IEEE Open Journal of the
Industrial Electronics Society

Received 5 July 2024; accepted 6 August 2024. Date of publication 8 August 2024;
date of current version 16 August 2024. The review of this article was arranged by Associate Editor María Dolores Valdés Peña.

Digital Object Identifier 10.1109/OJIES.2024.3440578

Explainable Model Prediction of Memristor

SRUTHI PALLATHUVALAPPIL (Graduate Student Member, IEEE), RAHUL KOTTAPPUZHACKAL, AND ALEX JAMES

School of Electronic Systems and Automation, Digital University Kerala, Trivandrum, Kerala 695317, India

CORRESPONDING AUTHOR: ALEX JAMES (e-mail: apj@ieee.org)

This work was supported by the university research grants of Ministry of Electronics and Information Technology, Government of India, and Department of IT and Electronics, Government of Kerala.

ABSTRACT System level simulation of neuro-memristive circuits under variability are complex and follow a black-box neural network approach. In realistic hardware, they are often difficult to cross-check for accuracy and reproducible results. The accurate memristor model prediction becomes critical to decipher the overall circuit function in a wide range of nonideal and practical conditions. In most neuro-memristive systems, crossbar configuration is essential for implementing multiply and accumulate calculations, that form the primary unit for neural network implementations. Predicting the specific memristor model that best fits the crossbar simulations to make it explainable is an open challenge that is solved in this article. As the size of the crossbar increases the cross-validation becomes even more challenging. This article proposes predicting the memristor device under test by automatically evaluating the *I–V* behavior using random forest and extreme gradient boosting algorithms. Starting with a single memristor model, the prediction approach is extended to memristor crossbar-based circuits explainable. The performance of both algorithms is analyzed based on precision, recall, f1-score, and support. The accuracy, macro average, and weighted average of both algorithms at different operational frequencies are explored.

INDEX TERMS Extreme gradient boost (XGBoost) predictor, memristor models, memristor crossbar, pinched hysteresis, random forest predictor.

I. INTRODUCTION

Memory resistors are a class of devices abbreviated as memristors [1]. It is the fourth basic circuit element that functionally relates the charge and linkage flux. Their properties differ from the other three fundamental devices by their nonvolatile memory effect, pinched hysteresis loop, scalability, programming capability, and compatibility with CMOS technology. It memorizes the latest attained conductance value even if the power supply is OFF. Due to these features, the application of memristors is wide in range, like in-memory computing, logic, neuromorphic computing, etc.

There are several models of memristors [2]. While designing the circuits, a mathematical model is used to show the behavior of the memristor [3], [4]. Compared to the behavior of physical devices, the model should be sufficiently accurate, simple, and computationally efficient. In addition, the model should be general so that it can be suitable for different technologies. The wide usage of different memristor models for circuit simulations makes them flexible for a wide range of applications. While using the models [5] in large circuits for high-end applications, it is challenging to cross validate.

Finding the efficient solution to several complex computational problems, evolving hardware neuromorphic computing architectures offer promising solutions. Memristors mimic synapses in neural network implementations, which change resistance state according to the applied voltage and memorize the latest attained resistance state. Like a matrix, the crossbar arrangement of memristors with selector transistors along rows and columns mimics weighted summation operations in neural network models. It offers different high-density architectures to implement the synaptic connections and neural network models. For this, the hardware circuit implementations based on different memristor models demand deciding the appropriate selection of these models to get maximum performance. In this proposed work, memristor models used in the circuit simulations within a black box are predicted using machine learning based on the dataset of pinched hysteresis.

This method has significant industrial applications in implementations of different neural network models, pattern recognition, in-memory computations, etc., by enabling identification of proper memristor models suitable for specific computations that helps to optimize the neuromorphic systems. The tasks like image classification, language processing, speech recognition, and robotics are flexible to be implemented with neuro memristive arrays with proper memristor models. The decision-making process while using large datasets in different fields like finance, health care, manufacturing industries, and edge computing applications using smart sensors and IoT can use this neuromorphic computing architectures. Choosing proper memristor models that can have high performance to specific applications is essential for having higher degree of accuracy in computations. This proposed approach provides proper cross-validation and prediction of memristor models so that the usage of those models with proper mapping to the application demands can be done.

This article focuses on explaining neuro-memristive circuits and systems through a cross-validation of the simulations, irrespective of the complexity of the model. Memristors represent a broad class of devices that can be modeled using a broad set of device models. This problem is very different to that of MOSFETs, where only one type or minor variant of the device is modeled. Over time, even if models are standardized for memristors, there will be a need to perform cross-validations as being a class of devices, several combinations of variability make system modeling complex and inaccurate. The conventional system of verifying the simulation results of emerging memristive devices that are yet to mature, using the experimental results, is replaced here to address the challenges associated with the accurate modeling due to different variability. Even though many models are emerging, the scientific community always needs to compromise for different properties associated with the physical devices like threshold voltage, state variable motion, area of hysteresis, different responses for different input stimuli, etc., due to the wide range of variability among this broad class of devices. This reverse engineering approach addresses the complexity of estimating the accuracy and functional behavior of physical characterization data due to the abovementioned issues. This proposed approach is helpful as more and more memristive devices are discovered.

A large majority of research using machine learning is for device modeling. However, this work diverges from this trend in applying machine learning in an entirely new application, where conventionally, cross-validation of simulation results is only done through experimental verification. This is an approach to cross validate the simulations using a machine learning approach.

The motivation of this work is to develop a technique for explaining neuro-memristive circuits and systems by validating circuit simulations done with emerging device models. Most memristor devices are difficult to model accurately due to device-to-device and cycle-to-cycle variability. Some examples of memristor models, their properties, device level, and simulation level challenges are illustrated in Table 1. Under such circumstances, the circuits built with idealistic models result in large output errors. Furthermore, as the devices have a range of variability, the experimental results are also difficult to conclude in estimating the desired functional behavior and accuracy of the design logic. This necessitates a simulation-based approach to cross-validate the functionality and accuracy of circuit designs with memristors.

Modeling memristive devices is significantly more complex than MOSFETs because memristors are devices with diverse material compositions, modes of operations, and structures. MOSFETs have relatively uniform structure and operation, making them feasible for standardized models like the Berkeley short-channel IGFET model. The memristor field is heterogeneous and includes various devices with different operation mechanisms. For example, the resistive random access memory, in which conductance states switch depending upon the formation and rupturing of conductive filament. Phase change memory (PCM), in which conductance states change according to change in state from crystalline to amorphous. Spin-transfer torque RAM in which the conductance state changes according to changes in the magnetic state due to the spin-transfer torque effect. The resistive switching and storage mechanisms are different for these devices. Hence, the behavior of memristive devices cannot be analyzed using a unified model. Detailed modeling approaches are needed to model this wide range of resistive switching mechanisms like ion migrations, magnetic effects, phase transitions, etc. For MOSFETS, the relationship between the electrical characteristics and the physical structure modeling approaches in CMOS devices is comparatively simple.

