applied sciences

MDPI

Article

Research on Device Modeling Technique Based on MLP Neural Network for Model Parameter Extraction

Haixia Kang [1,2], Yuping Wu [1,2,3,4,*], Lan Chen [1,3] and Xuelian Zhang [1,3]

[1] Institute of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China; kanghaixia@ime.ac.cn (H.K.); chenlan@ime.ac.cn (L.C.); zhangxuelian@ime.ac.cn (X.Z.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Beijing Key Laboratory of Three-Dimensional and Nanometer Integrated Circuit Design Automation Technology, Beijing 100029, China
[4] Key Laboratory of Microelectronic Devices & Integrated Technology, CAS, Beijing 100029, China
* Correspondence: wuyuping@ime.ac.cn; Tel.: +86-010-8299-5693

Abstract: The parameter extraction of device models is critically important for circuit simulation. The device models in the existing parameter extraction software are physics-based analytical models, or embedded Simulation program with integrated circuit emphasis (SPICE) functions. The programming implementation of physics-based analytical models is tedious and error prone, while it is time consuming to run the device model evaluation for the device model parameter extraction software by calling the SPICE. We propose a novel modeling technique based on a neural network (NN) for the optimal extraction of device model parameters in this paper, and further integrate the NN model into device model parameter extraction software. The technique does not require developers to understand the device model, which enables faster and less error-prone parameter extraction software developing. Furthermore, the NN model improves the extraction speed compared with the embedded SPICE, which expedites the process of parameter extraction. The technique has been verified on the BSIM-SOI model with a multilayer perceptron (MLP) neural network. The training error of the NN model is 4.14%, and the testing error is 5.38%. Experimental results show that the trained NN model obtains an extraction error of less than 6%, and its extraction speed is thousands of times faster than SPICE in device model parameter extraction.

Keywords: device model parameter extraction; device modeling; neural network; multilayer perceptron

Citation: Kang, H.; Wu, Y.; Chen, L.; Zhang, X. Research on Device Modeling Technique Based on MLP Neural Network for Model Parameter Extraction. *Appl. Sci.* **2022**, *12*, 1357. https://doi.org/10.3390/app12031357

Academic Editor: Stavros Souravlas

Received: 31 October 2021
Accepted: 24 January 2022
Published: 27 January 2022

1. Introduction

In the process of device model parameter extraction, the model parameter values are substituted into the device model for calculation. The error between the measured data of the device and the calculated data of the model is obtained and compared with the target value. The optimization engine will adjust the model parameter values and then substitute them into the model calculation. This process is carried out iteratively until the fitting error is lower than the target value, as shown in Figure 1. The device models in the existing parameter extraction software are manually implemented, physics-based analytical models, or embedded SPICE. The implementation of physics-based analytical models requires the developer to be skilled in device physics and have a deep understanding of the models. Moreover, the programming implementation of the physics-based analytical models is tedious and error prone and, therefore, the software development is time intensive. The development of physics-based analytical models is not generic, and needs programming for each device model individually. When the SPICE is called for simulation, it is necessary to parse the netlist, establish the matrix equations, and iteratively solve the matrix equations, etc. It is time consuming to run the device model evaluation for the device model parameter

extraction software by calling the SPICE, making the model parameter extraction less efficient.

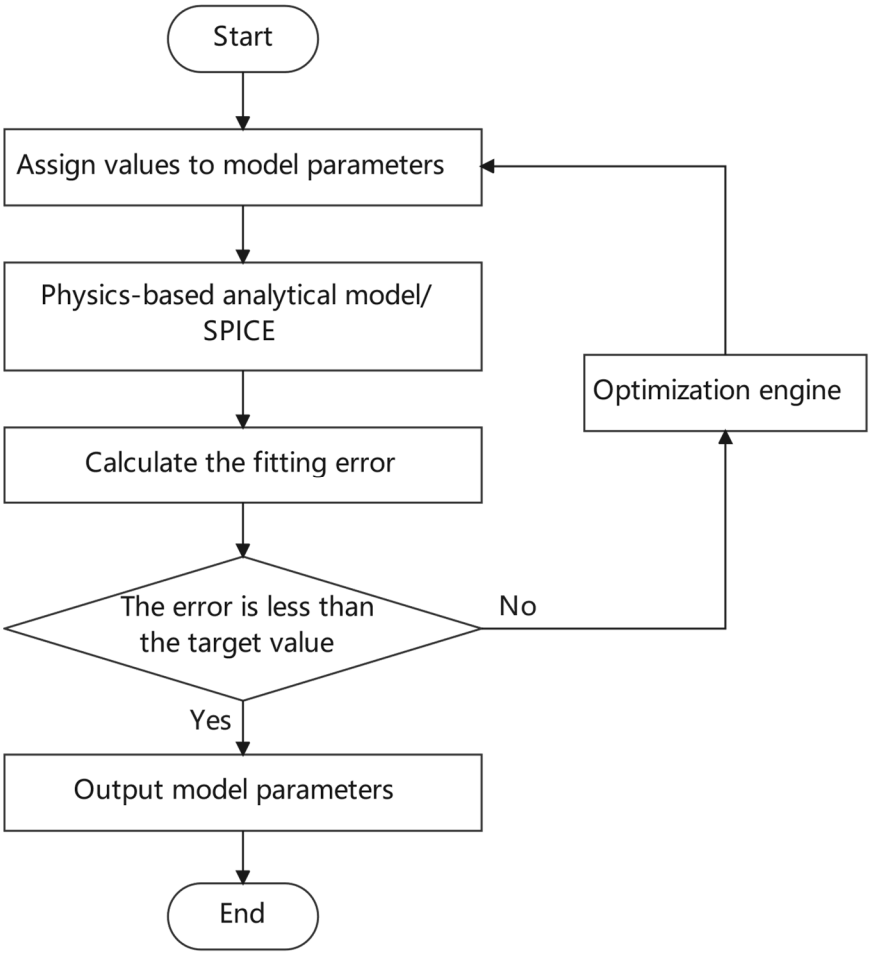

Figure 1. The process of model parameter extraction.

In recent years, NNs have been widely used due to their superior data processing capabilities. NNs provide smooth functions and infinite order derivatives that can approximate complex nonlinear functions [1,2]. Since the IV curves and other characteristics of devices are nonlinear, NNs have the potential to model them [3,4]. Currently, NNs have been applied in several modeling situations, such as small-signal modeling and microwave device modeling [5–8]. As the complexity of the device model increases, the number of model parameters goes up. Manually specifying new model parameters for complex device physics equations is much harder than adding new variables to the NN model [9].

The artificial neural network (ANN) is applied to model the device, with a preprocessing of the current and voltage to train the ANN model accurately [10]. A device model based on the NN acts as an intermediate between early device measurements and a later occurring compact model [11]. A fitness function based on key electrical characteristics (the off current and the subthreshold slope) is proposed to compensate for the R2 functions acting solely as the fitness function. A novel physics-inspired neural network (Pi-NN) approach for compact modeling is proposed in [12], where fundamental device physics are incorporated in the NN model to improve the modeling accuracy. In this paper, we propose using an MLP neural network instead of a manually implemented physics-based analytical model or embedded SPICE to build a device model and integrate it into the parameter extraction software. The NN uses the IV and CV curves, and other characteristics for modeling, and learns the complex relationship between the input and output by training. The NN modeling technique does not require understanding the device model,

which enables faster and less error-prone development than the implementation of the physics-based analytical model. Moreover, the NN model improves the extraction speed over SPICE, which can expedite the process of parameter extraction.

The NNs in these works [10–12] are only used for circuit simulation, and the inputs of the NNs only contain the voltages and the device geometries. We use the MLP neural network to build a device model for parameter extraction. The MLP neural network inputs include the voltages, device geometries, temperature, and model parameters. We perform a sensitivity analysis of the model parameters, where sensitivity means the influence degree of the model parameters on the characteristic curves. The model parameters with non-zero sensitivity are used as the inputs for the NN. The device modeling flow based on NN and the flow of device model parameter extraction using the NN model is shown in Figure 2.

