Improving Machine Learning-Based Modeling of Semiconductor Devices by Data Self-Augmentation

Zeheng Wang, Liang Li, Ross C. C. Leon, and Arne Laucht

Abstract—In the electronics industry, introducing Machine Learning (ML)-based techniques can enhance Technology Computer-Aided Design (TCAD) methods. However, the performance of ML models is highly dependent on their training datasets. Particularly in the semiconductor industry, given the fact that the fabrication process of semiconductor devices is complicated and expensive, it is of great difficulty to obtain datasets with sufficient size and good quality. In this paper, we propose a strategy for improving ML-based device modeling by data self-augmentation using variational autoencoder-based techniques, where initially only a few experimental data points are required and TCAD tools are not essential. Taking a deep neural network-based prediction task of the Ohmic resistance value in Gallium Nitride devices as an example, we apply our proposed strategy to augment data points and achieve a reduction in the mean absolute error of predicting the experimental results by up to 70%. The proposed method could be easily modified for different tasks, rendering it of high interest to the semiconductor industry in general.

Index Terms—Semiconductor Devices; Machine Learning; Data Augmentation; Gallium Nitride.

I. INTRODUCTION

ELECTRONIC Design Automation (EDA) has played a critical role in boosting developments in the electronics industry in the past several decades, by significantly reducing the design complexity and time consumption of the related tasks. Particularly in the area of semiconductor devices one of the EDA techniques – Technology Computer-Aided Design (TCAD) using the finite element method to consistently solve basic physics equations [1], such as the Poisson and Schrödinger equations – has liberated researchers and engineers from solving extraordinarily complicated math problems, and providing easy access to simulated results that can hardly be solved manually [2]. Moreover, TCAD has also paved a successful and efficient way to dramatically shrink the cost of experiments during device design, so that in most cases it is no longer necessary to fabricate numerous batches of devices in a cleanroom in order to optimize just one geometrical parameter.

However, with the rapid development of the semiconductor device industry TCAD starts to encounter many formidable challenges. Simulating the complex three-dimensional (3D) device structures consumes considerable computational resources, and strains the computational capability of conventional computer to its limit. Although many excellent models and methods have been developed to reduce the computational resource consumption and shorten the turn-around time (TAT) [3], exploring novel methodologies of TCAD is still a burning issue to relieve the contradiction between the accuracy and time consumption of sophisticated physics simulations (e.g. quantum effects). While large-scale, programable quantum computers might act as remedy in the future, they still have to overcome many barriers first [4].

To this end, Machine Learning (ML)-based solutions have attracted strong attention and have been successfully employed in many cases of device modeling [5]–[7]. All these promising applications benefit from the advantage of ML-based modeling, that is, the huge computational resource is only required in model training where a large amount of the neural units [8] or the regression weights [9] are set and calibrated by multiple algorithms. After the training, the model can work efficiently in a low resource consumption mode [10], and the required computational resource for semiconductor device modeling is within the capability of nowadays computer clusters. In other words, a compromise between the simulation efficiency and the computational capability in ML-based semiconductor modeling has already been achieved.

Nevertheless, the industry is approaching the end of Moore’s law [11], which means to learn the high-dimension relationships more comprehensively and exclusively between the inputs and outputs, it is of great importance to add more neurons or regression items into the model. As an unfortunate result, the input data instead of the computational resource that are more and more desperately needed to fully complete the model training [12]. However, due to the resource overhead in fabricating and testing devices, collecting a sufficient amount of experimental data becomes a big challenge. In short, experimental data are no longer sufficient to substantially

Zeheng Wang, Ross C. C. Leon, and Arne Laucht are with the School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney, New South Wales 2052, Australia (e-mail: zenwang@outlook.com).

Liang Li is with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China.

Corresponding author: Zeheng Wang.
complete model training.

