



Article

A Physics-Informed Automatic Neural Network Generation Framework for Emerging Device Modeling

Guangxin Guo , Hailong You , Cong Li *, Zhengguang Tang and Ouwen Li

School of Microelectronics, Xidian University, Xi'an 710071, China

* Correspondence: licong@xidian.edu.cn

Abstract: With the rapid development of semiconductor technology, traditional equation-based modeling faces challenges in accuracy and development time. To overcome these limitations, neural network (NN)-based modeling methods have been proposed. However, the NN-based compact model encounters two major issues. Firstly, it exhibits unphysical behaviors such as un-smoothness and non-monotonicity, which hinder its practical use. Secondly, finding an appropriate NN structure with high accuracy requires expertise and is time-consuming. In this paper, we propose an Automatic Physical-Informed Neural Network (AutoPINN) generation framework to solve these challenges. The framework consists of two parts: the Physics-Informed Neural Network (PINN) and the two-step Automatic Neural Network (AutoNN). The PINN is introduced to resolve unphysical issues by incorporating physical information. The AutoNN assists the PINN in automatically determining an optimal structure without human involvement. We evaluate the proposed AutoPINN framework on the gate-all-around transistor device. The results demonstrate that AutoPINN achieves an error of less than 0.05%. The generalization of our NN is promising, as validated by the test error and the loss landscape. The results demonstrate smoothness in high-order derivatives, and the monotonicity can be well-preserved. We believe that this work has the potential to accelerate the development and simulation process of emerging devices.

Keywords: emerging device modeling; neural network; physical informed; automated machine learning (AutoML); compact model; semiconductor device; circuit simulation; field-effect transistor (FET)



Citation: Guo, G.; You, H.; Li, C.; Tang, Z.; Li, O. A Physics-Informed Automatic Neural Network Generation Framework for Emerging Device Modeling. *Micromachines* **2023**, *14*, 1150. <https://doi.org/10.3390/mi14061150>

Academic Editor: Carlos Sampedro

Received: 10 March 2023

Revised: 25 May 2023

Accepted: 26 May 2023

Published: 29 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the development of semiconductor technology, fabricating and evaluating new transistors is time-consuming and expensive. Compact models serve as the bridge between device process technology and electronic integrated circuit (IC) design. It is essential to quickly complete transistor modeling accurately to save time and costs [1]. Standard compact models of transistors (e.g., BSIM-CMG [2], GSIM-IMG [3], and PSP [4]) are widely used in industry, but they have difficulty modeling new emerging devices. As transistors are scaled, more new unideal effects and quantum mechanical effects appear. These new challenges increase the difficulty of modeling new emerging devices for three reasons: (a) the traditional standard FET models cannot well-capture the electrical characteristics of emerging devices, (b) developing the physics-based model equation requires a long time and expertise, and (c) for equation-based models, it is still challenging to fully automate the model parameter extraction process while achieving a very high fitting accuracy [5]. In the previous studies [6–16], Neural Networks (NN) show promising accuracy in emerging device modeling. However, NN-based device modeling suffers from two main issues: unphysical behaviors and needing NN expertise [5]. The unphysical issues, such as un-smoothness, non-zero drain current (I_d) at $V_{DS} = 0$, and lacking monotonic dependency, are blocking the adoption of NN-based methods by the industry scene. The requirements of expertise issue consumes lots of time to try out an appropriate NN with high accuracy and lightweight.

Several approaches have been proposed to solve the unphysical issues of NN-based modeling. For instance, Li et al. [6] used a two-portion NN with different activation functions for V_d and V_g , but it can only handle the terminal voltage and not other electrical inputs such as gate length (L_g). Kao et al. [11] combined physics-based compact models (e.g., BSIM [2]) with NN output, but it relies on established compact models and is computationally complex. Wang et al. [5] used a symmetry transform function to obtain a smooth curve, but it cannot handle an unsymmetrical source/drain scene [17]. Huang et al. [18] incorporated a physical-relation-neural-network to map between device parameters and surface potential, then constructed I_d by mathematical equations, which may induce additional errors for emerging devices. Tung et al. [10] used a loss function to smooth the output, but this approach only works on oversampled data. When the input electrical parameters are increased, the oversample may result in an unacceptable data size. Few works incorporate the monotonic dependence on the NN network, for instance, drain/source current (I_{ds}) increases when V_{gs} increases. When the data sample is less or the variable relationship is not clear, it is necessary to set the monotonic dependency between the input and output electrical parameters of the NN.

The commonly used method to obtain an appropriate NN structure is based on the trial-and-error method. The issue of needing NN expertise has confused semiconductor background researcher a lot. Wang et al. [5] used SPICE simulation turn-around time to find an appropriate structure that balances accuracy and speed, which is less of a guidance and time-consuming process. They increased the NN size when the accuracy was low and reduced the NN size when the SPICE simulation time was high. This may lead to a loop when there is no solution. Tung et al. [10] tested the relationship between nodes number and speed using grid search. Additionally, they searched parameters using the trial-and-error method.

To summarize, the primary challenges of NN-based device modeling include the following: (1) Requiring expertise in neural networks to establish an appropriate structure. (2) Addressing unphysical issues associated with NN-based modeling, including: (a) Ensuring smooth differentiability of I_d with respect to V_{gs} and V_{ds} . (b) Establishing a monotonic I_d curve. (c) Ensuring that I_d equals 0 when V_d equals 0. (d) Incorporating both symmetric and asymmetric drain/source scenarios. (e) Leveraging existing device modeling knowledge.

In this paper, an Automatic Physical-Informed Neural Network (AutoPINN) generation framework is proposed, as shown in Figure 1. This framework is composed of two parts: Automatic Neural Network (AutoNN) and Physics-Informed Neural Network (PINN). Compared with other general NN methods, the PINN method has better physical behavior because of taking the physical information of device modeling into consideration. Compared with other general AutoNN methods, the AutoNN is optimized for our PINN regarding device modeling. It can substantially decrease the search time during the NN architecture optimization, according to the complexity of input device data.

This framework takes device data, semiconductor domain knowledge (e.g., monotonic relationship between V_{ds} and I_{ds}), and the optimization target as input. Then, it generates a device modeling neural network, with optimal structure and physical information embedded. The AutoNN assists PINN to find an optimal structure without human involvement. It can solve the expertise issue mentioned before. To overcome the unphysical issues, the PINN is introduced. The PINN embeds physical information with a few key technologies, such as Domain Transform, Smooth Loss Function, Monotonic Network Block, and Knowledge Transfer. The Domain Transform makes I_d smooth and differentiable to V_{ds} by increasing the density near the $V_{ds} = 0$. The functions have the ability to handle both symmetry and un-symmetry drain/source scene. It also transforms the optimization target to a new one, which can ease the burden of NN fitting. The Smooth Loss Function takes not only the optimization target, but also the derivatives and other factors into consideration. It makes the total I_d curve smooth and differentiable. The Monotonic Network Block is used to obtain the monotonic behavior by constraining the weight of NN as non-negative. The information from other devices can be transferred to new device modeling

by Knowledge Transfer. It can speed up the training convergence process and obtain better physical behavior.

