Multi-order Differential Neural Network for TCAD Simulation of the Semiconductor Devices

Zifei Cai*
1575803685@qq.com

AnAoxue Huang*
1353227546@qq.com

Yifeng Xiong*
327988214@qq.com

Dejiang Mu*
2741105136@qq.com

Xiangshui Miao*
miaoxs@hust.edu.cn

Xingsheng Wang*†
xswang@hust.edu.cn

ABSTRACT

Technology Computer Aided Design (TCAD) is a crucial step in the design and manufacturing of semiconductor devices. It involves solving physical equations that describe the behavior of semiconductor devices to predict various device parameters. Traditional TCAD methods, such as finite volume and finite element methods, discretize relevant physical equations to achieve numerical simulations of devices, significantly burdening the computation resources. For the first time, this paper proposes a novel method for TCAD simulation based on Physics-Informed Neural Networks (PINNs). We proposed multi-order differential neural network (MDNN), an improved Radial Basis Function Neural Network (RBFNN) model. By training MDNN, it achieves the couple solution of the Poisson equation and drift-diffusion equation under steady-state conditions, without the need for a pre-existing dataset. To the best of our knowledge, this marks the first instance of an ML-TCAD simulation that does not require any pre-existing data. For an example of PN junction diode, this method effectively simulates the basic physical characteristics of the device, with a self-consistent solution error of less than 1×10^{-5}.

1 INTRODUCTION

For the entire integrated circuit industry, Electronic Design Automation (EDA) tools play an indispensable role. TCAD is an essential component within the EDA toolchain, providing irreplaceable functions in the design and optimization process of devices. Currently, the integration of EDA and machine learning has become a trend, with an increasing number of studies exploring the combination of machine learning and TCAD [1-5].

*School of Integrated Circuits, Huazhong University of Science and Technology, Wuhan, Hubei, China.

† Xingsheng Wang is the corresponding author.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM 979-8-4007-0601-1/24/06…

https://doi.org/10.1145/3649329.3656215

These studies specifically include the use of trained Multilayer Perceptron (MLP) models to predict device current-voltage (I_D-V_G) [1,2] curves or employing Graph Neural Network (GNN) [3,4] models to predict the spatial distribution of physical quantities such as device's potential and carrier density. However, all these research achievements are based on data-driven methods. In other words, the realization of these outcomes still relies on the simulation results of traditional TCAD tools, such as finite volume and finite element methods, as their prerequisite conditions. To date, there have been no reported research achievements that directly employ machine learning techniques to self-consistently solve relevant physical model differential equations (e.g., Poisson equation, drift-diffusion equation, current continuity equation) for device simulation.

Now, let us shift our focus to the field of AI/ML for Science. It is evident that within the past 5 to 6 years, a plethora of research achievements have emerged that employ machine learning to directly solve partial differential equations (PDEs) [6-11]. The PDEs covered in these scientific publications span various areas, including electromagnetism [8], fluid mechanics, quantum mechanics [9], seismic wave propagation [10], and even the spread of the COVID-19 virus [11]. The origin of all these studies can be traced back to the introduction of Physics-Informed Neural Networks (PINNs).

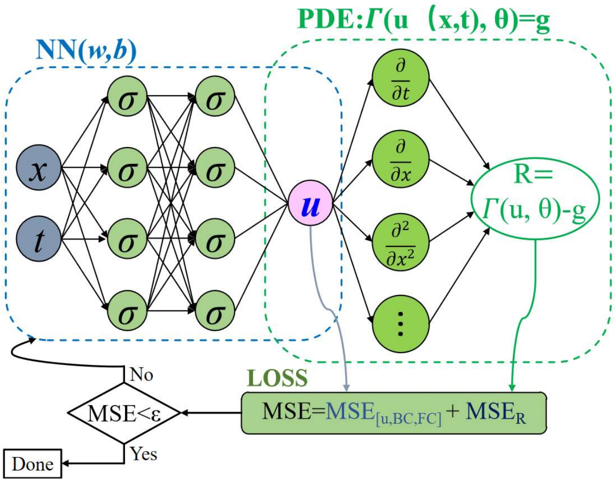

Figure 1: Schematic diagram of the PDE numerical solution process performed by PINN.