CMOS devices show linear behavior in their operational regions, whereas memristive devices are nonlinear devices that change their resistance based on the previous states or history of current and voltage. In amplifiers, the linear region operation of CMOS devices is used, and the memristors are used in applications like mimicking neural computations, neural synapses, etc, where nonlinear behaviors are advantageous. The CMOS behavior is described using three different equations corresponding to different regions of operations, whereas memristor models have state-dependent resistance values, and hence, the behavior is described using state-variable equations connecting the voltage–current relationship and internal state variable. Memristors are dynamic since their resistance is dependent on the previous state. Hence, the resistance will change over time based on the device's voltage and current history. Hence, memristors are dynamic in nature, whereas the CMOS devices exhibit stable and predictive behavior in circuit simulations. Leakage current effects also need to be considered in CMOS circuits.

Several mathematical models are needed for memristive circuit simulation and analysis. Cross-validation must be done across various device types and operating conditions that require complexity in the modeling process. Hence, this work emphasises the cross-validation of memristor models that continuously need to be refined and modified, which cannot be

TABLE 1. Properties and Challenges of Different Memristor Models

Memristor Models	Properties and Challenges
Joglekar and Biolek	There are a lot of discrepancies while comparing the results with the data obtained through physical characterization. When pulse-wave form is applied, in the positive regime, when conductivity increases, the size of the hysteresis loop increases, which is opposite to the trend seen in the characterization data.
Air Force Research Lab (AFRL) and metal-insulator-metal (MIM)	Correlate more closely to the characterization data, and there is a strong connection to the physical mechanisms within the device. For alternative voltage inputs, the results will not match the characterization data. Significant updation is needed with different device structures because these models are specific to a single fabricated device.
Hyperbolic Sine Models and the University of Michigan Model	The properties like the Schottky barrier at a metal-oxide interface and the diffusion of ions are modelled. Have a stronger correlation to physical memristor characterization data. These models describe the memristor functionalities in a more generalized and accurate way. However, they hardly correlate to the physical hardware.
Other generalized models	Less theoretical correlation to the physical mechanisms. Many models will not consider the threshold voltage of the physical memristor device. Unless the voltage across the memristor exceeds this threshold voltage, the hysteresis will not be seen. The state variable motion depends on the applied voltage's magnitude and polarity. This implies that the dynamics of Oxygen vacancy expansion and compression are different. Most models have the equivalent state variable motion in positive and negative directions, which is not true in actual cases.

unified like a more stable modeling environment of CMOS-based models.

Through this new approach, we propose that estimating the device model followed by using those models to build circuits can lead to better estimates in cross-validating the accuracy of circuit-level simulation results. Reverse engineering the model from circuit design using the proposed approach also leads to an efficient way to account for a wide range of device variability.

The programming mechanism also differs in different devices. Hence, proper programming methodology must be followed for different devices mapping neural weights. Predicting the models from the simulations will help develop the necessary programming strategies to tune the weights with minimum error, which will help reduce the relative current error and increase accuracy.

While doing the hardware implementation of neural networks, memristive crossbars emulate the weighted summation operations using the multiply and accumulate (MAC) operation carried in between input voltages to the crossbar and conductance values to which the memristors are programmed. The output current read through the columns of the crossbar is equal to the MAC result. Computations are performed within the memory array, enhancing speed, and energy efficiency. How precisely the weights are mapped to the conductance states will determine the accuracy. Minimum deviation from the target weight values should be ensured during mapping.

In edge computing applications in which the tasks performed are specific, the memristor models need to be familiar prior to use in those applications. Under different scenarios, selecting the memristive devices that are suitable for specific tasks helps optimize performance in different edge computing applications.

Prediction of models from simulations helps to improve the programming strategies, which differ among various memristive devices. This device-specific programming helps provide fine-tuning by properly mapping weights that will enhance neural network performance and offer accurate computations, reducing relative current errors.

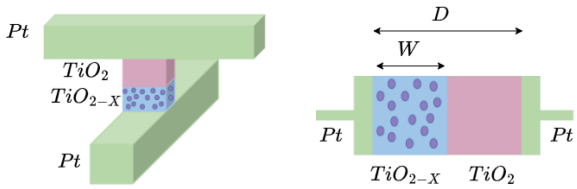

FIGURE 1. (a) HP ion drift model implementation illustrated in material physics. (b) Sandwich structure having two Pt electrodes, TiO_2 and TiO_{2-x}.

The rest of this article is organized as follows. Section II gives an introduction. Section II comprises the background of this article. Section III details the proposed model prediction approaches in the single memristor model and models in memristive crossbar arrays, followed by the analysis methodology and the results and discussion. Finally, Section IV concludes this article.

II. BACKGROUND

Most memristor models selected for this work are based on the memristor equations of the HP memristor model.

A. HP MEMRISTOR ION-DRIFT MODEL

The principle of resistance switching between two extreme values, Ron and Roff, the device's lowest and highest resistance, makes them mathematically flexible to model in different ways. The HP memristor model, an example of a metal–insulator–metal (MIM) device, is shown in Fig. 1.

When the Pt electrodes are excited externally, as shown in Fig. 2, the oxygen ions present in the doped region will drift to the undoped region under the influence of the electric field. This process will cause a shifting of the boundary between these two regions. This displacement in the boundary will cause a change in the resistance value also. If the structure is entirely covered by TiO_{2-x}, it is in its low-resistance state or maximum conductive (R_{on}). If

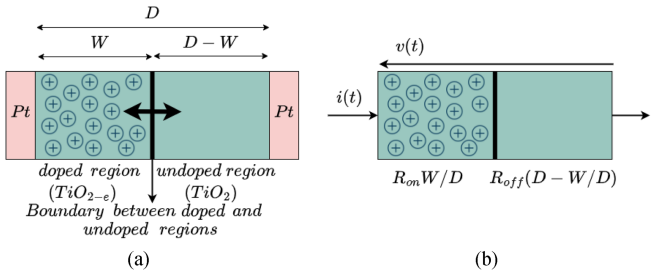

FIGURE 2. (a) Detailed device level structure of the HP memristor model. *W* is the width of the doped region, and *D* is the total device length. (b) Total resistance will be the effective resistance of the low-resistance region (doped region) and the high-resistance region (undoped region) .

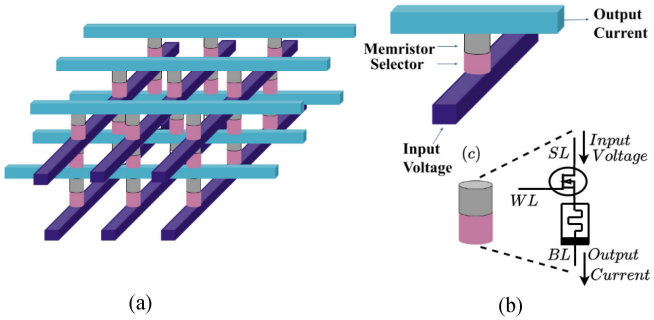

FIGURE 3. Memristor-crossbar architecture. (a) 3-D architecture of a memristive crossbar array obtained by stacking three layers of 2-D memristive crossbar array. (b) Memristor and selector pair. (c) Circuit connection showing the selector transistor and memristor in a crossbar array with WL (Word line), BL (Bit line), and SL (Source line); v_{in1}, v_{in2}, v_{in3}, and output currents i_1, i_2, i_3. Selector devices that need to be activated are applied with an input voltage greater than the threshold voltage. The Conductance of the memristor is denoted as g_{mn} for the *m*th row and *n*th column. The conductance of the selector transistor is given by g_T.

the structure is entirely covered by TiO_2, it is in its high-resistance state (R_{off}) or minimum conductive. The memristance is given by, $M(q) = R_{\text{on}}w(t)/D + R_{\text{off}}(1 - w(t)/D)$. The relation between the voltage and current is given by $V(t) = (R_{\text{on}}w(t)/D + R_{\text{off}}(1 - w(t)/D))i(t)$. Here w(t) is the width of the doped region, and D is the total width of the doped and undoped regions. The width w(t) is affected by i by, $dw/dt = \mu_v R_{\text{on}} i(t)/D$. Here μ_v is the dopant mobility. $w(t) = \mu_v R_{\text{on}} q(t)/D + w_0$. Here, q(t) is the charge injected in the time t.

