Generative Modeling of Semiconductor Devices for Statistical Circuit Simulation

Dominik Kasprowicz * and Grzegorz Kasprowicz

Posted Date: 26 April 2024

doi: 10.20944/preprints202404.1675.v1

Keywords: generative model; machine learning; variational autoencoder; VAE; semiconductor device modeling; process variability; MOSFET; Monte Carlo

 doi:10.20944/preprints202404.1675.v1

Article

Generative Modeling of Semiconductor Devices for Statistical Circuit Simulation

Dominik Kasprowicz [1,*] **and Grzegorz Kasprowicz** [2]

[1] Institute of Microelectronics and Optoelectronics, Warsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland; dominik.kasprowicz@pw.edu.pl
[2] Institute of Electronic Systems, Warsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland; grzegorz.kasprowicz@pw.edu.pl
* Correspondence: dominik.kasprowicz@pw.edu.pl

Abstract: Emerging semiconductor devices often lack accurate analytical models. The same is usually true of any devices working under extreme conditions like cryogenic temperatures. The usual workaround involves the use of approximation models, usually based on lookup tables or neural networks individually fitted to measurement data. In the case of experimental devices or ones working under extreme conditions, the number of units available for measurement is limited. As a result, the number of approximation-model instances is too small to enable a statistical simulation of even middle-sized circuits, which is a necessary step in integrated-circuit design since it provides the realistic picture of the circuit's behavior in the presence of manufacturing process variations. Approximation models using structure parameters as inputs do exist in the literature, but are only useful if the end user knows the statistical distributions of those parameters, which is not usually the case. We propose a technique based on generative machine learning, namely the variational autoencoder, that uses only a small sample of devices to capture the essential features of their I–V curves under process variations and subsequently generates an arbitrary number of similarly disturbed curves. The model trained on as few as 20 instances per device type is shown to precisely reproduce the distributions of period and power consumption of a ring oscillator.

Keywords: generative model; machine learning; variational autoencoder; VAE; semiconductor device modeling; process variability; MOSFET; Monte Carlo

1. Introduction

Statistical simulation, a.k.a. Monte Carlo simulation, is an essential step in integrated circuit design as it provides insight into the circuit's behavior under process variations. Several hundred simulations are usually necessary to capture the statistical properties of the circuit's crucial parameters like speed or power consumption. Reliable statistical simulation is only possible with models capable of faithfully reproducing the variability in the electrical characteristics of components due to process disturbances. In the case of mature semiconductor manufacturing processes, such statistical models are included in the process design kit (PDK). The crucial parameters of these models, like e.g. effective gate length or carrier mobility, are random variables whose covariance matrix is identified in the fab as part of the process monitoring.

This is unfortunately not the case with experimental semiconductor devices, which usually lack accurate analytical models. The same is usually true of devices fabricated in stable processes but working under extreme conditions, for example at cryogenic temperatures. Device models found in the PDKs are usually valid within a relatively narrow temperature range with no accuracy guarantee outside this range. Worse still, the very nature of commonly used analytical models precludes their use at extreme temperatures due to incomplete or inadequate modeling of the underlying physics and, in some cases, severe numerical problems like intrinsic carrier concentration being smaller than the smallest double precision number available on the simulation platform [1].

The usual workaround involves the use of approximation models, typically based on lookup tables ([2–5]) or artificial neural networks [6,7] individually fitted to measurement data. Prior to the design of a circuit, a number of test structures must be manufactured for electrical characterization. The

 doi:10.20944/preprints202404.1675.v1

approximation model is then individually fitted to the measurement data obtained from each device. In the case of emerging devices the number of instances available for measurement is usually limited. For devices operating at cryogenic temperatures the limiting factor is usually not the number of test structures as such but the throughput of the cooler necessary to bring them to the target temperature. As a result, the number of instances of the approximation model may be too small to enable a statistical simulation of large circuits.

To circumvent this limitation, a number of parametric approximation models have been proposed that take into account the actual values of disturbed process parameters. Such models require *training*, i.e. their internal metaparameters must be optimized so as to make the model general enough to correctly reproduce the I–V curves of any device as long as the process perturbations are contained within a given range. The model introduced in [8] is fitted to I–V data of the nominal device as well as to all "corner" devices, i.e. ones having one process parameter assuming its minimum or maximum value. I–V curves for other devices can then be obtained by interpolating between those extreme cases. This interpolation is done on a linear scale above the threshold voltage and on a log scale in deep subthreshold. In moderate inversion the weighted average of these two approaches is used. In [9], a lookup-table based model is provided for the nominal (i.e. deviation-free) device, while the I–V curves of disturbed devices are obtained by transforming the output of this nominal model with highly nonlinear functions of V_{GS} and V_{DS}. The parameters of those nonlinear functions are assumed to be linear or quadratic functions of the process-parameter deviations. Once the coefficients of those linear and quadratic functions are identified, it is very easy to reproduce I–V curves of a device, as long as the actual values of disturbed process parameters are known for this device. The Authors of [10] use an artificial neural network (ANN) to model the behavior of a nanosheet FET. The network's inputs include, apart from the device's terminal voltages, also selected design and process parameters like gate length or nanosheet thickness. A similar approach has been adopted in [11]. The most conceptually advanced works use ANNS known as autoencoders (a brief description of this architecture can be found in Section 2) to learn the nonlinear relationships between selected process parameters and I–V curves of p-i-n diodes [12] or I–V and C–V curves of FinFETs [13].

All the models presented in the above paragraph use as their inputs the actual values of process parameters for particular devices. This poses two problems. One is that using such a model for statistical simulation necessitates prior knowledge of the statistical distributions of process parameters. While this data is available internally at the semiconductor fab, it is very unlikely to be disclosed to outsiders. A more serious problem is that training such a model requires the actual values of process parameters for each of the devices whose I–V curves the model learns to reproduce. This means the training data cannot come from measurements of real devices, whose nanometer-scale dimensions make it impossible to reliably measure the physical parameters like the gate length or doping level. Instead, the training data must be obtained from TCAD simulations, where the designer is in control of all the device parameters. TCAD tools, however, are highly configurable in terms of how (and if at all) particular physical phenomena are to be modeled. As a result, it is not known how faithfully the TCAD output reproduces the response of an actual device.