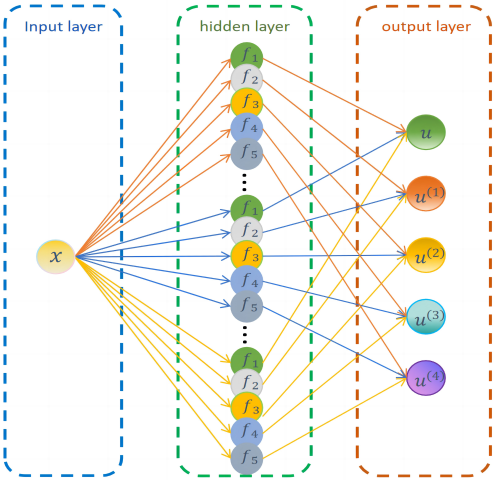

Figure 2: A schematic representation of the Multi-order Differential Neural Network (MDNN) model. In the diagram, lines with the same color indicate the same linear transformation. Neurons with the same color in the hidden layer represent the same type of adaptive kernel function.

PINN, proposed by Raissi et al. in 2017, is a machine learning method for obtaining numerical solutions to differential equations. Compared with the finite volume method and finite element method, PINN is a brand new numerical solution paradigm [6,7]. The basic idea of PINN is to define the loss function as the residual of the artificial neural network output with respect to the governing differential equation, boundary conditions, and initial conditions. Consequently, through methods such as gradient descent, the loss function value of the neural network is minimized during the training process. This enables the artificial neural network to satisfy the governing differential equation, boundary conditions, and initial conditions constraints as closely as possible. The output that satisfies these constraints represents the numerical approximation of the sought-after solution to the differential equation.

Compared to traditional numerical methods such as finite difference, finite volume, and finite element methods, PINN offers the following advantages. First, the method does not require complex mesh generation or the creation of matrix equations based on mesh structure for numerical solving. Second, by using numerical differentiation, the PINN model can output high-order partial derivatives of the approximate solution at any position within the domain. Third, PINN has been successfully applied to solve high-dimensional (100-dimensional or even higher) partial differential equations [12,13]. As a result, it holds promise for addressing the "curse of dimensionality" problem faced by Design-Technology Co-Optimization (DTCO) in simulating complex structured devices [1]. Last but not least, artificial neural networks are mathematically continuous models. Continuous models provide more flexibility, making them easier to adapt to complex forms of partial differential equations and more amenable to modifications and optimizations compared to discrete models [14]. For these reasons, our paper presents the first exploration of using PINN for TCAD simulation, filling a gap in this research direction.

In this paper, we introduce a Multi-order Differential Neural Network (MDNN) model and associated simulation and optimization algorithms for the self-consistent solution of Poisson's equation and drift-diffusion equations under steady-state conditions. By replacing the deep fully connected neural network (DNN) used in traditional PINN methods with a composite MDNN model, we obtain a novel machine learning approach for solving coupled Poisson's equation and carrier transport equations.

The structure of this paper is as follows: In Section 2, we provide a detailed description of the mathematical formulation of MDNN, the process of performing numerical simulations of PN junction diodes using MDNN, and the accuracy improvement algorithm based on the Residual Neural Network (Res-Net) concept. In Section 3, we present numerical simulation results of PN junction diodes under various conditions, the gradient descent process of the loss function, and compare the carrier and the potential distributions obtained through MD-PINN simulation with the corresponding simulation results from Sentaurus TCAD. The results demonstrate that the improved RBFNN model has the capability to accurately simulate semiconductor devices. Finally, Section 4 concludes the paper and provides an outlook on future research in Machine Learning-TCAD.

2 NEURAL NETWORK AND SIMULATION PROCESS

In this section, we will provide a detailed description of the proposed Multi-order Differential Neural Network Model (MDNN) and the specific process of utilizing MDNN for the simulation of semiconductor devices.

