Exploring Deep Learning For Semiconductor Device BSIM4 Modeling

Leo Chenwei Shao
Valley Christian School
San Jose, California
Leo.chenwei.shao@gmail.com

***Abstract*—A deep learning neural network is built and trained to extract semiconductor transistor BSIM4 model parameters for circuit simulation. The current-voltage (I-V) data with corresponding BSIM4 model parameters are generated from industry standard simulation tool LTSpice from Analog Device. I-V curves are feed to multiple layer neural networks as input. The model parameters used to generate I-Vs are used as the output. The neural network models are implemented with Pytorch. It shows that a neural network with 5 hidden layers can be trained within 1 hour with RTX 5060TI and gives excellent result.**

Keywords—Deep learning, Pytroch, Semiconductor transistor model, BSIM4, LTSpice

I. Introduction

Accurate semiconductor transistor models are fundamental to integrated circuit design, as they provide reliable current–voltage (I–V) relationships required for circuit simulation and verification. Modern integrated circuits often contain a large number of transistors interconnected to realize complex and highly specific functions. Consequently, the accuracy of transistor models directly impacts the predictability and performance of circuit designs.

For CMOS transistors fabricated using technology nodes ranging from 0.13 µm to 20 nm, the Berkeley Short-Channel IGFET Model 4 (BSIM4) [1] has been widely adopted as the industry-standard compact model. A CMOS transistor described by BSIM4 consists of four external terminals: drain, gate, source, and substrate. Given the bias voltages applied to the gate, drain, and substrate with respect to the source—namely V_{GS}, V_{DS}, and V_{BS}—the BSIM4 model computes the corresponding terminal currents at all four nodes.

The BSIM4 model is highly comprehensive, incorporating more than 200 parameters to capture a wide range of physical effects and device behaviors. This extensive parameter enables accurate modeling across varying transistor geometries, technology nodes, and fabrication processes. Fig. 1 illustrates a simple circuit schematic and Fig. 2 shows a portion of an example BSIM4 model. Each BSIM4 model is stored in a text file using a parameter assignment format of the form parameter = value. To construct a model for a specific transistor, device modeling engineers employ commercial software tools such as MBP and BSIMProPlus, which extract model parameters by fitting simulated I–V characteristics to measured device data.

This parameter extraction and curve-fitting process is often time-consuming and labor-intensive due to the complexity of the BSIM4 model and the strong interdependence among its parameters. Engineers must repeatedly adjust parameter values

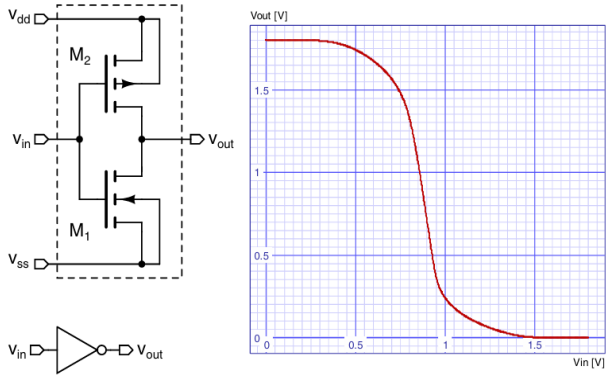

Figure 1. One simple circuit with two transistors, M1 and M2. In order to get accuarte relation between inout potental Vin and output potential Vout, circuit dimulators need accurate models like BSIM4 models for M1 and M2 to calculate currents from nodes with applied bias volatges.

```
* Short channel models from CMOS Circuit Design, Layout, and Simulation,
* 50nm BSIM4 models VDD=1V, see CMOSedu.com
*
.model  N_50n  nmos  level = 54
+binunit = 1          paramchk= 1          mobmod  = 0
+capmod  = 2          igcmod  = 1          igbmod  = 1          geomod  = 0
+diomod  = 1          rdsmod  = 0          rbodymod= 1          rgatemod= 1
+permod  = 1          acnqsmod= 0          trnqsmod= 0
+tnom    = 27         toxe    = 1.4e-009   toxp    = 7e-010     toxm    = 1.4e-009
+epsrox  = 3.9        wint    = 5e-009     lint    = 1.2e-008
+ll      = 0          wl      = 0          lln     = 1          wln     = 1
+lw      = 0          ww      = 0          lwn     = 1          wwn     = 1
+lwl     = 0          wwl     = 0          xpart   = 0          toxref  = 1.4e-009
+vth0    = 0.22       k1      = 0.35       k2      = 0.05       k3      = 0
+k3b     = 0          w0      = 2.5e-006   dvt0    = 2.8        dvt1    = 0.52
+dvt2    = -0.032     dvt0w   = 0          dvt1w   = 0          dvt2w   = 0
```

Figure 2. One BSIM model is coded in a text file. Here shows the beginning part of a BSIM4 model [2] which has mor than 200 parameters. Circuit simulator will parse the file and use the paramter values to calculate the current at each nodes.

and regenerate I–V curves to achieve acceptable agreement with measured data, making the process tedious and not productive.