This problem might potentially be alleviated by data augmentation, which is a technique that has attracted intensive attention in the semiconductor industry since 2019. Many intriguing works using TCAD-based data augmentation have been reported since then [13], [14], where experiment-calibrated TCAD has been introduced to generate the artificial data that could be fed into ML-based models. This approach could indeed provide an extreme boost to ML-based modeling, within the TCAD industry’s development. However, in the semiconductor industry, there still exist some problems that cannot be directly solved by TCAD, for example, the simulation of the formation of ohmic contacts in Gallium Nitride (GaN) devices. As a result, this also imposes a formidable challenge on the TCAD-based augmentation technique.

In this paper, we propose a Variational Autoencoder (VAE)-based data self-augmentation strategy to extend the size of the dataset, to improve the performance of ML-based modeling task of the semiconductor devices without requiring more experiments or calibration of TCAD tools. A modeling task of GaN ohmic contacts was introduced to verify the proposed method. The results show that after using the proposed strategy to augment data points, the mean absolute error of the prediction drops by up to 70%, indicating the proposed strategy is promising for device simulation and other applications in the semiconductor industry.

II. GUIDELINES FOR MANUSCRIPT PREPARATION

Generally, ML-based semiconductor device modeling could be simply considered as a procedure that links the device features (i.e., gate length, drain voltage, and annealing temperature) to the electrical performances (i.e., surface potential, saturation current, and on/off ratio). To explicitly demonstrate this procedure, the proposed data augmentation framework is separated into three main parts in this work: feature preprocessing module (sector I and II), data augmentation module (sector III), and ML-based modeling module (sector IV), which are schematically depicted in Fig. 1. As shown in the sector I of Fig. 1, the first step of modeling is to extract parameter data, namely device features, from the experiment results. After being extracted from real experimental samples, the data containing device features are transferred into vectors by the feature preprocessing module, as shown in the sector II of Fig. 1. Then, the whole transferred real experimental data are split into two parts, the training set (67% of the whole data) and the test set (33% of the whole data). The training set is used to build the generative models to realize the augmentation, as shown in sector III of Fig. 1. Then, this set is combined with the generated artificial data (augmented data) and fed into the deep neural network (DNN)-based model to realize semiconductor device modeling. Another part of the experimental data, the test set, is accordingly used to evaluate the generalization performance of that DNN model and the augmentation model.

For this work, the whole procedure of the augmentation and the verification could also be described step by step as follows and the technical details are described in the following subsections:

1) Extracting feature data of experiments and splitting the data into the test set and the training set.
2) Using the training set to train the VAE-based model.
3) Generating the artificial data and combine these data with the training set.
4) Using the combined dataset to train the corresponding DNN models.
5) Using the test set to test those two DNN models to calculate the Mean Absolute Error (MAE).
6) Repeating steps 2) – 5) five times and calculate the average MAE of these loops for discussion.

A. Feature Pre-processing

To minimize the influence of missing data points, all experimental data are filtered by a ‘data cleaning’ process such that the whole vector will be deleted if any vacancy exists. Then the experimental data are 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 are then transformed into a numerical vector using one-hot encoding [15], [16]. After encoding, each dimension of this numeric vector is centered at 0 with a standard deviation of 1 on each dimension. For the numerical input, the item $x$ of the data are directly standardized by Z-score to ensure the numerical input are centered to 0 with a standard deviation of 1 as well, following the equation below:

$$ z = \frac{x - \mu_x}{\sigma_x} $$ (1)
where \( \mu_x \) is mean of \( x \) and \( \sigma_x \) is the standard deviation of \( x \). Note that for simplicity the ohmic value data described in the following part of this paper are all standardized values but not original values.

B. Generative Model

Overall, the generative model aims to learn the joint probability distribution of a given dataset using unsupervised learning. As shown in Fig. 2 panels I and II, a successfully trained generative model could enable the data augmentation from the experimental data easily by interpolating some variations into the established generative model. In this paper we separate the generator of features and the generator of labels, meaning our generative model consists of two components: artificial feature generator and artificial label generator. For the artificial feature generator, we applied VAE, one of the most simple but powerful deep generative models, for the augmentation task to overcome the problem of insufficient training data in the task. And for artificial label generators, we apply a K-th Nearest Neighbor (KNN) regressor to generate corresponding labels, as a complementary of the VAE model.