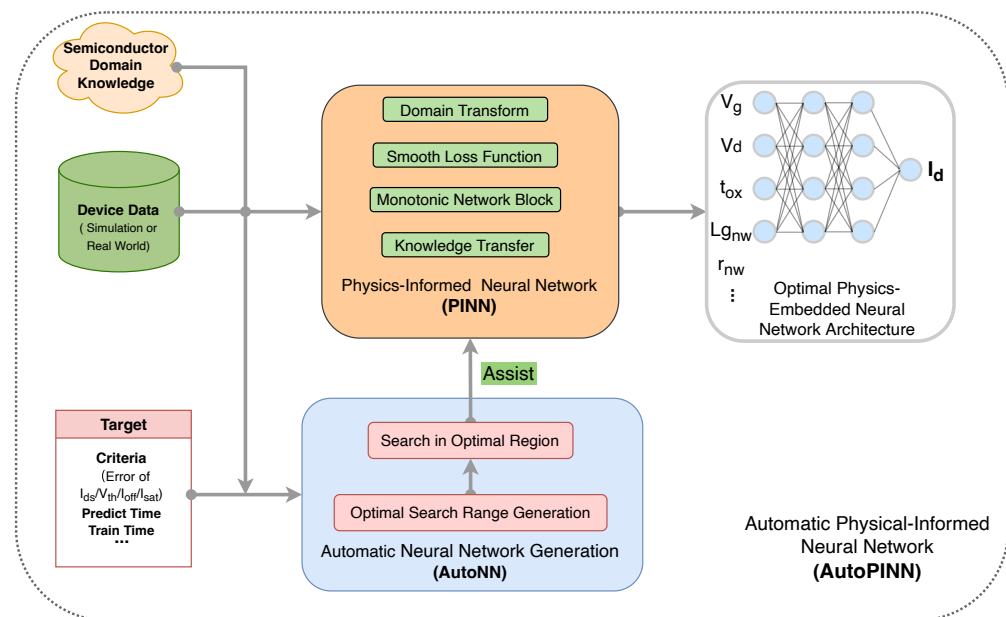


Figure 1. Physical-Informed automatic neural network generation framework. The PINN is embedded with physical information from device data and semiconductor knowledge. The AutoNN assists PINN to find an optimal structure to meet the target.

The contributions of this paper are summarized as follows:

1. In this paper, a physics-informed neural network (PINN) is proposed, which can embed physical device information into neural networks (NN) to overcome non-physical behaviors and improve accuracy in compact modeling. The techniques proposed include the Domain Transform functions, Smooth Loss Function, Transfer Knowledge, and Monotonic Network Block. These techniques aim to make NN-based modeling practical.
2. This paper proposes a two-step Automatic Neural Network (AutoNN) method for optimizing PINN structure. The framework involves two steps: (a) generating a small range of PINN parameters according to the complexity of electrical features, and (b) finding the optimal PINN structure based on accuracy and speed. The AutoNN assists PINN to improve accuracy without human involvement.
3. Evaluated on the TCAD-simulated gate-all-around transistor (GAAFET) device, this framework can achieve an error of less than 0.05%. The framework outperforms an ensemble learning result, achieving a 72.2% reduction in the error of the drain current (I_d) compared to the ensemble method.

The rest of this article is divided into the following sections. Section 2 presents several key techniques to embed physical information into NN. In Section 3, the optimization of PINN is described using a two-step AutoNN technique to find the optimal architecture based on user-defined targets. Then, in Section 4, we present the experimental results of our framework evaluated on the GAAFET data. Finally, in Section 5, we summarize our key conclusions.

2. Physical-Informed Neural Network

Although the neural network (NN) has the power of universal approximation, there are still some challenges to bringing NN-based device modeling methods to practical use. The most important is the non-physical behaviors of the NN-based model. This section focuses on the techniques proposed to embed physics information into NN to overcome the

barriers and improve the accuracy of device modeling. Physical behaviors play a crucial role in making the results more reasonable and practical. Moreover, it can induce accuracy improvement. The key techniques proposed, including the Domain Transform function, Monotonic Network Block, Smooth Loss Function, and Transfer knowledge, are designed to effectively integrate physical behaviors into NN-based device modeling methods.

2.1. Smooth Loss Function

The loss function is a critical factor in determining the accuracy of a neural network, as it guides the direction of optimization. It is also an intuitive way to incorporate physical information into the network. Therefore, it is essential to define an appropriate loss function that takes into account both accuracy and physical behavior. The proposed loss function for the PINN is defined in a smooth and accurate way in Equations (1) and (2).

$$Loss = Err(y_{true} - y_{pred}) + \alpha \times Err(e^{y_{true}} - e^{y_{pred}}) + \beta \times Err\left(\frac{Diff I_d}{Diff V_g}\right) + \gamma \times Err\left(\frac{Diff I_d}{Diff V_d}\right) \quad (1)$$

$$Err = RMSE \quad (2)$$

where α, β, γ are the weight that controls the importance of each component in the loss function. The first component aims to decrease the error in the logarithmic scale to accurately model the sub-threshold region. To improve the accuracy of the saturated region, the second component of Equation (1) considers the error in the original numerical scale. Additionally, the smoothness of the current-voltage ($I - V$) curve is an important physical behavior that can be integrated into the loss function by adding the derivative of I_d with respect to the input voltage (V_g and V_d) as the third and fourth components, respectively. It is important to mention that TCAD simulations or hardware measurements produce discrete numeric values, and therefore, a numerical approximation is employed to represent the derivative of I_d with respect to V_g and V_d .

The Gummel Symmetry Test (GST) is a well-established method used to evaluate the smoothness and symmetry of the current-voltage characteristics of a device. This method was first introduced by Gummel [19], and it has since become widely adopted in the field. Figure 2 shows a circuit with a GAAFET device, the GST involves setting a specific voltage (V_G) on the gate and varying another voltage (V_X) to measure the current (I_d) flowing through the device. Then, the smoothness and symmetry of the current-voltage curve can be assessed.

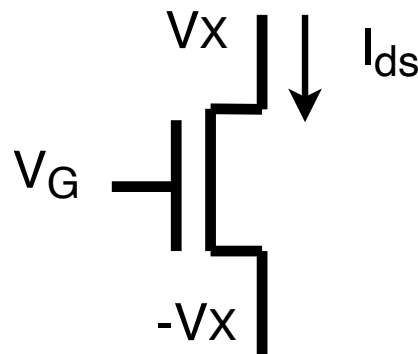


Figure 2. Gummel symmetry test setup circuit.

In the field of artificial neural networks, the use of the smooth loss function is beneficial in encouraging the network to generate smooth and continuous predictions as the input values vary. This is particularly important when the output is a function of multiple inputs, as small changes in one input can lead to significant changes in the output. In Figure 3, a comparison is made between two neural networks. Figure 3a shows the NN without a smooth loss function. Figure 3b shows the NN with a smooth loss function. It

is observed that the network with the smooth loss function can produce a smooth and relevant current-voltage curve, even for first-order derivatives with respect to V_X .