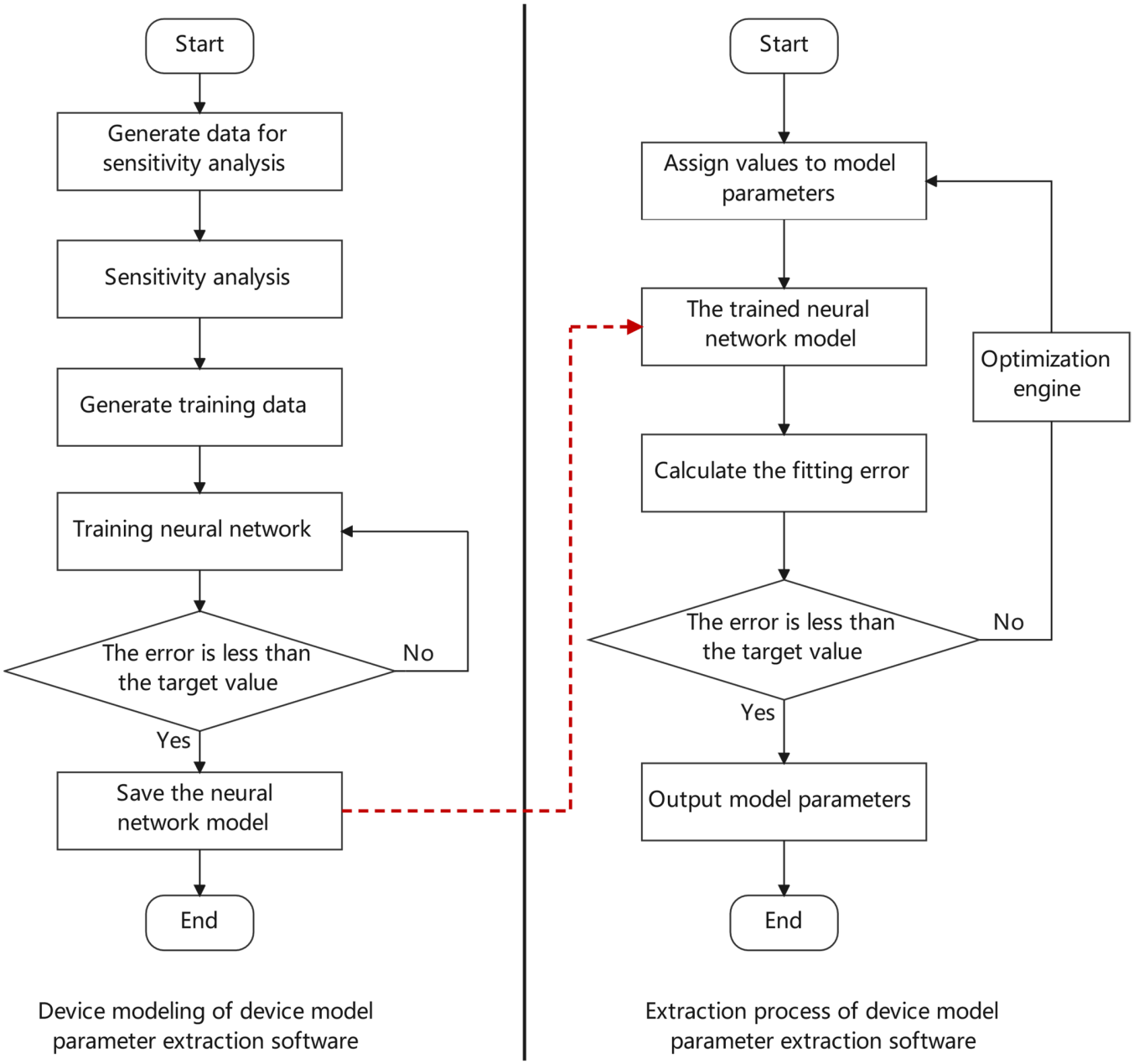

Figure 2. The device modeling flow based on NN and the flow of device model parameter extraction using the NN model.

The rest of this paper is organized as follows. Section 2 introduces the data preparation and preprocessing methods. Section 3 presents the training and testing work of the NN. In Section 4, we analyze the device modeling and model parameter extraction results. Section 5 summarizes the work.

2. Data Preparation and Pre-Processing

2.1. Data Preparation

This work uses current data for device modeling, defining the influence of the model parameters on the current as the sensitivity. Model parameters with non-zero sensitivity are taken as the inputs of the NN. The sensitivity calculation formulas are defined as follows:

$$A0[i] = |\frac{\Delta I_i}{I_i} / \frac{\Delta P}{P}| \quad (1)$$

$$S(A) = max(A0) \quad (2)$$

where I_i is current when the model parameter A is set to P, and ΔI_i denotes the current difference when the model parameter A is set to P and $P + \Delta P$ under the same working conditions. Array $A0$ represents the sensitivity array containing all current data. $S(A)$ represents the sensitivity of the model parameter A, which is the maximum value of array $A0$.

In this work, we take the BSIM-SOI model as the source for the training set and testing set [13]. The sensitivity analysis results of the BSIM-SOI model show 128 model parameters with non-zero sensitivity, which are used as the NN inputs. Some of the model parameters with high sensitivity are listed in Table 1. It took 23.9 h to generate the data needed for the sensitivity analysis and 36 s for the sensitivity analysis itself.

Table 1. Some model parameters of BSIM-SOI.

Model Parameters	Description
Bigc	Parameter for I_{gcs} and I_{gcd}
Nrecr0	Recombination non-ideality factor at reversed bias for source
Beta2	Third V_{ds} dependent parameter of impact ionization current
Vtun0	Voltage dependent parameter for tunneling current for source
Ntun	Reverse tunneling non-ideality factor for source

The extraction of device model parameters needs to take into account various working conditions (voltages, temperature) [14,15] and use a MOSFET of different sizes [16,17]. Therefore, the inputs of the NN need to contain six variables, namely, drain-source voltage (V_{ds}), gate-source voltage (V_{gs}), bulk-source voltage (V_{bs}), temperature (T), gate width (W), and gate length (L) in addition to the model parameters. Drain-source current (I_{ds}) is the output. The MLP neural network consists of an input layer, a few hidden layers, and an output layer, which are fully connected [18,19].

The 45 nm SOI process [20] was used to build the circuit. The minimum gate length value of the nmosbc-type MOSFET used in this work is 56 nm, while the minimum value of the gate width is 654 nm. For the training set, the gate length ranges from 56 nm to 1000 nm, the gate width from 654 nm to 1200 nm, and the temperature from −40 °C to 125 °C. The output characteristic (I_{ds}-V_{ds}) curves sweep from 0 to 1.0 V for the V_{ds} with a step of 0.05 V. The transfer characteristic (I_{ds}-V_{gs}) curves sweep from 0 to 1.0 V for the V_{gs} with a step of 0.05 V. The temperature, gate length, and gate width are scattered within the ranges. At the same time, the Latin hypercube sampling method was applied to the model parameters to ensure that their ranges were fully covered [21]. Tens of thousands of sets of IV curves were prepared as the training set. The time taken to generate the training set for the NN was 11.2 h.

A good NN model must have good generalization ability, i.e., it is required that the NN model performs well on the test data, which is different from the training data [22,23].

The inputs were extended to varying degrees when preparing the test set. W was taken from 654 nm to 1500 nm, L from 56 nm to 1200 nm, and T from −40 °C to 125 °C. The range of model parameters was extended to 1 to 1.1 times the original range. The test set takes a minor step for voltage. The output characteristic curves for V_{ds} sweep from 0 V to 1.0 V with a step of 0.025 V. The transfer characteristic curves for V_{gs} sweep from 0 to 1.0 V with a step of 0.025 V. More than 4000 sets of IV curves were prepared as the test set.

2.2. Data Pre-Processing

Input feature vectors have different magnitudes, among which large feature vectors can dominate the training, causing the network to fail to learn from other feature vectors. Thus, the inputs need to be normalized. The normalization of the inputs can facilitate gradient descent and accelerate convergence [24,25]. In this work, min–max normalization is applied to the input feature vectors to map them between 0 and 1.

The values of currents mainly range from 1×10^{-12} A to 1×10^{-3} A with a large span of magnitude. If the NN is trained using unprocessed current, the error is dominated by the large current range, and the errors in the small current range are not weighted enough, which will lead to a poor fit for small values. We applied a logarithmic operation to the training data to solve this problem, and errors in all regions are weighted equally. However, when V_{ds} is 0, I_{ds} is zero current, which cannot take logarithms, so the points with V_{ds} of 0 were removed from the training set and test set. At the same time, the voltage settings in the data preparation were adjusted, and the output characteristic curves were swept from 0.05 V to 1.0 V for V_{ds}. In the following work, the I_{ds} at V_{ds} of 0 was specified as 0 for the convenience of discussion.