The approach presented in this work requires no assumption as to the sources of process variations or even their statistical distributions. All that is required is a representative sample of $I_D(V_{GS}, V_{DS})$ characteristics, preferably coming from current-voltage measurements rather than TCAD simulations. The proposed approach is based on *generative modeling*, a machine-learning technique in which a model learns to generate new samples resembling the training data [14]. The training examples are treated as observations of a multivariate random variable whose distribution must be discovered and modeled. This distribution may be arbitrarily complex, in particular no assumptions as to normality are made. Once the training is completed, the model may be used to generate "synthetic" data points by sampling from this distribution. One of the earliest applications of generative modeling is creation of images of non-existent persons' faces, where an image is treated as a random vector, each of whose components represents the intensity of a distinct pixel. In the presented work each component of the

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 April 2024 doi:10.20944/preprints202404.1675.v1

data vector is the device's drain current I_D for a given bias (V_{GS}, V_{DS}). The entire data vector is thus a family of I–V curves of a single device sampled at predefined bias points.

The trained model can then be used to generate data of this kind in much larger quantities (thousands or more) than the size of the training set (under a hundred). This procedure can be seen as a case of *data augmentation*, where a small set of real data is used to generate synthetic data points to build a dataset large enough for a given application [15]. Since each data point is just a vector of discrete $I_D(V_{GS}, V_{DS})$ samples for a single device, an ANN-based model must be subsequently fitted to those samples to produce a continuous representation of the device's I–V curves for Spice simulation. The entire workflow is shown in Figure 1.

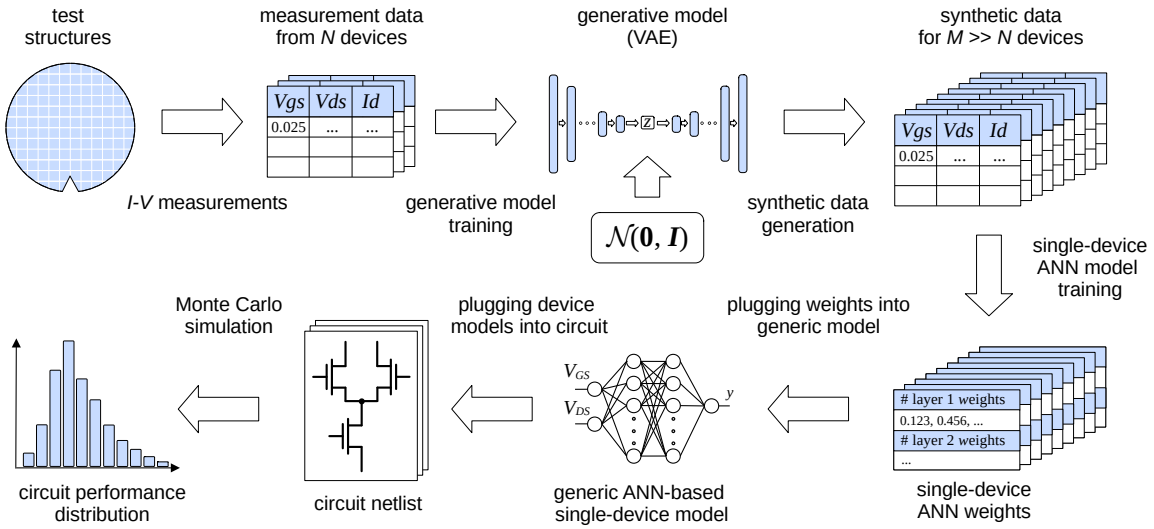


Figure 1. Details of the proposed procedure from test-structure I–V measurements through synthetic I–V data generation to statistical circuit simulation.

The rest of the paper is organized as follows. Section 2 briefly reviews the concept of variational autoencoder, i.e. the architecture of the generative model used in this work. Section 3 presents the setup of the experiment, while Section 4 summarizes its results.

2. Autoencoders and Variational Autoencoders: Theoretical Background

An autoencoder is a type of ANN used to find a lower-dimensional representation of its input data. The architecture of a typical autoencoder is sketched conceptually in Figure 2.

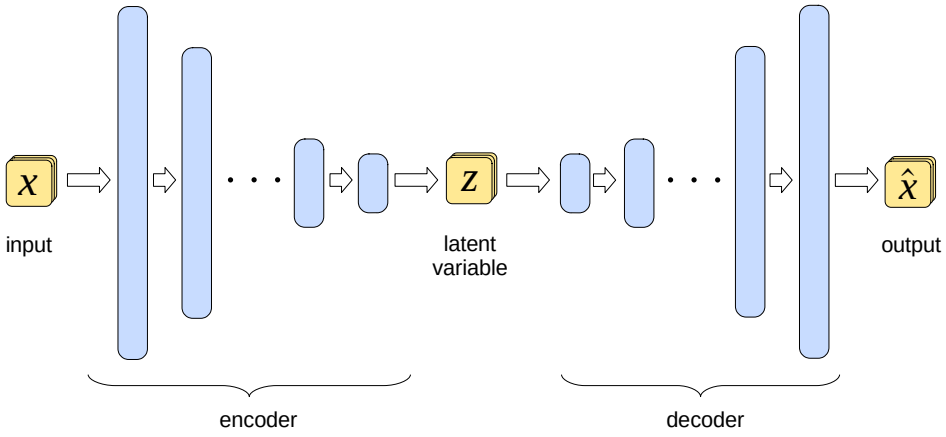


Figure 2. Architecture of an autoencoder. The blue bars represent layers of units (artificial neurons). The bar sizes reflect the varying number of units in subsequent layers, translating into the dimension of data processed by each layer.

 doi:10.20944/preprints202404.1675.v1

The high-dimensional input data vector is first propagated through subsequent layers of artificial neurons (a.k.a. *units*) in the part denoted as encoder, with each layer containing fewer units than its predecessor. This gradually reduces the dimensionality of the processed data, which can be seen as a form of lossy compression. The vector at the encoder output is referred to as the *latent representation* of the original data or just *latent variable*. This representation is subsequently processed by the block denoted as decoder, gradually increasing its dimensionality until it reaches the size of the original input vector. The training process involves minimizing the difference between the autoencoder's input and its output. This forces the encoder part to produce a latent representation that preserves the essential features of the input data at the expense of less important features and noise.

A variational autoencoder (VAE) is an extension of this concept, with the latent variable being forced during the training process to follow a known statistical distribution, usually the multidimensional standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ [16]. The architecture of the VAE used in this work is shown in Figure 3.