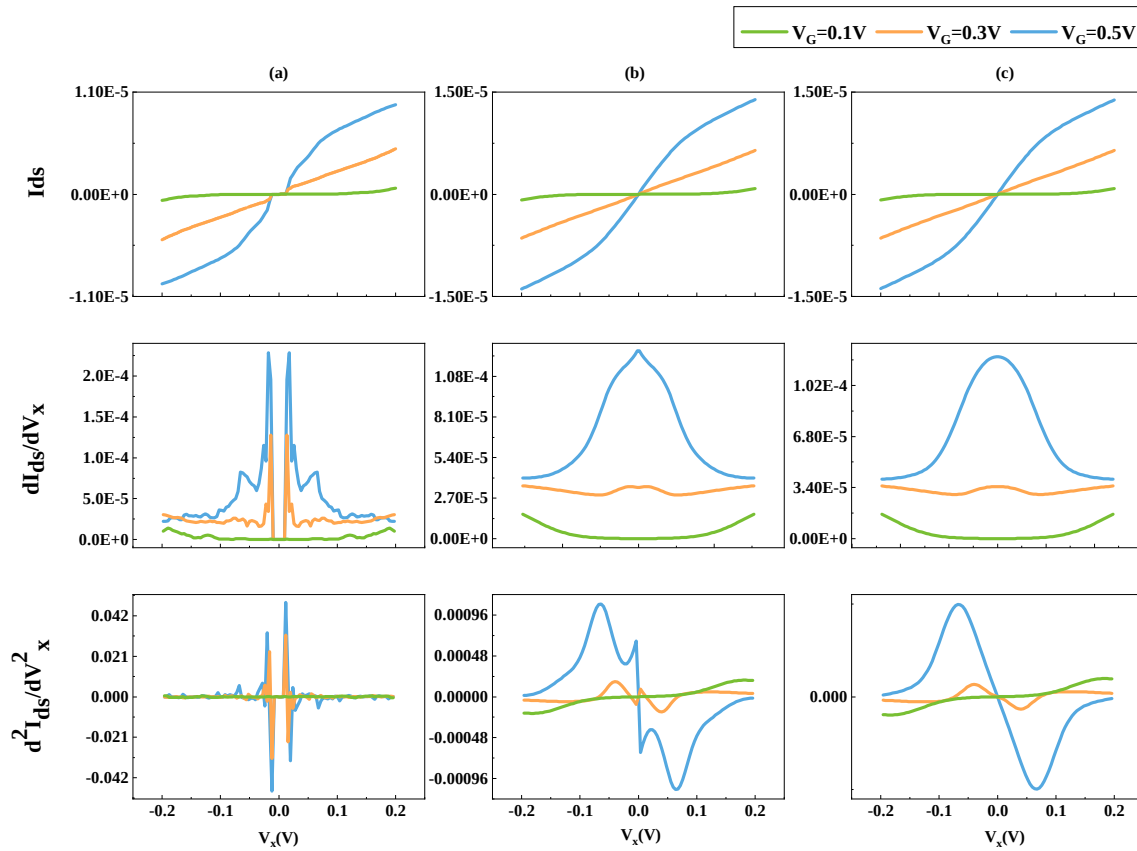


Figure 3. I_{ds} , first, and second derivative with aspect to V_X . (a) NN without the smooth loss function. (b) NN with the smooth loss function. (c) NN with smooth function and domain transform function.

However, a stripe is observed in the first-order and second-order derivatives of the predicted curve when V_X is near zero. This issue is addressed in the next section using a technique called Feature Domain Transform.

2.2. Domain Transform

The paper highlights that the most fundamental physical behavior of a device is zero current. The drain current (I_d) should be zero when the drain-source current (V_{DS}) is equal to zero. Additionally, the I_d in both the sub-threshold and saturation regions should achieve high accuracy. However, the sub-threshold I_d is too small to distinguish at the normal scale. Klemme et al. [9] used two separate nets, which may introduce discontinuities and non-smoothness near the connection. To address this issue, the paper proposes transforming I_d to the logarithmic scale. To ensure physical accuracy, the output of the NN (y) is defined in a way that constrains I_d to be zero when V_{DS} is zero, irrespective of the NN output. The transforming function is as follows in Equation (3).

$$y = \ln\left(\frac{I_{DS}}{V_{DS}}\right) \quad (3)$$

As mentioned earlier, the output of the NN is in the logarithmic scale, which is essential for sub-threshold region modeling. However, the training data are discretely sampled from TCAD simulation, and the change in I_d becomes very sharp in the logarithmic scale when V_D approaches zero, as shown in Figure 4. This poses a challenge for the NN as it tends to treat sharp changes as outliers.

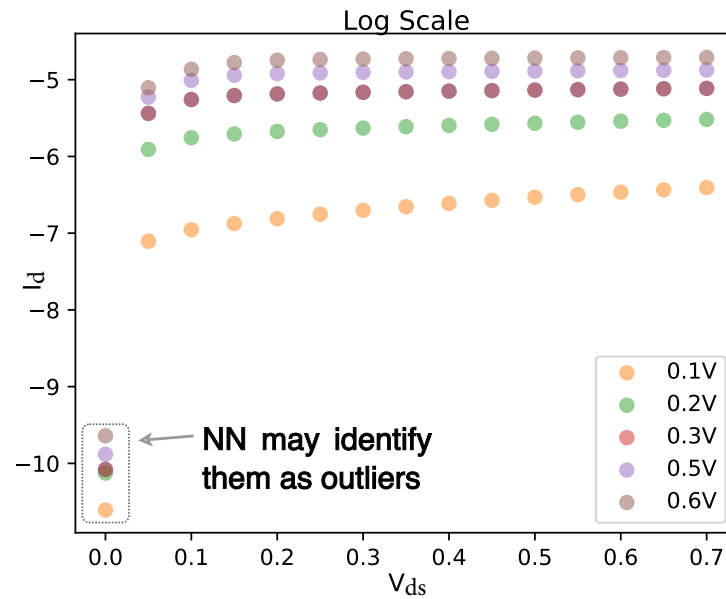


Figure 4. Drain current I_d in the logarithmic scale. The points in the left bottom may be identified as outliers.

To solve this problem, the paper proposes the Domain Transform function for V_{ds} and V_{gs} , as shown in Equations (4) and (5), respectively.

$$V_{ds_new} = \text{sign}(V_{ds}) \times ((V_{ds}^2 + \gamma^2)^{1/2} - \gamma) \quad (4)$$

$$\gamma = Q_1(\text{Range}(V_{ds}))$$

$$V_{gs_new} = V_{gs} + (V_{ds_new} - V_{ds})/2 \quad (5)$$

Here, γ represents the first quartile (Q_1) of the V_{ds} range. This function squeezes V_{ds} in Q_1 to increase data sample density, which is beneficial for fitting the trend when V_{ds} approaches zero. The $\text{sign}(V_{ds})$ factor allows the function to handle both the symmetry of I_d with respect to V_{ds} and the unsymmetrical cases [11]. Moreover, to balance the effects of V_{ds} transformation, a bias is added to V_{gs} . As the GST results shown in Figure 3c, the I_d when V_{ds} near zero is pretty smooth, even for the second-order derivative.

