Improving Semiconductor Device Modeling for Electronic Design Automation by Machine Learning Techniques

Zeheng Wang, *Member, IEEE*, Liang Li, Ross C. C. Leon, Jinlin Yang, Junjie Shi, Timothy van der Laan, and Muhammad Usman

***Abstract*— The semiconductors industry benefits greatly from the integration of machine learning (ML)-based techniques in technology computer-aided design (TCAD) methods. The performance of ML models, however, relies heavily on the quality and quantity of training datasets. They can be particularly difficult to obtain in the semiconductor industry due to the complexity and expense of the device fabrication. In this article, we propose a self-augmentation strategy for improving ML-based device modeling using variational autoencoder (VAE)-based techniques. These techniques require a small number of experimental data points and do not rely on TCAD tools. To demonstrate the effectiveness of our approach, we apply it to a deep neural network (DNN)-based prediction task for the ohmic resistance value in gallium nitride (GaN) devices. A 70% reduction in mean absolute error (MAE) when predicting experimental results is achieved. The inherent flexibility of our approach allows easy adaptation to various tasks, thus making it highly relevant to many applications of the semiconductor industry.**

***Index Terms*— Data augmentation, electronic design automation (EDA), gallium nitride (GaN), machine learning (ML), semiconductor devices.**

I. Introduction

ELECTRONIC design automation (EDA) has been crucial in advancing the semiconductors industry by simplifying design tasks and reducing their time consumption [1]. One particular EDA technique, technology computer-aided design (TCAD), has been especially useful in the area of semiconductor devices. TCAD solves basic physics equations using the finite element method, such as the Poisson and Schrödinger equations, which provides easy access to simulated results that would be difficult to solve manually [2], [3], [4]. In addition, TCAD has significantly reduced the cost of experiments during device design by avoiding them altogether [5].

Nevertheless, simulating complex 3-D device structures requires significant computational resources. While many models and methods have been developed to reduce resource consumption, exploring novel methodologies of TCAD remains a pressing issue to balance the accuracy and time consumption of sophisticated physics simulations. So far, machine learning (ML)-based solutions have been successfully employed in many device modeling cases and offer the advantage of low-resource consumption after model training [6], [7], [8], [9], [10]. However, with expanding size of the ML models, there is an increasing need for input data to fully complete model training [11].

TCAD-based data augmentation, a technique that has garnered significant attention in the semiconductor industry since 2019 [9], [12], [13], [14], has been employed to generate artificial data that can be fed into deep neural network (DNN)-based models. This approach could provide an expanded dataset and then significant boost to DNN-based modeling within the TCAD industry's development. However, many problems in the semiconductor industry cannot be directly solved by TCAD tools, such as the simulation of the formation of ohmic contacts in gallium nitride (GaN) devices, which imposes a formidable challenge on the TCAD-based augmentation technique.

Recently, a study by Sheelvardhan et al. [15] highlighted the potential of knowledge-based ML algorithms in overcoming the limitations of traditional ML-based approaches for semiconductor device modeling. By leveraging prior knowledge, these algorithms offer a promising solution to address the complexities associated with establishing and training ML models. This research represents a significant advancement toward the development of next-generation ML-based TCAD toolkits.

Manuscript received 17 July 2023; accepted 17 August 2023. Date of publication 30 August 2023; date of current version 2 January 2024. This work was supported in part by CSIRO's Impossible Without You Program. The review of this article was arranged by Editor H. Agarwal. *(Corresponding authors: Zeheng Wang; Timothy van der Laan; Muhammad Usman.)*

Zeheng Wang is with Data61, CSIRO, Clayton, VIC 3168, Australia, and also with CSIRO Manufacturing, Lindfield, NSW 2070, Australia (e-mail: zenwang@outlook.com).

Liang Li is with the Academy for Advanced Interdisciplinary Studies, Peking University, Beijing 100871, China.

Ross C. C. Leon is with Quantum Motion, N7 9HJ London, U.K.

Jinlin Yang is with the Department of Chemistry, National University of Singapore, Singapore 117543.

Junjie Shi is with the School of Materials Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia.

Timothy van der Laan is with CSIRO Manufacturing, Lindfield, NSW 2070, Australia (e-mail: tim.vanderlaan@csiro.au).

Muhammad Usman is with Data61, CSIRO, Clayton, VIC 3168, Australia (e-mail: muhammad.usman@csiro.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TED.2023.3307051.

Digital Object Identifier 10.1109/TED.2023.3307051

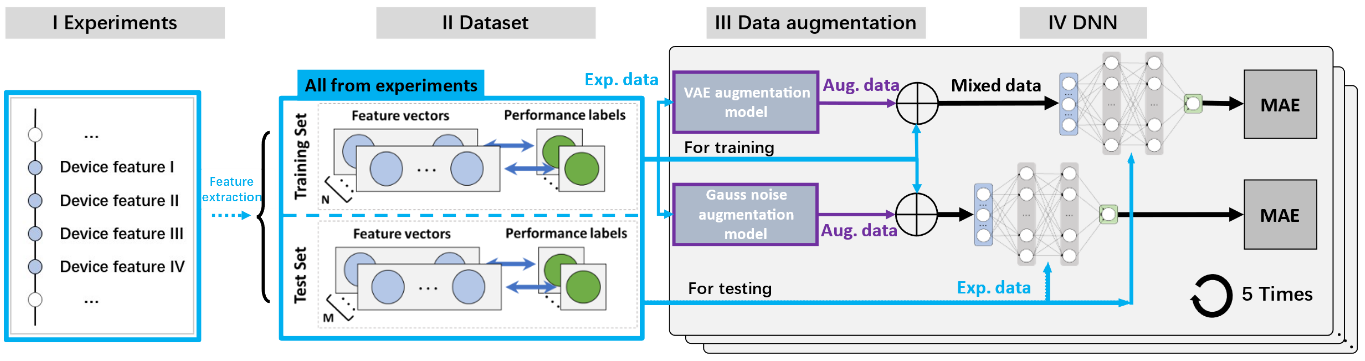

Fig. 1. Sketch of the whole procedure of augmentation-enhanced ML-based semiconductor device modeling–The data were extracted from literature containing experimental data and then were augmented by the augmentation model (VAE). The original data and the artificially augmented data were measured in a DNN-based prediction task. The data flows between each step are indicated by arrows. Note that the validation process was carried out with the experimental data that were extracted from the recently reported literature.

This article proposes a novel data self-augmentation strategy for expanding the size of semiconductor device datasets used for DNN-based modeling tasks, without requiring calibration of TCAD tools. Although several studies have made good attempts in using autoencoders as an unsupervised ML strategy to improve the classification tasks in the semiconductor industry [16], [17], using variational autoencoder (VAE) in boosting TCAD's performance of semiconductor devices is still unclear to the best of our knowledge. Our approach is based on the use of VAE [18], a type of generative model that can learn the underlying probability distribution of the dataset, and then generate new, synthetic data samples that are statistically similar to the original data. Specifically, we apply the data from our proposed strategy to the DNN-based modeling task—predicting the ohmic contacts of GaN devices, a challenging problem that cannot be solved directly using TCAD.