The dw/dt is the dynamic state variable, the drift velocity of the Oxygen vacancies. The integration of the expression $\mu_v R_{\text{on}} q(t)/D$ gives the value of w(t). Even $q(t) = 0$, the integrated output will equal a constant. This implies that even if the current flow is zero, the charge is constant, and resistance remains unchanged. The principle of nonvolatility satisfies here. Based on the migration of ions, the value of w varies between 0 and D. The drifting of the boundary region is interpreted by different window functions and equations that give different mathematical models of memristors. While modeling different memristors, state variable equations are substituted with the equation $x(t) = W(t)/D$. The state variable becomes a normalized quantity whose value lies between 0 and 1. $x(t) = 0$ for the minor conductive state and $x(t) = 1$ for the most conductive state. Window functions limit the motion of the state variable between 0 and 1.

B. MEMRISTIVE CROSSBAR ARRAY

In a crossbar architecture [6], [7], memristors are arranged in a matrix form, as shown in Fig. 3. Each row and column intersection consists of a memristive device [8] and a selector CMOS. The figure shows a 3-D crossbar array in which 2-D crossbar arrays are stacked vertically. Word lines (WL) in a crossbar are used to activate or deactivate selectors. Voltage values above the threshold voltage of transistors are applied through word lines to activate selectors. Source lines are used to feed the input voltages, and bit lines are used to read output currents. The output currents are the results of the MAC operation [9] between the input voltage and conductance of the memristive device.

The equation of the MAC operation is given by

$$i_j = \sum_{k=1}^{j} v_k g_{k,j} \tag{1}$$

where i_j is the output current, v_k is the input voltage, and $g_{k,j}$ is the conductance of the crossbar node.

C. ENSEMBLE LEARNING–RANDOM FOREST AND EXTREME GRADIENT BOOSTING

Ensemble learning techniques combine different learning algorithms to make more accurate predictions. Predictions from individual learning models are aggregated to form the final prediction. These algorithms efficiently handle nonlinearity and interactions and provide feature-importance, flexibility, and robustness to overfitting. Since the data collected for this study is susceptible to overfitting the model and shows a nonlinear relationship, we are focusing on the following two ensemble learning techniques.

Random forest [10], [11] is an ensemble prediction algorithm having a combination of tree predictors. The majority vote of all individual trees determines the final prediction of the input vector. Each tree casts one vote for the most frequently occurring class. In random forest, the Gini index, which measures the degree of impurity of an attribute to different classes, is used as an attribute selection measure.

For a given training set T, selecting one case randomly and assigning to a class C_i, the Gini index is expressed by

$$\sum \sum_{j \neq i} (f(C_i, T)/|T|)(f(C_j, T)/|T|) \tag{2}$$

where $f(C_i, T)/|T|$ is the probability that the selected case belongs to class C_i. To reduce the complexity of the tree and to prevent overfitting, pruning is used [12]. It removes the branches of the tree that do not contribute to accuracy, and the remaining branches are grown to the maximum.

Algorithm 1: Pseudocode for XGBoost Classifier.

Training on the data

Training data (X_{train}, y_{train}), number of boosting rounds (`num_rounds`), maximum depth of each tree (`max_depth`), learning rate (`eta`), subsample ratio of training instances (`subsample`), and column subsample ratio of features (`colsample_bytree`) XGBoost model

Procedure:

Initialize model with a constant value: `model = initial_prediction_value;`

for round ← 1 `num_rounds` **do**

`gradients = -gradient_of_loss(`y_{train}`, model.predict(`X_{train}`))`

`weak_model = fit_weak_model(`X_{train}`, gradients, max_depth, colsample_bytree);`

`update = eta * weak_model.predict(`X_{train}`);`

`model = model + update;`

Return: XGBoost model (`model`);

Prediction using the trained model

XGBoost model, Test data (`X_test`) Predicted class labels for `X_test`

Procedure:

`predictions = model.predict(`X_{test}`)`

Return: `predictions;`

In this approach, input voltages and the corresponding output currents are the features. Based on this data, the random forest prediction algorithm splits nodes to generate new trees and identifies the respective models.

The extreme gradient boosting (XGBoost) [13] is a scalable and efficient application of the gradient boosting framework. A weight will be assigned for each observation. This weight will be adjusted after training the predictor. The weight of the correctly classified observations is decreased, and misclassified observations are increased. Using the observations with modified weights, the subsequent predictor is trained, and the process is repeated to create a highly accurate model. The sum of prediction score $f_k(X_i)$ of all trees gives the estimated output $\hat{y}_i$of the gradient boosting tree model

$$\hat{y}_i = \sum_{k=1}^{K} f_k(X_i),\ f_k \in \Gamma \tag{3}$$

where Γ is the space of the regression tree, K is the number of regression trees, and X_i is the features corresponding to sample I.

This approach proposes estimating the device model and using those models to build circuits to get better estimates in cross-validating the accuracy of circuit-level simulation results. Reverse engineering the model from circuit design also leads to an efficient way to account for a wide range of device variability. A workflow of the proposed approach is shown in the Fig. 4.

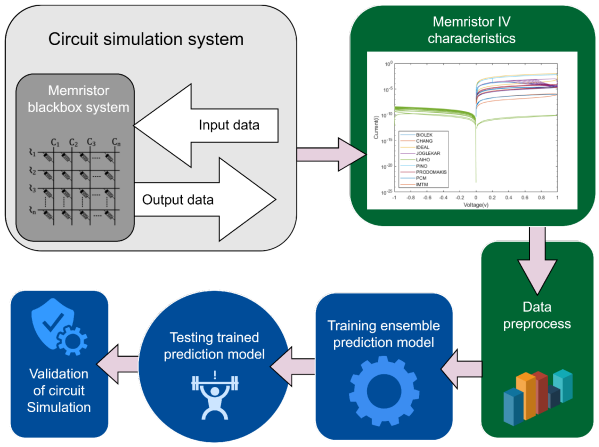

FIGURE 4. **Explainable neuro-memristive circuit system workflow.**

Here, nine memristor models are simulated in Spice. Each model is simulated with an input voltage of four different frequencies (0.5, 1, 5, and 10 Hz). The graph obtained by plotting the input voltage versus the logarithmic scale of output current gives nine different pinched hystereses, as shown in Fig. 5. This output data are collected for each model and prediction is performed using random forest and XGBoost techniques. For random forest, the parameters are trained with 100 trees for each dataset for the different frequencies applied. The prediction results for these nine models with four different frequencies of input voltages using random forest and XGBoost algorithms are analyzed by the factors precision, recall, f1-score, and support. The two prediction approaches, accuracy, macro average, and weighted average, are compared for different frequencies.