2.1 Multi-order Differential Neural Network Model

The Multi-order Differential Neural Network model proposed in this paper is shown in Figure 2. MDNN is a single-hidden-layer neural network model developed as an improvement upon the RBFNN (Radial Basis Function Neural Network). Its distinctive feature lies in its ability to accurately and efficiently output the partial derivatives of a specific quantity with respect to the input variables of the MDNN model. In the one-dimensional case, MDNN has one input term representing spatial position and a total of five output terms: u, $u^{(1)}$, $u^{(2)}$, $u^{(3)}$, $u^{(4)}$. Among them $u^{(1)}, u^{(2)}, u^{(3)}, u^{(4)}$ are the first to fourth-order partial derivatives of u with respect to the neural network input variable x, respectively.

Unlike traditional single-hidden-layer models such as RBFNN, the hidden layer of the MDNN model is composed of several groups of neurons, with each group containing five neurons with distinct kernel functions (activation functions). For the hidden layer, the five neurons within the same group share the same bandwidth parameters w and kernel function centers b. For the output layer, each group of neurons in the hidden layer also shares the same weights A. Meanwhile, the biases for the output layer are all set to zero. The only difference among these five neurons within the same group in the hidden layer is that they each have their own distinct adaptive kernel functions: $f_1(t)$, $f_2(t)$, $f_3(t)$, $f_4(t)$, $f_5(t)$. In order to make the five output terms of the MDNN model satisfy the required mathematical relationships, we chose the mathematical forms of these adaptive kernel functions to be as follows after extensive numerical experiments:

$$f_1(t) = \frac{\sqrt{\pi}\, t\, erf(t) + e^{-t^2}}{2w^2} \tag{1}$$

$$f_2(t) = \frac{\sqrt{\pi}\ erf(t)}{2w} \tag{2}$$

$$f_3(t) = e^{-t^2} \tag{3}$$

$$f_4(t) = -2wte^{-t^2} \tag{4}$$

$$f_5(t) = 2w^2(2t^2 - 1)e^{-t^2} \tag{5}$$

In the above equations (1)-(5), w represents the bandwidth parameters of the corresponding hidden layer neurons, and their specific values will change as the neural network training process progresses. Here, t ($t = w(x - b)$) represents the input to the kernel function (activation function) in the hidden layer neurons, and $erf(t)$ denotes the Gaussian error function.

Additionally, the hidden layer and the output layer of the MDNN model are not fully connected. All neurons of the same type in the hidden layer (i.e., having the same type of adaptive kernel function) are connected only to a specific output term in the output layer. In other words, each output term of the neural network implies the linear weighted sum of the output results of all neurons of the same type in the hidden layer. As a result, we obtain the mathematical expressions for the five output quantities of the constructed MDNN: $u, u^{(1)}, u^{(2)}, u^{(3)}, u^{(4)}$:

$$u = \sum_{i=1}^{n} A_i \left(\frac{\sqrt{\pi}\, w_i(x-b_i) erf\left(w_i(x-b_i)\right) + e^{-w_i^2(x-b_i)^2}}{2w_i^2} \right) \tag{6}$$

$$u^{(1)} = \sum_{i=1}^{n} A_i \left(\frac{\sqrt{\pi}\, erf\left(w_i(x-b_i)\right)}{2w_i} \right) \tag{7}$$

$$u^{(2)} = \sum_{i=1}^{n} A_i \left(e^{-w_i^2(x-b_i)^2} \right) \tag{8}$$

$$u^{(3)} = \sum_{i=1}^{n} A_i \left(-2w_i^2(x - b_i)e^{-w_i^2(x-b_i)^2} \right) \tag{9}$$

$$u^{(4)} = \sum_{i=1}^{n} A_i \left(2w_i^2\left(2w_i^2(x - b_i)^2 1\right)e^{-w_i^2(x-b_i)^2} \right) \tag{10}$$

In the above equations (6)-(10) A_i, w_i and b_i are the output layer weights, bandwidth parameters and kernel function centers of the i-th hidden layer neuron, respectively. n indicates the total number of neuron groups in the entire hidden layer.

