

To the Graduate Council:

I am submitting herewith a thesis written by Jack Hutchins entitled “Advancing Compact Modeling of Electronic Devices: Machine Learning Approaches with Neural Networks, Mixture Density Networks, and Deep Symbolic Regression.” I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Ahmedullah Aziz, Major Professor

We have read this thesis
and recommend its acceptance:

Ahmedullah Aziz

Garrett Rose

Catherine Schuman

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

To the Graduate Council:

I am submitting herewith a thesis written by Jack Hutchins entitled “Advancing Compact Modeling of Electronic Devices: Machine Learning Approaches with Neural Networks, Mixture Density Networks, and Deep Symbolic Regression.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Engineering.

Ahmedullah Aziz, Major Professor

We have read this thesis
and recommend its acceptance:

Ahmedullah Aziz

Garrett Rose

Catherine Schuman

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Advancing Compact Modeling of Electronic Devices: Machine Learning Approaches with Neural Networks, Mixture Density Networks, and Deep Symbolic Regression

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Jack Hutchins

May 2024

© by Jack Hutchins, 2024
All Rights Reserved.

Acknowledgements

I would like to extend my deepest gratitude to my advisor, Ahmedullah Aziz, whose expertise and insightful guidance have been indispensable throughout my research journey. His encouragement to pursue ambitious research questions has profoundly shaped my academic path, and for that, I am truly grateful. I am also immensely thankful to my parents for their unwavering love and support. Their belief in me and constant encouragement have been my strength throughout this journey. Additionally, I would like to thank all the members of the Nanoelectronic Devices and Integrated Circuits (NorDIC) Laboratory for their collaboration and camaraderie, which deeply enriched my research experience. Working alongside such dedicated and talented individuals has been both a privilege and a delight.

I am grateful for the opportunity to have parts of this thesis published in peer-reviewed journals. This has provided invaluable feedback and has significantly strengthened the research presented here. The work in chapter 2 was originally published in *IEEE Access* under the title "A generalized workflow for creating machine learning-powered compact models for multi-state devices" in October 2022. Chapter 3 was published in *Scientific Reports* under the title "Machine learning-powered compact modeling of stochastic electronic devices using mixture density networks" in March 2024.

This research was supported in part by seed funding from the AI Tennessee Initiative at the University of Tennessee, Knoxville.

This work was supported in part by the Air Force Research Laboratory under Agreement FA8750-21-1-1018.

Abstract

This thesis pioneers the integration of deep learning techniques into the realm of compact modeling, presenting three distinct approaches that enhance the precision, efficiency, and adaptability of compact models for electronic devices. The first method introduces a Generalized Multilayer Perception Compact Model, leveraging the function approximation capabilities of neural networks through a multilayer perception (MLP) framework. This approach utilizes hyperband tuning to optimize network hyperparameters, demonstrating its effectiveness on a HfOx memristor and establishing a versatile modeling strategy for both single-state and multistate devices.

The second approach explores the application of Mixture Density Networks (MDNs) to encapsulate the inherent stochasticity of electronic devices in compact models. By capturing complex probability distributions and employing an innovative sampling methodology based on inverse transform sampling, this method offers a more accurate representation of cycle-to-cycle variations in stochastic devices, as evidenced through experimental data from a heater cryotron.

Finally, the thesis ventures into deep symbolic regression to construct compact models that facilitate faster convergence in circuit simulations. This method employs a transformer-based neural network architecture to derive equations directly from device datasets, significantly streamlining the model creation process and enhancing simulation efficiency. The applicability and benefits of this approach are exemplified through the development of a compact model for a graphene nanoribbon field-effect transistor (GNRFET).

Collectively, these approaches embody a significant advancement in compact modeling, offering novel solutions that harness the power of deep learning to address the complexities and variabilities of modern electronic devices. This work not only contributes to the field of electronic device modeling but also opens new avenues for research in leveraging deep learning for enhanced computational simulations and model generalization.

Table of Contents

1	Introduction	1
1.1	Challenges with Lookup Table-Based Models	2
1.2	Neural Network-Based Models	3
1.3	Contributions of this Thesis	5
2	Generalized Multilayer Perception Compact Model	7
2.1	Introduction	7
2.2	Methods	9
2.3	Results and Analysis	12
2.4	Discussion	20
3	Mixture Density Networks for Stochastic Compact Models	22
3.1	Introduction	22
3.2	Results	24
3.2.1	Mixture Density Network	24
3.2.2	Model Sampling Methodology	26
3.2.3	Device Characteristics of Heater Cryotron	30
3.2.4	Heater Cryotron Model	33
3.3	Discussion	37
3.4	Methods	37
3.4.1	Heater Cryotron Fabrication	37
3.4.2	Heater Cryotron Characterization	38