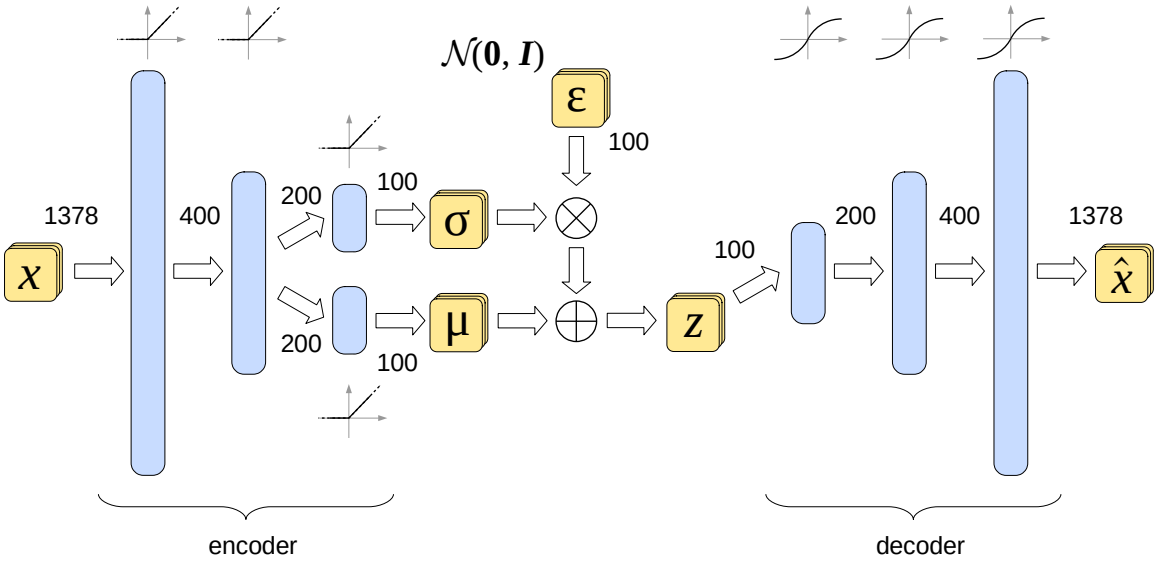


Figure 3. Architecture of the variational autoencoder used in this work. Also shown are the activation functions of all layers as well as the dimensions of the data vector at each stage. All layers are fully connected.

To achieve this goal, the VAE is trained so as to minimize the following loss function:

$$L_{VAE} = (1 - w_{KL})\, MSE(\boldsymbol{x}, \hat{\boldsymbol{x}}) + w_{KL}\, \mathcal{D}_{KL}\big(Q(\boldsymbol{z}) || \, \mathcal{N}(\mathbf{0}, \boldsymbol{I})\big). \tag{1}$$

MSE, referred to as the *reconstruction error*, is the mean squared error between the encoder input $\boldsymbol{x}$ and decoder output $\hat{\boldsymbol{x}}$. In the context of this paper, $\boldsymbol{x}$ is some function of the drain current (to be detailed in Section 3.2) and "mean" denotes an average taken over all the values of V_{GS} and V_{DS} as well as over all the observations in the minibatch. D_{KL} denotes the Kullback-Leibler divergence, i.e. a measure of dissimilarity between two statistical distributions, defined as

$$\mathcal{D}_{KL}\big(Q(\boldsymbol{z}) \,||\, P(\boldsymbol{z})\big) \equiv \mathbb{E}_{z\sim Q}\big(\log Q(\boldsymbol{z}) - \log P(\boldsymbol{z})\big). \tag{2}$$

The Kullback-Leibler divergence is a non-negative quantity dropping to zero if and only if distributions P and Q are identical. In our case $P(\boldsymbol{z})$ is the standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and is treated as the target. On the other hand, $Q(\boldsymbol{z})$ results from processing the input variable $\boldsymbol{x}$ with the encoder.

 doi:10.20944/preprints202404.1675.v1

Since the encoder is a neural network with trainable parameters (weights[1]), $Q(\boldsymbol{z})$ may be shaped almost at will. Making $\mathcal{D}_{KL}\big(Q(\boldsymbol{z})\,||\,\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\big)$ a component of the loss function subject to minimization brings $Q(\boldsymbol{z})$ possibly close to the standard normal distribution as the training progresses. The encoder is thus trained to transform the multidimensional input variable $\boldsymbol{x}$ into a lower-dimensional latent standard normal variable $\boldsymbol{z}$. The decoder, on the other hand, is trained to reverse this process, i.e. to transform $\boldsymbol{z}$ into a possibly accurate approximation $\hat{\boldsymbol{x}}$ of the VAE's input $\boldsymbol{x}$. The essential feature of these transformations is that if two input vectors $\boldsymbol{x}$ are close in value, they are supposed to produce similar values of $\boldsymbol{z}$. Furthermore, since the decoder is trained to reverse the transformation performed by the encoder, similar values of $\boldsymbol{z}$ are transformed into similar values of $\hat{\boldsymbol{x}}$. Thus, the decoder learns to perform a smooth, albeit highly nonlinear, interpolation between training examples.

Once the VAE is trained, the encoder may be discarded. The decoder is then stimulated with a random variable $\boldsymbol{z}$ drawn from the multidimensional standard normal distribution. If a value of $\boldsymbol{z}$ is used that has never appeared at the decoder's input during the training phase, the decoder will produce an output that is not identical with any of its training examples and yet shares their essential statistical properties. This way, the decoder stimulated with a source or random vectors drawn from $\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$, can be used as a generator of a multivariate random variable whose statistical distribution, although not given in an analytical form, is inferred from the training data. For this reason VAEs are widely used as *generative models*. While their original application was generating artificial faces, they are used in other fields like text processing [17], generating audio signals [18], creating level maps for video games [19] or mapping high-dimensional search spaces into lower-dimensional ones in order to facilitate optimization [20]. However, no information regarding the use of VAEs in semiconductor-device modeling is to be found in the literature.

In this work, a VAE-based model is used to generate I–V curves of a MOSFET with given nominal channel dimensions and manufactured in a process subject to perturbations. Please note that the variations of process parameters *need not* be known explicitly to successfully train the VAE. Also, the latent vector *cannot* be treated as values of any particular process parameters. In other words, no reverse engineering is performed and the confidentiality of the process-related data is not compromised.

3. Experimental Setup

To prove the usability of the VAE as a generative model of MOSFET I–V characteristics, a series of numerical experiments was performed. "Reference" samples used to train the model were generated using Spice simulation. The number of samples obtained this way was much greater than if they came from measurement of fabricated structures. This enabled a detailed visualization of histograms of selected circuit performances for subsequent comparison with the result obtained with the proposed model. Anyway, the source of reference data (measurement vs. simulation) does not influence the workflow presented below.