The reasons for choosing to construct the neural network model for use in the PINN method are as follows: Firstly, through multiple numerical experiments, we have found that compared to deep neural networks, single-layer neural networks like RBFNN are more likely to escape the "trap" of local optima when solving partial differential equations, thus converging to the desired results with fewer training iterations. Secondly, single-layer neural networks have stronger interpretability, lower training computational costs, and require fewer overall network parameters. In addition, we hope to take advantage of the excellent smoothness and fitting capabilities of Gaussian basis functions to approximate the numerical solution of carrier concentration distribution with steep gradients in space [15]. Finally, our design ensures the analyzability of the MDNN, allowing for efficient and accurate calculation of its output results' various orders of partial derivatives. This not only avoids a large amount of numerical differentiation computation but also ensures that numerical errors generated during the computation process are not progressively amplified with the increase in derivative order.

It is worth noting that the five output quantities of the neural network model proposed in this paper have realistic physical meanings in the subsequent device simulation and solution process. Due to the mathematical relationships between these five quantities, we only need to multiply them by specific constants to represent the five physical quantities in space: potential, electric field strength, charge density, gradient of charge density, and divergence of charge density. The distribution of these five physical quantities is precisely the result of interest in TCAD simulations.

2.2 Device Simulation Process

The core problem to be solved in TCAD for semiconductor device simulation is how to achieve self-consistent solutions for the coupled electric field equations and carrier transport equations [16]. Here, we choose to use the Poisson equation to describe the spatial distribution of the potential and the drift-diffusion equation under steady-state conditions as the carrier transport equation. The specific mathematical forms of these two types of equations in one-dimensional cases are described as follows:

$$\varepsilon \frac{d^2\psi}{dx^2} = -\rho = q(n - p - c) \tag{11}$$

$$D_p \frac{d^2p}{dx^2} + \mu_p \frac{dp}{dx} \cdot \frac{d\psi}{dx} + \mu_p p \frac{d^2\psi}{dx^2} = 0 \tag{12}$$

$$D_n \frac{d^2n}{dx^2} - \mu_n \frac{dn}{dx} \cdot \frac{d\psi}{dx} - \mu_n n \frac{d^2\psi}{dx^2} = 0 \tag{13}$$

In the equations, ψ, n, p represent the three variables to be solved: potential, electron concentration, and hole concentration. ρ is the charge density. c is the net charge density contribution of impurities in the device. ε and q represent the dielectric constant of the device material and the elementary charge, respectively. Finally D_n, μ_n, D_p and μ_n are the diffusion coefficient for electrons, the mobility of electrons, the diffusion coefficient for holes, and the mobility of holes, respectively.

The specific process of using the model combined by MDNNs to self-consistently solve these two coupled equations is shown in Figure 3.Given the material, doping conditions, and other physical properties of a device, we first defined three MDNN models: Model N, Model P, and Model C. Subsequently, we employed the output terms of Model C, Model N, and Model P to accurately

