Figs. 10-11. It is evident from the results that the models developed in H2O have been able to competently replicate the behavior for both training and testing sets, and simultaneously like models developed in MATLAB and Python mimic the measurement data behavior of GaN devices which validates the robustness of the modelling schemes. Next section includes a thorough discussion on the advantage and disadvantage of H2O framework for the modelling of the chosen devices.

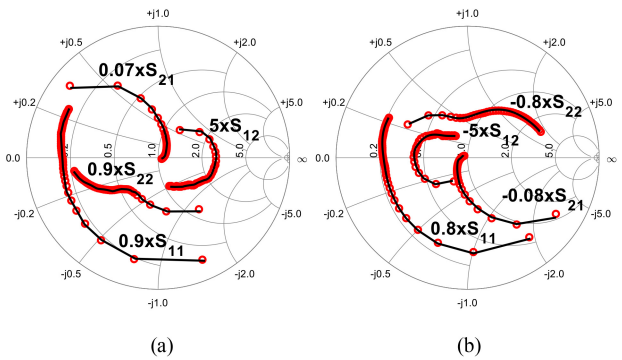

FIGURE 10. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Diamond HEMT at: (a) $V_{GS} = -1$ V and $V_{DS} = 12.5$ V and (b) $V_{GS} = 0$ V and $V_{DS} = 25$ V for the frequency range of 0.1 GHz to 40 GHz (ANN based models developed in R-H2O).

V. RESULTS AND DISCUSSION

This section analyzes the capability of ANN based models developed in MATLAB, Python and R for several metrics namely seamless integration with Advanced Design

TABLE 18. Evaluation of ANN based models developed in R-H2O for GaN-on-Diamond HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	5.01e-5	6.12e-5	2.01e-4	9.08e-5	7.43e-5	5.27e-5	1.03e-4	5.65e-5
MAE	All bias	3.91e-3	5.36e-3	3.87e-3	5.65e-3	2.56e-3	4.97e-3	6.48e-3	5.87e-3
% R-Squared	All bias	99.99	99.97	99.76	99.78	99.95	99.90	99.94	99.89
Training time (s)		83.64	87.17	82.38	80.17	82.54	87.21	124.33	122.12

TABLE 19. Evaluation of ANN based models developed in R-H2O for GaN-on-Diamond HEMT on the testing samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	5.49e-5	8.13e-5	2.15e-4	9.23e-5	7.79e-5	6.06e-5	1.51e-4	4.03e-4
MAE	All bias	4.09e-3	5.68e-3	4.03e-3	5.78e-3	2.92e-3	5.52e-3	7.58e-3	6.05e-3
% R-Squared	All bias	99.97	99.95	99.66	99.77	99.91	99.88	99.93	99.85
Prediction time (s)		0.1067	0.1161	0.1146	0.1007	0.1136	0.1057	0.1585	0.1565

TABLE 20. Evaluation of ANN based models developed in R-H2O for GaN-on-Si HEMT on the training samples.

		Real	Img.	Real	Img.	Real	Img.	Real	Img.
Metrics	Bias	S_{11}	S_{11}	S_{21}	S_{21}	S_{12}	S_{12}	S_{22}	S_{22}
MSE	All bias	7.73e-5	7.45e-5	3.23e-4	4.99e-5	6.01e-5	3.82e-4	3.35e-4	2.81e-4
MAE	All bias	6.21e-3	8.11e-3	2.01e-3	2.04e-3	5.35e-3	7.17e-3	8.23e-3	6.35e-3
% R-Squared	All bias	99.85	99.86	99.55	99.62	99.63	99.45	99.84	99.70
Training time (s)		106.19	108.01	107.04	109.82	109.15	110.79	135.68	134.57

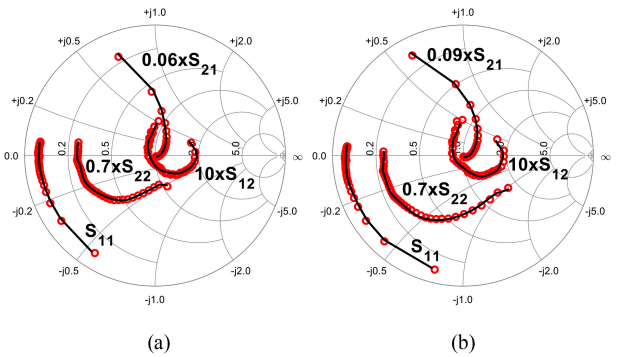

FIGURE 11. Measured (symbols) and simulated (lines) S-parameters for GaN-on-Si HEMT at: (a) $V_{GS} = -0.7$ V, $V_{DS} = 28$ V and $T = 50°C$ and (b) $V_{GS} = -1.4$ V, $V_{DS} = 48$ V and $T = 100°C$ for the frequency range of 0.1 GHz to 26 GHz (ANN based models developed in R-H2O).