3.1. Benchmark Circuit

The benchmark used to evaluate the proposed approach was a 5-stage ring oscillator (RO) built with nominally identical inverters. The performances of interest were oscillation period and power consumption. Both these figures depend on the "driving capability" of devices forming the inverters. The conventional figure of merit expressing a transistor's driving capability is I_{on}, defined as the drain-current value corresponding to both the drain and gate biased with the nominal supply voltage. As shown in Figure 4, this point is never reached, or even approached, during normal operation of a ring oscillator or clocked digital circuit. This stresses the importance of statistical characterization of a device across a wide range of biases rather than just collecting the statistics of I_{on} or any other single figure of merit.

---

[1] The term "weights" is used throughout this paper as shorthand for both weights *and* biases of ANN units.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 April 2024 doi:10.20944/preprints202404.1675.v1

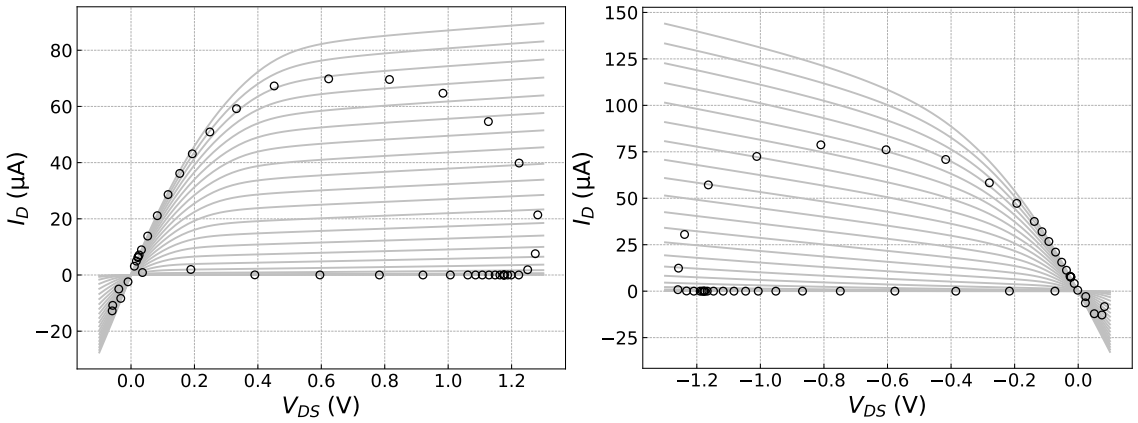


Figure 4. The circles represent (I_D, V_{DS}) samples collected at 5 ps intervals from one n-channel and one p-channel MOSFET (Figures **(a)** and **(b)**, respectively) during a single oscillation period, plotted against the respective device's output curves. The top curve corresponds to $|V_{GS}| = 1.2$ V (oscillator supply voltage), the spacing between curves is $\Delta V_{GS} = 50$ mV.

The inverters analyzed in this experiment were built with minimum-length 130nm-node MOSFETs with the channel width of 180 nm in the n-channel devices and 680 nm in the p-channel transistors. The channel widths were chosen so as to make the inverter's threshold voltage equal to exactly half of the supply voltage $V_{sup} = 1.2$ V. Three device parameters have been assumed to be subject to Gaussian variations: channel length L, threshold voltage V_{th0}, and source/drain resistance R_{DSW}. The mean values and standard deviations of those parameters are summarized in Table 1. The V_{th0} variation is used here as a substitute for deviations in a number of process parameters like gate material work function, gate dielectric permittivity, and the profile of threshold-adjust implant as well as halo/pocket implants.

Table 1. Mean values and standard deviations of the Gaussian distributions of the device parameters used in the ring-oscillator experiment.

Parameter	n-channel		p-channel	
	mean	sdt. dev.	mean	std. dev.
Channel length, L (nm)	130	10	130	10
Threshold voltage, V_{th0} (mV)	332	20	–350	20
Unit-width source/drain resistance, R_{DSW} ($\Omega \cdot \mu$m)	200	40	400	80

Two Monte Carlo experiments were conducted on such ring oscillators. In the *reference* simulations, the BSIM 3v3 MOSFET models were used. Then, they were replaced by ANN-based generated models. The details of model generation will be given in the subsequent subsections. Since the current version of the generative model only produces I–V data but not yet the C–V data, the ANN-based models had to somehow use the same terminal capacitances as BSIM models for fair comparison of dynamic properties of both models. To this end, the voltage-dependent BSIM capacitances had to be replaced with such fixed values as to assure the same oscillation period. The network of fixed capacitances used in each inverter is shown in Figure 5.

 doi:10.20944/preprints202404.1675.v1

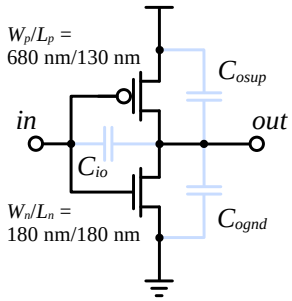


Figure 5. Architecture of single stage of the ring oscillator. The values of the equivalent capacitances are fitted separately for each supply voltage value.

C_{io} represents the sum of gate-to-channel- and gate-to-drain overlap capacitances of both devices. C_{ognd} represents the drain-to-substrate capacitance of the n-channel device shown in the Figure plus the gate-to-source overlap capacitance of the n-channel device in the next stage. C_{osup} plays a similar role in the p-channel devices. The values of those three equivalent capacitances have been tuned so as to produce a waveform identical with the one obtained with the BSIM capacitances enabled. Since MOSFET capacitances are highly voltage dependent, this procedure had to be repeated separately every time the supply voltage had been changed. The extracted equivalent capacitances were subsequently used both in the reference Monte Carlo simulations (using BSIM models) and, later, in the simulations with the proposed ANN-based models.

3.2. VAE-Based Generative Model of MOSFET I–V Data

The architecture of the model used to generate MOSFET I–V data is presented in Figure 3. The model has been implemented in Python using the Pytorch library [21]. A single data point in the training set was the family of $I_D(V_{GS}, V_{DS})$ samples from a single device. V_{GS} was swept from 0 V to 1.3 V with a fixed step of 25 mV, while V_{DS} was swept from 50 mV to 1.3 V with a 50 mV step. The procedure for P-channel devices was identical but all the voltages were negative. Thus, each device was described with $53 \times 26 = 1378$ numbers.