To calculate precision, recall, and f1-score, the following parameters are calculated from the confusion matrix [14], [15]. A confusion matrix is a tabular way of representing the performance of the prediction algorithm

- *True positive (*TP*):* Values predicted as positive and it is true.
- *True negative (*TN*):* Values predicted as negative and it is true.
- *False positive (*FP*):* Values predicted as positive and it is false.
- *False negative (*FN*):* Values predicted as negative and it is false.

Precision is found using the following equation:

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP})$$

Recall is calculated by the following equation:

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN}).$$

F1-score is measured using the equation

$$\text{F1-score} = (2*\text{Recall}*\text{Precision})/(\text{Recall}+\text{Precision}).$$

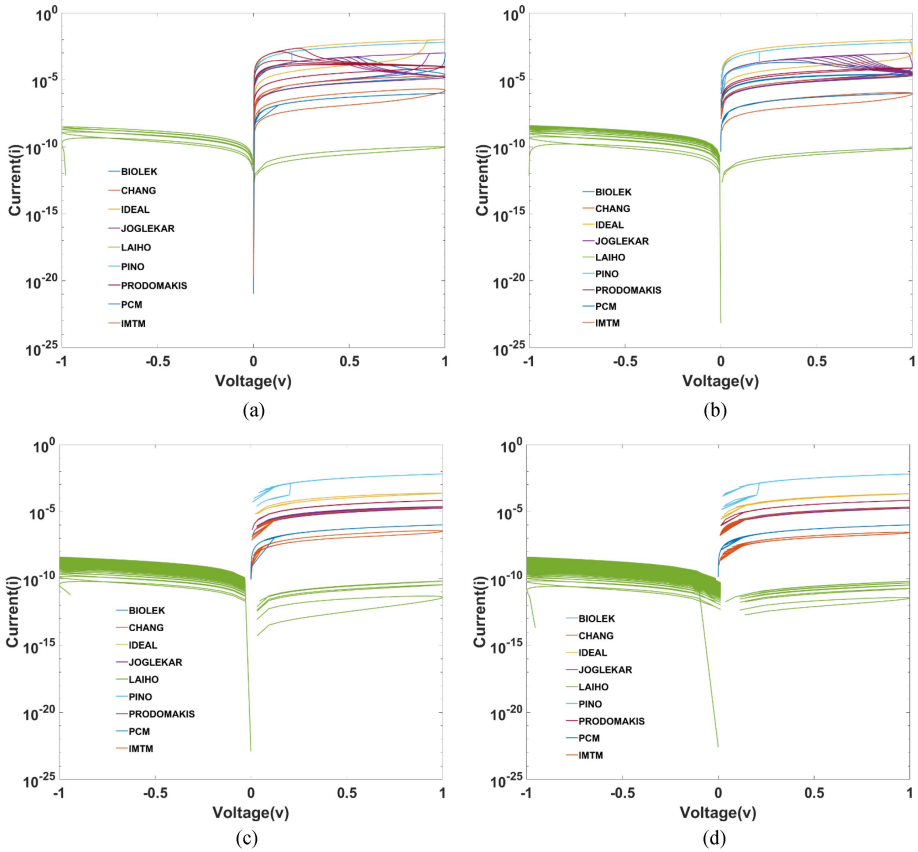

FIGURE 5. BIOLEK: Biolek Model, UMM: University of Michigan Model, IDEAL: Ideal Memristor Model, JOGLEKAR: Joglekar Model, GHSM: General Hyperbolic Sine Model, AFRLM: Air Force Research Lab Model, PRODOMAKIS: Prodromakis Model, PCM: Phase Change Memory, IMTMS: Insulator to Metal Transition Memristive Systems, Pinched hysteresis of nine memristor models with (a) 0.5 Hz input voltage frequency, (b) 1 Hz input voltage frequency, (c) 5 Hz input voltage frequency, and (d) 10 Hz input voltage frequency.

For performing prediction in a memristive crossbar instead of a single memristor, memristors are arranged row and columnwise in different dimensions (2×2, 4×4, 8×8 and 16×16). The simulated spice data of different input voltages through rows and different output currents through the columns are used to predict using random forest and XGBoost. Performance is analyzed based on the factors of precision, recall, f1-score, and support. The accuracy, macro average, and weighted average of the two prediction approaches are compared for different crossbar dimensions.

D. RANDOM FOREST ALGORITHM ON MEMRISTOR DATA

A random forest algorithm is used for prediction and regression problems. It combines multiple decision trees to form a forest. To predict, a random subset of the input data and a random subset of the input features are used to train each decision tree. Aggregation of the decisions of all trees gives the final decision of prediction. Here, we use a technique known as bagging that reduces overfitting and improves accuracy by combining the predictions of multiple decision trees formed from bootstrapped training data samples. The pseudocode for random forest is given in Algorithm 1.

E. XGBOOST ALGORITHM ON MEMRISTOR DATA

The XGBoost algorithm is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. Decision trees are base learners for the XGBoost or eXtream gradient boosting prediction. It controls overfitting by using a more regularized model. This makes it more accurate and faster than traditional gradient boosting. The pseudocode for XGBoost is given in Algorithm 2.

III. RESULTS AND DISCUSSION

The nine memristor models with four different frequencies used for the prediction are shown in Fig. 5. The predictor may not capture the relevant information if the number of features is too small. If the number of features is too large, the predictor may overfit the training data, leading to poor generalization performance. Since we are only using two features, input voltage and output current, the predictor highly depends on the data. In the dataset, different models show similar readings of input voltage and output current (at the pinched point). Here, the overall data are set into 70% of training data and 30% of test data.

The initial dataset is split for training and testing. The testing dataset contains randomly selected data for each model. After training to predict the model, this testing dataset is

TABLE 2. Performance Analysis of Random Forest Predictor on Single Memristor Circuit Simulations

Memristor Model	0.5 Hz				1 Hz				5 Hz				10 Hz			
	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.75	0.80	0.77	426	0.92	0.84	0.88	427	0.71	0.86	0.78	1306	0.85	0.84	0.85	1366
UMM	0.66	0.94	0.78	298	0.73	0.89	0.80	462	0.72	1.00	0.83	1423	0.94	0.81	0.87	1348
IDEAL	0.93	0.93	0.93	338	0.97	0.99	0.98	633	1.00	1.00	1.00	950	1.00	1.00	1.00	1393
IMTMS	0.88	0.89	0.89	170	0.93	0.95	0.94	321	1.00	0.97	0.98	1123	0.96	0.99	0.98	2271
JOGLEKAR	0.78	0.74	0.76	565	0.93	0.91	0.92	932	0.82	0.66	0.73	1342	0.86	0.84	0.85	1302
GHSM	0.93	0.76	0.84	214	0.97	0.90	0.93	404	1.00	0.86	0.93	1236	0.84	0.93	0.88	1348
PCM	0.71	0.40	0.51	138	0.72	0.66	0.69	456	1.00	0.51	0.68	756	0.91	0.93	0.92	1380
AFRLM	0.98	0.98	0.98	1227	1.00	0.99	0.99	2509	1.00	1.00	1.00	13105	1.00	1.00	1.00	28767
PRODOMAKIS	0.88	0.82	0.85	439	0.97	0.98	0.97	431	1.00	1.00	1.00	1382	1.00	1.00	1.00	1377