3. Training and Testing the MLP Neural Network

This MLP neural network leverages the backpropagation (BP) algorithm for training. The training process is divided into forward propagation and backpropagation. Forward propagation is a series of linear and activation operations using the hyperparameters with the inputs, and the output results are calculated backward from the input layer. Backpropagation uses the output layer error to optimize the hyperparameters [26,27].

The NN employs the adaptive momentum estimation (Adam) algorithm to accelerate the convergence [28], and the initial learning rate of the Adam algorithm is set to 0.01. The mean square root of the relative error (RR) is defined as the performance metrics, which is defined as follows:

$$RR = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\frac{I_{nn}^{i} - I_{sim}^{i}}{I_{sim}^{i}}\right|^{2}} \quad (3)$$

where I_{sim}^{i} denotes the current value of SPICE simulation and I_{nn}^{i} represents the predicted value of the NN.

The IV curves can be divided into subthreshold and non-subthreshold regions according to the relationship between V_{gs} and the threshold voltage (V_{th}). The relationship between the current and voltage in the subthreshold region is exponential, and the equation is provided as follows:

$$I_{ds} = \mu_0 \times \frac{W_{eff}}{L_{eff}} \times \sqrt{\frac{q\varepsilon_{si}N_{ch}}{2\phi_s}} \times v_t^2 \times \left(1 - \exp\left(-\frac{V_{ds}}{v_t}\right)\right) \times \exp\left(\frac{V_{gs} - V_{th} - V_{off}}{nv_t}\right) \quad (4)$$

where μ_0 represents the mobility, W_{eff} is the effective channel width, L_{eff} is the effective channel length, q is the electronic charge, ε_{si} is the dielectric constant of silicon, N_{ch} is the substrate doping concentration, ϕ_s is the surface potential, v_t is the thermal voltage, V_{off} is the offset voltage, and n is the subthreshold swing parameter. The relationship between the current and voltage in the non-subthreshold region is linear, and the equation is provided as follows:

$$I_{ds} = W_{eff} v_{sat} C_{ox} \times (V_{gs} - V_{th} - A_{bulk} \times V_{dsat}) \times \left(1 + \frac{V_{ds} - V_{dsat}}{V_A}\right) \times (1 + \frac{V_{ds} - V_{dsat}}{V_{ASCBE}}) \quad (5)$$

where v_{sat} is the saturation velocity, C_{ox} is the gate oxide capacitance, A_{bulk} the body charge effect parameter, V_{dsat} the drain-source saturation voltage, V_A the Early voltage, and V_{ASCBE} is the substrate current-induced body effect (SCBE) corresponding to the Early voltage [29]. Among the common NN activation functions, sigmoid types are used in preference, as they have smooth derivatives to meet the requirements for high-order derivatives in circuit simulations.

Among the sigmoid-type functions, the commonly used ones are log-sigmoid and tanh-sigmoid functions, which can map $(-\infty, +\infty)$ inputs to the steps (0, 1) and $(-1, +1)$, respectively. The equations of log-sigmoid and tanh-sigmoid functions are shown in Equations (6) and (7), respectively.

$$log - sigmoid = \frac{1}{1 + e^{-x}} \quad (6)$$

$$tanh - sigmoid = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (7)$$

Two activation functions are used to model the device separately. From Figure 3, we can observe that log-sigmoid fits better than tanh-sigmoid. Therefore, the activation function of this NN eventually uses the log-sigmoid. During the forward propagation of the NN, log-sigmoid controls the data within 0 and 1, and the data change smoothly without data diffusion, so log-sigmoid obtains a higher model accuracy.

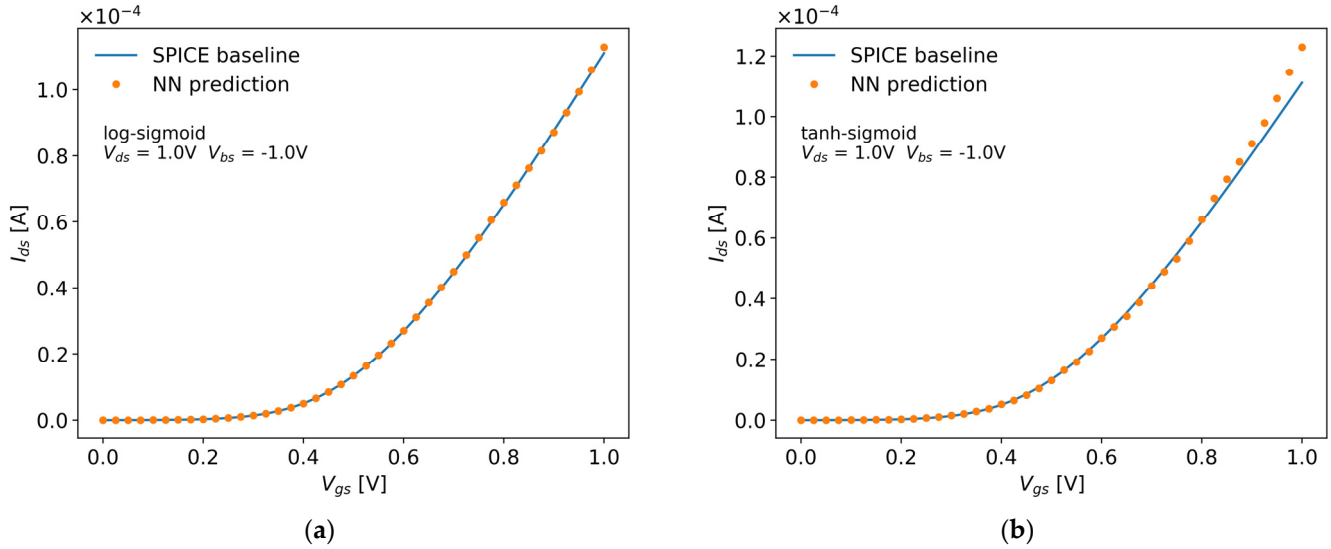

Figure 3. Transfer curves with different activation functions: (**a**) fitting curves of log-sigmoid; (**b**) fitting curves of tanh-sigmoid. Easy to see: In the small voltage range, both curves fit quite well. However, when using the tanh-sigmoid function, there are obvious errors in the higher-value range.

Some criteria were used in this study to better estimate the trained NN models and avoid overfitting. Those criteria include the Akaike Information Criterion (AIC), the corrected Akaike Information Criterion (*AICc*), and the Bayesian Information Criterion (*BIC*).

The *AIC* method attempts to find the model that explains the data best with the smallest number of parameters. The preference model is the model with the lowest AIC

value [30]. Under the assumption that the model errors are normally distributed, the value of *AIC* can be obtained by the following equation:

$$AIC = 2k + nln(RSS/n) \quad (8)$$

where k is the number of parameters for the model to be estimated, n is the sample size, and *RSS* is the residual sum of squares from the estimated model. The equation of *RSS* is as follows:

$$RSS = \sum_{i=1}^{N} \left(I_{nn}^{i} - I_{sim}^{i} \right)^2 \quad (9)$$

AICc is a correction of the *AIC*, which performs better when the sample size is smaller [31]. The equation of *AICc* is as follows:

$$AICc = AIC + \frac{2k(k+1)}{n-k-1} \quad (10)$$

The *BIC* penalizes parameters more heavily than the *AIC*. For any two estimated models, the model with the lower value of *BIC* is preferred [32]. The equation of *BIC* is as follows:

$$BIC = k \times ln(n) + nln(RSS/n) \quad (11)$$

The value of RR is obtained based on the relative error. Since there is a large magnitude difference in the currents of different regions, using the relative error can show the fit of the NN model more intuitively. The value of RR can reflect the fitting quality of the IV curves, so when the values of RR of the different NN models differ widely, we prefer the NN model that generates the smallest RR value. When the RR values of different NN models are close to each other, we select the optimal model by referring to the *AIC*, *AICc*, and *BIC* values.

The *AIC*, *AICc*, and *BIC* values were calculated for each NN model, and the results are shown in Table 2. RE-10% in Table 2 represents the proportion of data with a relative error (RE) greater than 10% in the fitting data of the NN model. NN1, NN2, and NN3 are the optimal models corresponding to the *AIC*, *AICc*, and *BIC*, respectively. NN1, NN4, and NN5 are NN models with the smallest RR values.