Using I_D as the VAE input works fine for V_{GS} above the threshold voltage V_{th0}. However, I_D values below the threshold are many orders of magnitude smaller, so they have hardly any contribution to the training loss function. As a consequence, the VAE does not even attempt to minimize the reconstruction error in subthreshold. A common solution is to make the ANN reproduce the logarithm of I_D rather than I_D itself. Such a transformation, however, brings the above- and below-threshold values to the same order of magnitude and, as a consequence, almost equalizes the *relative* error across the whole V_{GS} range. This is not desirable, because relative-error levels that are acceptable for the off-current are too high for the on-current. Therefore, in this work the subthreshold regime has been slightly deprioritized by defining each component of the VAE input vector in the following way:

$$x(V_{GS}, V_{DS}) = \frac{V_{GS} - V_{GS\,\mathrm{min}}}{V_{GS\,\mathrm{max}} - V_{GS\,\mathrm{min}}} \ln \frac{I_D(V_{GS}, V_{DS})}{I_0}, \tag{3}$$

where the normalizing current $I_0 = 10$ fA. This value was chosen because it is an order of magnitude below the smallest current observed in the dataset. Therefore this choice of I_0 guarantees that the argument of the logarithm in (3) is always greater than unity, which leads to positive logarithm values.

Various variants of the VAE architecture have been tested, namely:

- Number of layers in both decoder and encoder ranging from three to five.
- The number of units in the encoder's input layer and decoder's output layer varying between 100 and 800, with all the other layers modified accordingly.
- The following combinations of activation functions in the decoder and encoder:
  - ReLU in all layers,

 doi:10.20944/preprints202404.1675.v1

  - sigmoid in all layers,
  - sigmoid in the last layer, ReLU in the rest,
  - ReLU in the last layer, sigmoid in the rest.
- Convolutional layers instead of fully connected ones.

The parameters of the final VAE can be found in Figure 3. All the layers are fully connected. While convolutional layers are commonly used in VAEs whose goal is generation of images, this approach turned out to produce extremely noisy results in our case. The final encoder architecture consists of three layers, whose output vectors have a length of 400, 200, and 100, respectively, with the last number being the size of the latent variable z. All the units in the encoder have the ReLU activation function. The layers in the decoder expand the latent variable to 200, 400, and finally 1378 dimensions. All the units in the decoder have the sigmoid activation function. The batch size used in training was 25, except for the smallest training set consisting of 20 examples, where the batch size had to be reduced to 5. Increasing the batch size over 25 items led to greater reconstruction errors.

A validation set of 1000 devices was used all along the training process to monitor both components of the loss, i.e. the reconstruction error and Kullback-Leibler divergence. Typical learning curves for those two error components are shown in Figure 6. The training was interrupted after both error components saturated. This happened after 5,000 to 40,000 epochs, with smaller training sets requiring more epochs. The entire process took about five minutes on a desktop computer with computations offloaded to an NVIDIA GeForce GTX 1660 SUPER GPU.

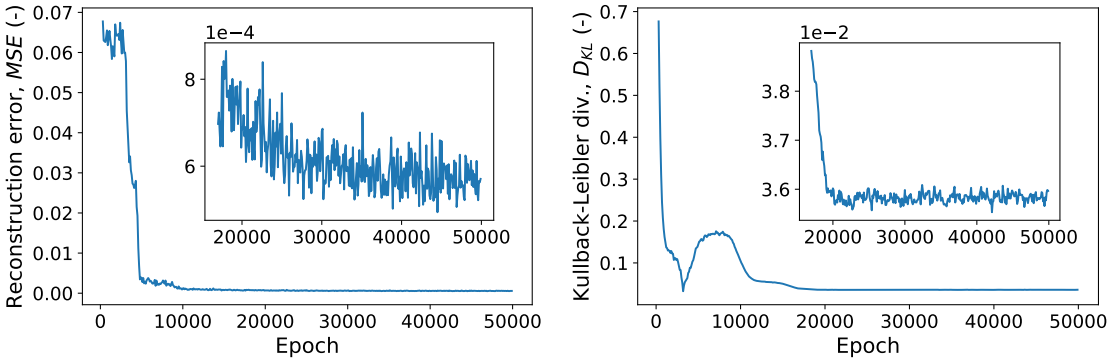


Figure 6. (a) VAE reconstruction error and **(b)** Kullback-Leibler divergence (KLD) on the validation dataset evaluated in the training phase every 100$^{\text{th}}$ epochs. The insets show that the KLD saturates after approximately 20,000 epochs, while the reconstruction error takes approximately twice as long to reach its minimum.

The optimal contribution of Kullback-Leibler divergence w_{KL} in the loss function (see equation (1)) was experimentally found to be 0.01. Larger values of w_{KL} excessively prioritized the normality of latent variable over reconstruction accuracy. The I–V curves reconstructed by the decoder in this case either showed excessive ripples or, while preserving satisfactory smoothness, deviated heavily from the input data. On the other hand, reducing w_{KL} below 0.01 did not visibly improve the quality of reconstructed I–V curves while making the latent-variable distribution significantly deviate from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, thus precluding later use of the decoder as a source of I–V curves sharing the statistical properties with the training data.

After training, the VAE was once again used to encode a subset of the validation dataset. The Henze-Zirkler test for multivariate normality was subsequently performed on the latent variable to make sure its distribution was sufficiently close to normal [22]. The test passed over 90% of the time, which shows that the latent variable is indeed close to normal.

Figure 7 shows example I–V curves generated by a VAE trained on a dataset of 50 devices and subsequently stimulated with six standard normal vectors. The curves are qualitatively similar to

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 April 2024 doi:10.20944/preprints202404.1675.v1

actual MOSFET curves (presented in Figure 4 (a)) both on the linear scale (top row) and logarithmic scale (bottom row). Ripples are very small and are further smoothed out at the next stage, where ANN-based approximation models are fitted individually to the I–V curves of each device (see the following Section).