In recent years, deep learning–based neural networks have demonstrated significant potential in modeling complex nonlinear relationships between inputs and outputs. Their flexibility, scalability, and ability to leverage advanced optimization algorithms make them well suited for automated curve fitting tasks. Previous work by the Berkeley research

979-8-3315-7111-5/26/$31.00 ©2026 IEEE

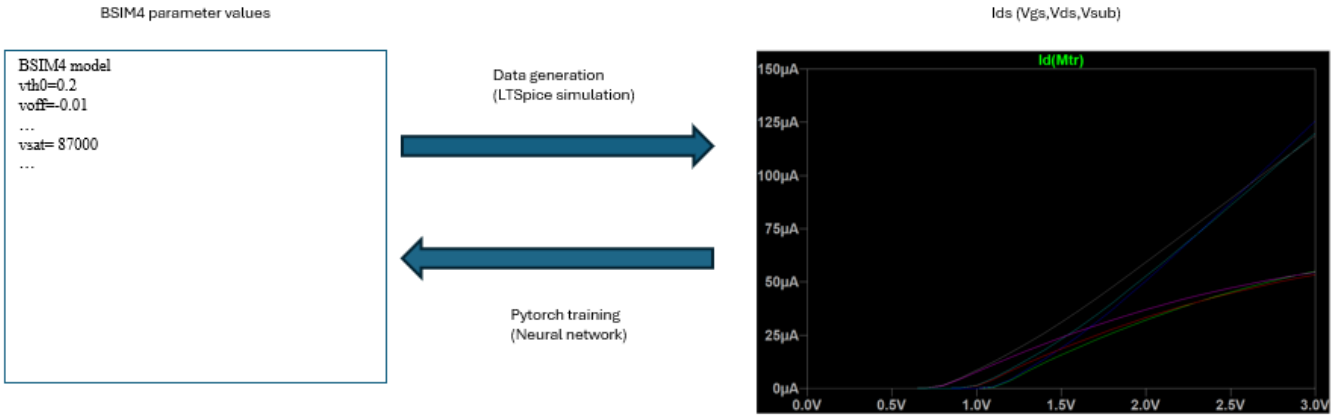

Figure 3. Overview of the neural network approach for BSIM4 model parameter extraction. To generate data sample for Pytorch neural network training, one randomly generated BSIM4 model will used by LTSpice [5] to generate one sets of I-V data. For Pytorch neural network , the input data is I-V data from LTSpice and the output target are the corresponding BSIM4 parameter values.

group has successfully applied deep learning techniques to advanced transistor models such as the BSI-CMG model [3], indicating the viability of this approach for compact modeling applications.

In this paper, a deep learning–based modeling framework is developed using PyTorch [4] to approximate the behavior of the BSIM4 model. As BSIM4 is applicable to semiconductor technology nodes ranging approximately from 0.13 µm to 22 nm, it covers a broad class of transistors used in analog, RF, and mixed-signal applications, including power management circuits and CMOS image sensors. Many of these devices are highly customized, and in certain cases, semiconductor foundries such as TSMC do not provide complete BSIM4 models for such transistors. Under these circumstances, a data-driven modeling approach based on deep learning offers a practical and efficient alternative.

The primary objectives of this work are to evaluate the effectiveness of deep learning networks in fitting BSIM4 I–V characteristics, to determine the network size required to achieve satisfactory modeling accuracy, and to assess the computational resources necessary to complete training within a reasonable time frame. Through this investigation, the feasibility and potential advantages of deep learning–based compact transistor modeling for practical circuit design applications are systematically examined.

II. Modeling Detail

The objective of the deep learning models is to take the current–voltage data of a specific transistor device as input and directly predict (or extract) values of a set of BSIM4 model parameters capable of reproducing the same I–V at input. Although the BSIM4 model contains approximately 200 parameters, this study does not attempt to predict the complete parameter set. Instead, it focuses on nine key parameters that have the most significant impact on transistor behavior. In semiconductor manufacturing, transistor characteristics are often adjusted through variations in ion implantation doses, and these nine parameters are particularly sensitive to such process tuning. The selected parameters are listed on the right side of Fig. 4.