1) Artificial Feature Generator

The artificial feature generator is realized by the VAE which is a variant of the automatic encoder combining variational inference with the conventional autoencoder framework, so it also consists of two parts: encoder and decoder. The encoding-decoding process could efficiently realize the dimensionality reduction, which will emphasize the preferred features of the input and suppress the less-important features to minimize the interference \[17\], \[18\]. The encoder encodes, denoted as \( Enc(x) \), input \( x \) into a latent representation \( z \) with a parameter \( \theta \), which is denoted by \( q_\theta(z|x) \), while the decoder reconstructs, denoted as \( Dec(z) \), the data distribution \( \bar{x} \) from the given \( z \), which is depicted as follows:

\[
\begin{align*}
    z &= Enc(x) - q_\theta(z|x) \\
    \bar{x} &= Dec(z) - p(x|z)
\end{align*}
\]

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

\[
p(X) = \prod_{i=1}^{N} p(x_i|z)p(z)
\]

The Eq. (4) can be written in a more general form:

\[
p(X) = \int p(x|z)p(z)dz
\]

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 posterior probability \( p(z|x) \) that could be modeled by a simpler distribution \( q_\theta(z|x) \). Then, the problem is converted into minimizing the difference between those two distributions using KL divergence \[19\]:

\[
KL(q_\theta(z|x)||p(z|x)) = \mathbb{E}[\log q_\theta(z|x) - \log p(x|z) - \log p(z)]
\]

The optimal approximate posterior is thus:

\[
q_\theta(z|x) = \arg \min_{q_\theta} KL(q_\theta(z|x)||p(z|x))
\]

Note that \( p(x) \) appears in the formula, which is intractable as mentioned above. Considering the following function named Evidence Lower Bound (ELBO) \[20\]:

\[
\begin{align*}
    ELBO(\theta) &= \mathbb{E}_{q_\theta} \left[ \log p(x|z) + \log q_\theta(z|x) - \log p(z) \right] \\
    &= \mathbb{E}_{q_\theta} \left[ \log p(x|z) - \log p(z) \right] + \mathbb{E}_{q_\theta} \left[ \log q_\theta(z|x) \right] - \log p(z)
\end{align*}
\]
\[ \text{ELBO} (\theta) = -\mathbb{E}[\log p(x | z) + \log p(z) - \log q_{\theta}(z | x)] \]
\[ = \mathbb{E}[\log p(x | z)] - KL (q_{\theta}(z | x) || p(z)) \]  
(8)

According to the Eqs. (6) and (8), the Eq. (5) can be rewritten as a log likelihood function:
\[ \log p(x) = KL (q_{\theta}(z | x) || p(z)) + \text{ELBO} (\theta) \]  
(9)

Note that \( x \) is known, \( \log p(x) \) is constant. Besides, the KL divergence is always greater than or equal to zero according to Jensen’s inequality [21]. Therefore, minimizing the KL divergence is equivalent to maximizing \( \text{ELBO} (\theta) \), since no datapoint shares its latent \( z \) with another datapoint in VAE. We can write this function for a single datapoint as:
\[ \text{ELBO} (\theta)_i = \mathbb{E}[\log p(x_i | z_i)] - KL (q_{\theta}(z_i | x_i) || p(z_i)) \]  
(10)

where the first term forces the decoder to learn to reconstruct the data from latent representation \( z \), the second term can be viewed as a constraint, which measures the loss of using \( q_{\theta}(z_i | x_i) \) to represent \( p(z_i) \). According to [18], the \( p(z_i) = \mathcal{N}(0, I) \) is assumed as a standard normal distribution, while the \( q_{\theta}(z_i | x_i) = \mathcal{N}(\mu(x_i), \Sigma(x_i)) \), where \( \mathcal{N} \) represents a normal distribution, \( I \) is an identity matrix, \( \mu \) and \( \Sigma \) are arbitrary deterministic functions that can be learned from data. Therefore, the second term \( KL (q_{\theta}(z_i | x_i) || p(z_i)) \) can be rewritten as:
\[ \frac{1}{2} \sum_k [\Sigma(x)_k + \mu^2(x)_k - 1 - \log \Sigma(x)_k] \]  
(11)