To validate the effectiveness of our approach, we used experimental data extracted from the literature to train the VAE, which was then used to generate augmented data that were combined with the experimental data for training the DNN model (for dataset details, see our previous work [19]). The results demonstrate that our data augmentation strategy significantly reduces the mean absolute error (MAE) of the prediction by up to 70% for AlGaN/GaN devices, when compared with the model using experimental data only. This finding highlights the potential of our approach for enhancing the accuracy and robustness of device simulation tools in the semiconductor industry.

II. Methods for Data Augmentation and Verification

In this article, we propose a data self-augmentation strategy based on a VAE to improve the performance of ML-based modeling. Generally, ML-based device modeling aims to link device features, such as gate length, drain voltage, and annealing temperature, to performance, such as surface potential, saturation current, and ON/OFF ratio [20], [21], [22]. Our proposed data augmentation framework is divided into three main parts: a feature preprocessing module (sectors I and II), a data augmentation module (sector III), and an ML-based modeling module (sector IV), as shown in Fig. 1. The first step of modeling is to extract parameter data, i.e., device features, from experimental results, as illustrated in sector I. The extracted data are then transformed into vectors by the feature preprocessing module, as shown in sector II. The experimental data are split into a training set (67% of the data) and a test set (33% of the data) before establishing the VAE model. Each dataset's data points were chosen randomly. The training set is used to build the generative models for augmentation, as depicted in sector III. The generated artificial data are combined with the training set and fed into the DNN-based model for semiconductor device performance modeling (sector IV). The test set (only contains experimental data) is used to evaluate the generalization ability of the DNN and the augmentation model.

It is noteworthy that we adopted a linear unit structure of VAE with minimal parameters in this study. By doing so, only the crucial parameters of the real distribution are learned through backpropagation, akin to principal component analysis. This approach allows us to partially relax the size requirements of the training dataset, but meanwhile, it may introduce inaccuracy, which requires further study and is not the scope of this study.

The steps of the proposed data augmentation strategy and verification are described as follows, with technical details provided in the subsequent subsections.

1) Extract feature data from experiments and split the data into a test set and a training set.
2) Train the VAE-based model using the training set.
3) Generate artificial data and combine it with the training set.
4) Train the corresponding DNN models using the combined dataset.
5) Test the DNN model using the test set and calculate the MAE.
6) Repeat steps 2)–5) five times and calculate the average MAE for further discussion.

A. Feature Preprocessing

To minimize the influence of missing data points, all experimental data were filtered by a "data cleaning" process, this deletes whole vectors if any vacancy exists. Then, the

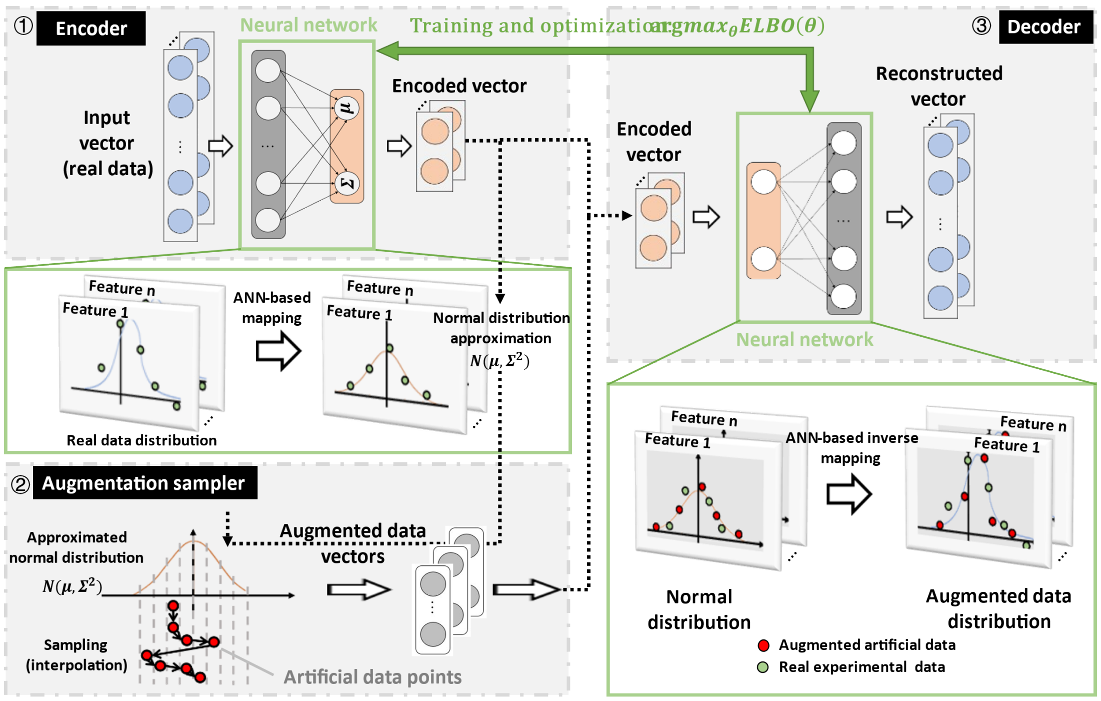

Fig. 2. Schematic procedure of artificial data generation. The typical VAE model was adopted for generating the features, while for generating the corresponding labels, the nearest neighbor algorithm was used. The green frames indicate the process of data transformation in latent spaces, while the frames with gray backgrounds show the structure of the augmentation model.

experimental data were divided into two parts for the next step: the numerical part and the text-based part. These two parts are handled by different processes: for the text-based input, such as the name of material layers, the data were then transformed into a numerical vector using one-hot encoding [19], [23]. For the numerical input, the item x of the data was directly standardized by z-score to ensure that the numerical input was centered to 0 with a standard deviation of 1, following the equation given next:

$$z = \frac{x - \mu_x}{\sigma_x} \tag{1}$$

where μ_x is the mean of x and σ_x is the standard deviation of x. Note that, for simplicity, the ohmic value data described in the following sections are all standardized values, not original values.

B. Generative Model

The primary goal of a generative model is to learn the joint probability distribution of a given dataset through unsupervised learning. This enables successful data augmentation from experimental data by interpolating variations into the trained generative model (shown in panels I and II of Fig. 2). In this study, we used a two-component generator consisting of an artificial feature generator and an artificial label generator. To overcome the challenge of insufficient training data in the augmentation task for the artificial feature generator, we utilized the VAE, which is a simple yet powerful deep generative model. In addition, we applied the Kth nearest neighbor (KNN) regressor to generate the corresponding labels. This approach offers an effective solution to the problem of insufficient training data in data augmentation and represents a significant contribution to the field of generative modeling.