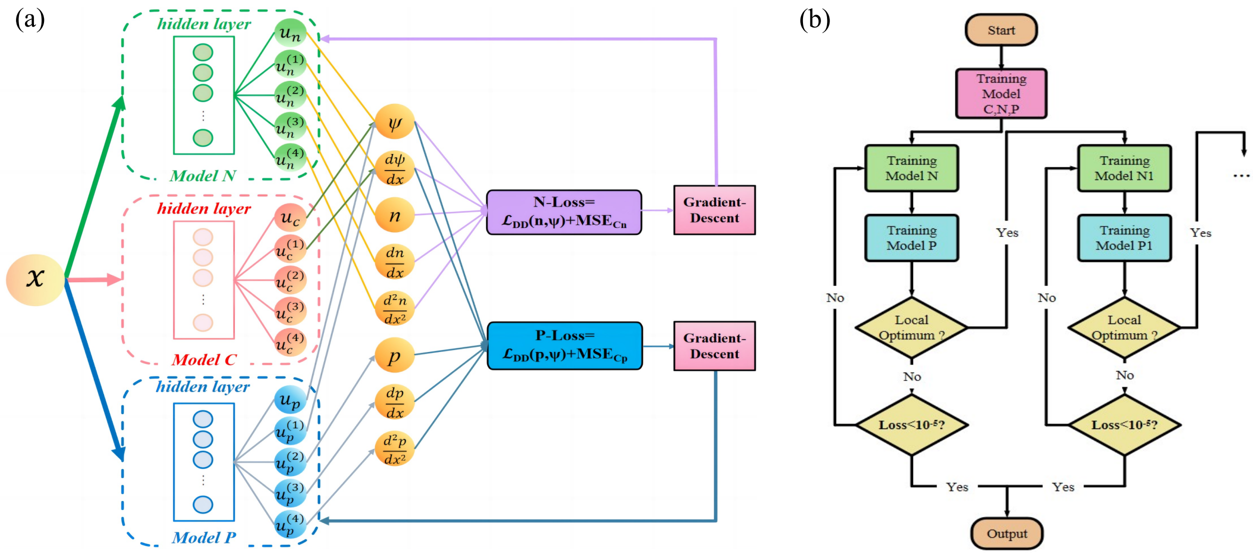

Figure 3: The left diagram (3.a) illustrates the schematic of the combined MDNN model used for self-consistent solution of coupled physical equations, while the right diagram (3.b) represents the specific process of training the combined MDNN model according to the optimization algorithm proposed in this paper to achieve device simulation.

approximate the distributions of net impurity concentration, donor impurity concentration, and acceptor impurity concentration in the device as initial trial solutions. Next, we fixed Model C, ensuring that its internal parameters remained constant throughout the subsequent processes. We then utilized the output terms of Model N and Model P as electron and hole concentrations, respectively[1].

Consequently, physical quantities such as electric potential, electric field intensity, carrier concentration, carrier concentration gradient, and carrier concentration divergence can be successively represented by the output terms of the three MDNN models as follows:

$$\psi = q\left(u_n - u_p - u_c\right)/\varepsilon + Ax + B \tag{14}$$

$$E = -\frac{d\psi}{dx} = -q\left(u_n^{(1)} - u_p^{(1)} - u_c^{(1)}\right)/\varepsilon + A \tag{15}$$

$$n = u_n^{(2)} \quad p = u_p^{(2)} \quad c = u_c^{(2)} \tag{16}$$

$$\frac{dn}{dx} = u_n^{(3)} \quad \frac{dp}{dx} = u_p^{(3)} \tag{17}$$

By leveraging the inherent mathematical relationships between the five quantities obtained from the MDNN outputs, we ensure that the Poisson equation for the electric field is always strictly satisfied throughout the solving process. Simultaneously, to ensure that the obtained potential distribution satisfies not only the constraints of the control equation (11) within the domain but also the boundary conditions, two adjustable undetermined coefficients are introduced here. The specific values of these coefficients are determined by the combined output of the three MDNN models at the boundary. As a result, we have successfully reduced the number of physical quantities to be determined in the TCAD simulation process from three (electric potential, electron concentration, and hole concentration), as required by the currently used Newton-Raphson and Gummel methods, to two (electron concentration and hole concentration).

The next task is to make the output distribution of Model P and Model N satisfy the control equation (12) and control equation (13), as well as their respective boundary conditions. Overall, we aim to achieve this objective by utilizing the gradient-descent method, originating from the basic idea of the PINN (Physics-Informed Neural Network) method. However, numerical experiments indicate that if Model P and Model N are trained simultaneously in parallel, the model is very likely to fall into a local optimum. The value of the loss function will be difficult to quickly decrease to a value close to 0.

Therefore, we draw inspiration from the Gummel method and choose to train Model N and Model P alternately [16]. During the training process, the model being trained only needs to approach the equation and boundary conditions that implicitly contain its output items through gradient descent. For example, when training Model N, the loss function only includes terms related to Equation (13) and the electron concentration boundary conditions. When training a particular model, the parameters of the other models remain unchanged[2]. Furthermore, based on the experience of numerical experiments, this paper chose the AdamW optimizer to train the relevant models.