In device modeling, there are several electrical targets that must be achieved. In this paper, we evaluate the error for four targets: threshold voltage (V_{th}), saturation drain current (I_{dsat}), off-state current (I_{off}), and drain current (I_{ds}). The metrics used to measure the error are Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Percent Error (RMSPE), as shown in Equation (6).

$$RMSPE(I_{ds}) = \sqrt{\frac{1}{m} \sum_{i=1}^m \left(\frac{\log(I_{ds, True, i}) - \log(I_{ds, Pred, i})}{\log(I_{ds, True, i})} \right)^2} \quad (6)$$

where m is the total number of samples, i is the i -th sample, $I_{ds, True, i}$ is true I_{ds} value of the i -th test point and $I_{ds, Pred, i}$ is the predicted I_{ds} value of the i -th test point.

The impact of the γ parameter in Equation (4) on accuracy and complexity is presented in Table 1. As the quartile increases, the errors of V_{th} and I_{dsat} increase. However, if the quartile is less than the second quartile, the error increase in all metrics is limited. The training time remains almost the same, taking into account the influence of running conditions. The prediction time may slightly increase due to the transformation function used.

Table 1. Experimental results on different γ parameter.

Metrics	No Transform	First Quartile (Ours)	Second Quartile	Third Quartile
MAE (V_{th})	0.6	0.9	2.3	2.7
MAPE (I_{dsat})	0.37%	0.35%	0.65%	0.62%
MAPE (I_{off})	1.30%	1.22%	1.30%	1.28%
RMSPE (I_{ds})	0.05%	0.05%	0.06%	0.06
Training Time (s)	234.05	217.88	223.6	231.4
Prediction Time (s)	0.71	0.85	0.8	0.83

2.3. Monotonic Network Block

The NN-based model lacks the ability to enforce the desired monotonic dependence between input and output, which is commonly observed in device characteristics. For instance, in a device, the drain current (I_d) exhibits a monotonic relationship with the gate voltage (V_g), and the on-state current (I_{on}) decreases as the gate length (L_g) increases. To overcome this limitation, a Monotonic Block has been proposed to incorporate the knowledge of monotonic device characteristics into NNs. The weights of the Monotonic Block are non-negative, while the weights of the Normal Block range from negative infinity to positive infinity. The non-negative constraint is enforced by squaring the weights, as illustrated in Equation (7).

$$o_j^l = \sigma(\sum_k (W_{jk}^l)^2 o_k^{l-1} + b^l) \quad (7)$$

where o_j^l is the the j -th output of the layer l , σ is the active function, W_{jk}^l is the j -th weight of the layer l connected to k -th weight of the previous layer, and the b^l is the bias of the layer l .

The overall NN architecture includes three input groups: None-Monotonic, Positive, and Negative correlation features, which are combined by the Monotonic Block to produce the output I_d , as shown in Figure 5. This architecture can constrain the correlation effectively to ensure that physical behaviors are not broken, which is critical in device modeling.

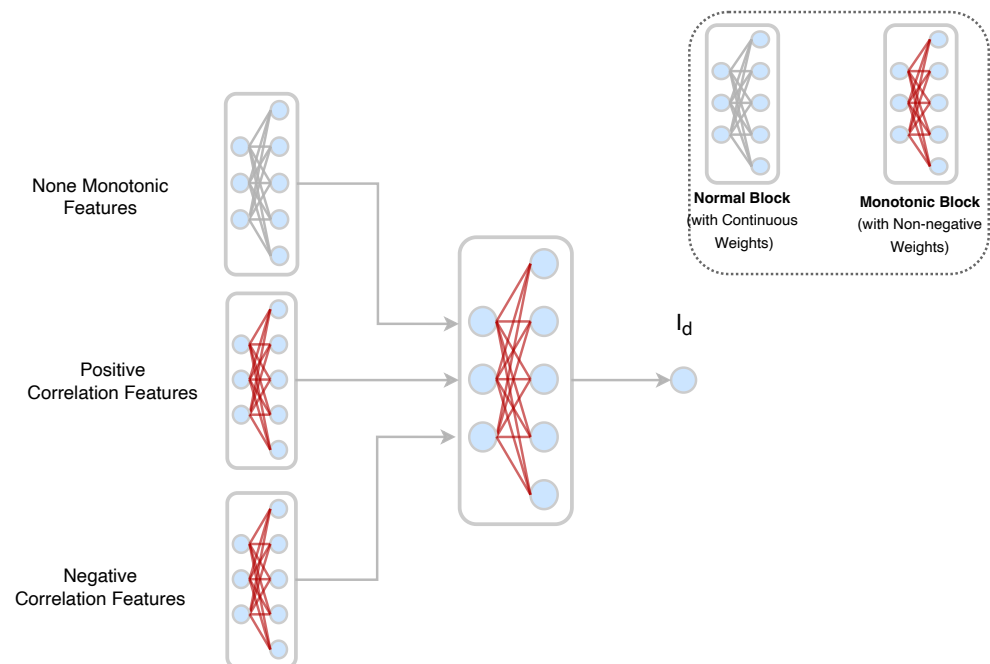


Figure 5. The NN with monotonic block. The block with a gray line represents a normal NN block and the block with a red line represents the monotonic block. The lines in red represent the non-negative weights. The lines in gray represent the continuous values.

2.4. Knowledge Transfer

When a new device is designed, the NN-based compact model cannot leverage previous learning and must train from scratch. This can be time-consuming and the training accuracy cannot be guaranteed. To solve this problem, transfer learning is processed here. By transferring knowledge learned from previous device models to new ones, the training process can be expedited, and the accuracy of the model can be improved. For instance, the knowledge learned from modeling a Planar can be transferred to modeling a GAAFET. Setting the initial weight accordingly can still improve the accuracy of the new model, regardless of whether the NN architecture is the same as the previous one or if only a portion of the architecture is shared. This improvement can be attributed to the shared similarities in their physical behaviors. In Figure 6, transfer learning can significantly improve fitting accuracy after 100 training epochs. To further improve the knowledge transfer quality, some optimization techniques can be utilized, such as fine-tuning a portion of layers and a new learning rate scheduling method [20].