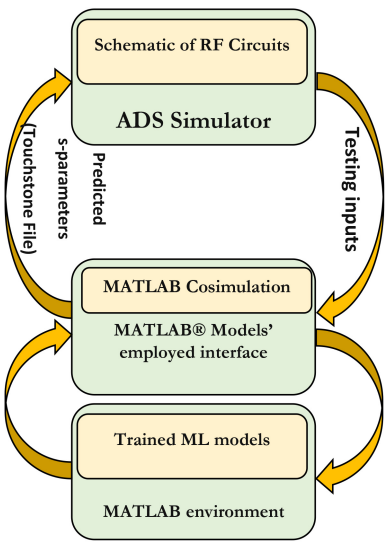

FIGURE 12. The block diagram depicting the interfacing of ML algorithms implemented in MATLAB with ADS.

System (ADS), generalization capability and training and prediction speed (see Table 22). Furthermore, it also discusses software environments related properties such as support and documentation, user-friendly interface, ease in the development of the models, open-access, and cost. Integrating the ANN based models with CAD tools, such as Keysight ADS, is one of the most fundamentals and vital stages of ANN-driven circuit design. Therefore, the programming languages which can be comfortably integrated with ADS is critical. In this context, ANN based models developed in MATLAB and Python have advantage over models developed in R. Models developed in MATLAB and Python can be seamlessly integrated with ADS as shown in Figs. 12 and 13. In essence, MATLAB has the ability to create an interface between MATLAB and ADS Ptolemy. The input-output function is executed with the aid of MATLAB blocks. In contrast, Python has recourse to ADS Datalink. In both cases, the trained ANN models are provided with the testing inputs through the interfacing mediums. A program will take those inputs, perform the predictions and outputs in the form of a touchstone file which is fed to ADS with the help of the interfacing mediums. However, all platforms, MATLAB, Python and R support a simple in-house procedure, where response of the testing condition can be directly print in a touchstone file and subsequently can be used in CAD tools. Moreover, (4)–(7) can

TABLE 21. Evaluation of ANN based models developed in R-H2O for GaN-on-Si HEMT on the testing samples.

Metrics	Bias	Real S_{11}	Img. S_{11}	Real S_{21}	Img. S_{21}	Real S_{12}	Img. S_{12}	Real S_{22}	Img. S_{22}
MSE	All bias	1.19e-4	1.63e-4	3.56e-4	5.34e-5	7.36e-5	7.25e-4	3.54e-4	4.14e-4
MAE	All bias	6.92e-3	8.85e-3	1.99e-3	2.39e-3	5.72e-3	7.31e-3	8.79e-3	7.58e-3
% R-Squared	All bias	99.76	99.64	99.50	99.46	99.51	99.13	99.78	99.56
Prediction time (s)		0.2034	0.2106	0.2174	0.2123	0.2015	0.2085	0.2645	0.2604

TABLE 22. Outcomes of the developed ANN based models for the small-signal modelling of GaN HEMTs, and survey of the different software environments.

Parameters	MATLAB (LM)	Python-Keras (ADAM)	Python-PyTorch (ADAM)	Python-Scikit-learn (ADAM)	R-H2O (ADAM)
ADS compatibility	MATLAB-ADS Ptolemy and In-house	ADS Datalink and In-house	ADS Datalink and In-house	ADS Datalink and In-house	In-house
Generalization-MSE (GaN-on-Diamond HEMT)	1.39e-4	1.29e-4	1.43e-4	1.48e-4	1.42e-4
Generalization-MSE (GaN-on-Si HEMT)	1.94e-4	1.78e-4	2.68e-4	3.23e-4	2.82e-4
Training speed (s) (GaN-on-Diamond HEMT)	119.28	2470.83	7.355	137.16	93.69
Training speed (s) (GaN-on-Si HEMT)	443.22	7133.76	32.401	420.25	115.15
Prediction speed (s) (GaN-on-Diamond HEMT)	0.0142	0.3736	0.0017	0.0064	0.1215
Prediction speed (s) (GaN-on-Si HEMT)	0.020	1.3267	0.0026	0.0194	0.2223
User-support	Available	Available	Available	Available	Available
Open source and cost	Closed-source and not free	Open-source and free	Open-source and free	Open-source and free	Open-source and free