[1] Of course, it is essential to perform appropriate normalization of input and output quantities such as impurity concentration, electron concentration, hole concentration, and device dimensions to avoid issues such as data overflow.

[2] It is worth mentioning that the number of epochs required for continuous training of a specific model should not be excessive, especially in the early stages of the entire solving process. In this paper, when solving the PN junction diode example, only 50 epochs were used for continuous training of a specific model.

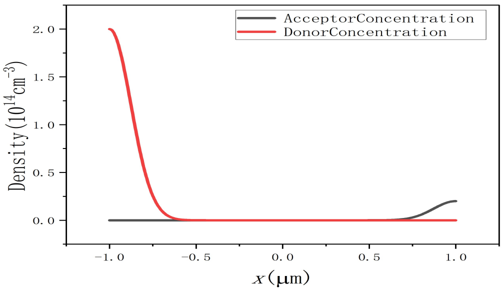

Figure 4: Schematic diagram of the impurity distribution in the PN junction diode to be simulated.

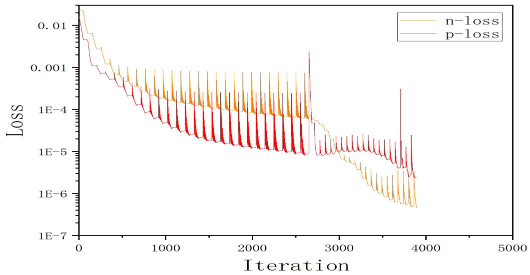

Figure 5: Schematic diagram of the changes in the loss function value of the MDNN composite model during the simulation of the PN junction diode under a 0.4V forward bias. Here, n-loss and p-loss represent the errors in the electron spatial distribution and hole spatial distribution outputs from the MDNN composite model, respectively[3].

Similar to training other neural network models, the MDNN model may also fall into a local optimum after multiple gradient descent processes when simulating semiconductor devices. To address this issue, we propose the following solution [17]: replace a single large-scale combined MDNN model with multiple smaller-scale combined MDNN models. First, train one of the combined models (model N, model P). Once it is found that the value of the loss function remains unchanged after a certain number of gradient descents, switch to a new combined model (model N1, model P1) to solve the residual of the fixed simulation results of the original model at that time. Then, repeat this process continuously until the sum of the loss functions representing the solving error reaches within the allowable error range. The effect of this solution is shown in Figure 5. The schematic diagram of all the training processes mentioned above can be seen in Figure 3.b.

In this way, we have achieved the self-consistent solution of the coupled electric field equation and carrier transport equation through the machine learning method based on the MDNN model. As a result, we have completed the simulation of one-dimensional semiconductor devices without the need for any prior experience data.

[3] By using lightly doped diodes, we can observe a wide range of carrier distributions under various operating conditions, which can help to better evaluate and validate the performance of the proposed MDNN model in simulating different scenarios.

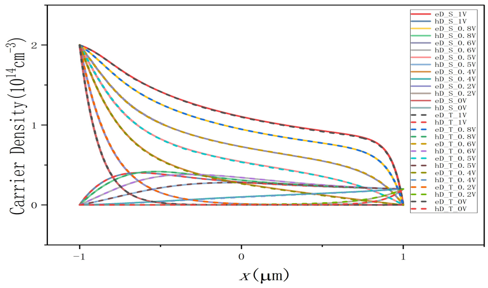

Figure 6: Schematic diagram of the simulation results of electron and hole distributions inside the PN junction diode.

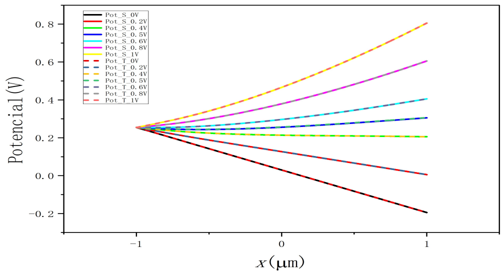

Figure 7: Schematic diagram of the simulation results of potential inside the PN junction diode. In the legend of Figure 6 and Figure 7, eD and hD represent electron concentration and hole concentration, respectively; the solid line (S) represents the results output by Sentaurus TCAD; the dashed line (T) represents the results output by the MDNN model. The value at the end of the legend indicates the applied bias voltage.