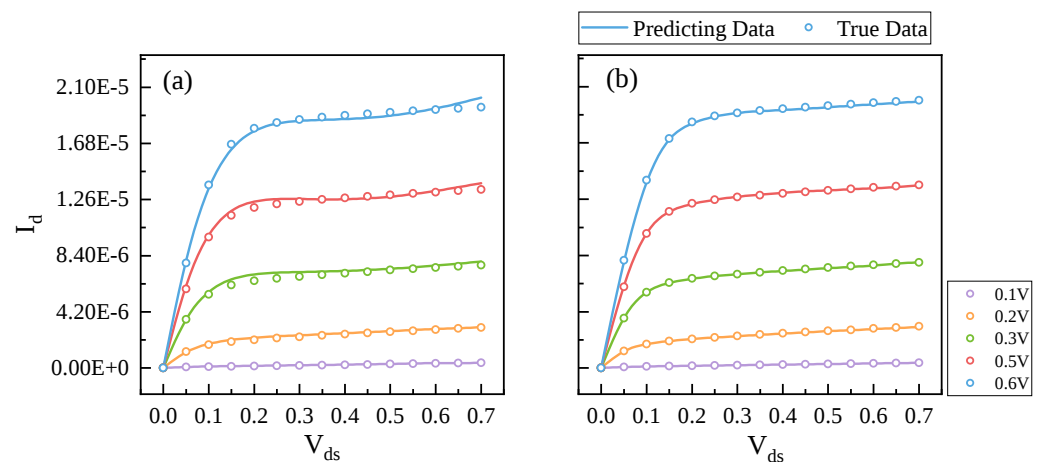


Figure 6. Prediction results of NN. (a) Results without knowledge transfer. (b) Results with knowledge transfer. The lines represent the prediction results. The dots represent the simulation data.

3. Automatic Neural Network Generation Framework

The physical behavior of device modeling is guaranteed by the Physics-Informed Neural Networks (PINN) proposed in Section 2. Another difficulty in NN-based device modeling is achieving accuracy. Obtaining high accuracy often requires considerable time and expertise in Machine Learning (ML), because the different NN parameters have significant impacts on accuracy. To simplify the trial-and-error process, we propose a two-step automatic NN generation flow to obtain an optimal architecture for PINN, as shown in Figure 7. The *Optimal Search Range Generation* first gives a range of NN architecture that is suitable for accuracy. Then, a search in the range with feedback will be performed to find an optimal architecture.

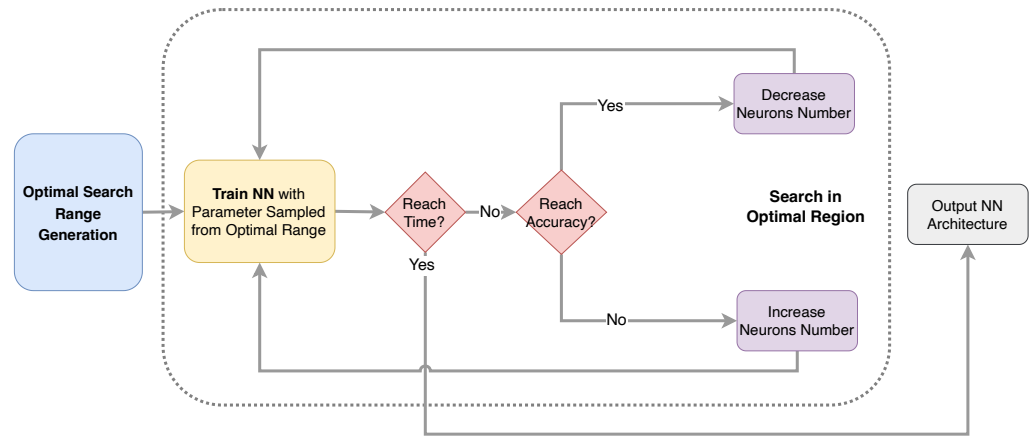


Figure 7. The automatic NN architecture generation flow.

3.1. Optimal Search Range Generation

To ensure high accuracy in device modeling, the complexity of the NN architecture must match the complexity of the problem. If the NN is too powerful, it may overfit, while if it is too simple, it may underfit. To reduce the search range and find an appropriate NN architecture, we propose the *Optimal Search Range Generation* in Equation (8), inspired by the Vapnik–Chervonenkis dimension, a neural network learnability metric [21].

$$N = \prod_{i=1}^m \log_b(n_i + b) \times e^{\sqrt{\text{corr}_i}} / s \quad (8)$$

$$\text{Range}_{l1} = N \pm \max(N \times 0.2, N_{\min})$$

Here, N represents the center of search range result. m is the input feature number, n_i is the number of samples for feature i , and corr_i is the correlation coefficient of the feature i . The base factor is denoted by b and the scale factor by s , and we set 4 as the default. The resulting range of layer one is denoted by Range_{l1} , while the range of layer two is half of Range_{l1} . The $N_{\min}$ is set to 8.

3.2. Search in Optimal Region

After determining the optimal search range for the NN architecture, the next step is to find the optimal accuracy while taking into account constraints on prediction time and other criteria. In this search process, meeting the desired prediction time is the primary condition. If the NN achieves the desired accuracy within the given prediction time constraint, the number of neurons is decreased to further reduce the prediction time. On the other hand, if the desired accuracy is not achieved, the number of neurons is randomly increased within the optimal search range in order to improve accuracy.

The error changes at the AutoNN procedure for GAAFET device modeling is shown in Figure 8, where the red rectangle represents the optimal search range generated using the proposed Equation (8). The final goal is to obtain an optimal architecture that balances accuracy and lightweight. The results demonstrate that the *Optimal Search Range* can provide a suitable NN architecture. Furthermore, it has been observed that having a low number of neurons in the first layer of the NN hampers its ability to extract sufficient information for achieving high accuracy. On the other hand, increasing the number of neurons in the first layer can lead to overfitting, particularly when the NN's representational power becomes excessively high. Moreover, based on the distribution of loss, it is evident that layer 1 of the NN has a greater influence compared to layer 2. This suggests that the initial layer plays a crucial role in capturing and representing the essential features and patterns in the data, while the subsequent layers may further refine and process this information.

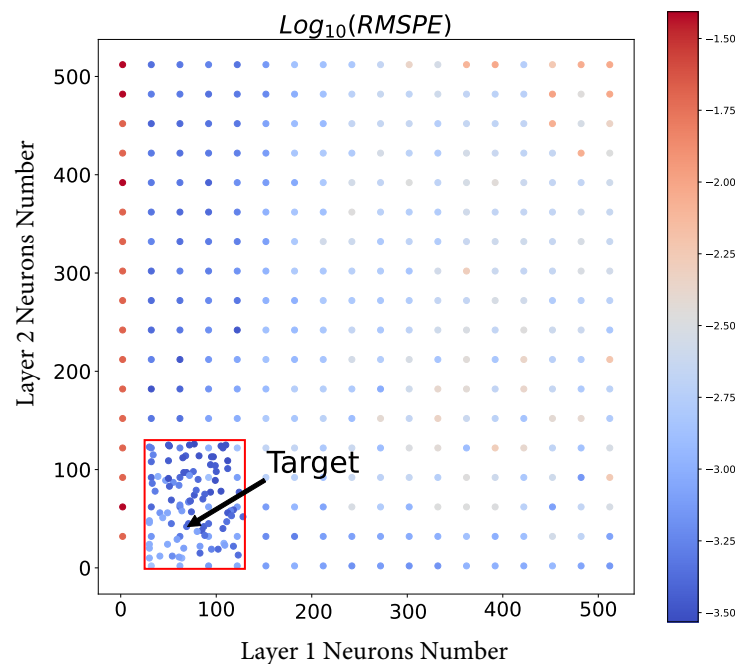


Figure 8. The error changes at the automatic NN structure searching process. The rectangle in red is generated by the *Optimal Search Range Generation* operation. The target is found by *Search in Optimal Region* operation.