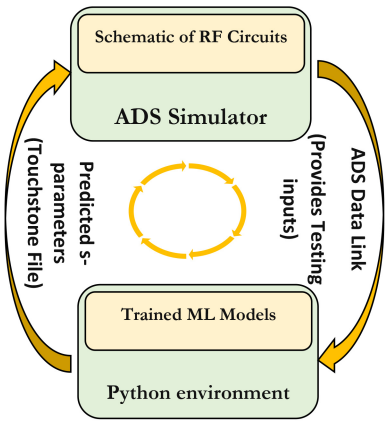

FIGURE 13. The block diagram depicting the interfacing of ML algorithms implemented in Python with ADS.

be used to compute the response (S-parameters) in CAD tools.

As mentioned earlier, the generalization capability refers to the models' accuracy on the unseen testing set. We computed averaged error over all S-parameters for each model. The outcomes are given below:

- MATLAB (GaN-on-Diamond HEMT)—1.39e-4
- MATLAB (GaN-on-Si HEMT)—1.94e-4
- Keras (GaN-on-Diamond HEMT)—1.29e-4
- Keras (GaN-on-Si HEMT)—1.78e-4
- PyTorch (GaN-on-Diamond HEMT)—1.43e-4
- PyTorch (GaN-on-Si HEMT)—2.68e-4
- Scikit-learn (GaN-on-Diamond HEMT)—1.48e-4
- Scikit-learn (GaN-on-Si HEMT)—3.23e-4
- H2O (GaN-on-Diamond HEMT)—1.42e-4
- H2O (GaN-on-Si HEMT)—2.82e-4

From this analysis, it is evident that the all the developed models in different software environments have shown excellent generalization ability.

Training and prediction speed, here, convey the time taken by the respective software environment to train on the training set and predict on the testing set, respectively. In addition, training speed denotes the time taken by the respective model to train using the optimal topology only. However, speed depends on the computer specification. The specification of the computer we are using to develop and test all the models are as follows: Processor-Intel Xeon W-2135 CPU @ 3.70GHz; Installed RAM-128 GB; Windows Edition-Windows 10 Enterprise; Version-21H2. To illustrate the training and prediction speed we computed the average time taken by each library or platform over all S-parameters (see Table 22). The results corresponding to the training speed are as follows:

- MATLAB (GaN-on-Diamond HEMT and GaN-on-Si HEMT)—119.28 s and 443.22 s
- Keras (GaN-on-Diamond HEMT and GaN-on-Si HEMT)—2470.83 s and 7133.76 s
- PyTorch (GaN-on-Diamond HEMT and GaN-on-Si HEMT)—7.355 s and 32.401 s
- Scikit-learn (GaN-on-Diamond HEMT and GaN-on-Si HEMT)—137.16 s and 420.25 s
- H2O (GaN-on-Diamond HEMT and GaN-on-Si HEMT)—93.69 s and 115.15 s

Here, it is imperative to state that the models developed in MATLAB are trained using LM algorithm whilst the models developed in Python (Keras, PyTorch and Scikit-learn) and R (H2O) are trained using ADAM optimizer. MATLAB provides excellent training speed owing to the standard mathematical implementation, vectorized and matrix operations, and highly optimized toolboxes and built-in functions whereas R (H2O), being an in-memory platform have shown very good training speed. Moreover, the models developed in Keras took the most time to train and predict for both GaN devices since it is a high level API and requires background processing.

MATLAB is built, supported and maintained by MathWorks. Therefore, it is not free and is a closed-source platform. MathWorks websites host user support and documentation for MATLAB programs and codes. There are online communities like MATLAB Answers that helps online questions. In contrast, Python is a free and an open source platform. Unlike MATLAB, Python has a number of highly efficient open source libraries to develop the ANN algorithms. Python's ANN libraries are well supported by leading corporate including Google and Facebook. The libraries have excellent user support and documentation. Akin to Python, R is also a free and open source software environment. It comprises of many packages which are freely accessible and available in the comprehensive R archive network (CRAN) and also provides documentation and user support. Nevertheless, for the GaN devices, all the platforms render enough contiguous memory block to store the data and process them effectively.