In the Eq. (11), the \( \mu(x)_k \) and \( \Sigma(x)_k \) denote the \( k \)-th element of vectors \( \mu(x) \) and \( \Sigma(x) \). Thus, the training target of the VAE is actually approaching the maximum of the ELBO function shown below, which is also labeled in Fig. 2 between the panel I and III:
\[ \arg \max_{\theta} \text{ELBO} (\theta) = \sum_i \text{ELBO} (\theta)_i \]  
(12)

and it could be explicitly expressed by \( \mu \) and \( \Sigma \) as follows:
\[ \sum_k \left[ (\bar{x}_k - \bar{x}_k)^2 + \frac{1}{2} (\Sigma(x)_k + \mu^2(x)_k - 1 - \log \Sigma(x)_k) \right] \]  
(13)

where \( \bar{x}_k \) is the \( k \)-th element of reconstructed data vectors \( \tilde{x}_i \). Considering the encoder and decoder processes, this \( \tilde{x}_i \) can be formulated as the following form:
\[ \tilde{x}_i = \text{Dec} (\text{Enc}(x_i)) \]  
(14)

In this proposed strategy, we use a linear neural network to construct the encoder of the VAE model by which the input \( x_i \) is encoded into the latent representation \( z_i \). Therefore, other linear neural network could also be constructed for decoding \( z_i \) into \( \tilde{x}_i \).

Mathematically, for the encoder \( \text{Enc}(x_i) \) we have:
\[ \text{Enc}(x_i) = q_{\theta}(x_i | x_i) \sim \mathcal{N} (\bar{\mu}(x_i), \bar{\Sigma}(x_i)) \]  
(15)

According to Eq. (2) and (3), \( \bar{\mu}(x) \) and \( \bar{\Sigma}(x) \) should therefore be encoded as the following forms by the linear neural network:
\[ \bar{\mu}(x_i) = w_2 \sigma (x_i w_1 + \nu_1) \]
\[ \bar{\Sigma}(x_i) = w_3 \sigma (x_i w_1 + \nu_2) \]  
(16)

where \( w_1 \) is the weight of the first layer in the encoder neural network. \( w_2 \) and \( w_3 \) are the weight of the second layer in the encoder neural network, \( \sigma \) is the non-linear activation function. Then the latent representation \( z_i \) can be written as:
\[ z_i = \bar{\mu}(x_i) + \bar{\Sigma}(x_i) \]  
(17)

Similarly, for the decoder \( \text{Dec}(z_i) \) we have:
\[ \text{Dec}(z_i) = p(x_i | z_i) \sim \mathcal{N} (\bar{\mu}(x_i), \bar{\Sigma}(x_i)) \]  
(18)

according to Eqs. (2) and (16), \( \bar{\mu}(x_i) \) and \( \bar{\Sigma}(x_i) \) should be decoded as the following form by neural network:
\[ \bar{\mu}(x_i) = w_5 \sigma (z_i w_4) \]
\[ = w_3 \sigma (w_2 \sigma (x_i w_1) + w_3 \sigma (x_i w_1))w_4 \]  
(19)

\[ \bar{\Sigma}(x_i) = w_6 \sigma (z_i w_4) \]
\[ = w_6 \sigma (w_2 \sigma (x_i w_1) + w_3 \sigma (x_i w_1))w_4 \]  
(20)

where \( w_4 \) is the weight of the first layer in the decoder neural network, \( w_5 \) and \( w_6 \) are the weight of second layer in the decoder neural network, the activation function \( \sigma \) here is the same activation function as that of the encoder neural network.