1) Artificial Feature Generator: The artificial feature generator is realized by the VAE. This is a variant of the automatic encoder combining variational inference with a conventional autoencoder framework. Thus, the VAE consists of two parts: an encoder and a decoder. The encoding–decoding process efficiently realizes dimensionality reduction, which emphasizes the preferred features of the input and suppresses the less-important features to minimize interference [24], [25]. The encoder encodes, denoted as Enc($\boldsymbol{x}$), input $\boldsymbol{x}$ into a latent representation z with a parameter θ, which is denoted by $q_\theta(z\,|\,\boldsymbol{x})$. The decoder reconstructs, denoted as Dec(z), the data distribution $\tilde{\boldsymbol{x}}$ from the given z, which is depicted as follows:

$$z = \mathrm{Enc}(\boldsymbol{x}) \sim q_\theta(z\,|\,\boldsymbol{x}) \tag{2}$$
$$\tilde{\boldsymbol{x}} = \mathrm{Dec}(z) \sim p(\boldsymbol{x}\,|\,z). \tag{3}$$

Given a dataset $\boldsymbol{x} = \{x_1, \ldots, x_N\}$, where N is the number of samples, the target of the generative model is to maximize the probability $p(\boldsymbol{X})$

$$p(\boldsymbol{X}) = \sum_{i=1}^{N} p(\boldsymbol{x}_i\,|\,z)\,p(z) \tag{4}$$

in which $p(z)$ is the probability distribution of the encoded latent representations, which is unknown. The purpose of the VAE is to infer $p(z)$ from the ideal posterior probability $p(z\,|\,\boldsymbol{x})$. It could be replaced by a simpler normal distribution $q_\theta(z\,|\,\boldsymbol{x})$. Then, the problem is converted into minimizing the difference between those two distributions using Kullback–Leibler (KL) divergence [26]

$$\begin{aligned}\mathrm{KL}(q_\theta(z\,|\,\boldsymbol{x})||p(z\,|\,\boldsymbol{x}))\\ &= \mathbb{E}\big[\log q_\theta(z\,|\,\boldsymbol{x}) - \log p(\boldsymbol{x}\,|\,z) - \log p(z)\big]. \end{aligned} \tag{5}$$

By applying the following function named evidence lower bound (ELBO) [27]

$$\begin{aligned}\mathrm{ELBO}(\theta) &= -\mathbb{E}\big[\log p(\boldsymbol{x}\,|\,z) + \log p(z) - \log q_\theta(z\,|\,\boldsymbol{x})\big]\\ &= \mathbb{E}\big[\log p(\boldsymbol{x}\,|\,z)\big] - \mathrm{KL}(q_\theta(z\,|\,\boldsymbol{x})||p(z)). \end{aligned} \tag{6}$$

Equation (4) can be rewritten as a log-likelihood function

$$\log p(x) = \mathrm{KL}(q_\theta(z\,|\,\boldsymbol{x})||p(z\,|\,\boldsymbol{x})) + \mathrm{ELBO}(\theta). \tag{7}$$

Note that $\boldsymbol{x}$ is known, $\log p(\boldsymbol{x})$ is constant. Besides, the KL divergence is always greater than or equal to zero according to Jensen's inequality [28]. Therefore, minimizing the KL divergence is equivalent to maximizing $\mathrm{ELBO}(\theta)$. Since no datapoint shares its latent z with another datapoint in VAE, we can write this function for a single datapoint as follows:

$$\mathrm{ELBO}(\theta)_i = \mathbb{E}\big[\log p(\boldsymbol{x}_i\,|\,z_i)\big] - \mathrm{KL}(q_\theta(z_i\,|\,\boldsymbol{x}_i)||p(z_i)). \tag{8}$$

Thus, the training target of the VAE is actually approaching the maximum of the ELBO function shown next, which is also labeled in Fig. 2 between panels I and III

$$\arg\max_\theta \mathrm{ELBO}(\theta) = \sum_i \mathrm{ELBO}(\theta)_i. \tag{9}$$

According to [25], $p(z_i) = \mathcal{N}(0, I)$ is assumed as a standard normal distribution, while $q_\theta(z_i\,|\,\boldsymbol{x}_i) = \mathcal{N}(\mu(\boldsymbol{x}_i), \Sigma(\boldsymbol{x}_i))$, where $\mathcal{N}$ represents a normal distribution, I is an identity matrix, and μ and Σ are arbitrary deterministic functions that can be learned from data. Thus, this can be explicitly expressed by μ and Σ as follows:

$$\sum_{i,k}\left[\left(x_i^k - \tilde{\boldsymbol{x}}_i^k\right)^2 + \frac{1}{2}\left(\Sigma(\boldsymbol{x}_i)_k + \mu^2(\boldsymbol{x}_i)_k - 1 - \log\Sigma(\boldsymbol{x}_i)_k\right)\right] \tag{10}$$

where $\tilde{\boldsymbol{x}}_i^k$ is the kth element of reconstructed data vectors $\tilde{\boldsymbol{x}}_i$ and $\mu(\boldsymbol{x}_i)_k$ and $\Sigma(\boldsymbol{x}_i)_k$ denote the kth element of vectors $\mu(\boldsymbol{x}_i)$ and $\Sigma(\boldsymbol{x}_i)$, respectively. Considering the encoder and decoder processes, this $\tilde{\boldsymbol{x}}_i$ can be formulated as the following form:

$$\tilde{\boldsymbol{x}}_i = \mathrm{Dec}(\mathrm{Enc}(\boldsymbol{x}_i)). \tag{11}$$

In this proposed strategy, we used a linear neural network to construct the encoder of the VAE model by which the input x_i is encoded into the latent representation z_i. Thereafter, another linear neural network can be constructed for decoding z_i into $\tilde{\boldsymbol{x}}_i$.

Mathematically, for $\mathrm{Enc}(\boldsymbol{x}_i)$, we have

$$\mathrm{Enc}(\boldsymbol{x}_i) = q_\theta(z_i\,|\,\boldsymbol{x}_i) \sim \mathcal{N}\big(\tilde{\mu}(\boldsymbol{x}_i), \tilde{\Sigma}(\boldsymbol{x}_i)\big). \tag{12}$$

According to (2) and (3), $\tilde{\mu}(\boldsymbol{x}_i)$ and $\tilde{\Sigma}(\boldsymbol{x}_i)$ should therefore be encoded through the following forms by the linear neural network

$$\begin{cases}\tilde{\mu}(\boldsymbol{x}_i) = \boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\\ \tilde{\Sigma}(\boldsymbol{x}_i) = \boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\end{cases} \tag{13}$$

where $\boldsymbol{w}_1$ is the weight of the first layer in the encoder neural network, $\boldsymbol{w}_2$ and $\boldsymbol{w}_3$ are the weight of the second layer in the encoder neural network, and σ is the nonlinear activation function. The latent representation z_i can then be expressed as follows:

$$z_i = \tilde{\mu}(\boldsymbol{x}_i) + \tilde{\Sigma}(\boldsymbol{x}_i). \tag{14}$$