4. Experimental Results and Discussion

4.1. Environment Setup

Our framework was evaluated on the open-source GAAFET dataset [22]. The values for V_{dd} and V_{dlin} were 0.5 V and 0.1 V, respectively. Table 2 presents the boundaries and sample number for each of the five input parameters. The dataset contained 98,175 samples, which were split into 68,595 samples (70%) for training and 29,580 (30%) for testing.

Table 2. The electrical parameters boundaries and sample number of a GAAFET dataset.

Parameters	Lower Boundary	Upper Boundary	Sample Number
l_{gnw}	10	20	11
r_{nw}	2	5	7
t_{ox}	0.5	1.5	5
V_g	0	0.7	15
V_d	0	0.7	15

4.2. AutoPINN Physical Behaviors

To check the smoothness, Figure 3 exhibits the Gummel Symmetry Test results of different NNs. The results of the default NN implemented in Pytorch targeting tabular data [23] are shown in Figure 3a. This is a NN-based model without physical information embedded. It is suffering unphysical behaviors: I_d is not smooth and differentiable, and it is not monotonic, meaning it does not consistently increase or decrease, and it reaches zero at an early stage when V_{ds} is not zero. After some physical information is embedded, the prior work [10] shows better results. However, it is also unsmooth near $V_{ds} = 0$, as shown in Figure 3b. Ours shows promising physical behaviors as shown in Figure 3c.

To check the monotonic, Figure 9 shows the monotonic relationship between inputs and output. After adding a negative constraint on I_{ds} and L_g , the curve becomes monotonic and smooth from Figure 9a,b. Figure 9c,d show the relationship between V_{ds} and I_{ds} without and with monotonic block. The block obviously solves the no-monotonic at saturation region.

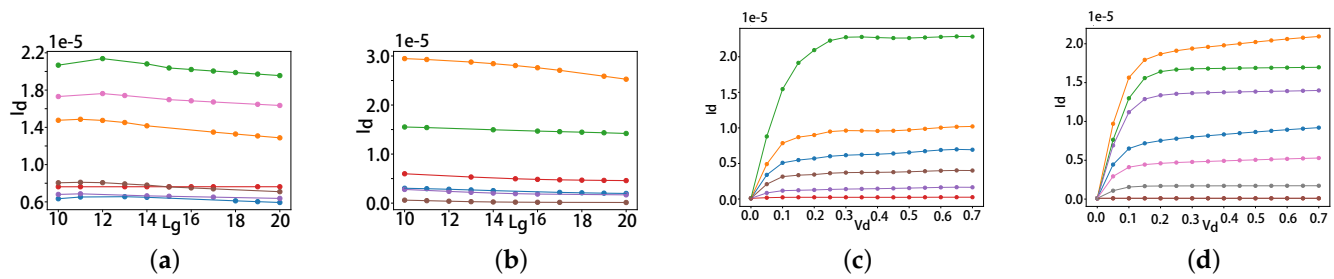


Figure 9. Relationship between (a) I_{ds} and L_g without monotonic block. (b) I_{ds} and L_g with monotonic block. (c) I_{ds} and V_{ds} without monotonic block. (d) I_{ds} and V_{ds} with monotonic block. The index is a randomly selected data sample number in the public dataset [22].

4.3. AutoPINN Accuracy

Our framework was compared with several existing models that specifically target universal tabular data. These models include FastAI [24], Pytorch NN model [23], and the ensemble learning model released by Autogluon [25]. Additionally, we compared our framework with the prior work released by Tung et al. in 2022 [10]. Table 3 shows the experimental results. Compared to the best-performing model among them, our AutoPINN can reduce the MAE of V_{th} by 95%, and MAPE of I_{dsat} and I_{off} by 15% and 73%, respectively. In addition, the total curve of I_{ds} can be reduced by 72%, as shown in Figure 10.

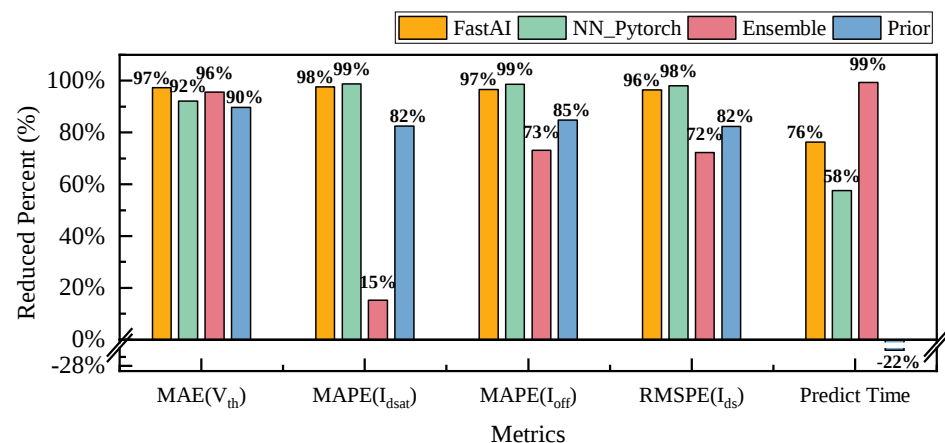


Figure 10. The reduced percent compared to FastAI, NN proposed by Pytorch, and Ensemble learning.

Table 3. Experimental results compared to FastAI, NN proposed by Pytorch, and Ensemble learning.

Metrics	FastAI [24]	NN_Pytorch [23]	Ensemble [25]	Prior [10]	Ours	Reduced Percent Compared to Ensemble
$MAE(V_{th})$	0.0351	0.0118	0.0209	0.009	0.0009	95.6%
$MAPE(I_{dsat})$	0.1487	0.2865	0.0041	0.02	0.0035	15.2%
$MAPE(I_{off})$	0.3542	0.9003	0.0454	0.08	0.0122	73.1%
$RMSPE(I_{ds})$	1.48%	2.69%	0.19%	0.3%	0.05%	72.2%
Training Time (s)	38.23	301.30	476.10	124	217.88	54.2%
Prediction Time (s)	3.61	2.01	123.25	0.7	0.85	99.3%

The total I_{ds} curve error can be substantially reduced to 0.05% by our AutoPINN. It should be noted that other machine learning algorithms have a tendency to perform un-physical behaviors that are not acceptable for real applications. The prior work [10] was evaluated using their default settings. The accuracy of it is less than the ensemble learning. It reflects that the AutoNN is necessary to obtain high accuracy.