Hence, for our model the training target function \( \arg \max_{\theta} \text{ELBO} (\theta) \) could be obtained by combining Eqs. (13), (14), and Eqs. (19) and (20):
\[ \text{argmax}_{\theta} \text{ELBO} (\theta) = \sum_i \text{ELBO} (\theta)_i \]
\[ + \frac{1}{2} \sum_k \left( \left( \frac{\bar{x}_k - \text{Dec} \left( \text{Enc}(x_i) \right) \right)^2}{w_2 \sigma (x_i w_1) + w_3 \sigma (x_i w_1)} \right) - 1 \]
\[ \sum_{i,k} \left( \left( \frac{\bar{x}_k - \text{Dec} \left( \text{Enc}(x_i) \right) \right)^2}{w_2 \sigma (x_i w_1) + w_3 \sigma (x_i w_1)} \right) - 1 \]  
(21)

The training process of the VAE is iterating this \( \arg \max_{\theta} \text{ELBO}(\theta) \), which has been explicitly expressed in Eq. (21) regarding the proposed tasks, till it yields the proper weights for the training dataset \( x \). In other words, during this training process, the network keeps recording the information of the data distribution into those weights \( w_\theta \). Once the weights have been confirmed, the artificial data \( \tilde{x}_i \) could be generated from a normal distribution and could subsequently be fed into the decoder to create an artificial datapoint (augmented data, not appeared in \( x \)) with similar distribution of \( x \), according to Eqs. (19) and (20). These processes ensure that the proposed VAE could successfully generate the artificial data which feature the proximate distribution of the original data.

After being trained by the experiment data, the artificial feature generator could then be utilized to generate artificial device features. The artificial device performances could be generated by the artificial label generator which will be introduced in the next subsection. The procedure of training, optimizing, and utilizing the VAE model could also be schematically understood through Fig. 2.

2) Artificial label Generator

The artificial label generator is for assigning outputs (device performances) for the data generated by the artificial feature generator (device features). Since the real features and the artificial features share the same space, the artificial performance data could be assumed as neighborhoods of the real performance data in the same space as well. Thus, the nearest neighbor algorithm is applied to generate the performance value for an artificial feature by analyzing the real performance data. The detailed procedure is as follows.

First, we calculate all the distances between artificial features and real features by Euclidean metric [22]:
\[ d(\hat{x}_i, x_j) = \sum_{k=1}^{M} (\hat{x}_{ik} - x_{jk})^2 \]  
(22)
where $d(\hat{x}_i, x_j)$ denotes the distance between $i$-th artificial feature $\hat{x}_i$ and $j$-th real feature $x_j$, $M$ is the dimension of $x_j$.

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

$$\hat{y}_i = \frac{1}{\sum_{r=1}^{S} \frac{1}{d(\hat{x}_i, x_r)}} \sum_{r=1}^{S} \frac{y_r}{d(\hat{x}_i, x_r)}$$  \hspace{1cm} (23)

where $\hat{y}_i$ is the generated output of the $i$-th artificial feature $\hat{x}_i$; $S$ denotes the top $S$ nearest neighbors to the $\hat{x}_i$; and $y_r$ is the real output value of $r$-th nearest real feature. In other words, the artificial label generator first calculates the distance between an artificial vector and its nearest real data vectors and then creates its corresponding artificial label by evaluating the real labels of its nearest neighbors, regarding those calculated distances, as expressed in Eq. (23).

C. Methods for Verification

A DNN-based regression model is adopted in this paper to verify the artificial augmented data, as sketched in Fig. 1 sector IV. The model can be depicted in the following form:

$$(y = \sigma(Hw_b + b_h))$$ \hspace{1cm} (24)

$$(H = \sigma(Xw + b_i))$$

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

For the experimental dataset, we adopted a set of data of metal-semiconductor ohmic contact resistance that are tested and extracted from experimentally fabricated n-type GaN, p-type GaN, and AlGaN/GaN heterojunction devices [16]. The dataset contains the ohmic values and their corresponding fabrication recipes including metal layers, annealing temperature, annealing time, and annealing gas. This dataset has the following features: 1) few data points, 2) complicated fabrication process, 3) low fabrication recipe variation, 4) hard to be simulated in TCAD. Therefore, this dataset is ideal for the verification of the proposed self-augmentation model.

Before training the DNN for the prediction task, ten different scales of augmentations were generated individually, where the multiples (compared to the training dataset) of the number of the augmented data points vary from 1 (the same as the training data) to 10 (ten times more than the training data).