Table 2. Estimation results of different NN models.

No.	NN	AIC	AICc	BIC	RR	RE-10%
1	134-160-160-160-1	-3.6964×10^6	-3.5891×10^6	-2.9608×10^6	5.38%	2.02%
2	134-120-120-120-1	-3.6945×10^6	-3.6624×10^6	-3.2394×10^6	6.18%	3.81%
3	134-100-100-100-1	-3.6564×10^6	-3.6401×10^6	-3.3174×10^6	6.53%	5.83%
4	134-150-150-150-1	-3.6832×10^6	-3.6031×10^6	-3.0238×10^6	5.48%	2.67%
5	134-190-190-190-1	-3.5421×10^6	-3.2832×10^6	-2.5539×10^6	5.66%	2.95%

Although NN2 and NN3 achieved optimality under the *AICc* and *BIC* criteria, respectively, their RR values were significantly higher than the NN1 model. The equations for *AIC* and *BIC* can be divided into two items: the first one is the complexity penalty of the model ($2k$ for *AIC*, $k \times ln(n)$ for *BIC*), and the second one is the accuracy penalty of the model ($n \times ln(RSS/n)$). The first item is mainly used to avoid the overfitting of the model, and the second item considers the fitting of the model itself. The first item is a penalty to the second item, and the model with the fewest parameters is selected under the condition of satisfying the model validity and reliability, so the complexity must be considered after satisfying the high accuracy first. The difference in the accuracy penalty between the different NN models refers to the difference in RSS. The RSS value is 3.7910×10^{-5}, 5.2934×10^{-5}, and 7.5426×10^{-5} for the NN1, NN2, and NN3 models, respectively. There is a significant difference in the accuracy of the three NN models, and the accuracy of the NN1 model is

higher than the other two models. Therefore, the NN2 and NN3 models were not selected as the optimal model.

The RR values for the NN1, NN4, and NN5 models were close, so we further evaluated them using the *AIC*, *AICc*, and *BIC* criteria. The *AIC*, *AICc*, and *BIC* values for NN5 are larger than those of NN1 and NN4, while the *AIC*, *AICc*, and *BIC* values for NN1 and NN4 are closer to each other. The RSS value is 3.7910×10^{-5} and 4.4666×10^{-5} for the NN1 and NN4 models, respectively. Therefore, we eventually choose the NN1 model for further work. According to the RE-10% in Table 2, when the NN1 model fits the data, an RE of nearly 98% of the data is within 10%. It took 1.7 h to train the NN1 model, with a training error of 4.14% and a testing error of 5.38%.

4. Results and Discussion

4.1. Error and Performance Analysis

The mean value μ of the RE for the training data is 2.81%, and the standard deviation σ of the RE is 3.04%. The training data with an RE greater than μ + 3σ accounted for 0.89% of all training data. We performed an analysis on the training data with an RE greater than μ + 3σ and counted the distributions of temperature, gate width, and gate length. The statistical results are shown in Figures 4–6, respectively.

According to Figure 4, it can be seen that for the training data with an RE greater than μ + 3σ, the proportions of different bars are not very different. The proportion of the first bar in Figure 5 is lower than that of the other bars, which is because the gate width range of this bar is smaller. The overall distribution of the gate width is uniform when the gate width range is considered. From Figure 6, it can be observed that the proportion of data with a small gate length is very large, and the proportion almost decreases with the increase in the gate length. It indicates that the NN model fits less well to the data with a small gate length. Due to the short-channel effect, more complex device physics mathematics are involved around the minimum gate length [10], so it is reasonable that the error is slightly higher at the smaller gate length.

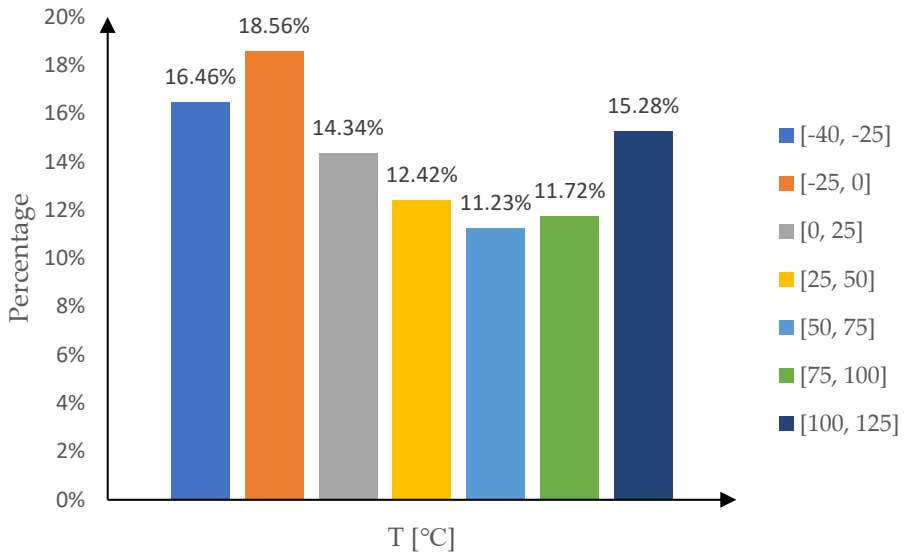

Figure 4. Temperature distribution of the training data with an RE greater than μ + 3σ. The training data with an RE greater than μ + 3σ accounted for 0.89% of all training data.

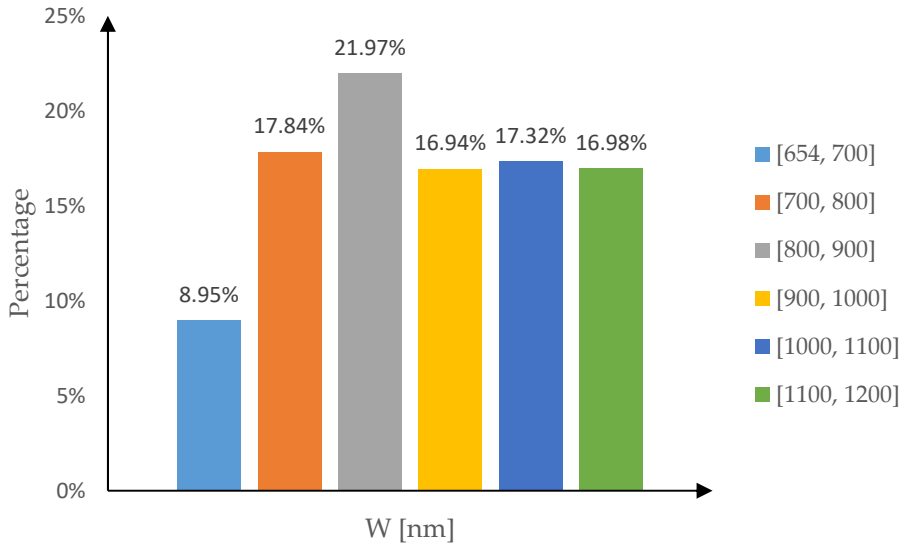

Figure 5. Gate width distribution of the training data with an RE greater than μ + 3σ. The training data with an RE greater than μ + 3σ accounted for 0.89% of all training data.

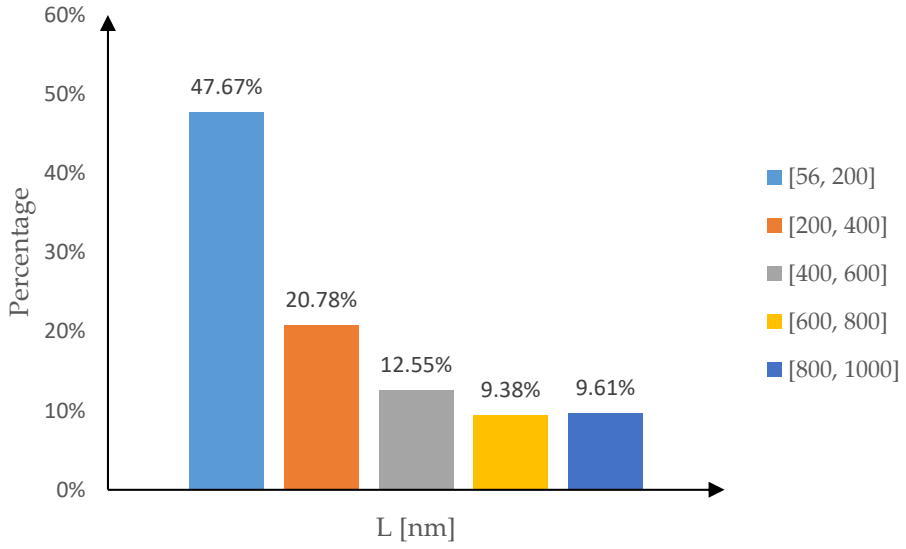

Figure 6. Gate length distribution of the training data with an RE greater than μ + 3σ. The training data with an RE greater than μ + 3σ accounted for 0.89% of all training data.