For neural network training and testing, both the training and testing datasets are generated using Analog Devices LTspice [5], a widely used and freely available SPICE circuit simulation tool. Given a specified transistor device and its associated BSIM4 model, LTspice is used to generate $I_{ds} - V_{gs}$ and I_{DS} - V_{DS} curves. Starting from a nominal BSIM4 model, the values of the selected nine parameters are varied individually. For each resulting BSIM4 model, corresponding I_{DS} -V_{GS} and I_{DS} -V_{DS} characteristics are simulated using LTspice.

To maintain a controllable neural network size in this study, the transistor width and length are fixed. Nevertheless, the proposed framework can be readily extended to include variable device dimensions as additional input features. Each training and testing data sample therefore consists of two components: (1) the I–V curve data and (2) the corresponding values of the nine BSIM4 model parameters.

When sampling BSIM4 parameter values, each BSIM4 parameter is constrained by predefined lower and upper bounds. If each parameter were discretized into ten values within its allowable range, the total number of parameter combinations would be 10^9, which is computationally infeasible. Instead, for each data sample, nine random numbers uniformly distributed in the interval $[0, 1]$ are generated. These random values are linearly mapped to the corresponding parameter ranges. For example, a random value of 0 corresponds to the lower bound of the threshold voltage parameter V_{TH0}, while a value of 1 corresponds to its upper bound. Intermediate values generate parameter values within the specified range. By this way, one 9-digit random number generates one set of values for the nine BSIM4 parameters.

Using this approach, a total of 50,000 training samples are generated. The testing dataset consists of 500 samples, each containing a unique set of BSIM4 parameter values and the corresponding I–V data. For the I-V of each sample, LTspice simulates I_{DS} - V_{GS} curves under multiple drain–source and substrate bias conditions, as well as I_{DS} - V_{DS} curves under

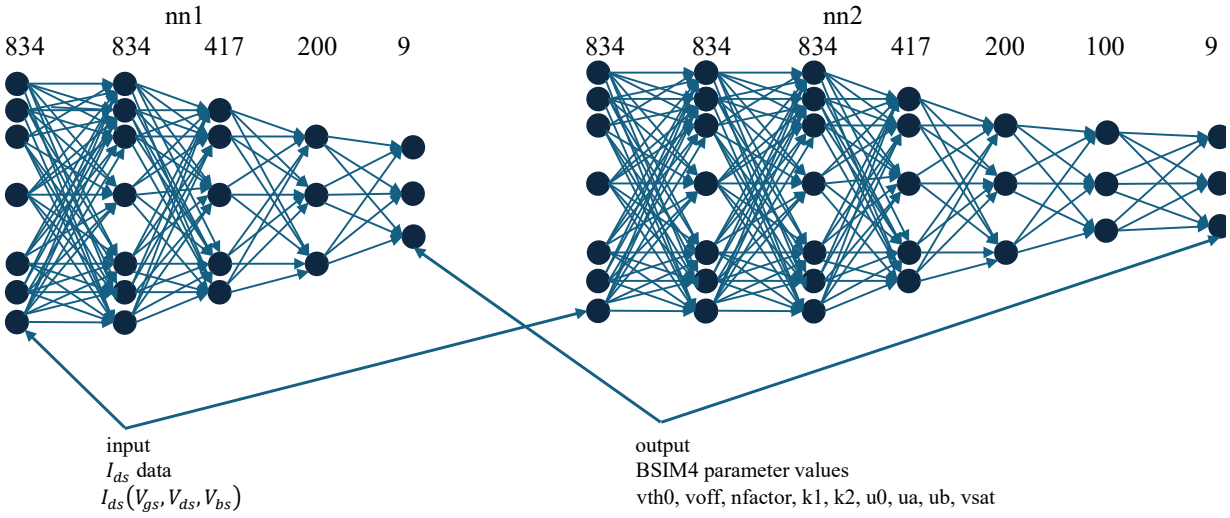

Figure 4. Two neural networks nn1 and nn2 are implemented. Each network has 834 neurons for the input layer which is corresponding to one set of Ids data under different Vgs, Vds, and Vsb bias. The output has 9 neurons for the nine BSIM4 parameters. The integer numbers above the networks are the neuron numbers at each layer. Between each neighboring layers, there is ReLU activation.

different gate–source and substrate biases, while keeping the nine BSIM4 parameters constant.