After data augmentation, the DNN-based network for the ohmic resistance prediction was trained by a batch of random combinations of the experimental data and the artificial data and then was tested by comparing it to the test data (exclusively experimental). Each DNN model was trained by the augmented data with different scales, and the performance of this model was tested by the test set. To be scientific, an average of five times this process was then executed and analyzed. For comparison, Gaussian noise augmentation was used as a control group. Ten different scales of augmentations were also generated from standard Gaussian distribution, and the model was trained and tested for each scale, similar to the model for experimental data.

To estimate the performance of the augmented dataset and the model, the solution is to measure the MAE of the prediction through a well-trained DNN-based ML model. The MAE is applied here to measure the performance of the regression model. To avoid the case that the DNN-based model learns some latent patterns of the augmented data, which will lead to a bias to the augmented data, we use only real experimental data for testing.

In general, it is pretty difficult to directly analyze the established augmentation model. Moreover, so far, no standard index or figure-of-merit could be adopted to explicitly evaluate the artificially generated data. Therefore, we here introduce three steps to evaluate our augmentation strategy:

1) Map the augmented data into a lower dimension with experimental data for intuitive visualization to evaluate the augmented data.
2) Analyze the Pearson $r$ to evaluate the similarity between the prediction results from the augmented data and the experimental data.
3) Evaluate the Mean Absolute Error of the prediction task using augmented data.

III. RESULT AND DISCUSSION

A. Augmented Data Visualization

The distributions of generated data with real data are organized as kernel density plots, as shown in Fig. 3, by applying an Uniform Manifold Approximation and Projection (UMAP) algorithm to project the data from a high-dimensional parameter space into a 2-dimensional plane [23]. It can be seen that the artificially generated data (shown as density color map) is located in proximity to the real data (shown as circles), which implies that the generated data carries some realistic information, similar to the real data. Besides, the high-concentration positions of the real data are largely not overlapped by the augmented data in Fig. 3, which means that the augmented data are not repeating the real data’s pattern but, to some extent, compensate the not-sufficient real data. In other words, the augmented data can extend the occupied area in the data space with a deliberate pattern, providing more comprehensive sampling points for ML-based tasks.

Also, it can be observed that the kernel density of the augmented data is not condensed altogether in a small range but dispersed over a big region. This could suggest, from another aspect, that the established augmentation model has successfully extracted the realistic patterns from the experimental data and then reasonably generated a more comprehensive artificial pattern containing enough additional realistic information.
B. Pearson r of Augmented Data

To further investigate and evaluate the proposed augmentation model, the data of the ohmic resistance values from predictions and experiments are plotted in Fig. 4. These predicted Ohmic resistance values are the outputs from the DNN-based prediction model. The evaluation of this model and its training process is discussed in the next subsection. Note, in Fig. 4, for both VAE-augmented and Gaussian noise augmented datasets, their sizes are ten times as many data points as the training data.

In Fig. 4, it can be seen that in all these three types of substrates the Pearson r of the model using the VAE to augment data is higher than the model using the Gaussian noise. Additionally, the Pearson r of augmented data in all three groups are all similar to the real experiment data, implying the generated augmentation data brings sufficient similar information to the real data.

The Pearson r of the VAE-based data and noise-based data versus the augmentation scales (in multiples) are shown in Fig. 5, where except only one point in the p-GaN group all the r-index of VAE-based data are higher than that of noise-based data. A trend that the Pearson r keeps reducing in noise-based data as data pool scaling up could be seen in Fig. 5 (b) and (c). This trend may indicate that more noise-based data points lead to lower relativity because the noise keeps diluting the data pool and equivalently the realistic information keeps fading. However, in Fig. 5 (a) this trend is not apparent. This could be attributed to the experimental data pool of the AlGaN/GaN group that is relatively sophisticated so that the augmented noise cannot obviously change the data pattern. On the contrary, the VAE-based data’s Pearson r is stable during 10 times multiplying, which is as expected, meaning the proposed augmentation model does not inject negative influence on the data pattern during augmentation, or in other words, the data pool is not being diluted.