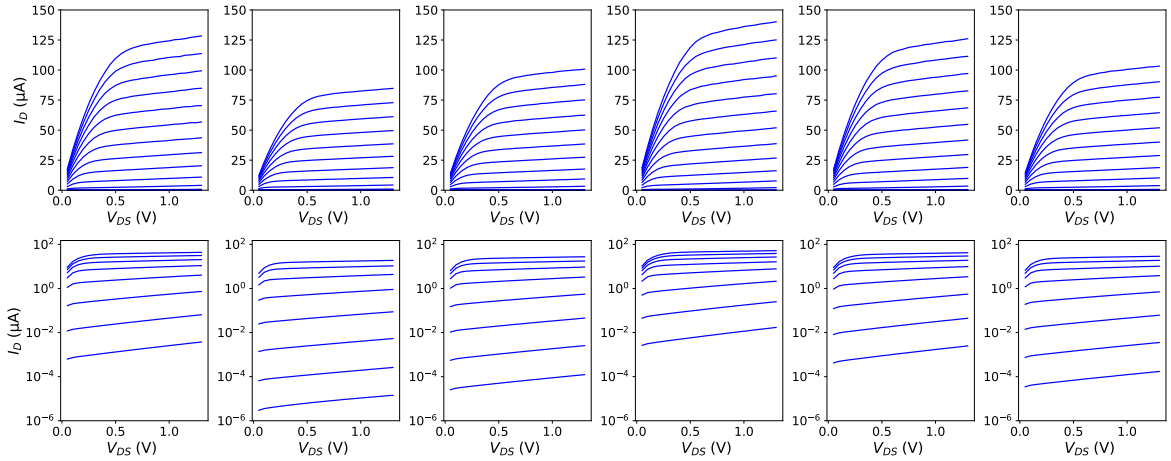


Figure 7. Example sets of I–V curves generated by the presented VAE plotted on the linear (top row) and logarithmic scale (bottom row). The curves correspond to bias ranges $V_{GS} = [0\,\text{V}, 1.3\,\text{V}]$ (top) and $V_{GS} = [0\,\text{V}, 0.7\,\text{V}]$ (bottom) with a V_{GS} step of 100 mV.

3.3. ANN-Based Models of Single Devices

The VAE presented in the previous Section generates a device's I–V response as a set of discrete points. To make this model usable, some interpolation between them is necessary. The approach taken in this work uses ANNs fitted individually to every device. These ANNs were composed of the following layers:

- Input layer: 2 linear units receiving V_{GS} and V_{DS} as their inputs.
- First hidden layer: 16 sigmoid units.
- Second hidden layer: 16 sigmoid units.
- Output layer: 1 linear unit producing response $\hat{y}$ (see below).

For reasons outlined in Section 3.2, the ANN output was a function of the logarithm of the device drain current I_D. This, however, posed a problem at $I_D = 0$, which takes place at $V_{DS} = 0$. Therefore, this particular bias was not used in the training set. To make the model correctly reproduce a MOSFET's behavior at zero drain-to-source bias anyway, it is common to train the ANN using the following expression (see e.g. [7,10] or [11]):

$$y = \ln \frac{I_D}{V_{DS}}. \tag{4}$$

At the inference stage, when the model is used in the simulator to produce a response $\hat{y}$, this expression is solved for I_D:

$$\hat{I}_D = V_{DS} \cdot \exp(\hat{y}), \tag{5}$$

which forces $\hat{I}_D = 0$ at $V_{DS} = 0$. This approach has been adopted in this work. However, like in the VAE presented in Section 3.2, it resulted in excessive prioritization of the subthreshold region, leading to unacceptably high error levels above the threshold. This effect was reversed by using the following sigmoid weighting function:

$$\sigma(V_{GS}) = \frac{1}{1 + \exp(-V_{GS}/V_s)}\,, \tag{6}$$

 doi:10.20944/preprints202404.1675.v1

where $V_s = 0.1$ V for n-channel devices and -0.1 V for p-channel devices. Finally, (4) and (6) were combined to form the loss function used at the training stage:

$$L_{ANN} = \sum_{\forall(V_{GS},V_{DS})} \sigma(V_{GS}) \frac{(y-\hat{y})^2}{y^2}. \tag{7}$$

The ANNs training was performed in Pytorch. The training of ANN models of individual devices can be greatly speed up by exploiting the relative similarity of the I–V responses of all the devices of the same channel type and size, which implies similar values of the weights of their models. This allows reusing the weights of the first successfully trained model as an initial guess in training the models of all the subsequent devices of the same type. While training the ANN model from scratch lasts on average 31 seconds per device on a desktop computer, this time is reduced to 4.2 seconds when reusing a previously generated model as the initial guess. Thus, this simple technique provides a speedup of almost 7.5 times.

The model of individual devices (referred to as *single-device model*) is split in two parts. A generic Verilog-AMS file is common to all devices, while weights, distinct to individual devices, are stored in separate files included into the main file during simulation. These two parts can then be loaded into a circuit simulator like Synopsys HSPICE® (used in this experiment) or Cadence Spectre®. For every device instance and every Monte Carlo run a new set of weights is read into the generic model.

4. Results

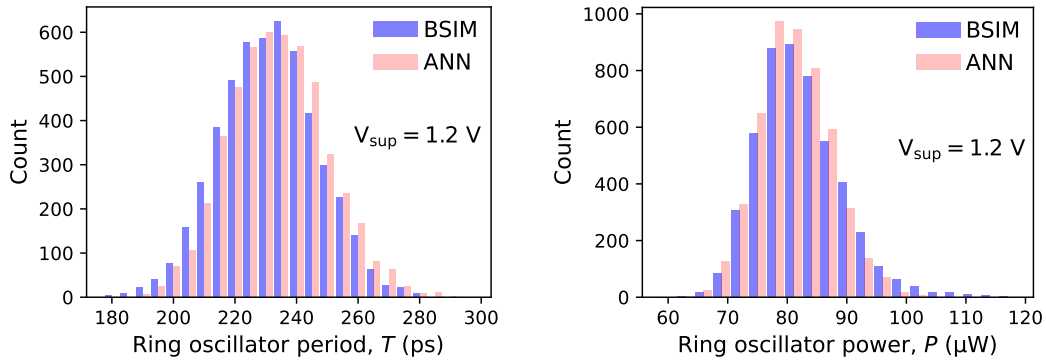
The setup presented in the previous Section was used in a series of Monte Carlo experiments to assess the feasibility of the proposed approach. Of particular interest was the influence of the training set size on the accuracy of reproducing the statistical distribution of RO period and power consumption. Training sets of 20, 50, 150, and 500 instances were tried for each device type. In a real-world setting, the proposed generative model will have to be trained based on measurement data coming from a relatively small set of test structures. This is in contrast to more "conventional" tasks like image generation, where a model is trained on datasets containing thousands or tens of thousands images.[2]

Two experiments were run, differing in the ring oscillator supply voltage. In the first experiment, the nominal supply voltage for the 130 nm MOSFET node $V_{sup} = 1.2$ V was used. In the other, V_{sup} was reduced to 0.4 V, which corresponds to devices operating slightly above the threshold voltage or, for strong positive deviations in $|V_{th0}|$, in subthreshold (see Table 1 for mean values and standard deviations of V_{th0}). 5,000 ROs were generated in either experiment, each with unique ANN-based models of n-channel and p-channel devices. By way of reference, another 5,000 ROs were simulated with BSIM models. Histograms of the RO period and power consumption are presented in Figure 8 ($V_{sup} = 1.2$ V) and Figure 9 ($V_{sup} = 0.4$ V).