TABLE 3. Performance Analysis of XGBoost Predictor on Single Memristor Circuit Simulations

Memristor Model	0.5 Hz				1 Hz				5 Hz				10 Hz			
	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.72	0.71	0.71	431	0.86	0.74	0.80	415	0.61	0.87	0.72	1334	0.87	0.94	0.91	1349
UMM	0.91	0.98	0.94	301	0.98	0.86	0.91	418	1.00	1.00	1.00	1437	0.99	1.00	1.00	1335
IDEAL	0.91	0.86	0.88	380	0.95	0.97	0.96	626	1.00	1.00	1.00	913	1.00	1.00	1.00	1360
IMTMS	0.69	0.90	0.78	165	0.76	0.90	0.82	339	0.97	0.76	0.85	1069	0.95	0.97	0.96	2245
JOGLEKAR	0.75	0.73	0.74	565	0.85	0.81	0.83	904	0.75	0.56	0.64	1354	0.95	0.84	0.89	1380
GHSM	0.99	0.99	0.99	190	1.00	1.00	1.00	412	1.00	1.00	1.00	1198	1.00	1.00	1.00	1312
PCM	0.95	0.80	0.87	148	0.87	0.98	0.92	467	1.00	0.99	0.99	724	1.00	0.99	1.00	1348
AFRLM	0.98	0.96	0.97	1223	0.99	0.98	0.98	2603	1.00	1.00	1.00	13308	1.00	1.00	1.00	28852
PRODOMAKIS	0.76	0.78	0.77	412	0.90	0.97	0.93	391	0.99	1.00	1.00	1286	1.00	1.00	1.00	1371

Algorithm 2: Pseudocode for Random Forest Classifier.

Training on the data

Training data (X_{train}, y_{train}), number of trees (`num_trees`), max depth of each tree (`max_depth`) Random forest model

Procedure:

Initialize an empty list to store the trees: `forest = []`;

for $i \leftarrow 1$ `num_trees` **do** `subset_X, subset_y = random_subset(`$X_{\text{train}}, y_{\text{train}}$`);`

```
tree = build_decision_tree(subset_X, subset_y, max_depth);
forest.append(tree);
```

Return: Random Forest model (`forest`);

Prediction using the trained model

Random Forest model, Test data (`X_test`) Predicted class labels for `X_test`

Procedure:

Initialize an array to store the predictions: `predictions = [];`

for each `tree` in `forest` **do** `prediction = predict_with_tree(tree, X_test);`

`predictions.append(prediction);`

Return: Majority vote of `predictions`;

used as input. After testing both algorithms in various data, the performance is visualized using a confusion matrix and analyzed using the parameters precision, recall, and f1-score based on the support for each model. Precision is the ratio of the number of true positives to the number of elements labeled to belong to the positive class. The ratio between the number of true positives and the total number of elements that belongs to the positive class that gives recall and f1-score is calculated by taking the harmonic mean of precision and recall. Support represents the number of samples of true responses lying in the class. The overall performance is evaluated using accuracy, macro average, and weighted average. In macro average, all classes equally contribute to the final averaged matrix, and in weighted average, each class's contribution to the average is weighted by its size.

The random forest prediction technique in which the parameters are trained with 100 trees is used to perform prediction. Tables 2 and 3 show the performance analysis of the random forest and XGBoost predictors for these nine models with four different frequencies of input voltages analyzed by the factors precision, recall, f1-score, and support using the confusion matrix shown in Figs. 6 and 7, respectively. Most of the models give high accuracy while using both prediction techniques. Support is a significant factor considering the prediction parameters for individual memristor models. Depending on the support, both algorithms show varying performance parameters but are still well enough to identify the model successfully. The comparison of overall accuracy, macro average, and weighed average of random forest and XGBoost for single memristor models at four different frequencies 0.5, 1, 5, and 10 Hz are illustrated by the Fig. 8. The performance of both approaches enhances with frequency. Maximum accuracy is achieved at a higher frequency. This implies that even though there are similar points in the dataset, the two predictors can produce a better accuracy in predicting or identifying the model.

Even though the overall prediction has good accuracy in most cases, it would be better to fine-tune the dataset since the data contains similar values from the pinched area of the *I–V* curve. In memristor models, the pinched region gives the voltage and current data as zero. Every memristor model will have this region, an essential condition for being in the class of memristors. Such training with a dataset excluding the pinched area may improve the predictions. Also, we can

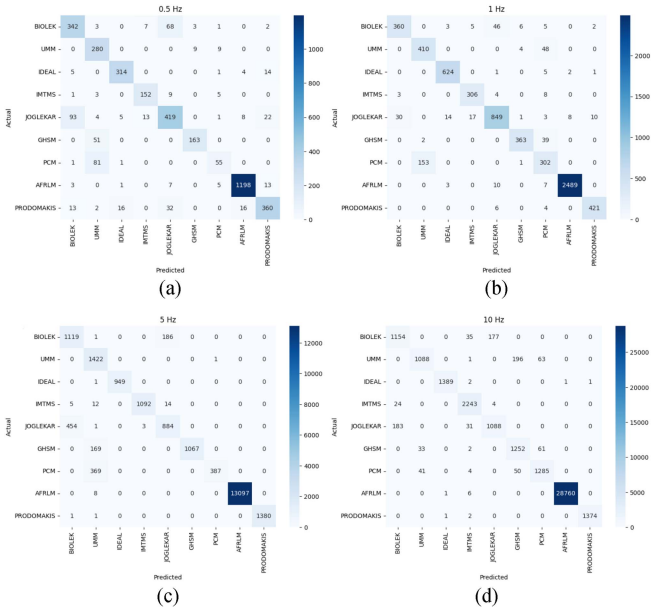

FIGURE 6. Random forest prediction confusion matrix for single memristor simulation for (a) 0.5-Hz input voltage frequency, (b) 1-Hz input voltage frequency, (c) 5-Hz input voltage frequency, and (d) 10-Hz input voltage frequency.