Similarly, for $\mathrm{Dec}(z_i)$, we have

$$\mathrm{Dec}(z_i) = p(\boldsymbol{x}_i\,|\,z_i) \sim \mathcal{N}(\mu(\boldsymbol{x}_i), \Sigma(\boldsymbol{x}_i)). \tag{15}$$

According to (2) and (13), $\mu(\boldsymbol{x}_i)$ and $\Sigma(\boldsymbol{x}_i)$ should be decoded as the following form by the neural network:

$$\begin{aligned}\mu(\boldsymbol{x}_i) &= \boldsymbol{w}_5\sigma(z_i\boldsymbol{w}_4)\\ &= \boldsymbol{w}_5\sigma((\boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1) + \boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1))\boldsymbol{w}_4) \end{aligned} \tag{16}$$

$$\begin{aligned}\Sigma(\boldsymbol{x}_i) &= \boldsymbol{w}_6\sigma(z_i\boldsymbol{w}_4)\\ &= \boldsymbol{w}_6\sigma((\boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1) + \boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1))\boldsymbol{w}_4) \end{aligned} \tag{17}$$

where $\boldsymbol{w}_4$ is the weight of the first layer in the decoder neural network, $\boldsymbol{w}_5$ and $\boldsymbol{w}_6$ are the weight of second layer in the decoder neural network, and the activation function σ here is the same activation function as that of the encoder neural network.

The training target function $\arg\max_\theta \mathrm{ELBO}(\theta)$ for our model was obtained by combining (10), (11), (16), and (17)

$$\sum_{i,k}\left[\begin{gathered}\left(x_i^k - \mathrm{Dec}\big(\mathrm{Enc}\big(x_i^k\big)\big)\right)^2\\ +\frac{1}{2}\left(\begin{gathered}\left(\boldsymbol{w}_6\sigma\left(\left(\begin{gathered}\boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\\ +\boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\end{gathered}\right)\boldsymbol{w}_4\right)\right)_k\\ +\left(\left(\boldsymbol{w}_5\sigma\left(\left(\begin{gathered}\boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\\ +\boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\end{gathered}\right)\boldsymbol{w}_4\right)\right)_k\right)^2 - 1\\ -\log\left(\boldsymbol{w}_6\sigma\left(\left(\begin{gathered}\boldsymbol{w}_2\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\\ +\boldsymbol{w}_3\sigma(\boldsymbol{x}_i\boldsymbol{w}_1)\end{gathered}\right)\boldsymbol{w}_4\right)\right)_k\end{gathered}\right)\end{gathered}\right]. \tag{18}$$

The VAE is a neural network used for generating artificial data that have a similar distribution to a given dataset. The training process involves iterating the maximization of ELBO: $\arg\max_\theta \mathrm{ELBO}(\theta)$ [as expressed in (18)] until the network produces appropriate weights $\boldsymbol{w}_i$ for the training dataset. During this process, the weights record information about the data distribution. This allows the VAE to generate artificial data (represented by $\tilde{z}_i$) with a distribution similar to the original dataset $\boldsymbol{x}$ [as shown in (16) and (17)]. This process ensures that the VAE can successfully generate artificial data that closely match the original dataset. Once trained, the artificial feature generator can create artificial device features, and the artificial label generator can generate artificial device performance. For a better understanding of the training process and usage of the VAE model, refer to Fig. 2.

2) Artificial Label Generator: The artificial label generator is a critical component of the data generation process, responsible for assigning device performances to the data generated by the artificial feature generator. Since the artificial and real features occupy the same space, the artificial performance data can be assumed to be in the same neighborhood as the real performance data. To generate performance values for artificial features, the nearest neighbor algorithm is employed to analyze the real performance data. This algorithm compares the artificial features with the real feature dataset to find the most similar real features and then assigns the corresponding performance value to the artificial features. In this way, the artificial label generator ensures that the generated artificial device performance closely matches the performance values of real devices. The detailed procedure for using the nearest neighbor algorithm to assign performance values to artificial features is explained as follows.

First, we calculate all the distances between artificial features and real features by an Euclidean metric [29]

$$d\left(\grave{\boldsymbol{x}}_i, \boldsymbol{x}_j\right) = \sum_{k=1}^{M} \left(\grave{\boldsymbol{x}}_{ik} - x_{jk}\right)^2 \quad (19)$$

where $d(\grave{\boldsymbol{x}}_i, \boldsymbol{x}_j)$ denotes the distance between the ith artificial feature $\grave{\boldsymbol{x}}_i$ and the jth real feature $\boldsymbol{x}_j$ and M is the dimension of $\boldsymbol{x}_j$.

Then, we introduce this distance into the performance data space. The nearest neighbor algorithm is applied to calculate the outputs of artificial features using the distance of features as follows:

$$\grave{y}_i = \frac{\sum_{r=1}^{S} \frac{1}{d(\grave{\boldsymbol{x}}_i, \boldsymbol{x}_r)} y_r}{\sum_{r=1}^{S} \frac{1}{d(\grave{\boldsymbol{x}}_i, \boldsymbol{x}_r)}} \quad (20)$$

where $\grave{y}_i$ is the generated output of the ith artificial feature $\grave{\boldsymbol{x}}_i$, S denotes the top S nearest neighbors to $\grave{\boldsymbol{x}}_i$, and y_r is the real output value of the rth nearest real feature. The artificial label generator first calculates the distance between an artificial vector and its nearest real data vectors.

Then, the generator creates the corresponding artificial label by evaluating the real labels of its nearest neighbors (regarding the calculated distances) as expressed in (20). Note that the VAE models for AlGaN/GaN, n-GaN, and p-GaN data were trained separately in this article, considering the intrinsic differences in the materials.

C. Methods for Verification

A DNN-based regression model is adopted in this article to verify the artificial augmented data (Fig. 1, sector IV). The model can be described in the following form:

$$\begin{cases} \boldsymbol{y} = \sigma(\boldsymbol{H}\boldsymbol{w_h} + \boldsymbol{b_h}) \\ \boldsymbol{H} = \sigma(\boldsymbol{X}\boldsymbol{w_i} + \boldsymbol{b_i}) \end{cases} \quad (21)$$

where $\boldsymbol{y} \in \boldsymbol{R}^m$ are the measured values, $\boldsymbol{w_h} \in \boldsymbol{R}^n$ are the weight of the hidden layer, $\boldsymbol{b_h} \in \boldsymbol{R}^n$ is the bias of hidden layer, $\boldsymbol{w_i} \in \boldsymbol{R}^t$ are the weight of input layers, $\boldsymbol{b_i} \in \boldsymbol{R}^t$ is the bias of input layer, $\boldsymbol{X} \in \boldsymbol{R}^{m\times n}$ is the data matrix combined with the real samples and artificial samples, and σ is the nonlinear activation function.