The mean value μ of RE for the test data is 3.89%, and the standard deviation σ of RE is 3.71%. The test data with an RE greater than μ + 3σ accounted for 1.50% of all test data. The distributions of temperature, gate width, and gate length for the test data with an RE greater than μ + 3σ are shown in Figures 7–9, respectively. According to Figures 7 and 8, it can be seen that for temperature and gate width, the proportions of different bars are not very different. According to Figure 9, it can be seen that the proportion of data with a small gate length is higher. The NN model fits less well to the training data for small gate lengths and, therefore, fits the test data for small gate lengths less well.

A statistical analysis of the IV curves for the training data was performed. It was found that for the I_{ds}–V_{gs} curves, the error of V_{gs} from 0 to 0.2 V was slightly higher than the other regions of the curves. It is basically in the subthreshold region when V_{gs} is between 0 and 0.2 V. According to Equation (4), it is known that the relationship between current and voltage in the subthreshold region is exponential. The exponential relation has infinite order derivatives, and its Taylor series expansion is composed of infinite polynomials. According

to Equation (5), the relationship between current and voltage in the non-subthreshold region is polynomial. When using the same NN model for subthreshold and non-subthreshold regions, the finite polynomial for the non-subthreshold region is easier to fit and, therefore, its modeling accuracy is higher. The errors were counted separately for the subthreshold and non-subthreshold data in the training data using Equation (3), with 4.77% for the subthreshold region and 3.83% for the non-subthreshold region. At the same time, the errors were counted separately for the subthreshold and non-subthreshold data in the test data using Equation (3), with 6.89% for the subthreshold region and 4.79% for the non-subthreshold region.

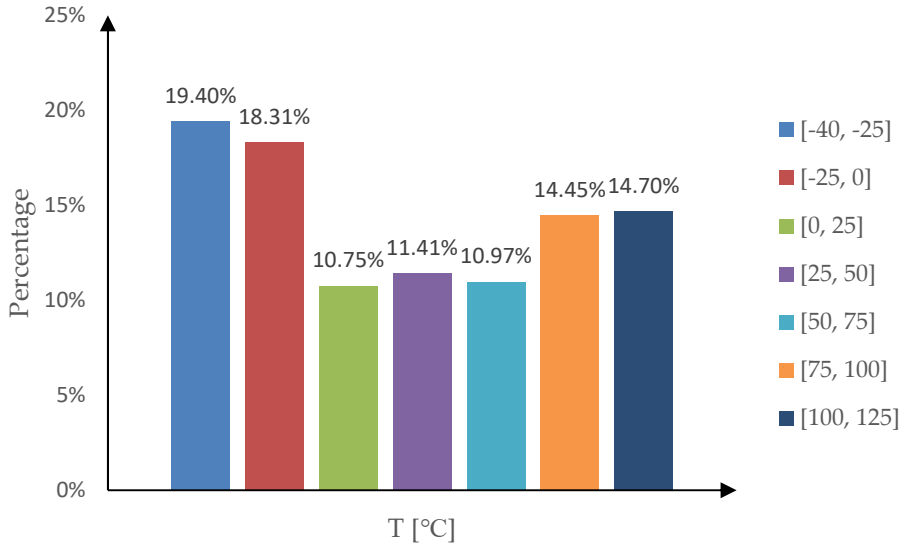

Figure 7. Temperature distribution of the test data with an RE greater than μ + 3σ. The test data with an RE greater than μ + 3σ accounted for 1.50% of all test data.

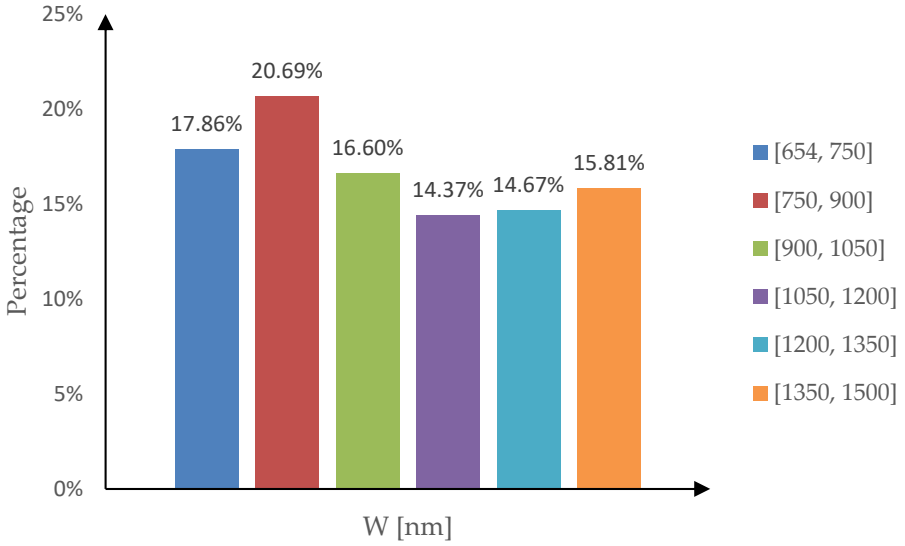

Figure 8. Gate width distribution of the test data with an RE greater than μ + 3σ. The test data with an RE greater than μ + 3σ accounted for 1.50% of all test data.

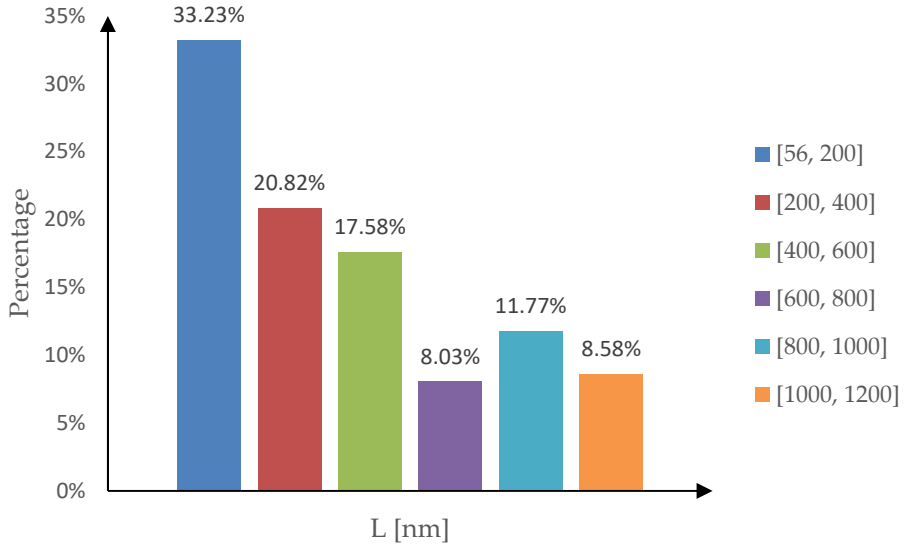

Figure 9. Gate length distribution of the test data with an RE greater than μ + 3σ. The test data with an RE greater than μ + 3σ accounted for 1.50% of all test data.

Multiple sets of IV curve data from a device need to be collected to extract different model parameters when extracting [27]. Thus, the NN model needs to fit multiple sets of IV curves from the device well. Ten randomly selected sets of inputs are shown in Table 3, and the IV curve fit for the first three sets of inputs are shown in Figures 10–12, where the curves indicate the IV curves from the SPICE simulation, and the dots indicate the predicted value of the NN. The fitting errors of the ten sets of inputs are shown in Table 4, where RR-all represents the error of the whole region. RR-sub and RR-nonsub represent the error in the subthreshold and non-subthreshold regions, respectively. The results show that the overall fit is good, demonstrating that the NN model achieves accurate modeling of the IV curves for the BSIM-SOI model.