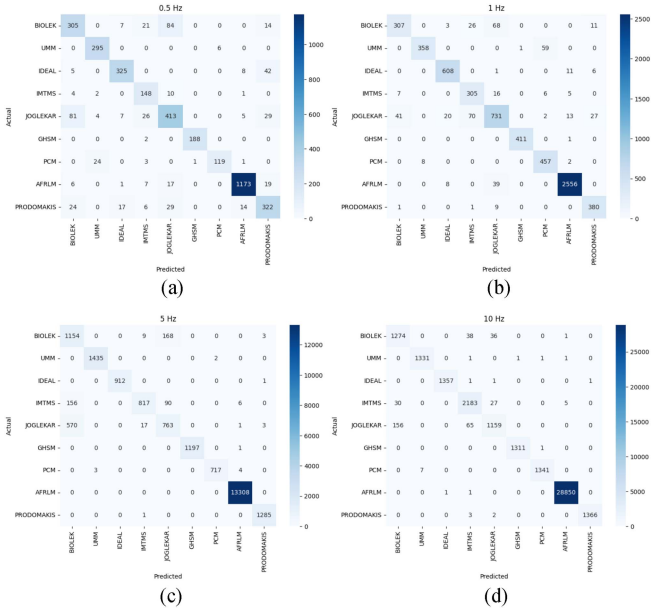

FIGURE 7. XGBoost prediction confusion matrix for single memristor simulation for (a) 0.5-Hz input voltage frequency, (b) 1-Hz input voltage frequency, (c) 5-Hz input voltage frequency, and (d) 10 Hz input voltage frequency.

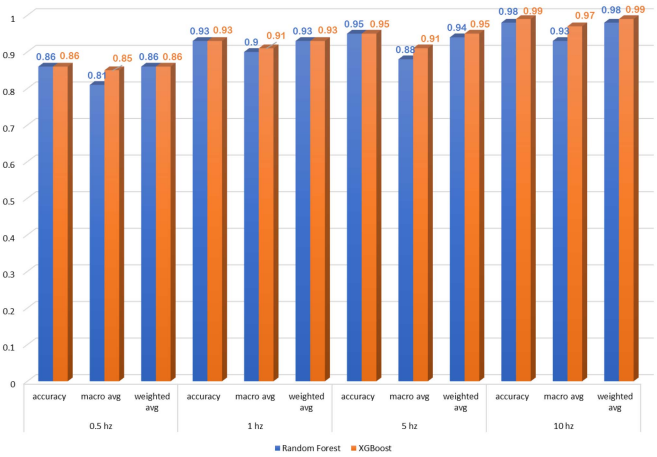

FIGURE 8. Accuracy, macro average, and weighted average while using Random forest and XGBoost for single memristor models at frequencies 0.5, 1, 5, and 10 Hz.

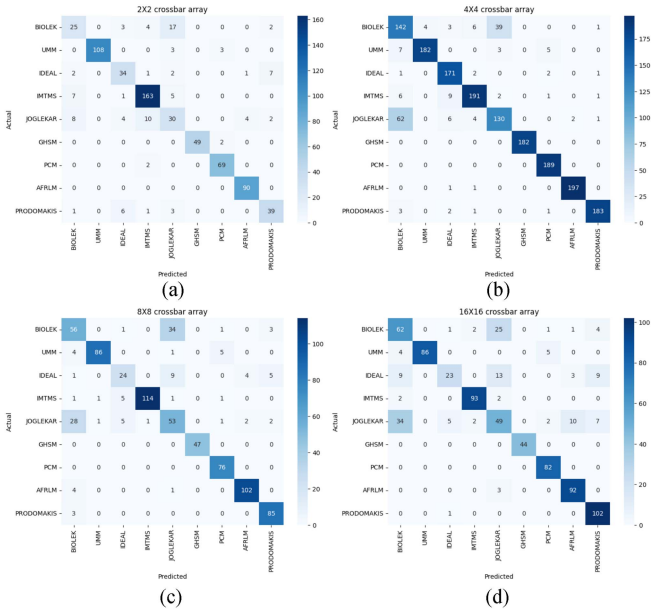

FIGURE 9. Random forest prediction confusion matrix for memristor crossbar array simulation of (a) 2 × 2 crossbar array, (b) 4 × 4 crossbar array simulation, (c) 8 × 8 crossbar array, and (d) 16 × 16 crossbar array.

combine results from multiple algorithms to enhance the prediction.

The performance analysis of random forest prediction and XGBoost prediction for 2 × 2 crossbar, 4 × 4 crossbar, 8 × 8 crossbar, and 16 × 16 crossbar based on the confusion matrix shown in the Figs. 9 and 10 are illustrated in the Tables 4–7, respectively. According to the performance parameters, support plays a significant role in predicting the model. Since the data from each model are close enough for higher prediction accuracy, more input data points are required for higher accuracy. Accuracy, macro average, and weighted average are compared for the two prediction approaches for the above four dimensions, as shown in Fig. 11. Random forest and XGBoost gave more than 80% overall accuracy in four cases. In some cases, random forest performed better than the XGBoost algorithm. But this can vary depending on the input data. Both approaches perform well enough to predict the model in most cases.

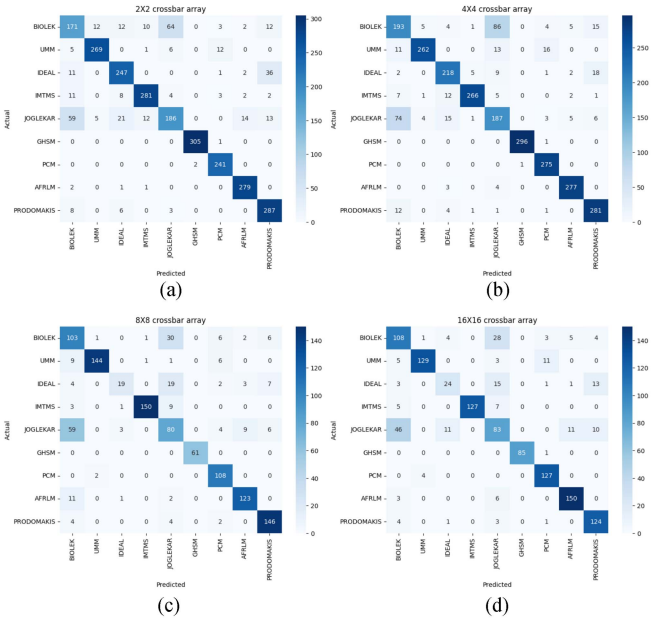

FIGURE 10. XGBoost prediction confusion matrix for memristor crossbar array simulation of (a) 2 × 2 crossbar array, (b) 4 × 4 crossbar array, (c) 8 × 8 crossbar array, and (d) 16 × 16 crossbar array simulation.

TABLE 4. Performance Analysis of Random Forest and XGBoost Algorithm on 2 × 2 Crossbar

Memristor Model	Random Forest				XGBoost			
	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.58	0.49	0.53	51	0.64	0.60	0.62	286
UMM	1.00	0.95	0.97	114	0.94	0.92	0.93	293
IDEAL	0.71	0.72	0.72	47	0.84	0.83	0.83	297
IMTMS	0.90	0.93	0.91	176	0.92	0.90	0.91	311
JOGLEKAR	0.50	0.52	0.51	58	0.71	0.60	0.65	310
GHSM	1.00	0.96	0.98	51	0.99	1.00	1.00	306
PCM	0.93	0.97	0.95	71	0.92	0.99	0.96	243
AFRLM	0.95	1.00	0.97	90	0.93	0.99	0.96	283
PRODOMAKIS	0.78	0.78	0.78	50	0.82	0.94	0.88	304

TABLE 5. Performance Analysis of Random Forest and XGBoost Algorithm on 4 × 4 Crossbar

Memristor Model	Random Forest				XGBoost			
	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.64	0.76	0.69	193	0.65	0.62	0.63	313
UMM	0.98	0.95	0.97	195	0.96	0.87	0.91	302
IDEAL	0.88	0.97	0.92	150	0.85	0.85	0.85	255
IMTMS	0.98	0.90	0.94	225	0.97	0.90	0.94	294
JOGLEKAR	0.76	0.62	0.68	219	0.61	0.63	0.62	295
GHSM	0.99	1.00	1.00	196	1.00	1.00	1.00	297
PCM	0.98	1.00	0.99	189	0.91	1.00	0.95	276
AFRLM	0.97	0.99	0.98	196	0.95	0.98	0.96	284
PRODOMAKIS	0.95	0.98	0.96	181	0.88	0.94	0.90	300

A. VARIABILITY ANALYSIS

The proposed work focuses on memristive devices which are emerging and yet to mature. Modeling those devices with higher accuracy is challenging due to different variabilities like cycle-to-cycle, device-to-device, defective points in the array, etc. Memristor is not a single device; several devices fall under the broad category of memristors. Hence, the problem is more complex than models which are CMOS-based. Because of the mentioned issues, memristive circuits with idealistic models can have output errors.