This model contains four layers, including an input layer for inputting the real features or artificial features, two hidden layers, and an output layer. The input layer has the same number of neurons as the length of those device features. The output layer contains only one neuron for predicting the device's electric performance. The two hidden layers that have more than 50 units are designed to fit the complex relationships between device features and their performance.

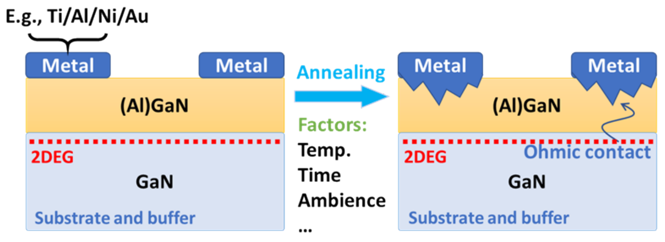

Fig. 3. Typical ohmic contact structure of GaN device and the simplified fabrication process flow.

We used an experimental dataset of metal–semiconductor ohmic contact resistance, extracted from fabricated n-type GaN, p-type GaN, and AlGaN/GaN heterojunction devices (the dataset details can be found in our previous work [19]). The device structure and the process are represented in Fig. 3. The dataset includes resistance values and their corresponding fabrication recipes, such as metal layers, annealing temperature, annealing time, and annealing gas. This dataset is ideal for verifying our proposed self-augmentation model. This is because the dataset has few data points, a complicated fabrication process (also consuming significant foundry time), and low fabrication recipe variation and is difficult to simulate in TCAD.

Before training the DNN for the prediction task, we generated ten different scales of augmentations individually. These ranged from the same number of data points as the training dataset to ten times more. We then trained a DNN-based network for ohmic resistance prediction using a batch of random combinations of experimental and artificial data. We then tested its performance against the test data, which exclusively consisted of experimental data. Each DNN model was trained using augmented data at different scales, and we performed this process five times to ensure the accuracy. For comparison, we used Gaussian noise augmentation as a control group. These noise-based data are generated by adding noise from the standard Gaussian distribution to the experimental data. Note that the DNN models for AlGaN/GaN, n-GaN, and p-GaN data were trained separately.

To evaluate the performance of the augmented dataset and the model, we measured the MAE of the prediction using a well-trained DNN-based ML model. The test set is from experimental data only, so we believe that benchmarking the final MAE provides a reasonable strategy for assessing the VAE model's performance. Validation of results could be improved by comparison to new experimental data, which will be the focus of future work. To avoid bias toward the augmented data, we only used real experimental data for testing. There is, however, no standard index or figure of merit to evaluate the artificially generated data. Therefore, we used three steps to evaluate our augmentation strategy as follows.

1) We mapped the augmented data into a lower dimension with experimental data for intuitive visualization.

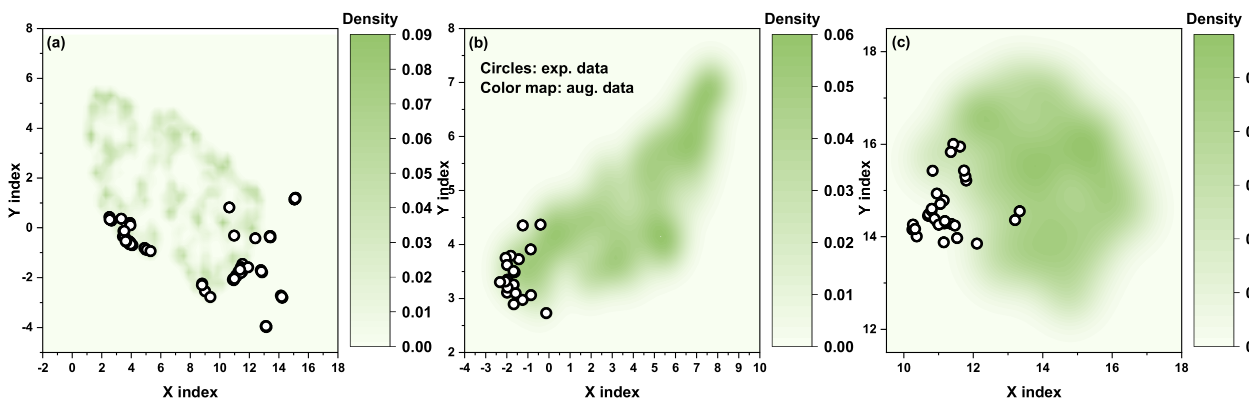

Fig. 4. Visualization of experimental data and augmented data, projected to a 2-D plane through a UMAP algorithm, for the resistance value of ohmic contacts on (a) AlGaN/GaN heterojunction, (b) n-type GaN, and (c) p-type GaN substrates. All data for training the VAE were extracted from experiments (for dataset details see our previous work [19]).

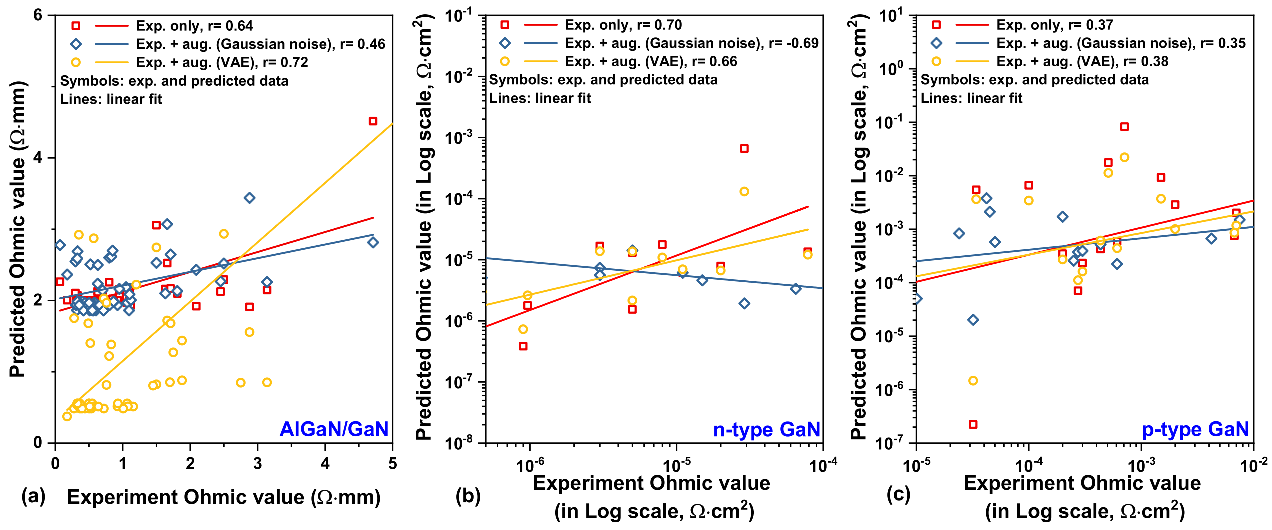

Fig. 5. Pearson *r* of the predicted and experimental ohmic resistance values of (a) AlGaN/GaN, (b) n-type GaN, and (c) p-type GaN substrates. The augmented data are ten times larger than the experimental data. Note: fitting lines are from individual prediction process, which is not averaged. For the details of the mean values, kindly refer to Fig. 6.