3 NUMERICAL RESULTS

To verify the effectiveness of the machine learning simulation method proposed above, the simulation result of a one-dimensional graded PN junction diode will be presented as an example. This PN junction diode has a size of 2 micrometers, is made of single-crystal silicon material, and operates at a temperature of 300 K. The donor impurities and acceptor impurities inside the device are phosphorus and boron, respectively, and the concentration distribution of both in space satisfies the Gaussian distribution function. The peak concentrations of phosphorus and boron are located at the two electrodes on both ends of the diode. The specific distribution is shown in Figure 4.

We used both Sentaurus TCAD and the MDNN combined model to simulate the same device. In the MDNN combined model used for this simulation, the hidden layers of Model N, Model P, and Model C all have 120 groups, totaling 600 neurons. The Sentaurus TCAD simulation of this device used a grid with 1,032 grid points. The input dataset for the MDNN combined model simulation consists of the coordinates of 800 uniformly distributed sample points inside the diode. We used the AdamW optimizer with a learning rate of 0.0015 to train the MDNN combined model. All code was compiled and run in the PyTorch environment of Pycharm. The changes in the loss function value of the MDNN composite model during the simulation process are illustrated in Figure 5. The comparison of the simulation results output by the MDNN combined model and Sentaurus TCAD under different bias conditions is shown in Figure 6 and Figure 7.

As can be observed from Figure 5, the strategy of alternately training Model N and Model P effectively leads to a steady improvement in the accuracy of the numerical solution output by the composite model. With the introduction of a new model to compensate for errors the issue of the model getting trapped in local optima is resolved, and the total error of the output result is rapidly reduced to an extremely low level. From Figure 6 and Figure 7, it can be clearly seen that the results obtained from our machine learning simulation method are almost identical to the simulation results of the traditional commercial TCAD. The average numerical error between the two is less than 1%. Moreover, the numerical errors of the final output of the MDNN combined model for the Poisson equation (Equation (11)) and the drift-diffusion equation (Equation (12) and Equation (13)) are both less than 1×10^{-5}. These results fully demonstrate the effectiveness of our proposed MDNN model and related algorithms in implementing TCAD simulations.

Finally, it is worth mentioning that all the neural model training processes in this paper were carried out on a workstation with a CPU configuration of an 8-core 12th gen Intel Core i3-12100. Without using GPU acceleration or high-performance servers, the time required for simulating each diode example is around 15-20 minutes, simply comparable or less than traditional TCAD simulations.

4 CONCLUSION

In summary, by replacing the deep neural network (DNN) used in traditional PINN methods with the combined MDNN model, this paper has achieved machine learning-TCAD simulation without the need for any prior experience data for the first time. The results show that the simulation accuracy of this method in the one-dimensional case is comparable to that of mainstream commercial software. We believe that this method has a far greater prospect than the current achievements.

Implementing the simulation of planar devices and quantum-limited effects directly through machine learning is the issue we are currently focusing on. In addition, accelerating neural network training through hardware-software co-design [18] and optimizing multi-model strategies are also key to making machine learning-TCAD truly applicable to industrial practice. Here, we hope that more colleagues will join us in exploring this new direction.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (NSFC) under grants U2341221 and 62274070, Hubei Province Science and Technology Major Project under Grant 2022AEA001, Huazhong University of Science and Technology under grants 2023JCYJ042 and 2019kfyXJJS048, and the Hubei Key Laboratory of Advanced Memories.

REFERENCES

[1] C. Jeong *et al.*, "Bridging TCAD and AI: Its Application to Semiconductor Design," *IEEE Trans Electron Devices*, vol. 68, no. 11, pp. 5364–5371, Nov. 2021, doi: 10.1109/TED.2021.3093844.