In our opinion, all platforms namely MATLAB, Python and R are equipped with user-friendly interfaces. However, visualization of the results and statistical analysis are easier in MATLAB and R as opposed to Python. Both ANN algorithms implemented in MATLAB and Python are easier to build owing to their simple syntax. However, MATLAB's ANN implementation is even simpler than Python because of the inbuilt functions and toolboxes. In general, R is a little difficult for the beginners. But with some experience, it can become easier to write the codes for ANN algorithms.

VI. CONCLUSION

This work investigated ANN algorithms implemented in MATLAB, Python and R programming languages for the small-signal modelling of GaN HEMTs. Then it rigorously examined the applicability, strengths and limitations of each programming framework to ascertain the suitability in distinct modelling settings. Initially, ANN based models in MATLAB, Python (Keras, Pytorch and Scikit-learn) and R (H2O) were developed to model the GaN HEMTs. Then, the developed models were assessed using MSE, MAE, R^2 and inferences were made. Finally, the developed models were also assessed for ADS compatibility, generalization capability, training and prediction speed, and software environments are surveyed for support and documentation, user-friendly interface, ease in the development of models, open-access and cost. We identified that the ANN based models developed in MATLAB (LM), Python (ADAM) and R (ADAM) provided accurate behavioral models for both GaN devices. It also come to notice that the models developed in MATLAB (LM), PyTorch (ADAM), Scikit-learn (ADAM) and H2O (ADAM) are time efficient. Moreover, we also found that the models developed in Keras (ADAM) is the least time efficient among all the tested models in this paper for both GaN devices.

REFERENCES

[1] P. M. Tomé, F. M. Barradas, L. C. Nunes, J. L. Gomes, T. R. Cunha, and J. C. Pedro, "Characterization, modeling, and compensation of the dynamic self-biasing behavior of GaN HEMT-based power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 69, no. 1, pp. 529–540, Jan. 2021.

[2] F. M. Ghannouchi and M. S. Hashmi, *Load-Pull Techniques With Applications to Power Amplifier Design*. Dordrecht, The Netherlands: Springer, 2012.

[3] J. B. King and T. J. Brazil, "Nonlinear electrothermal GaN HEMT model applied to high-efficiency power amplifier design," *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 1, pp. 444–454, Jan. 2013.

[4] G. Lv et al., "A fully integrated C-Band GaN MMIC doherty power amplifier with high efficiency and compact size for 5G application," *IEEE Access*, vol. 7, pp. 71665–71674, 2019.

[5] W. Hu and Y.-X. Guo, "An evolutionary multilayer perceptron-based large-signal model of GaN HEMTs including self-heating and trapping effects," *IEEE Trans. Microw. Theory Techn.*, vol. 70, no. 2, pp. 1146–1156, Feb. 2022.

[6] Y. Xu et al., "A scalable large-signal multiharmonic model of AlGaN/GaN HEMTs and its application in C-band high power amplifier MMIC," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 8, pp. 2836–2846, Aug. 2017.

[7] Z. Wen et al., "A quasi-physical compact large-signal model for AlGaN/GaN HEMTs," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 12, pp. 5113–5122, Dec. 2017.

[8] A. U. H. Pampori, S. A. Ahsan, and Y. S. Chauhan, "Modeling the impact of dynamic fin-width on the I–V, C–V and RF characteristics of GaN fin–HEMTs," *IEEE Trans. Electron Devices*, vol. 69, no. 5, pp. 2275–2281, May 2022.

[9] M. Q. Geng et al., "Modified small-signal behavioral model for GaN HEMTs based on support vector regression," *Int J. RF Microw. Comput. Aided Eng.*, vol. 31, no. 9, 2021, Art. no. e22774,

[10] F. Y. Huang, X. S. Tang, Z. N. Wei, L. H. Zhang, and N. Jiang, "An improved small-signal equivalent circuit for GaN high-electron mobility transistors," *IEEE Electron Device Lett.*, vol. 37, no. 11, pp. 1399–1402, Nov. 2016.

[11] G. Crupi et al., "High-frequency extraction of the extrinsic capacitances for GaN HEMT technology," *IEEE Microw. Wireless Compon. Lett.*, vol. 21, no. 8, pp. 445–447, Aug. 2011.

[12] Z. Wen et al., "An efficient parameter extraction method for GaN HEMT small-signal equivalent circuit model," *Int. J. Numer. Model.*, vol. 30, Jan./Feb. 2017, Art. no. e2127.