2) We analyzed the Pearson r to evaluate the similarity between the prediction results from the augmented data and the experimental data.
3) We evaluated the MAE of the prediction task using augmented data.

III. Results and Discussion

A. Augmented Data Visualization

Fig. 4 shows the kernel density plots that organize the distributions of the generated data with the real data. The uniform manifold approximation and projection (UMAP) algorithm was used to project the data from a high-dimensional parameter space into a 2-D plane [30]. The artificially generated data are shown as a density color map, and the real data are shown as circles. It is observed that the artificially generated data are located in proximity to the real data. This indicates that it carries realistic information similar to the real data. In addition, in Fig. 4, the high-concentration positions of the real data do not largely overlap the augmented data. This implies that the augmented data do not repeat the real data's pattern but (to some extent) compensates for the insufficient real data. The augmented data therefore can extend the occupied area in the data space, with a deliberate pattern, providing more comprehensive sampling points for ML-based tasks.

Furthermore, it can be observed that the kernel density of the augmented data is not condensed altogether in a small range but is dispersed over a large region. This observation suggests that the established augmentation model has successfully extracted the realistic patterns from the experimental data. It also suggests that it has reasonably generated a more comprehensive artificial pattern containing sufficient additional realistic information.

B. Pearson r of Augmented Data

To investigate and evaluate the proposed augmentation model, we plotted the ohmic resistance values obtained from predictions and experiments in Fig. 5. These predicted values were generated by the DNN-based prediction model. We discuss the evaluation of this model and its training process in the next section. Note that in Fig. 5, both the VAE-augmented

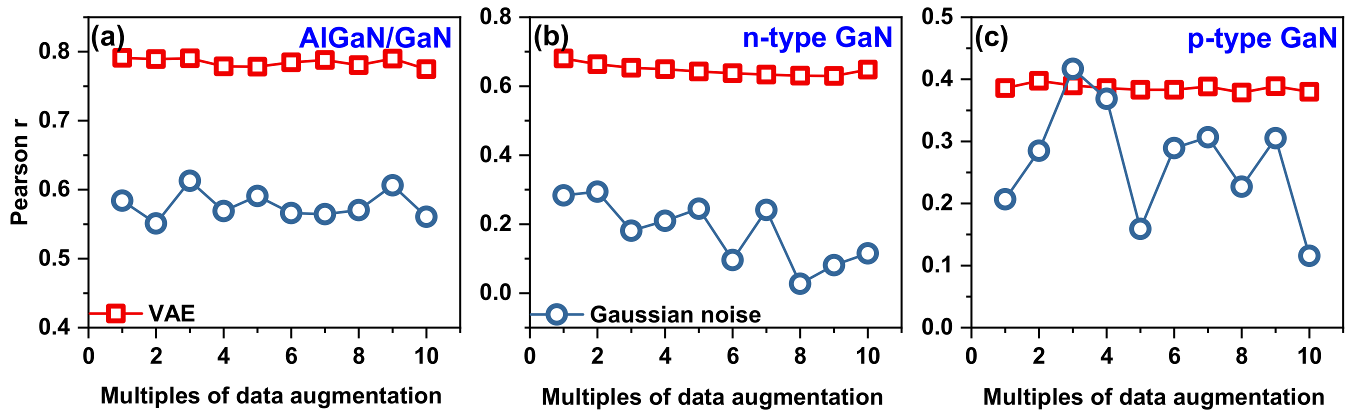

Fig. 6. Mean values of Pearson *r* (reflecting the correlation between the real data and the predicted data) using VAE-based model, noise-based model, and duplicating the experimental data (the test set) with different augmentation scales for (a) AlGaN/GaN heterojunction, (b) n-type GaN, and (c) p-type GaN substrates. Note: each data point is averaged from fivefold cross validation.

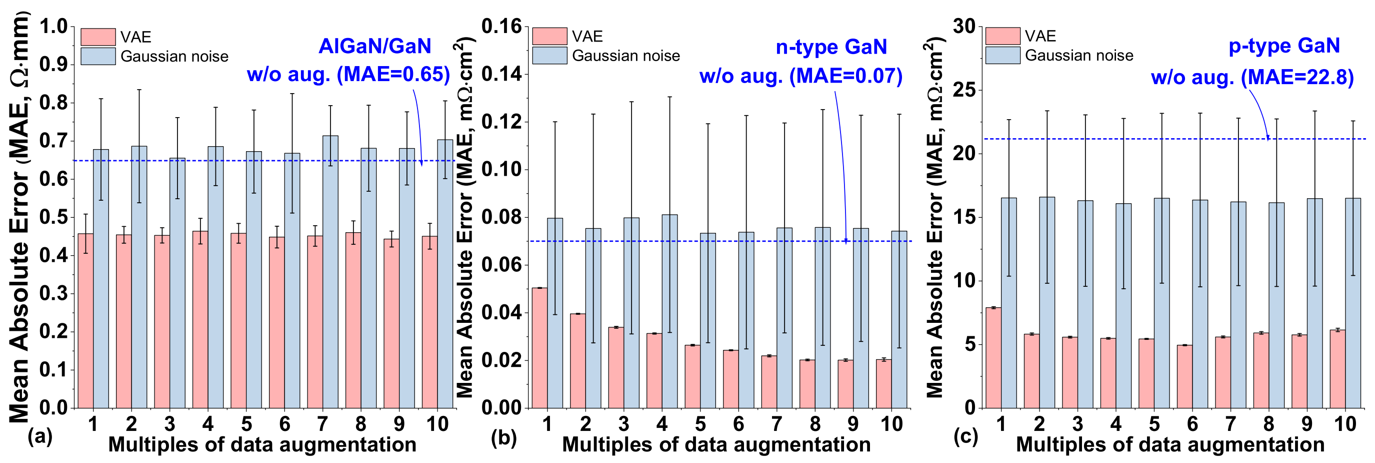

Fig. 7. MAE of the prediction after five times testing, using pure experimental data, and different sizes of VAE- and noise-based data on (a) AlGaN/GaN, (b) n-type GaN, and (c) p-type GaN substrates. Each bar represents five repeated testing processes.

and Gaussian noise augmented datasets have ten times more data points than the training data.

In Fig. 5, we observe that for all three types of substrates, the Pearson correlation coefficient of the model using VAE-augmented data is higher than that using Gaussian noise augmented data. The Gaussian noise can even result in a negative slope of the correlation sometimes (see n-GaN data). This is because augmenting the data with Gaussian noise may introduce patterns that are opposite to the actual data, and these patterns are subsequently learned by the ANN. Moreover, the Pearson correlation coefficient of the augmented data in all three groups is similar to real experiment data, indicating that the generated augmentation data provide sufficient information similar to the real data.