[2] S.-C. Han and S.-M. Hong, "Deep Neural Network for Generation of the Initial Electrostatic Potential Profile," in *2019 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, IEEE, Sep. 2019, pp. 1–4. doi: 10.1109/SISPAD.2019.8870521.

[3] H. Dhillon, K. Mehta, M. Xiao, B. Wang, Y. Zhang, and H. Y. Wong, "TCAD-Augmented Machine Learning With and Without Domain Expertise," *IEEE Trans Electron Devices*, vol. 68, no. 11, pp. 5498–5503, Nov. 2021, doi: 10.1109/TED.2021.3073378.

[4] P. Aleksandrov, A. Rezaei, T. Dutta, N. Xeni, A. Asenov, and V. Georgiev, "Convolutional Machine Learning Method for Accelerating Nonequilibrium Green's Function Simulations in Nanosheet Transistor," *IEEE Trans Electron Devices*, vol. 70, no. 10, pp. 5448–5453, Oct. 2023, doi: 10.1109/TED.2023.3306319.

[5] W. Jang, S. Myung, J. M. Choe, Y.-G. Kim, and D. S. Kim, "TCAD Device Simulation With Graph Neural Network," *IEEE Electron Device Letters*, vol. 44, no. 8, pp. 1368–1371, Aug. 2023, doi: 10.1109/LED.2023.3290930.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," vol. 1711, no. 10561, Nov. 2017.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations," Nov. 2017.

[8] A. Khan and D. A. Lowther, "Physics Informed Neural Networks for Electromagnetic Analysis," *IEEE Trans Magn*, vol. 58, no. 9, pp. 1–4, Sep. 2022, doi: 10.1109/TMAG.2022.3161814.

[9] X.-K. Wen, G.-Z. Wu, W. Liu, and C.-Q. Dai, "Dynamics of diverse data-driven solitons for the three-component coupled nonlinear Schrödinger model by the MPS-PINN method," *Nonlinear Dyn*, vol. 109, no. 4, pp. 3041–3050, Sep. 2022, doi: 10.1007/s11071-022-07583-4.

[10] S. Karimpouli and P. Tahmasebi, "Physics informed machine learning: Seismic wave equation," *Geoscience Frontiers*, vol. 11, no. 6, pp. 1993–2001, Nov. 2020, doi: 10.1016/j.gsf.2020.07.007.

[11] H. Hu, C. M. Kennedy, P. G. Kevrekidis, and H.-K. Zhang, "A Modified PINN Approach for Identifiable Compartmental Models in Epidemiology with Application to COVID-19," *Viruses*, vol. 14, no. 11, p. 2464, Nov. 2022, doi: 10.3390/v14112464.

[12] C. Wang, S. Li, D. He, and L. Wang, "Is L2 Physics-Informed Loss Always Suitable for Training Physics-Informed Neural Network?," Jun. 2022.

[13] K. Tang, X. Wan, and C. Yang, "DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations," *J Comput Phys*, vol. 476, p. 111868, Mar. 2023, doi: 10.1016/j.jcp.2022.111868.

[14] W. E and B. Yu, "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems," *Commun Math Stat*, vol. 6, no. 1, pp. 1–12, Mar. 2018, doi: 10.1007/s40304-018-0127-z.

[15] Z. Uddin, S. Ganga, R. Asthana, and W. Ibrahim, "Wavelets based physics informed neural networks to solve non-linear differential equations," *Sci Rep*, vol. 13, no. 1, p. 2882, Feb. 2023, doi: 10.1038/s41598-023-29806-3.

[16] H. K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations," *IEEE Trans Electron Devices*, vol. 11, no. 10, pp. 455–465, Oct. 1964, doi: 10.1109/T-ED.1964.15364.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[18] D. Song et al., "Mitigate IR-Drop Effect by Modulating Neuron Activation Functions for Implementing Neural Networks on Memristor Crossbar Arrays," *IEEE Electron Device Letters*, vol. 44, no. 8, pp. 1280–1283, Aug. 2023, doi: 10.1109/LED.2023.3285916.