As shown in Figures 8 and 9, the statistical distributions of the oscillation period and power consumption display a significant skewness. This is especially true at lower supply voltages, i.e. with devices working in weak and moderate inversion, where the relationship between the drain current and the parameters subject to variations (mostly channel length and threshold voltage) is close to exponential. Such distributions cannot be accurately described with just the mean value and standard deviation. More information is conveyed by percentiles. The X-th percentile, here denoted as pX, is defined as the value that is not exceeded by the random variable more often than X-percent of the time. Thus, $p50$ is the median, while $p99$ is the value exceeded only by the top 1% cases. Selected percentiles of RO period and power for both values of V_{sup} are given in Table 2.

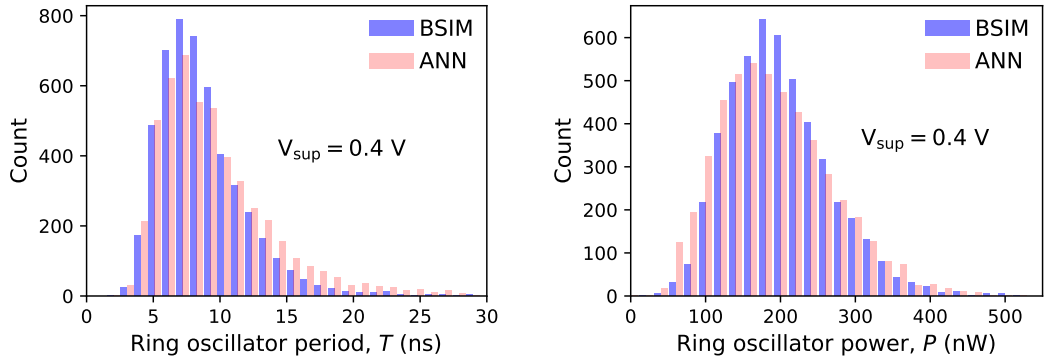
[2] By way of example, the training dataset of the MNIST database of images of handwritten digits, frequently used a benchmark for many machine-learning algorithms, including generative models, contains 6,000 images of each digit.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 April 2024 doi:10.20944/preprints202404.1675.v1



(a) **(b)**

Figure 8. Histograms of **(a)** period and **(b)** power consumption of 5,000 ring oscillators described in Section 3.1. The ANN-based device models were generated by VAEs trained using 50 instances per device type. BSIM results serve as reference.



(a) **(b)**

Figure 9. Histograms of **(a)** period and **(b)** power consumption of 5,000 ring oscillators with devices operating in moderate inversion ($V_{\text{sup}} = 0.4$ V). The ANN-based device models were generated by VAEs trained using 50 instances per device type. BSIM results serve as reference.

Table 2. Selected percentiles of the ring-oscillator period and power consumption based on 5,000 Monte Carlo Spice runs. BSIM results (bottom row) serve as reference. Other results were obtained with the proposed procedure using a generative model trained on datasets of varying sizes.

Training set size	$V_{\text{sup}} = 1.2$ V						$V_{\text{sup}} = 0.4$ V					
	Period (ps)			Power (μW)			Period (ns)			Power (nW)		
	p50	p90	p99	p50	p90	p99	p50	p90	p99	p50	p90	p99
20	234.92	252.64	268.64	82.20	89.43	97.08	7.80	11.60	17.34	223.33	310.59	394.87
50	235.54	256.67	274.96	82.19	90.29	97.59	9.04	15.62	25.84	192.22	304.45	411.75
150	233.07	254.17	272.17	83.12	91.27	98.29	8.42	13.55	21.26	206.22	307.91	407.74
500	232.35	253.30	270.77	83.63	92.43	100.97	7.80	13.56	27.89	222.89	328.22	438.61
BSIM	234.04	254.88	271.16	83.12	93.01	106.80	8.63	13.46	21.11	202.77	303.48	417.68

5. Discussion

The histograms of ring-oscillator period and power consumption obtained with the proposed model are in good agreement with BSIM-based simulations. Thus, the generative model reproduces well the statistical properties of the I–V curves of devices used for its training. The agreement is satisfactory even for near-threshold operation ($V_{sup} = 0.4$ V), where the relationship between the

 doi:10.20944/preprints202404.1675.v1

perturbed device parameters and the drain current is close to exponential, i.e. strongly nonlinear. As a result, the values of period and power span an entire order of magnitude and their distributions show a pronounced skewness. The numerical data presented in Table 2 show that even the top percentiles $p90$ and $p99$, corresponding to the long right tails, can be predicted with a reasonable accuracy. This is important because the right tails of power- and period distributions directly influence the parametric yield of a digital circuit. For training sets of 150 devices all percentiles are estimated within 5 percent of their true value (and in most cases within 2 percent). For training sets of 20 and 50 devices the errors are closer to 10 percent and, for $p99$, sometimes larger. It must be remembered, however, that $p99$ is defined as the value that is exceeded only by 1 percent of the population. Estimating this quantity with a model trained on just 20 or 50 observations is necessarily prone to error. Still, the number of training cases used in this experiment is several orders of magnitude smaller than what is normally used in image-generation applications. Unlike images of faces, digits or other objects, MOSFET drain current plotted in the (V_{GS}, V_{DS}) space forms a relatively smooth surface. Also, differences between individual cases are quite subtle. Thus, a relatively small set of training examples may capture all the necessary statistical properties of the entire population. Interestingly, there is no clear relationship between the accuracy of period and power estimation and training-set size. Both going below 50 cases and above 150 cases seems to increase the error.

Please note that the I–V curves were sampled on a fixed uniform (V_{GS}, V_{DS}) grid. Lowering the RO operating voltage from 1.2 V to 0.4 V means that samples corresponding to two-thirds of V_{GS} biases and the same proportion of V_{DS} biases were left unused. Still, even in such conditions the histograms obtained with the proposed model look very close to those coming from the BSIM simulations.

A single RO simulation took an average of 161 ms with the BSIM model and 886 ms with the ANN-based model. Thus, the currently used Verilog-AMS-based implementation of the proposed model is approximately 5.5 times slower than BSIM. One may expect that implementing the model directly in the simulator source code would significantly speed up the simulation.