TABLE 6. Performance Analysis of Random Forest and XGBoost Algorithm on 8 × 8 Crossbar

Memristor Model	Random Forest				XGBoost			
	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.58	0.59	0.58	95	0.53	0.69	0.60	149
UMM	0.98	0.90	0.93	96	0.98	0.89	0.94	161
IDEAL	0.69	0.56	0.62	43	0.79	0.35	0.49	54
IMTMS	0.99	0.93	0.96	123	0.99	0.92	0.95	163
JOGLEKAR	0.54	0.57	0.55	93	0.55	0.50	0.52	161
GHSM	1.00	1.00	1.00	47	1.00	1.00	1.00	61
PCM	0.90	1.00	0.95	76	0.84	0.98	0.91	110
AFRLM	0.94	0.95	0.95	107	0.90	0.90	0.90	137
PRODOMAKIS	0.89	0.97	0.93	88	0.88	0.94	0.91	156

TABLE 7. Performance Analysis of Random Forest and XGBoost Algorithm on 16 × 16 Crossbar

2*Memristor Model	Random Forest				XGBoost			
	precision	recall	f1-score	support	precision	recall	f1-score	support
BIOLEK	0.56	0.65	0.60	96	0.62	0.71	0.66	153
UMM	1.00	0.91	0.95	95	0.96	0.87	0.91	148
IDEAL	0.77	0.40	0.53	57	0.60	0.42	0.49	57
IMTMS	0.96	0.96	0.96	97	1.00	0.91	0.95	139
JOGLEKAR	0.53	0.45	0.49	109	0.57	0.52	0.54	161
GHSM	1.00	1.00	1.00	44	1.00	0.99	0.99	86
PCM	0.91	1.00	0.95	82	0.88	0.97	0.92	131
AFRLM	0.87	0.97	0.92	95	0.90	0.94	0.92	159
PRODOMAKIS	0.84	0.99	0.91	103	0.82	0.93	0.87	133

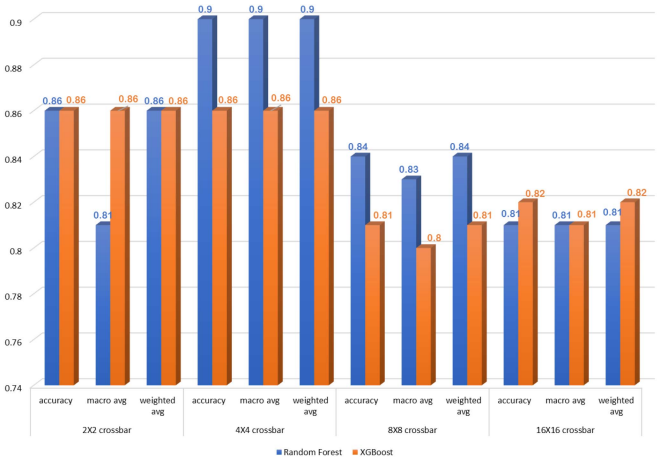

FIGURE 11. Accuracy, macro average, and weighted average while using random forest and XGBoost for crossbar architectures of dimensions 2 × 2, 4 × 4, 8 × 8, and 16 × 16.

Variability in memristors can be analyzed by considering the change in the conductance states in the device and crossbar level. The variability can be induced by changing the conductance states of the memristive devices in the crossbar. The analysis can be done by evaluating changes in the column's current measured through the crossbar columns. Since our dataset contains the input voltages to the crossbar rows and output currents from the crossbar columns, to analyze the impacts of changed crossbar column currents in predicting the memristor models, the relative current error after inducing variability in conductance states needs to be considered.

There are two test cases for the same: Case 1: $G + V * G$ and Case 2: $G - V * G$. Here, V is the variability percentage, and G is the actual conductance.

The simulations are done for conductance variations with three variability percentages: 20%, 30%, and 40%. The conductance values of the memristive devices in the crossbar

TABLE 8. Accuracy Obtained After Inducing Variabilities of 20%, 30%, and 40%

Dimension	Variability Ratio (%)	Accuracy (Random Forest)(%)	Accuracy (XGBoost) (%)
4×4	0	90	86
	+20	79	76
	-20	78	79
	+30	81	81
	-30	77	79
	+40	77	75
	-40	81	77
8×8	0	84	81
	+20	80	79
	-20	83	79
	+30	75	77
	-30	81	81
	+40	78	78
	-40	82	81
16×16	0	81	82
	+20	77	75
	-20	81	78
	+30	79	77
	-30	82	79
	+40	81	78
	-40	81	80

are changed by inducing the abovementioned percentages of variabilities. This updated dataset contains the input voltages, updated current values are again fed to the classifier, and classification reports are analyzed. Table 8 details the impact of these induced variabilities in predicting the memristor models using the random forest and XGBoost algorithms.

When we consider memristive devices in crossbars in a neuro system, like a neural network implementation, these changes in conductance states and relative current error, which gives the difference between ideal and measured column currents, are more important, which impacts the accuracy of implementations. Due to ageing effects, the R_{ON} and R_{OFF} values may get changed. This corner analysis shows the impacts of relative current error on the accuracy of neural network implementations. In neural network implementations, weights are mapped to conductance values to perform MAC, equivalent to weighted summation. Hence, the accuracy depends on how accurately the weights are mapped to conductance states. If more conductance states are available, the weights will be mapped more precisely, and the accuracy attained will also be higher. Ageing of memristors will cause the vanishing of different conductance states, which will cause degradation in accuracy due to a lack of precise mapping. The relative current error gives the impact of variability in the neural system. When R_{ON} increases and R_{OFF} decreases, the number of available conductance states decreases, and mapping of weights cannot be done precisely, which leads to degradation in the accuracy. When R_{ON} decreases and R_{OFF} increases, more conductance states will be available; hence relative current error decreases.

Due to different variabilities, it is difficult to estimate the functional behavior and accuracy of the experimental results. The simulation-based approach is used to cross-validate memristor circuit designs regarding accuracy and functionality to address the above issue. A wide range of variabilities in the devices are addressed using this reverse engineering approach, which is useful today and for the long term as more and more memristive devices are discovered.