Table 3. Ten sets of inputs.

No.	T (°C)	W (nm)	L (nm)	Bigc ($(Fs^2/g)^{0.5}m^{-1}V^{-1}$)	Nrecr0 (-)	Beta2 (V)	Vtun0 (V)	Ntun (-)
1	43	804	705	0.004556	3.035	0.06692	0.5217	1.179
2	40	742	364	0.003386	3.300	0.08025	0.4921	1.046
3	65	936	1055	0.002766	2.795	0.07883	0.5806	1.346
4	−26	1178	254	0.006996	2.529	0.04238	0.3751	1.016
5	31	758	463	0.007036	2.991	0.03853	1.1110	1.410
6	28	1237	182	0.009446	2.240	0.03931	0.5623	0.997
7	−5	1211	405	0.006202	2.665	0.07713	0.8417	1.049
8	101	762	363	0.003523	2.175	0.06655	1.1560	1.284
9	61	1412	834	0.001907	3.303	0.06904	1.0025	1.497
10	6	902	111	0.008876	3.252	0.02890	1.1886	1.484

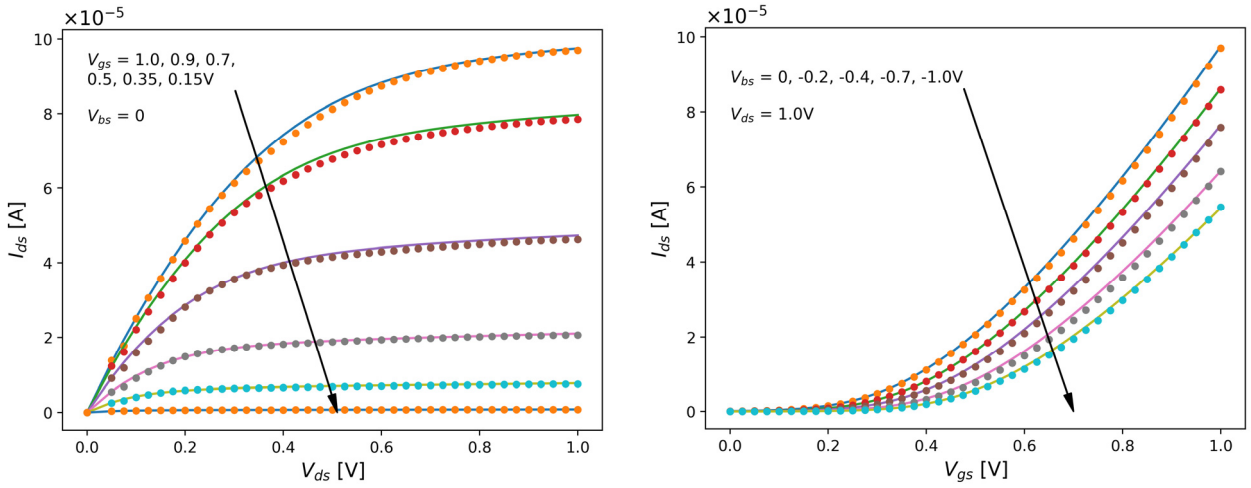

Figure 10. IV fitting curves for the first set of inputs: (**a**) I_{ds}–V_{ds} curves at different V_{gs} when V_{bs} = 0; (**b**) I_{ds}–V_{gs} curves at different V_{bs} when V_{ds} = 1.0 V.

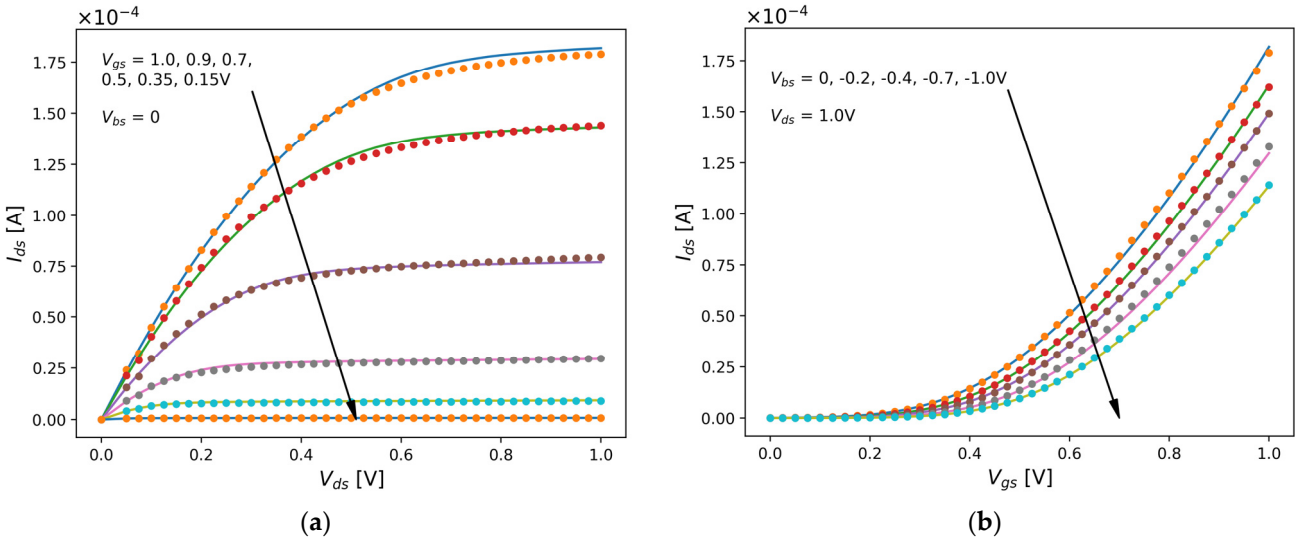

Figure 11. IV fitting curves for the second set of inputs: (**a**) I_{ds}–V_{ds} curves at different V_{gs} when V_{bs} = 0; (**b**) I_{ds}–V_{gs} curves at different V_{bs} when V_{ds} = 1.0 V.

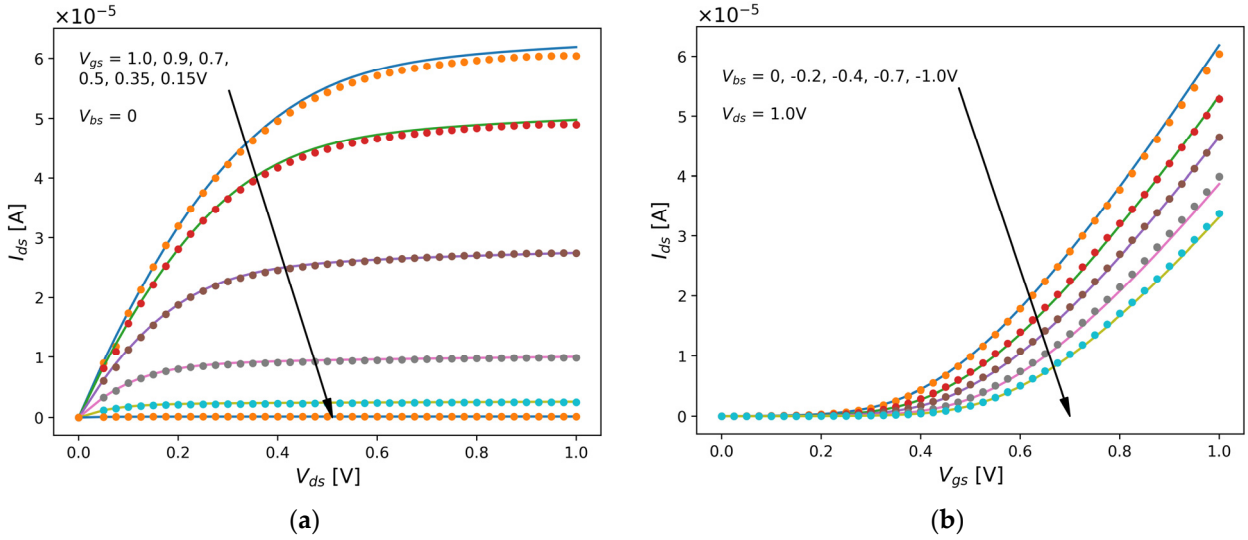

Figure 12. IV fitting curves for the third set of inputs: (**a**) I_{ds}–V_{ds} curves at different V_{gs} when V_{bs} = 0; (**b**) I_{ds}–V_{gs} curves at different V_{bs} when V_{ds} = 1.0 V.