[13] S. Khandelwal et al., "ASM GaN: Industry standard model for GaN RF and power devices—Part 1: DC CV and RF model," *IEEE Trans. Electron Devices*, vol. 66, no. 1, pp. 80–86, Jan. 2019.
[14] A. Jarndal, S. Husain, M. Hashmi, and F. M. Ghannouchi, "Large-signal modeling of GaN HEMTs using hybrid GA-ANN, PSO-SVR, and GPR-based approaches," *IEEE J. Electron Devices Soc.*, vol. 9, pp. 195–208, 2021.
[15] W. Hu et al., "An accurate neural network-based consistent gate charge model for GaN HEMTs by refining intrinsic capacitances," *IEEE Trans. Microw. Theory Techn.*, vol. 69, no. 7, pp. 3208–3218, Jul. 2021.
[16] J. Cai et al., "Bayesian inference-based behavioral modeling technique for GaN HEMTs," *IEEE Trans. Microw. Theory Techn.*, vol. 67, no. 6, pp. 2291–2301, Jun. 2019.
[17] 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.
[18] J. Cai, J. King, C. Yu, J. Liu, and L. Sun, "Support vector regression-based behavioral modeling technique for RF power transistors," *IEEE Microw. Wireless Compon. Lett.*, vol. 28, no. 5, pp. 428–430, May 2018.
[19] A. Jarndal, S. Husain, and M. Hashmi, "Genetic algorithm initialized artificial neural network based temperature dependent small-signal modeling technique for GaN high electron mobility transistors," *Int. J. RF Microw. Comput.-Aided Eng.*, vol. 31, no. 3, 2021, Art. no. e22542
[20] S. Husain, M. Hashmi, and F. M. Ghannouchi, "Comprehensive investigation and comparative analysis of machine learning-based small-signal modelling techniques for GaN HEMTs," *IEEE J. Electron Devices Soc.*, vol. 10, pp. 1015–1032, 2022.
[21] Z. Marinković et al., "A review on the artificial neural network applications for small-signal modeling of microwave FETs," *Int. J. Numer. Model.*, vol. 33, no. 3, 2020, Art. no. e2668.
[22] A. Majumder, S. Chatterjee, S. Chatterjee, S. S. Chaudhari, and D. R. Poddar, "Optimization of small-signal model of GaN HEMT by using evolutionary algorithms," *IEEE Microw. Wireless Compon. Lett.*, vol. 27, no. 4, pp. 362–364, Apr. 2017.
[23] A. Jarndal, S. Husain, and M. Hashmi, "On temperature-dependent small-signal modelling of GaN HEMTs using artificial neural networks and support vector regression," *IET Microw. Antennas Propag.*, vol. 15, no. 8, pp. 937–953, 2021.
[24] A. Jarndal, X. Du, and Y. Xu, "Modelling of GaN high electron mobility transistor on diamond substrate," *IET Microw. Antennas Propag.*, vol. 15, no. 6, pp. 661–673, 2021.
[25] Q. Wu et al., "Performance comparison of GaN HEMTs on diamond and SiC substrates based on surface potential model," *ECS J. Solid-State Sci. Technol.*, vol. 6, no. 12, pp. Q171–Q178, 2017.
[26] Q. Wu et al., "A scalable multiharmonic surface-potential model of AlGaN/GaN HEMTs," *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 3, pp. 1192–1200, Mar. 2018.
[27] Y. Chen et al., "Temperature-dependent small signal performance of GaN-on-diamond HEMTs," *Int. J. Numer. Model.*, vol. 33, no. 3, 2020, Art. no. e2620.
[28] A. Edwards, B. Geller, and I. C. Kizilyalli, "Extracting a nonlinear electrothermal model for a GaN HFET," *Microw. J.*, vol. 51, no. 2, pp. 144–154, Mar. 2018.
[29] *Agilent N5242A PNA-X Microwave Network Analyzer: Operating and Service Manual*, Agilent Technol., Santa Clara, CA, USA, 2014.
[30] A. Gulli and S. Pal, *Deep learning with Keras*. Birmingham, U.K.: Packt Publ. Ltd., 2017.
[31] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[32] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. NIPS Workshop*, Long Beach, CA, USA, 2017, pp. 1–4.
[33] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.
[34] A. Candel, and E. LeDell. "Deep Learning with H2O." 2020. [Online]. Available: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/DeepLearningBooklet.pdf
[35] "H2O: Scalable machine learning platform." 2020. [Online]. Available: https://github.com/h2oai/h2o-3