C. Mean Absolute Error of Prediction Task

Fig. 6 shows the MAE of the augmented data from the VAE-based model and the noise-based model. As can be seen from the figure, the DNN model fed by VAE augmented data outperforms the model fed by Gaussian noise augmented data in all three device groups, in terms of the averaged MAE. Importantly, for n-type GaN and p-type GaN, the MAE of the VAE-based model first decreases then flattens as the size of the augmented data increases, while the MAE level of the noise-
based model stays at the same level as the pure experimental data (without any augmentation, gray dashed lines) remaining unchanged except showing small fluctuations. This could be attributed to the different levels of contribution to the additional realistic information provided from VAE-based and noise-based data. Because the VAE-based data successfully exhibit the realistic data pattern, the MAE therefore drops. And this extracted pattern features an accuracy limit or a systematic error, which flattens the MAE in a larger augmentation scale. Adding more experimental data to train the VAE may further reduce this error.

On the opposite, the noise-based model can only contribute a random pattern into the DNN model, therefore the MAE is not being enhanced but just being shifted. From another aspect, the MAE from the noised-based data spans a huge range, but the VAE-based data could provide a more confined MAE. This proves again that the augmented data from the VAE-based model are more patterned and less random than the noise-based data.

Interestingly, in Fig. 6 (a), the MAE of the VAE-based data does not show the same trend as its counterparts in Fig. 6 (b) and (c) but shows similar features as the noise-based data, although the mean MAE stays lower than the noise group. Given that the size of the experimental data in this group is larger than others, the reason why the augmentation is, to some extent, unsuccessful could be: 1) the augmentation model is not robust enough so that the model cannot extract sufficient patterns from a large dataset; 2) the augmentation model can only extract partial information from the training data especially when the experimental data of this group is following several different patterns, which could be seen in Fig. 3 (a) – compared to other two groups the circles are more dispersed.

Fig. 7 gives more intuitive results of the MAE improvement provided by the augmentation. Except for the AlGaN/GaN group, the augmented data significantly improves the prediction performance particularly in the n-GaN and p-GaN group: the improvement is around 70 percent for both.

Although the proposed augmentation could significantly improve the MAE of the modeling, the influence factors that contribute most to the enhancement are still not clear and need to be explored further. But through this study, it could be suggested that the nature of the distribution of the experimental data may act a key role. Similarly, 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, which is worth studying in the future as well.
Fig. 7. MAE improvement in test process of different augmentation scale.

IV. CONCLUSION

In this research, we proposed 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 can support its functionality. The evidence suggests that the established augmentation model could successfully extract the realistic patterns from the experimental data, leading to a set of high-quality augmented data. As a result, verified by the prediction task using a DNN-based model, the proposed strategy could significantly improve the performance of the DNN model, where a maximum of around 70% drop of MAE had been obtained. Therefore, we believe this strategy could benefit the next generation simulation and modeling in the semiconductor industry.

REFERENCES

[1] T. Ma, V. Moroz, R. Borges, and L. Smith, “TCAD: Present state and future challenges,” in 2010 International Electron Devices Meeting, Dec. 2010, pp. 15.3.1-15.3.4. doi: 10.1109/IEDM.2010.5703367.

[2] Y.-C. Wu and Y.-R. Jhan, “Introduction of Synopsys Sentaurus TCAD Simulation,” in 3D TCAD Simulation for CMOS Nanoelectronic Devices, Singapore: Springer Singapore, 2018, pp. 1–17. doi: 10.1007/978-981-10-3066-6_1.

[3] 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.

[4] A. Laucht et al., “Roadmap on quantum nanotechnologies,” Nanotechnology, vol. 32, no. 16, p. 162003, Apr. 2021. doi: 10.1088/1361-6528/abb333.

[5] H. Carrillo-Nuñez, N. Dinitrova, A. Asenov, and V. Georgiev, “Machine Learning Approach for Predicting the Effect of Statistical Variability in Si Junctionless Nanowire Transistors,” IEEE Electron Device Letters, vol. 40, no. 9, pp. 1366–1369, 2019. doi: 10.1109/LED.2019.2931839.