When preparing the dataset for PyTorch, the simulated I_{DS} values for each sample are concatenated into a single one-dimensional input vector. The concatenation order of the I_{DS}-$V_{GS}$and I_{DS}- V_{DS} curves is kept consistent across all samples. The voltage bias information is stripped. To use the neural network for prediction or extraction of its BSIM4 model values from I-V data, the I-V data must be measured by the exact same bias range and sweeping steps. In this work, each I–V sample contains a total of 834 I_{DS} data points, resulting in an input layer with 834 neurons. The output layer contains nine neurons, corresponding to the nine BSIM4 parameters to be predicted (or extracted).

The neural networks are implemented using PyTorch with GPU acceleration enabled. As illustrated in Fig. 4, two network architectures are investigated: one with three hidden layers and another with five hidden layers. Fully connected linear layers are used throughout the networks, with ReLU activation functions applied between layers. The loss function is defined as the mean squared error (MSE), and optimization is performed using the Adam optimizer with amsgrad=True to improve numerical stability. The learning rate is set to 0.001, and the training dataset is loaded using a batch size of 20.

III. Results

The Pytorch training is performed on a system with an AMD Threadripper 9960X with 128 GB of DDR5 memory, an NVDIA RTX 3050 with 6GB, and an NVIDIA RTX 5060 Ti GPU with 16 GB. The operating system is Rocky Linux 8. By setting the 'device' option to either 'cpu', 'cuda:0', or 'cuda:1', Pytroch trains the network with 9960X, 5060Ti, and 3050 separately. GPU-accelerated training is conducted using NVDIA CUDA 12.9.79 [6]. 500 epochs training of Three type training devices can finish within one hour.

The time to finish 500 epochs training of nn2 is listed in Table 1 for three devices. RTX 5060 Ti is about 2.2X faster than RTX 3050. This time ratio is inversely closer to the total memory bandwidth ratio (~2.7) than TFLOPS ratio (~3.5) of these two GPUs. This may suggest that the memory bandwidth is a bottleneck for the nn2 training with GPUs here. 9960X is the slowest though its total memory bandwidth is the 2nd highest. The training time of 9960X is only 1.6X slower than 3050. This may suggest the 2304 cores in RTX 3050 are not very efficient for the nn2 training due to its memory bandwidth or each AMD 9960X core is much more capable than one 3050 core. It would be very interesting to test these hypotheses by different neural network training, heavy on computation or heavy on data transfer.

Table 1. Training Time Comparison Between Devices

	Core	TFLOPS	Total memoery bandwidth	Training time of 500 epochs
Threadripper 9960X	24	n/a	180GB/s	3390s
RTX 3050	2304	6.774	168GB/s	2100s
RTX 5060Ti	4608	23.7	448GB/s	935s

Fig. 5 presents the testing error as a function of training epoch for both nn1 and nn2. The models are trained using 50,000 training samples, and the reported testing error is evaluated on the 500 testing data samples. Both neural network architectures exhibit a rapid reduction in error during the initial training epochs, followed by a slower rate of improvement beyond approximately 200 epochs. The seven-layer network nn2 shows reduced error fluctuation at higher epoch counts compared to the five-layer network nn1. Given the highly nonlinear and complex nature of BSIM4 I–V relationships, the deeper nn2 architecture is better suited for mapping I–V characteristics to BSIM4 model parameters.

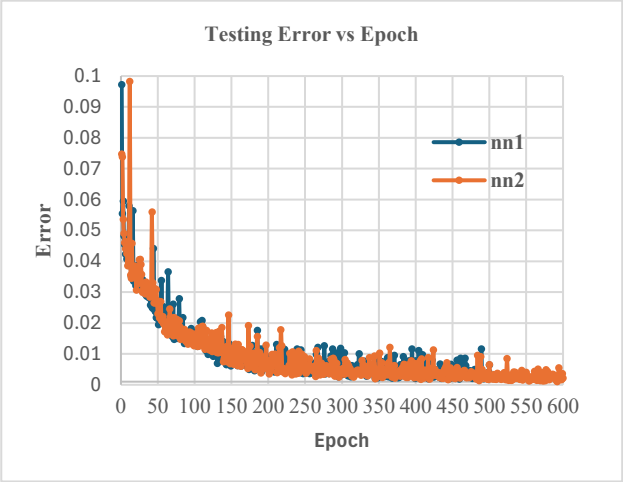

Figure 5. Testing error of two neural networks vs epoch. nn1 has three hidden layers and nn2 has 5 hidden layers. nn2 network has lightly smoother decrease of error vs epoch. After 300 epochs, the error decrease becomes very slow for both networks.