Table 4. Errors of ten sets of inputs.

No.	RR-All	RR-Sub	RR-Nonsub
1	4.59%	6.83%	4.01%
2	3.05%	4.02%	2.54%
3	3.13%	3.67%	2.87%
4	4.54%	6.58%	3.39%
5	5.20%	6.06%	4.81%
6	6.68%	7.30%	6.41%
7	2.95%	3.15%	2.91%
8	3.01%	5.08%	2.43%
9	2.98%	3.63%	2.83%
10	5.91%	6.11%	5.87%

The time comparison between the NN model prediction and SPICE simulation is shown in Table 5, comparing the time to generate 5000, 10,000, 15,000, and 20,000 sets of IV curves. The results show that the trained NN model is thousands of times faster than SPICE.

Table 5. Time comparison between NN model prediction and SPICE simulation.

Number of IV Curves	SPICE (s)	NN Model (s)	Multiplier of Speed Increase
5000	19,934.43	20.67	964.41
10,000	39,822.99	30.95	1286.69
15,000	59,729.98	46.12	1295.10
20,000	79,630.48	61.04	1304.56

4.2. Device Model Parameter Extraction

The trained NN model is used for parameter extraction. Measured results include data of various device sizes. The gate length included 56 nm, 100 nm, 200 nm, 300 nm, 400 nm, and 500 nm, and the gate width included 654 nm, 700 nm, 800 nm, 900 nm, and 1000 nm. The devices were subjected to different operating temperatures. We used Equation (3) to calculate the error between the measured result and simulation result using the extracted model parameters. The errors of the five sets of measured data are 4.88%, 5.71%, 5.62%, 5.78%, and 5.18%. Figures 13 and 14 show part of the fitting curves of the first two sets of

measured data, where the curves indicate the measured results, and the dots indicate the simulation results of the extracted model parameters. The results show a good fit.

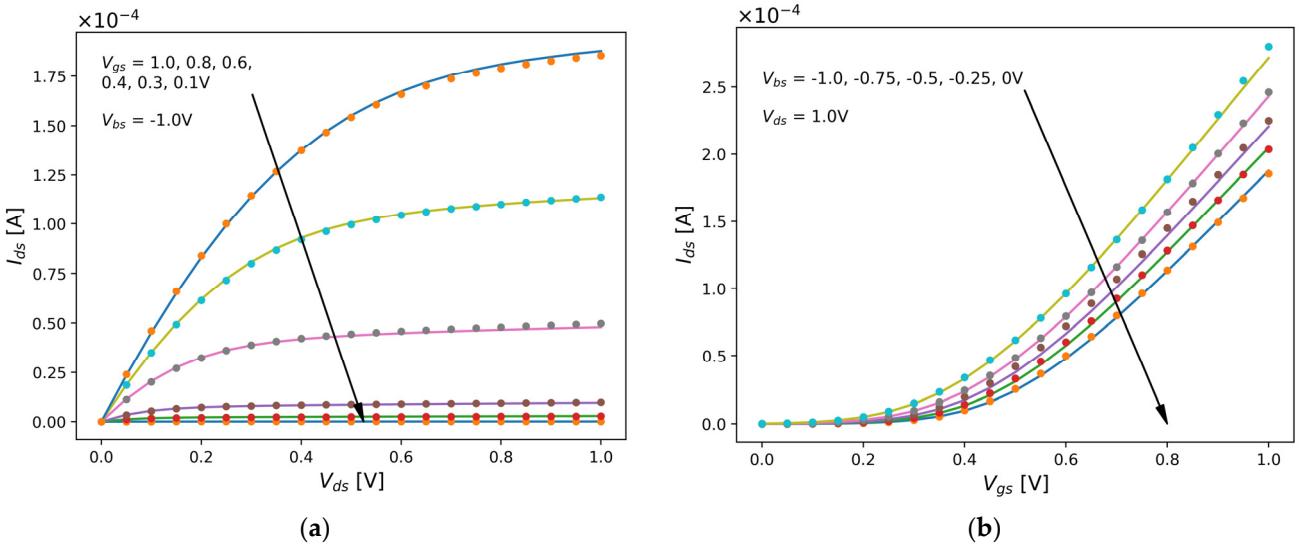

Figure 13. IV fitting curves of the first set of measured data: (**a**) I_{ds}–V_{ds} curves with T = 0 °C, W = 654 nm, L = 200 nm; (**b**) I_{ds}–V_{gs} curves with T = 0 °C, W = 654 nm, L = 200 nm.

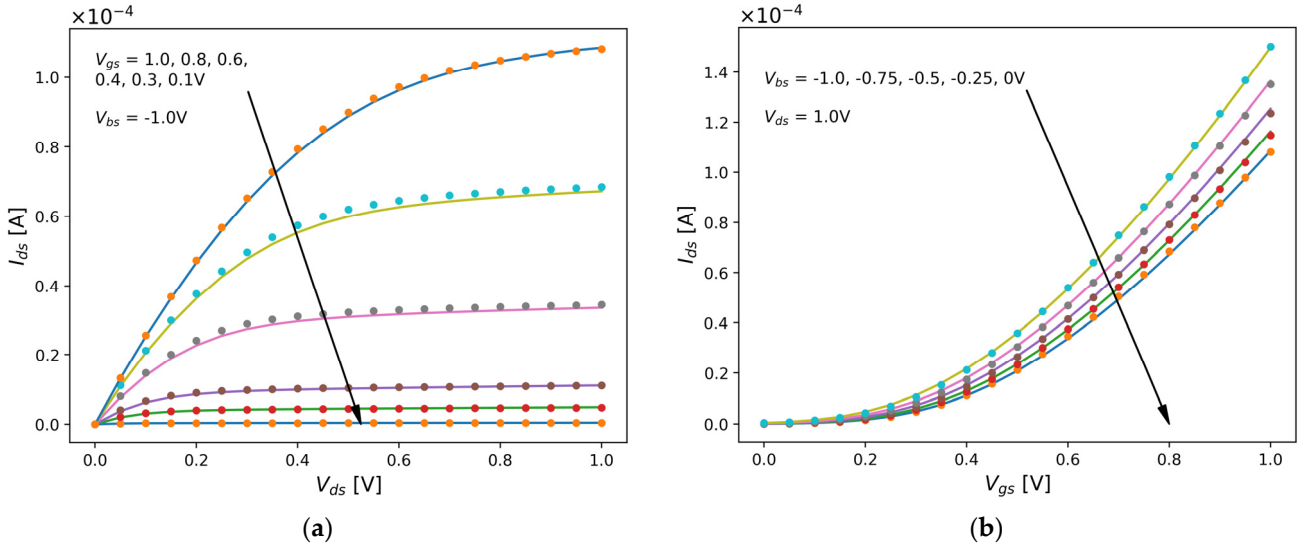

Figure 14. IV fitting curves of the second set of measured data: (**a**) I_{ds}–V_{ds} curves with T = 25 °C, W = 800 nm, L = 500 nm; (**b**) I_{ds}–V_{gs} curves with T = 25 °C, W = 800 nm, L = 500 nm.

We used SPICE and the NN model to extract parameters from the measured data of a single device. The extraction error of the SPICE was 3.27%, and it took 4.4 h. The extraction error of the NN model was 4.28%, and it took 8.6 s. Both the SPICE and the NN model worked on the same CPU processor.

5. Conclusions

In this paper, a novel device modeling technique based on NNs for the optimal extraction of device model parameters is proposed and verified with the BSIM-SOI model. This technique does not require developers to manually implement physics-based analytical models, which vastly accelerates the developing of parameter extraction software. The development of the program related to the physics-based analytical models in the model

parameter extraction software takes about six months, while the development of the NN model takes less than one week.

The MLP neural network used in this paper obtained 4.14% in overall training error, with 3.83% in the non-subthreshold and 4.77% in the subthreshold regions. The NN model has good generalization ability. The overall test error is 5.38% of more than four thousand IV curves, with 6.89% in the subthreshold and 4.79% in the non-subthreshold regions. The trained NN model is used for parameter extraction, and the extraction error is less than 6%. Compared with the embedded SPICE, the extraction speed of the NN model is thousands of times faster.