Prediction time is also significantly improved. When evaluated on the 29,580 test data samples with 116 calls, each call evaluating 255 samples, our AutoPINN only took 0.85 s to

execute. Ours reduced the time by 99% compared to the ensemble learning model. The multiple calls used here aim to account for the warm-up time of the model.

4.4. AutoPINN Generalization

The generalization refers to the ability to fit unseen data, i.e., test data. The generalization problem arose from the training and testing data usually having different distributions. There are two methods to evaluate the generalization of the NN. One is the NN accuracy on test data, which is the most important metric. Another is the loss landscape. The NN accuracy on test data is promising in Section 4.3. This section mainly talks about the loss landscape to reveal the generalization of NN.

The loss landscape is an intuitive way to visualize the generalization. The NN with a flat loss landscape has a better generalization than the sharp one [26,27]. To generate the loss landscape, Equation (9) is a widely adopted method.

$$\tau(\alpha, \beta; \theta^*) = L(\theta^* + \alpha\delta + \beta\eta) \quad (9)$$

where θ^* is the normalized weight of trained NN, and δ and η are two random directions of the parameter θ^* . The α and β are the factors applied in two directions. They control how far away from original parameter θ^* . Varying α and β from -1 to 1 is used to obtain the loss and draw the loss landscape.

Figure 11a presents the loss landscape of a not well-optimized Neural Network structure, i.e., (5,8,8,8,1). The minimum number of this loss landscape is ten while our well-optimized NN only has one minimum, as shown in Figure 11b. The more local minimum number in the loss landscape, the harder to achieve convergence in the training process. Ours is also flatter than the not well-optimized NN. The good accuracy on test data and the flat loss landscape shows that our NN model can achieve high generalization.

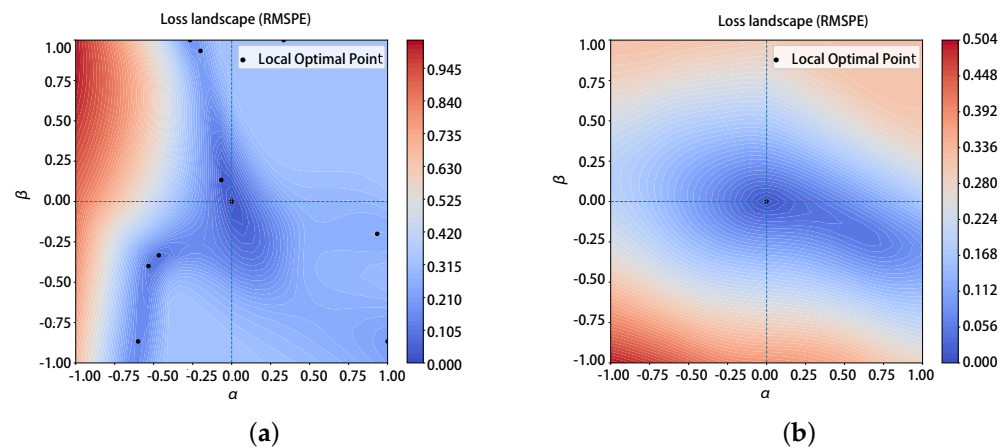


Figure 11. Loss landscape visualization. The black dots represent local optimal points. (a) Loss landscape of a not well-optimized Neural Network structure; (b) loss landscape of our well-optimized Neural Network structure.

5. Conclusions

This paper presents a novel framework called AutoPINN for NN-based semiconductor device modeling. AutoPINN solves two major challenges: unphysical behaviors and the requirement for NN expertise. The framework consists of two components: PINN and AutoNN.

PINN is introduced to tackle unphysical issues by incorporating physical information using several key technologies. There are a few key technologies used here. The *Domain Transform* ensures that the current–voltage relationship (I_d vs. V_{gs} and V_{ds}) is smooth and differentiable by transforming them with higher density near $V_{ds} = 0$. This transformation handles both symmetric and asymmetric drain/source scenarios. It also transforms the optimization target to simplify NN fitting. The *Smooth Loss Function* considers not only the

optimization target, but also derivatives and other factors to ensure a smooth and differentiable I_d curve. The *Monotonic Network Block* enforces non-negativity constraints on the NN weights to achieve monotonic behavior. *Knowledge Transfer* enables the transfer of modeling information and training from other devices, facilitating faster training convergence and improved physical behavior.

AutoNN assists PINN in finding an optimal structure without requiring human expertise. It generates an optimal search range for the NN architecture and optimizes accuracy while considering constraints such as prediction time and other criteria.

The effectiveness of the AutoPINN framework is demonstrated through experiments on a GAAFET device. The results show high accuracy while maintaining a lightweight model. To ensure generalization, validation results on sample data as well as the loss landscape are utilized to confirm the approach's ability to generalize well. The authors believe that this work has the potential to accelerate the development and simulation processes of emerging devices.

Author Contributions: Conceptualization, G.G.; funding acquisition, H.Y. and C.L.; investigation, G.G., Z.T. and O.L.; methodology, H.Y., C.L. and Z.T.; project administration, H.Y. and C.L.; software, G.G.; supervision, H.Y., C.L. and Z.T.; validation, O.L.; writing—original draft, G.G.; writing—review and editing, G.G., H.Y., C.L. and Z.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the cooperation project between Xidian University and Beijing Microelectronics Technology Institute, in part by the Project of Science and Technology on Reliability Physics and Application Technology of Electronic Component Laboratory (under Grant 6142806210302), in part by the China National Key R&D Program (Grant No. 2022YFF0605800) in part by the 111 Project of China (Grant No. 61574109), and in part by the National Natural Science Foundation of China (Grant No. 62234010).

Data Availability Statement: The training and testing dataset is available on GitHub [22]. The other data presented in this study are available on request from the corresponding author.

Acknowledgments: We would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript. It is important to acknowledge the contributions and support of those who have helped in the research and preparation of the paper. We would like to express our gratitude to the Integrated Circuit EDA Elite Challenge Contest and Primarius Technologies Co., Ltd. [28] for providing the TCAD simulation data. Special thanks to Xiaoxu Cheng from Primarius Technologies Co., Ltd. for their assistance. We would also like to acknowledge the contributions of Ruihua Xue for providing the beautiful pictures used in the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
ANN	Artificial Neural Network
NN	Neural Network
AutoNN	Automatic Neural Network Generation
PINN	Physical-Informed Neural Network
AutoPINN	Automatic Physics-Informed Neural Network Generation
L_g	Length of Gate
r_{nw}	radius of nanwire
t_{ox}	Thickness of oxide
V_g	Gate Voltage
V_d	Drain Voltage