IV. CONCLUSION

The proposed work classifies and predicts the memristor models used in circuit simulations with only available data of inputs and outputs. The efficient hardware neuromorphic computing systems for different industrial applications can be implemented with higher degree of performance by choosing memristor models that suit specific applications. Exploring other advanced classifiers for prediction and cross-validation can enlarge the boundaries for industrial applications in which this explainable AI approach can be used. In this article, we propose two ensemble learning techniques, random forest and XGBoost, to cross validate circuit simulations for different models of memristors. These proposed prediction models estimate the memristor model from a circuit simulation's voltage and current measurements. In the detailed examinations, we found that the predictive models could perform with high accuracy in various configurations of the circuit design simulations. The final analysis is based on the accuracy, precision, and f1-score obtained from the confusion matrix. The input voltage frequency was a key component in the accuracy of the prediction models. The prediction model's accuracy increased with frequency. In various crossbar simulations, the prediction models performed with high accuracy. In some cases, random forest was able to perform better than XGBoost. From the final analysis, we concluded that random forest and XGBoost work well given large homogeneous training data and are relatively robust to outliers.

The proposed prediction methods cross validate the memristor circuit simulations, ensuring accurate results concerning the memristor model. This method helps precisely analyze the circuit's *I-V* characteristics and reverse engineering the circuit only from the output measurements. When there is confusion on which memristor model should be used for a desired input and output, this cross-validation system can be successfully implemented to explain the black-box mystery.

AUTHOR CONTRIBUTIONS

S. Pallathuvalappil and R. Kottapuzhackal, equally contributed to lead the experimental setup, analysis, and writing. A. James led the problem formulation, experimental design, funding, and writing of the paper.

REFERENCES

[1] L. Chua, "Memristor-the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971.
[2] Z. Biolek, D. Biolek, and V. Biolkova, "Spice model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.
[3] P. Sheridan and W. Lu, "Memristors and memristive devices for neuromorphic computing," in *Memristor Networks*. Berlin, Germany: Springer, 2014, pp. 129–149.
[4] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnol.*, vol. 8, no. 1, pp. 13–24, 2013.

[5] S. Kvatinsky, K. Talisveyberg, D. Fliter, A. Kolodny, U. C. Weiser, and E. G. Friedman, "Models of memristors for spice simulations," in *Proc. IEEE 27th Conv. Elect. Electron. Engineers Isr.*, 2012, pp. 1–5, doi: 10.1109/EEEI.2012.6377081.
[6] A. P. James and L. O. Chua, "Analog neural computing with super-resolution memristor crossbars," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 68, no. 11, pp. 4470–4481, Nov. 2021.
[7] I. Vourkas, D. Stathis, G. C. Sirakoulis, and S. Hamdioui, "Alternative architectures toward reliable memristive crossbar memories," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 1, pp. 206–217, Jan. 2016.
[8] T. Prodromakis and C. Toumazou, "A review on memristive devices and applications," in *Proc. IEEE 17th Int. Conf. Electron. Circuits Syst.*, 2010, pp. 934–937.
[9] J. Chen, J. Li, Y. Li, and X. Miao, "Multiply accumulate operations in memristor crossbar arrays for analog computing," *J. Semicond.*, vol. 42, no. 1, 2021, Art. no. 013104.
[10] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, "An assessment of the effectiveness of a random forest classifier for land-cover classification," ISPRS J. Photogrammetry Remote Sens., vol. 67, pp. 93–104, 2012.
[11] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," in *Proc. 8th Int. Conf. Mach. Learn. Data Mining Pattern Recognit.*, Berlin, Germany, 2012, pp. 154–168.
[12] V. Y. Kulkarni and P. K. Sinha, "Pruning of random forest classifiers: A survey and future directions," in *Proc. Int. Conf. Data Sci. Eng.*, 2012, pp. 64–68.
[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, pp. 1189–1232, 2001.
[14] S. Haghighi, M. Jasemi, S. Hessabi, and A. Zolanvari, "PyCM: Multiclass confusion matrix library in Python," *J. Open Source Softw.*, vol. 3, no. 25, 2018, Art. no. 729.
[15] M. Heydarian, T. E. Doyle, and R. Samavi, "MLCM: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19083–19095, 2022.

SRUTHI PALLATHUVALAPPIL (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication from the University of Calicut, Malappuram, India, in 2014, and the M.Tech. degree in embedded systems from the Vellore Institute of Technology, Vellore, India, in 2017. She is currently working toward the Ph.D. degree with the School of Electronics Systems and Automation, Digital University of Kerala, Mangalapuram, Kerala, India.

She is currently involved in a few projects related to hardware-based low power memristive network implementation. Her research interests include low-power resistive memory networks for AI and also memristive analog circuits, multibit logic memories, 3-D integration, and neuromorphic computing systems.

RAHUL KOTTAPPUZHACKAL received the B.Sc. degree in physics from MG University, Kottayam, India, in 2020, and the M.Sc. degree in computer science from Digital University Kerala, Thiruvananthapuram, India, in 2023.

He is currently a Research Assistant with the School of Electronics Systems and Automation, Digital University of Kerala, Mangalapuram, Kerala, India. He is also working as an AI Engineer with India Graphene Engineering and Innovation Centre, Kochi, Kerala. He currently involved in the areas of machine learning, memristive systems, and neuromorphic computing systems. His research interests include the applications of machine learning for memristor-based circuits.

ALEX JAMES received the Ph.D. degree from Griffith University, Nathan, QLD, Australia, in 2008.

He is currently a Professor and Dean (Academic) with the Kerala University of Digital Sciences, Innovation and Technology (aka Digital University Kerala), Mangalapuram, Kerala, India. He is the Professor-in-Charge of the Maker Village, and the Chief Investigator with the India Innovation Centre for Graphene, Kochi, Kerala. His research interests include AI-neuromorphic systems (software and hardware), VLSI, and image processing.

Dr. James is a Life Member of the Association for Computing Machinery, Senior Fellow of higher education academy (HEA), Fellow of British Computer Society (BCS), and Fellow of Institution of Engineering and Technology. He was the recipient of IEEE Outstanding Researcher by IEEE Kerala Section for 2022, Kairali Scientist Award for Physical Science in 2021, and Best Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I in 2021. He was an Editorial Board Member of *Information Fusion* from 2010 to 2014, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM 1: REGULAR PAPERS from 2018 to 2023, and has been serving as an Associate Editor for IEEE ACCESS since 2017, *Frontiers in Neuroscience* since 2022, IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS since 2022, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS since 2024, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR ARTIFICIAL INTELLIGENCE since 2024. He has also been an Associate Editor-in-Chief for IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS since 2024. He is a member of IEEE Circuits and Systems Society (CASS) Technical Committee (TC) on Nonlinear Circuits and Systems; IEEE Consumer Technology Society TC on Quantum in Consumer Technology; TC on Machine learning, Deep learning, and AI in CE; IEEE CASS TC on Cellular Nanoscale Networks and Memristor Array Computing; and IEEE CASS Special Interest Group on AgriElectronics. He was the founding Chair of IEEE CASS Kerala chapter, a Member of IET Vision and Imaging Network, and is currently a Member of BCS' Fellows Technical Advisory Group.