Author Contributions: Conceptualization, H.K., Y.W. and L.C.; methodology, H.K.; software, H.K.; validation, H.K.; investigation, H.K. and Y.W.; data curation, H.K.; writing—original draft preparation, H.K.; writing—review and editing, H.K., Y.W. and X.Z.; supervision, Y.W.; funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Research on Modeling, Analysis and Optimization Technologies for ULP Circuit (E1GW047002).

Data Availability Statement: Not applicable.

Acknowledgments: The first author, H.K., hereby acknowledges the Institute of Microelectronics of Chinese Academy of Sciences (IMECAS) and the EDA Center.

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

References

1. Jain, A.K.; Mao, J.; Mohiuddin, K.M. Artificial neural networks: A tutorial. *Computer* **2015**, *29*, 31–44. [CrossRef]
2. Canziani, A.; Paszke, A.; Culurciello, E. An Analysis of Deep Neural Network Models for Practical Applications. *arXiv* **2016**, arXiv:1605.07678.
3. Root, D.E. Future Device Modeling Trends. *IEEE Microw. Mag.* **2012**, *13*, 45–59. [CrossRef]
4. Zhang, Z.; Wang, R.; Chen, C.; Huang, Q.; Wang, Y.; Hu, C.; Wu, D.; Wang, J.; Huang, R. New-Generation Design-Technology Co-Optimization (DTCO): Machine-Learning Assisted Modeling Framework. In Proceedings of the 2019 Silicon Nanoelectronics Workshop (SNW), Kyoto, Japan, 9–10 June 2019; pp. 1–2.
5. Zhang, A.; Gao, J. InP HBT Small Signal Modeling based on Artificial Neural Network for Millimeter-wave Application. In Proceedings of the 2020 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), Hangzhou, China, 7–9 December 2020; pp. 1–3.
6. Liu, W.; Na, W.; Feng, F.; Zhu, L.; Lin, Q. A Wiener-Type Dynamic Neural Network Approach to the Modeling of Nonlinear Microwave Devices and Its Applications. In Proceedings of the 2020 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), Hangzhou, China, 7–9 December 2020; pp. 1–3.
7. Wang, S.; Roger, M.; Sarrazin, J.; Lelandais-Perrault, C. Hyperparameter Optimization of Two-Hidden-Layer Neural Networks for Power Amplifiers Behavioral Modeling Using Genetic Algorithms. *IEEE Microw. Wirel. Compon. Lett.* **2019**, *29*, 802–805. [CrossRef]
8. Liu, W.; Zhu, L.; Feng, F.; Zhang, W.; Liu, G.J.M. A Time Delay Neural Network Based Technique for Nonlinear Microwave Device Modeling. *Micromachines* **2020**, *11*, 831. [CrossRef] [PubMed]
9. Xu, J.; Root, D.E. Advances in artificial neural network models of active devices. In Proceedings of the IEEE Mtt-s International Conference on Numerical Electromagnetic & Multiphysics Modeling & Optimization, Limoges, France, 6–8 July 2022; pp. 1–3.
10. Zhang, L.; Chan, M. Artificial neural network design for compact modeling of generic transistors. *J. Comput. Electron.* **2017**, *16*, 825–832. [CrossRef]
11. Klemme, F.; Prinz, J.; Santen, V.M.v.; Henkel, J.; Amrouch, H. Modeling Emerging Technologies using Machine Learning: Challenges and Opportunities. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2–5 November 2020; p. 1367.
12. Li, M.; İrsoy, 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]
13. Sheu, B.J.; Scharfetter, D.L.; Ko, P.K.; Jeng, M.C.J.S.-S.C. BSIM: Berkeley short-channel IGFET model for MOS transistors. *IEEE J. Solid-State Circuits* **1987**, *22*, 558–566. [CrossRef]
14. Petrosyants, K.O.; Ismail-zade, M.R.; Sambursky, L.M.; Dvornikov, O.V.; Lvov, B.G.; Kharitonov, I.A. Automation of parameter extraction procedure for Si JFET SPICE model in the −200 . . . +110 °C temperature range. In Proceedings of the 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT), Moscow, Russia, 14–16 March 2018; pp. 1–5.
15. Cortes-Ordonez, H.; Haddad, C.; Mescot, X.; Romanjek, K.; Ghibaudo, G.; Estrada, M.; Cerdeira, A.; Iñiguez, B. Parameter Extraction and Compact Modeling of OTFTs From 150 K to 350 K. *IEEE Trans. Electron Devices* **2020**, *67*, 5685–5692. [CrossRef]

16. Leonhardt, A.; Ferreira, L.F.; Bampi, S. Nanoscale FinFET global parameter extraction for the BSIM-CMG model. In Proceedings of the 2015 IEEE 6th Latin American Symposium on Circuits & Systems (LASCAS), Montevideo, Uruguay, 24–27 February 2015; pp. 1–4.
17. Tanaka, C.; Matsuzawa, K.; Miyata, T.; Adachi, K.; Hokazono, A. Investigation of BSIM4 parameter extraction and characterization for Multi Gate Oxide-Dual Work Function (MGO-DWF)-MOSFET. In Proceedings of the 2016 Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS), Vienna, Austria, 25–27 January 2016; pp. 96–99.
18. Allan, P.J.A.N. Approximation theory of the MLP model in neural networks. *Acta Numer.* **1999**, *8*, 143–195.
19. Karlik, B.; Olgac, A.V. Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. *Int. J. Artif. Intell. Expert Syst.* **2011**, *1*, 111–122.
20. Koh, R.; Yamagami, S.; Lee, J.W.; Waka Ba Yashi, H.; Saito, Y.; Ogura, A.; Narihiro, M.; Arai, K.; Takemura, H.; Mogami, T. Soi Mosfet. U.S. Patent 6933569 B2, 23 August 2005.
21. Iman, R.L.; Davenport, J.M.; Zeigler, D.K. *Latin Hypercube Sampling (Program User's Guide). [LHC, in FORTRAN]*; Sandia Labs.: Albuquerque, NM, USA, 1980.
22. Yang, Y.; Li, C. Quantitative analysis of the generalization ability of deep feedforward neural networks. *J. Intell. Fuzzy Syst.* **2021**, *40*, 4867–4876. [CrossRef]
23. Choudhury, T.A.; Hosseinzadeh, N.; Berndt, C.C. Improving the Generalization Ability of an Artificial Neural Network in Predicting In-Flight Particle Characteristics of an Atmospheric Plasma Spray Process. *J. Therm. Spray Techn.* **2012**, *21*, 935–949. [CrossRef]
24. Shalabi, L.A.; Shaaban, Z. Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix. In Proceedings of the 2006 International Conference on Dependability of Computer Systems, Szklarska Poreba, Poland, 25–27 May 2006; pp. 207–214.
25. Tai, Q.Y.; Shin, K.S. GA-based Normalization Approach in Back-propagation Neural Network for Bankruptcy Prediction Modeling. *J. Intell. Inf. Syst.* **2010**, *16*, 1–14.
26. Hecht-Nielsen, R. Theory of the Backpropagation Neural Network. *Int. 1989 Jt. Conf. Neural Netw.* **1989**, *1*, 593–605.
27. Villa, A.; Duch, W.; Erdi, P.; Masulli, F.; Palm, G. *Artificial Neural Networks and Machine Learning—ICANN 2012*; Springer: Berlin, Germany, 2011.
28. Kingma, D.; Ba, J.J.C.S. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
29. Cheng, Y. *BSIM3v3 Manual*; Department of Electrical Engineering & Computer Science, University of California: Berkeley, CA, USA, 1996.
30. Akaike, H. Information theory as an extension of the maximum likelihood principle. In Proceedings of the Second International Symposium on Information Theory, Budapest, Hungary, 2–8 September 1971; pp. 267–281.
31. Sugiura, N. Further analysts of the data by akaike' s information criterion and the finite corrections. *Commun. Stat. Theory Methods* **1978**, *7*, 13–26. [CrossRef]
32. Xue, Y.; Wang, W.; Zhu, H.; Zheng, G.; Cui, Z. An Inversion Method for Shallow Water Geoacoustic Parameters Based on BIC Criterion. In Proceedings of the 2021 OES China Ocean Acoustics (COA), Harbin, China, 14–17 July 2021; pp. 192–196.