[6] N. Hari, M. Ahsan, S. Ramasamy, P. Sanjeevikumarak, A. Albarbar, and F. Blaiberg, “Gallium Nitride Power Electronic Devices Modeling Using Machine Learning,” IEEE Access, vol. 8, pp. 119654–119667, 2020. doi: 10.1109/ACCESS.2020.3005437.

[7] 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 Transactions on Microwave Theory and Techniques, vol. 64, no. 8, pp. 2519–2528, 2016. doi: 10.1109/TMTT.2016.2586055.

[8] A. Krogh, “What are artificial neural networks?,” Nature Biotechnology, vol. 26, no. 2, pp. 195–197, 2008. doi: 10.1038/nbt1386.

[9] Y. Liang, D. Niu, and W.-C. Hong, “Short term load forecasting based on feature extraction and improved general regression neural network model,” Energy, vol. 166, pp. 653–663, 2019. doi: https://doi.org/10.1016/j.energy.2018.10.119.

[10] Q. Guo et al., “An Empirical Study Towards Characterizing Deep Learning Development and Deployment Across Different Frameworks and Platforms,” in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 810–822. doi: 10.1109/ASE.2019.00080.

[11] I. L. Markov, “Limits on fundamental limits to computation,” Nature, vol. 512, no. 7513, pp. 147–154, 2014. doi: 10.1038/nature13570.

[12] L. Zhang and M. Chan, “Artificial neural network design for compact modeling of generic transistors,” Journal of Computational Electronics, vol. 16, no. 3, pp. 825–832, 2017. doi: 10.1007/s10825-017-0984-9.

[13] Y. S. Bankapalli and H. Y. Wong, “TCAD Augmented Machine Learning for Semiconductor Device Failure Troubleshooting and Reverse Engineering,” in 2019 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), 2019, pp. 1–4. doi: 10.1109/SISPAD.2019.8870467.

[14] 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 2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), 2020, pp. 351–354. doi: 10.23919/SISPAD49475.2020.9241654.

[15] A. V. Uriarte-Arcia, L. López-Yáñez, and C. Yáñez-Márquez, “One-Hot Vector Hybrid Associative Classifier for Medical Data Classification,” PLOS ONE, vol. 9, no. 4, p. e95715, 2014.

[16] Z. Wang, L. Li, and Y. Yao, “A Machine Learning-Assisted Model for GaN Ohmic Contacts Regarding the Fabrication Processes,” IEEE Transactions on Electronic Devices, pp. 1–8, 2021. doi: 10.1109/TED.2021.3063213.

[17] Y. Wang, H. Yao, and S. Zhao, “Auto-encoder based dimensionality reduction,” Neurocomputing, vol. 184, pp. 232–242, 2016. doi: https://doi.org/10.1016/j.neucom.2015.08.104.

[18] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” arXiv preprint arXiv:1312.6114, 2013.

[19] F. Perez-Cruz, “Kullback-Leibler divergence estimation of continuous distributions,” in 2008 IEEE International Symposium on Information Theory, 2008, pp. 1666–1670. doi: 10.1109/ISIT.2008.4595271.

[20] M. D. Hoffman and M. J. Johnson, “Elbow surgery: yet another way to carve up the variational evidence lower bound,” in Workshop in Advances in Approximate Bayesian Inference, NIPS, 2016, vol. 1, p. 2

[21] J. J. Ruel and M. P. Ayres, “Jensen’s inequality predicts effects of environmental variation,” Trends in Ecology & Evolution, vol. 14, no. 9, pp. 361–366, 1999. doi: https://doi.org/10.1016/S0169-5347(99)01664-X.

[22] L. Wang, Y. Zhang, and J. Feng, “On the Euclidean distance of images,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1334–1339, 2005. doi: 10.1109/TPAMI.2005.165.

[23] 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, Phyllosotila burmeisteri),” Animals, vol. 10, no. 8. 2020. doi: 10.3390/ani10081406.