References

1. 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. [\[CrossRef\]](#)
2. Duarte, J.P.; Khandelwal, S.; Medury, A.; Hu, C.; Kushwaha, P.; Agarwal, H.; Dasgupta, A.; Chauhan, Y.S. BSIM-CMG: Standard FinFET compact model for advanced circuit design. In Proceedings of the ESSCIRC Conference 2015—41st European Solid-State Circuits Conference (ESSCIRC), Graz, Austria, 14–18 September 2015; pp. 196–201. [\[CrossRef\]](#)
3. Khandelwal, S.; Chauhan, Y.S.; Lu, D.D.; Venugopalan, S.; Ahosan Ul Karim, M.; Sachid, A.B.; Nguyen, B.Y.; Rozeau, O.; Faynot, O.; Niknejad, A.M.; et al. BSIM-IMG: A Compact Model for Ultrathin-Body SOI MOSFETs With Back-Gate Control. *IEEE Trans. Electron Devices* **2012**, *59*, 2019–2026. [\[CrossRef\]](#)
4. Gildenblat, G.; Li, X.; Wu, W.; Wang, H.; Jha, A.; Van Langevelde, R.; Smit, G.; Scholten, A.; Klaassen, D. PSP: An Advanced Surface-Potential-Based MOSFET Model for Circuit Simulation. *IEEE Trans. Electron Devices* **2006**, *53*, 1979–1993. [\[CrossRef\]](#)
5. Wang, J.; Kim, Y.H.; Ryu, J.; Jeong, C.; Choi, W.; Kim, D. Artificial Neural Network-Based Compact Modeling Methodology for Advanced Transistors. *IEEE Trans. Electron Devices* **2021**, *68*, 1318–1325. [\[CrossRef\]](#)
6. Li, M.; Irsoy, O.; Cardie, C.; Xing, H.G. Physics-Inspired Neural Networks for Efficient Device Compact Modeling. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **2016**, *2*, 44–49. [\[CrossRef\]](#)
7. Zhang, L.; Chan, M. Artificial neural network design for compact modeling of generic transistors. *J. Comput. Electron.* **2017**, *16*, 825–832. [\[CrossRef\]](#)
8. Lei, Y.; Huo, X.; Yan, B. Deep Neural Network for Device Modeling. In Proceedings of the 2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM), Kobe, Japan, 13–16 March 2018; pp. 154–156. [\[CrossRef\]](#)
9. Klemme, F.; Prinz, J.; van Santen, V.M.; Henkel, J.; Amrouch, H. Modeling Emerging Technologies using Machine Learning: Challenges and Opportunities. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual, 2–5 November 2020; pp. 1–9.
10. Tung, C.T.; Kao, M.Y.; Hu, C. Neural Network-Based I-V and C-V Modeling With High Accuracy and Potential Model Speed. *IEEE Trans. Electron Devices* **2022**, *69*, 6476–6479. [\[CrossRef\]](#)
11. Kao, M.Y.; Kam, H.; Hu, C. Deep-Learning-Assisted Physics-Driven MOSFET Current-Voltage Modeling. *IEEE Electron Device Lett.* **2022**, *43*, 974–977. [\[CrossRef\]](#)
12. Butola, R.; Li, Y.; Kola, S.R. A Machine Learning Approach to Modeling Intrinsic Parameter Fluctuation of Gate-All-Around Si Nanosheet MOSFETs. *IEEE Access* **2022**, *10*, 71356–71369. [\[CrossRef\]](#)
13. Park, C.; Vincent, P.; Chong, S.; Park, J.; Cha, Y.S.; Cho, H. Hierarchical Mixture-of-Experts approach for neural compact modeling of MOSFETs. *Solid-State Electron.* **2023**, *199*, 108500. [\[CrossRef\]](#)
14. Jeong, H.; Woo, S.; Choi, J.; Cho, H.; Kim, Y.; Kong, J.T.; Kim, S. Fast and Expandable ANN-Based Compact Model and Parameter Extraction for Emerging Transistors. *IEEE J. Electron Devices Soc.* **2023**, *11*, 153–160. [\[CrossRef\]](#)
15. Yang, Q.; Qi, G.; Gan, W.; Wu, Z.; Yin, H.; Chen, T.; Hu, G.; Wan, J.; Yu, S.; Lu, Y. Transistor Compact Model Based on Multigradient Neural Network and Its Application in SPICE Circuit Simulations for Gate-All-Around Si Cold Source FETs. *IEEE Trans. Electron Devices* **2021**, *68*, 4181–4188. [\[CrossRef\]](#)
16. Myung, S.; Kim, J.; Jeon, Y.; Jang, W.; Huh, I.; Kim, J.; Han, S.; Baek, K.H.; Ryu, J.; Kim, Y.S.; et al. Real-Time TCAD: A new paradigm for TCAD in the artificial intelligence era. In Proceedings of the 2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), Kobe, Japan, 23 September–6 October 2020; pp. 347–350. [\[CrossRef\]](#)
17. Jagannathan, H.; Anderson, B.; Sohn, C.W.; Tsutsui, G.; Strane, J.; Xie, R.; Fan, S.; Kim, K.I.; Song, S.; Sieg, S.; et al. Vertical-Transport Nanosheet Technology for CMOS Scaling beyond Lateral-Transport Devices. In Proceedings of the 2021 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 11–16 December 2021; pp. 26.1.1–26.1.4. [\[CrossRef\]](#)
18. Huang, S.; Wang, L. MOSFET Physics-Based Compact Model Mass-Produced: An Artificial Neural Network Approach. *Micromachines* **2023**, *14*, 386. [\[CrossRef\]](#) [\[PubMed\]](#)
19. Mcandrew, C. Validation of MOSFET model Source–Drain Symmetry. *IEEE Trans. Electron Devices* **2006**, *53*, 2202–2206. [\[CrossRef\]](#)
20. Alqudah, A.M.; Alquraan, H.; Qasmieh, I.A. Segmented and Non-Segmented Skin Lesions Classification Using Transfer Learning and Adaptive Moment Learning Rate Technique Using Pretrained Convolutional Neural Network. *J. Biomim. Biomater. Biomed. Eng.* **2019**, *42*, 67–78. [\[CrossRef\]](#)
21. Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M.K. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM* **1989**, *36*, 929–965. [\[CrossRef\]](#)
22. AlforDeviceModeling. AlforDeviceModeling/Semiconductor_Device_Data. 2023. Available online: https://github.com/AlforDeviceModeling/Semiconductor_Device_Data (accessed on 15 March 2023).
23. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimselshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
24. Fastai. 2023. Available online: <https://github.com/fastai/fastai> (accessed on 15 March 2023).
25. Erickson, N.; Mueller, J.; Shirkov, A.; Zhang, H.; Larroy, P.; Li, M.; Smola, A. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv* **2020**, arXiv:2003.06505.
26. Li, H.; Xu, Z.; Taylor, G.; Studer, C.; Goldstein, T. Visualizing the Loss Landscape of Neural Nets. *arXiv* **2018**, arXiv:1712.09913.

27. Keskar, N.S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; Tang, P.T.P. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *arXiv* **2017**, arXiv:1609.04836.
28. Primarius Technologies Co., Ltd. Primarius, Memory EDA Leader. 2023. Available online: <https://www.primarius-tech.com> (accessed on 15 March 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.