To sum up, the proposed method seems attractive because I–V characterization of 20-50 transistors is often feasible even in cases of emerging devices or ones working in a cryogenic cooler. Extension of this method for C–V curves would enable taking into account deviations of MOSFETs' capacitances, thus making the statistical simulation even more realistic.

Author Contributions: Conceptualization, D.K.; methodology, D.K.; software, D.K.; validation, D.K.; formal analysis, D.K.; investigation, D.K.; resources, D.K.; data curation, D.K.; writing—original draft preparation, D.K.; writing—review and editing, D.K.; visualization, D.K.; supervision, D.K.; project administration, G.K.; funding acquisition, G.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Centre for Research and Development of Poland, grant QUANTERAII/1/80/SIQCI/2022.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Datasets available on request.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial neural network
KLD	Kullback-Leibler divergence
MOSFET	Metal-oxide-semiconductor field-effect transistor
MSE	Mean squared error
PDK	Process design kit
RO	Ring oscillator
VAE	Variational autoencoder

 doi:10.20944/preprints202404.1675.v1

References

1. Beckers, A.; Jazaeri, F.; Enz, C. Cryogenic MOS Transistor Model. *IEEE Transactions on Electron Devices* **2018**, vol. 65, no. 9, 3617–3625. https://doi.org/10.1109/TED.2018.2854701.
2. Kimura, M; Inoue, S.; Shimoda, T. Table Look-Up Model of Thin-Film Transistors for Circuit Simulation Using Spline Interpolation with Transformation by y=x+log(x). *IEEE Transactions on CAD* **2002**, vol. 21, no. 9, 1101–1104. https://doi.org/10.1109/TCAD.2002.801090.
3. Yang, B.; McGaughy, B. An Essentially Non-Oscillatory (ENO) High-Order Accurate Adaptive Table Model for Device Modeling. In Proceedings of the 41th Design Automation Conference, San Diego, CA, USA, 7-11 June 2004, 864–867.
4. Bourenkov, V.; McCarthy, K.G.; Mathewson, A. MOS Table Models for Circuit Simulation. *IEEE Transactions on CAD* **2005**, vol. 24, no. 3, 352–362. https://doi.org/10.1109/TCAD.2004.842818.
5. Thakker, R.A.; Sathe, C.; Sachid, A.B. et al. A Novel Table-Based Approach for Design of FinFET Circuits. *IEEE Transactions on CAD* **2009**, vol. 28, no. 7, 1061–1070. https://doi.org/10.1109/TCAD.2009.2017431.
6. Xu, J.; Gunyan, D., Iwamoto, M et al. Drain-Source Symmetric Artificial Neural Network-Based FET Model with Robust Extrapolation Beyond Training Data. In Proceedings of the 2007 IEEE/MTT-S International Microwave Symposium, Honolulu, HI, USA, 3-8 June 2007, 2011–2014. https://doi.org/10.1109/MWSYM.2007.380244.
7. Wang, J.; Kim, Y.-H.; Ryu, J. et al. Artificial Neural Network-Based Compact Modeling Methodology for Advanced Transistors. *IEEE Transactions on Electron Devices* **2021**, vol. 68, no. 3, 1318–1325. https://doi.org/10.1109/TED.2020.3048918.
8. Wang, J.; Xu, N.; Choi, W. et al. A Generic Approach for Vapturing Process Variations in Lookup-Table-Based FET Models. In Proceedings of the 2015 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), Washington, DC, 9-11 September 2015, 309–312. https://doi.org/10.1109/SISPAD.2015.7292321.
9. Kasprowicz, D. Table-Based Model of a Dual-Gate Transistor for Statistical Circuit Simulation. *IEEE Transactions on CAD* **2019**, vol. 38, no. 8, 1493–1500. https://doi.org/10.1109/TCAD.2018.2852756.
10. Woo, S.; Jeong, H.; Choi, J.; Cho, H.; Kong, J.-T.; Kim, S. Machine-Learning-Based Compact Modeling for Sub-3-nm-Node Emerging Transistors. *Electronics* **2022**, 11, 2761. https://doi.org/10.3390/electronics11172761.
11. Lyu, Y.; Chen, W.; Zheng, M. et al. Machine Learning-Assisted Device Modeling With Process Variations for Advanced Technology. *IEEE Journal of the Electron Devices Society* **2023**, vol. 11, 303–310. https://doi.org/10.1109/JEDS.2023.3277548.
12. Mehta, K.; Raju, S.S.; Xiao, M. et al. Improvement of TCAD Augmented Machine Learning Using Autoencoder for Semiconductor Variation Identification and Inverse Design. *IEEE Access* **2020**, vol. 8, pp. 143519–143529. https://doi.org/10.1109/ACCESS.2020.3014470.
13. Mehta, K.; Wong. Prediction of FinFET Current-Voltage and Capacitance-Voltage Curves Using Machine Learning with Autoencoder. *IEEE Electron Device Letters* **2021**, vol. 42, no. 2, 136–139. https://doi.org/10.1109/LED.2020.3045064.
14. Foster, D. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*, 1st ed.; Publisher: O'Reilly Media, 2019.
15. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* **2019**, 6. https://doi.org/10.1186/s40537-019-0197-0.
16. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the 2nd International Conference on Learning Representations, Banff, Canada, 14-16 April 2014.
17. Miao, Y; Yu, L; Blunsom, P. Neural Variational Inference for Text Processing. In Proceedings of the 33rd International Conference on Machine Learning ICML'16, New York City, USA, 19-24 June 2016, vol. 48, 1727–1736
18. Karamatlı, E.; Cemgil, A.T; Kırbız, S. Audio Source Separation Using Variational Autoencoders and Weak Class Supervision. *IEEE Signal Processing Letters* **2019**, vol. 26, no. 9, 1349–1353. https://doi.org/10.1109/LSP.2019.2929440.
19. Mak, H.W.L.; Han, R.; Yin, H.H.F. Application of Variational AutoEncoder (VAE) Model and Image Processing Approaches in Game Design. *Sensors* **2023**, 23, 3457. https://doi.org/10.3390/s23073457.

 doi:10.20944/preprints202404.1675.v1

20. Touloupas, K.; Sotiriadis, P.P. Mixed-Variable Bayesian Optimization for Analog Circuit Sizing through Device Representation Learning. *Electronics* **2022**, 11, 3127. https://doi.org/10.3390/electronics11193127.
21. Pytorch library for deep learning in Python. Available online: https://pytorch.org (accessed 6 March 2023).
22. Henze, N.; Zirkler, B. (1990). A class of invariant consistent tests for multivariate normality. *Communications in statistics-Theory and Methods*, **1990**, 19(10), 3595-3617.