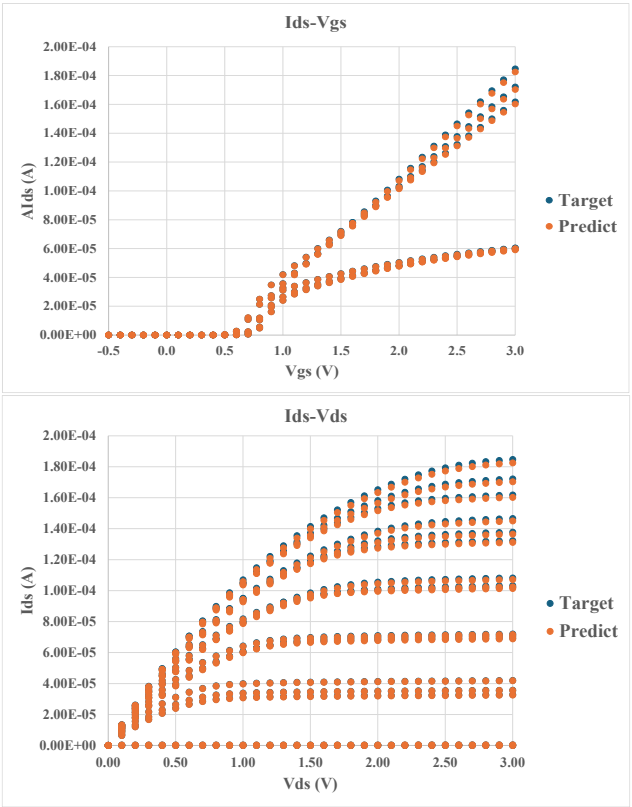

Figure 6. Comparison of I-V data for one testing sample. The target I-V data is used at the input of the nn2. Its output, extracted BSIM4 values, are used to re-generate the predict I-V data in LTSpice.

Fig. 6 shows the I-V data comparison. One I-V sample is input to nn2 to extract the values of the BSIM4 parameters. The extracted BSIM4 model is used by LTSpice to generate the predict I-V data. The original and predict I-V curves match very well. This confirms the nn2 is trained very accurately and its output is useable for accurate circuit simulation.

IV. Conclution

This work shows that neural networks is powerful and efficient for BSIM4 parameter extraction from I-V data. Simple linearly connected neural network with 5 hidden layers can achieve good result. The trained neural networks take I-V data and extract BSIM4 model parameter values to re-produce I-V with high accuracy. The neural network training is not very demanding for computer hardware. This work only extracts 9 parameters. Extension to more parameters is straight forward and will not increase training time exponentially. The approach can be used for local fine tuning of high fidelity device BSIM4 models, or fast generation of models for devices with mild process tuning. It can be easily integrated into semiconductor device simulation tools like Synopsys Sentaurus TCAD to generate BSIM models from TCAD I-V simulation data.

The next step will improve the way to generate I-V data. Current method uses LTSpice to generate I-V data for each BSIM4 parameter set is not very time efficient. For each data sample, Python code will call external LTSpice.exe in command line to start I-V circuit simulation. It takes many hours to generate 50k data samples. If more parameters are added to neural network, more data samples are needed. Current calling LTSpice approach will become bottleneck. Better way is to directly call BSIM4 code [1] to calculate the I-V data.

References

[1] BSIM4 model. URL https://bsim.berkeley.edu/models/bsim4/.

[2] BSIM4 model example. URL https://cmosedu.com/cmos1/cmosedu_models.txt.

[3] F. Chavez, C.-T. Tung, M.-Y. Kao, C. Hu, J.-H. Chen, and S. Khandelwal, "Deep learning-based I-V global parameter extraction for BSIM-CMG", *Solid-State Electronics*, vol. 209, Nov. 2023.

[4] Meta Pytorch 2.9.1, 2025. URL https://pytorch.org/projects/pytorch/.

[5] Analog Device LTSpice simulator 26.0.1, 2025. URL https://www.analog.com/en/resources/design-tools-and-calculators/ ltspice-simulator.html

[6] Nvidia Cuda 12.9.1, 2025. URL https://developer.nvidia.com/cuda-12-9-1-download-archive.