The Pearson correlation coefficient of VAE-based data and noise-based data versus the augmentation scales (in multiples) is shown in Fig. 6 (mean values), where all but one point in the p-GaN group have higher r-index values for VAE-based data than for noise-based data. We observe a trend in Fig. 6(b) and (c), where the Pearson correlation coefficient decreases with an increase in noise-based data points. This indicates that more noise-based data points lead to lower relativity, as the noise dilutes the data pool and fades the realistic information. This trend is not apparent in Fig. 6(a), possibly due to the relatively sophisticated experimental data pool of the AlGaN/GaN group, where the augmented noise cannot significantly alter the data pattern. On the contrary to the noise-based data, the Pearson correlation coefficient of VAE-based data remains stable during ten times of multiplication, indicating that the proposed augmentation model does not inject any negative influence on the data pattern during augmentation. Thus, the data pool is not observed to be diluted.

C. MAE of Prediction Task

Fig. 7 shows the MAE of the VAE- and noise-based augmented data. The VAE-based model outperforms the noise-based model in all three device groups. For n-type GaN and p-type GaN, the MAE of the VAE-based model decreases initially and then flattens as the size of the augmented data increases. The MAE level of the noise-based model remains the same as the pure experimental data (without any augmentation, gray dashed lines) except for showing small fluctuations. This difference in behavior is attributed to the different levels of contribution of additional realistic information provided by VAE- and noise-based data. The VAE-based data successfully exhibit the realistic data pattern, leading to a decrease in

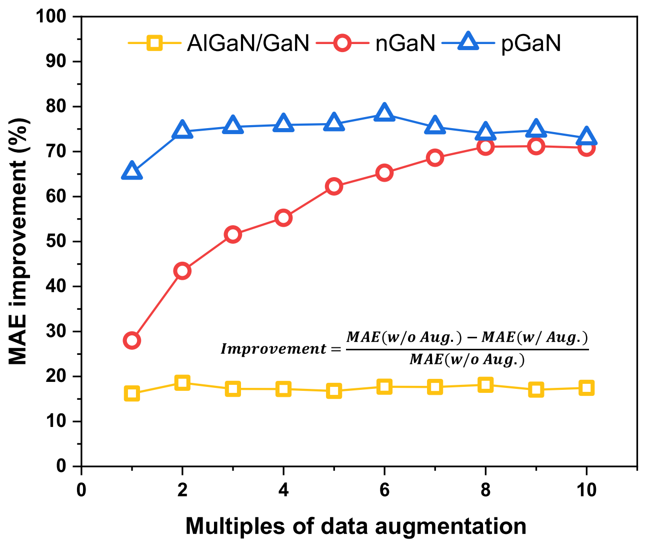

Fig. 8. MAE improvement in test process of different augmentation scales.

MAE. This pattern does, however, feature an accuracy limit (or systematic error), which eventually flattens the MAE at large augmentation scales. Increasing the amount of experimental data used to train the VAE would reduce this error further. Comparatively, the noise-based model can only contribute a random pattern to the DNN model, resulting in a shift in MAE. Notably, the error bar of the MAE of the noise-based data spans a huge range, whereas the VAE-based data provide a more confined MAE distribution in each multiple. This suggests that the augmented data from the VAE-based model are more patterned and less random than the noise-based data.

Interestingly, in Fig. 7(a), the MAE of the VAE-based data does not show the same trend as its counterparts in Fig. 7(b) and (c). Instead, it exhibits similar features to the noise-based data, although the mean MAE remains lower than the noise group. The reason for this could be that the size of the experimental data in this group is larger than in the other groups, and the augmentation model is not robust enough to extract sufficient patterns from such a large dataset. Alternatively, the augmentation model may only be able to extract partial information from the training data. This could especially be the case when the experimental data of this group are following several different patterns, as can be seen in Fig. 4(a) where the circles are more dispersed than in the other two groups.

Fig. 8 provides more intuitive results of the MAE improvement provided by the augmentation. The augmented data significantly improve the prediction performance in the n-GaN and p-GaN groups, with an improvement of over 70%.

Although the proposed augmentation significantly improves the MAE of the modeling, the factors that contribute most to the enhancement are still not clear and require further exploration. It is understood that the size of the experimental data will strongly influence the modeling results. Also, this study suggests that the nature of the distribution of the experimental data may play a key role in this regard. Moreover, the source of the performance difference between AlGaN/GaN-type and other types of data could also be attributed to the features of the experimental data distribution, and this warrants further investigation.

Furthermore, exploring the in-depth mechanisms behind the superior performance of the VAE is indeed an important aspect. However, this study's aim is to showcase the advantages of generating more data at a lower cost, as it reduces the reliance on the limited old dataset. In this context, achieving higher accuracy using lower cost data already fulfills the expectations of this study.

IV. Conclusion

We have proposed and tested a VAE-based data self-augmentation strategy to relieve the contradiction between the accuracy and the insufficient training data in ML-based semiconductor device modeling. In this strategy, no additional TCAD simulation is required and only a few experimental data points are needed for functionality. The testing suggests that the established augmentation model could successfully extract realistic patterns from the experimental data, leading to a set of high-quality augmented data that were able to be seamlessly fed into the DNN model used. As a result, this strategy could significantly improve the performance of the DNN model, where a maximum of more than a 70% drop of MAE was obtained. We therefore believe that this strategy could benefit the next-generation EDA simulations and modeling in the semiconductor industry.

Acknowledgment

Zeheng Wang would like to thank the support for the preliminary results of this work from Prof. Arne Laucht and the University of New South Wales.

References

[1] D. Macmillen, R. Camposano, D. Hill, and T. W. Williams, "An industrial view of electronic design automation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 12, pp. 1428–1448, Dec. 2000, doi: 10.1109/43.898825.

[2] T. Ma, V. Moroz, R. Borges, and L. Smith, "TCAD: Present state and future challenges," in *IEDM Tech. Dig.*, San Francisco, CA, USA, Dec. 2010, pp. 15.3.1–15.3.4, doi: 10.1109/IEDM.2010.5703367.

[3] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. Vienna, Austria: Springer, 1984. [Online]. Available: https://www.google.com.au/books/edition/Analysis_and_Simulation_of_Semiconductor/EE4HlRZTYi4C?hl=en

[4] Y.-C. Wu and Y.-R. Jhan, "Introduction of synopsys sentaurus TCAD simulation," in *3D TCAD Simulation for CMOS Nanoeletronic Devices*. Singapore: Springer, 2018, pp. 1–17, doi: 10.1007/978-981-10-3066-6_1.

[5] K. Mehta, S. S. Raju, M. Xiao, B. Wang, Y. Zhang, and H. Y. Wong, "Improvement of TCAD augmented machine learning using autoencoder for semiconductor variation identification and inverse design," *IEEE Access*, vol. 8, pp. 143519–143529, 2020, doi: 10.1109/ACCESS.2020.3014470.

[6] H. Carrillo-Nuñez, N. Dimitrova, A. Asenov, and V. Georgiev, "Machine learning approach for predicting the effect of statistical variability in Si junctionless nanowire transistors," *IEEE Electron Device Lett.*, vol. 40, no. 9, pp. 1366–1369, Sep. 2019, doi: 10.1109/LED.2019.2931839.

[7] N. Hari, M. Ahsan, S. Ramasamy, P. Sanjeevikumar, A. Albarbar, and F. Blaabjerg, "Gallium nitride power electronic devices modeling using machine learning," *IEEE Access*, vol. 8, pp. 119654–119667, 2020, doi: 10.1109/ACCESS.2020.3005457.

[8] A.-D. Huang, Z. Zhong, W. Wu, and Y.-X. Guo, "An artificial neural network-based electrothermal model for GaN HEMTs with dynamic trapping effects consideration," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 8, pp. 2519–2528, Aug. 2016, doi: 10.1109/TMTT.2016.2586055.

[9] M. Usman, Y. Z. Wong, C. D. Hill, and L. C. L. Hollenberg, "Framework for atomic-level characterisation of quantum computer arrays by machine learning," *npj Comput. Mater.*, vol. 6, no. 1, p. 19, Mar. 2020, doi: 10.1038/s41524-020-0282-0.

[10] K. Mehta and H.-Y. Wong, "Prediction of FinFET current-voltage and capacitance-voltage curves using machine learning with autoencoder," *IEEE Electron Device Lett.*, vol. 42, no. 2, pp. 136–139, Feb. 2021, doi: 10.1109/LED.2020.3045064.

[11] L. Zhang and M. Chan, "Artificial neural network design for compact modeling of generic transistors," *J. Comput. Electron.*, vol. 16, no. 3, pp. 825–832, Sep. 2017, doi: 10.1007/s10825-017-0984-9.

[12] Y. S. Bankapalli and H. Y. Wong, "TCAD augmented machine learning for semiconductor device failure troubleshooting and reverse engineering," in *Proc. Int. Conf. Simulation Semiconductor Processes Devices (SISPAD)*, Sep. 2019, pp. 1–4, doi: 10.1109/SISPAD.2019.8870467.

[13] S. S. Raju, B. Wang, K. Mehta, M. Xiao, Y. Zhang, and H.-Y. Wong, "Application of noise to avoid overfitting in TCAD augmented machine learning," in *Proc. Int. Conf. Simul. Semiconductor Processes Devices (SISPAD)*, Sep. 2020, pp. 351–354, doi: 10.23919/SISPAD49475.2020.9241654.

[14] H. Dhillon, K. Mehta, M. Xiao, B. Wang, Y. Zhang, and H. Y. Wong, "TCAD-augmented machine learning with and without domain expertise," *IEEE Trans. Electron Devices*, vol. 68, no. 11, pp. 5498–5503, Nov. 2021, doi: 10.1109/TED.2021.3073378.

[15] K. Sheelvardhan, S. Guglani, M. Ehteshamuddin, S. Roy, and A. Dasgupta, "Machine learning augmented compact modeling for simultaneous improvement in computational speed and accuracy," *IEEE Trans. Electron Devices*, early access, Mar. 9, 2023, doi: 10.1109/TED.2023.3251296.

[16] S.-K.-S. Fan, C.-Y. Hsu, C.-H. Jen, K.-L. Chen, and L.-T. Juan, "Defective wafer detection using a denoising autoencoder for semiconductor manufacturing processes," *Adv. Eng. Informat.*, vol. 46, Oct. 2020, Art. no. 101166, doi: 10.1016/j.aei.2020.101166.

[17] D.-Y. Liao, C.-Y. Chen, W.-P. Tsai, H.-T. Chen, Y.-T. Wu, and S.-C. Chang, "Anomaly detection for semiconductor tools using stacked autoencoder learning," in *Proc. Int. Symp. Semiconductor Manuf. (ISSM)*, Dec. 2018, pp. 1–4, doi: 10.1109/ISSM.2018.8651179.

[18] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019, doi: 10.1561/2200000056.

[19] Z. Wang, L. Li, and Y. Yao, "A machine learning-assisted model for GaN ohmic contacts regarding the fabrication processes," *IEEE Trans. Electron Devices*, vol. 68, no. 5, pp. 2212–2219, May 2021, doi: 10.1109/TED.2021.3063213.

[20] J. Xu et al., "A review on AI for smart manufacturing: Deep learning challenges and solutions," *Appl. Sci.*, vol. 12, no. 16, p. 8239, Aug. 2022, doi: 10.3390/app12168239.

[21] E. Afacan, N. Lourenço, R. Martins, and G. Dündar, "Review: Machine learning techniques in analog/RF integrated circuit design, synthesis, layout, and test," *Integration*, vol. 77, pp. 113–130, Mar. 2021, doi: 10.1016/j.vlsi.2020.11.006.

[22] T.-L. Wu and S. B. Kutub, "Machine learning-based statistical approach to analyze process dependencies on threshold voltage in recessed gate AlGaN/GaN MIS-HEMTs," *IEEE Trans. Electron Devices*, vol. 67, no. 12, pp. 5448–5453, Dec. 2020, doi: 10.1109/TED.2020.3032634.

[23] A. V. Uriarte-Arcia, I. López-Yáñez, and C. Yáñez-Márquez, "One-hot vector hybrid associative classifier for medical data classification," *PLoS ONE*, vol. 9, no. 4, Apr. 2014, Art. no. e95715.

[24] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, Apr. 2016, doi: 10.1016/j.neucom.2015.08.104.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[26] F. Perez-Cruz, "Kullback–Leibler divergence estimation of continuous distributions," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 1666–1670, doi: 10.1109/ISIT.2008.4595271.

[27] M. D. Hoffman and M. J. Johnson, "ELBO surgery: Yet another way to carve up the variational evidence lower bound," in *Proc. Workshop Adv. Approx. Bayesian Inference, NIPS*, 2016, p. 2.

[28] J. J. Ruel and M. P. Ayres, "Jensen's inequality predicts effects of environmental variation," *Trends Ecol. Evol.*, vol. 14, no. 9, pp. 361–366, 1999, doi: 10.1016/S0169-5347(99)01664-X.

[29] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1334–1339, Aug. 2005, doi: 10.1109/TPAMI.2005.165.

[30] R. M. Parra-Hernández, J. I. Posada-Quintero, O. Acevedo-Charry, and H. F. Posada-Quintero, "Uniform manifold approximation and projection for clustering taxa through vocalizations in a neotropical passerine (rough-legged tyrannulet, Phyllomyias burmeisteri)," *Animals*, vol. 10, no. 8, p. 1406, Aug. 2020, doi: 10.3390/ani10081406.