3.4.3	Model Creation	38
4	Compact Modeling using Deep Symbolic Regression	40
4.1	Introduction	40
4.2	Deep Symbolic Regression	42
4.3	Results	45
5	Conclusions	48
5.1	Challenges and Limitations	49
5.2	Future Work	50
5.3	Final Thoughts	51
	Bibliography	52
	Vita	60

List of Figures

1.1	Lookup table model of a simple quadratic equation. (a) Shows a piecewise linear lookup table model, (b) shows the same model with noise in the data to represent real-world data.	4
1.2	(a) Derivative of the piecewise linear lookup table mode vs ideal derivative with no noise, (b) derivative with added noise.	4
2.1	Flowchart of our ML-powered compact modeling approach from TCAD simulation data to a completed compact model in Verilog-A.	10
2.2	Neural network architecture: Blue indicates input, green shows the hidden layers with ReLU activation, and orange shows the output layer with linear activation.	10
2.3	Hyperband algorithm for automatically tuning hyperparameters with successive halving.	11
2.4	Flowchart of the steps to prepare the Verilog-A model and its execution.	13
2.5	(a) Device structure and (b) I-V characteristics of an oxide memristor.	13
2.6	Fit of our baseline model on several values of capture cross-section (a)-(e) in linear scale and (f) in logarithmic scale.	16
2.7	Interpolation performance of our model and a LUT-based model for an untrained value of capture cross-section.	19
2.8	Performance evaluation of the model using experimental data with (a) sweep-to-sweep and (b) device-to-device variations.	21

3.1	Overview of the machine learning-powered modeling approach for the stochastic behavior of heater cryotron. After training the mixture density network with the device characteristics obtained from experiments, the weights are extracted which are eventually used to create the Verilog-A-based compact model for circuit/system-level simulations.	27
3.2	Prediction of Switching characteristics of heater cryotron. (a) A false-colored scanning electron microscope (SEM) image of heater cryotron device consisting two superconducting nanowires of tungsten silicide (WSi) and a dielectric spacer of SiO_2 . (b)-(c) Illustration of the gate-current-controlled switching mechanism. An enough gate current (I_G) switches the gate nanowire to its resistive state and generates thermal phonos which eventually causes the switching of the channel from its superconducting to resistive state. (d) Switching model for critical current evaluated at the mean (μ) and plus or minus 2 standard deviations (σ) of the resultant distribution and (e) retrapping current of heater cryotron switching model evaluated at the same points. . . .	31
3.3	Model validation for load voltage (V_L) against the experiment. The predicted vs experimental distributions of V_L (dropped across a $1\text{ k}\Omega$ load resistor) at a bias current of $23.5\text{ }\mu A$ and gate current of (a) $1.35\text{ }\mu A$, (b) $1.4\text{ }\mu A$, (c) $1.45\text{ }\mu A$, (d) $1.5\text{ }\mu A$, (e) $1.55\text{ }\mu A$, and (f) $1.6\text{ }\mu A$	35

3.4	Switching probabilities at varying bias currents and transient simulation results. The predicted vs experimental switching probability for gate current (I_G) between 0 μA and 3 μA at a bias current (I_B) of (a) 14 μA , (b) 16.5 μA , (c) 23.5 μA , (d) 28 μA , and (e) 33 μA . The increase in I_B leads to a reduction in the gate switching current. Transient dynamics of (f) bias current and (g) gate current which cause switching of the channel after exceeding the corresponding thresholds. (h) Switching is shown and validated with the experiment by the time dynamics of the load voltage (V_L) at mean (μ) and plus or minus 2 standard deviations (σ). (i) A zoomed-in view of V_L shown in (h). . .	36
4.1	Overview of our modeling approach.	44
4.2	Performance of model compared to data for output characteristics. . .	47
4.3	Inverter simulation using our GRNFET model (a) and GNRFET inverter diagram (b).	47

Chapter 1

Introduction

In the rapidly evolving domain of nanoelectronics, the need for highly accurate and efficient simulation tools has never been more critical. At the heart of these tools lie compact models of devices, which serve as the bridge between the abstract, mathematical description of electronic behavior and the tangible, physical world of semiconductor devices [Gildenblat \(2010\)](#). This thesis explores a new methodology of compact models using neural networks, proposing a paradigm shift in how we approach the simulation of nanoelectronic devices.

Compact models are indispensable in circuit simulation, providing an efficient but accurate representation of electronic devices. These models translate the complex physics of semiconductor materials and device operation into mathematical equations, enabling the simulation of circuits without necessitating detailed physical descriptions of each component. This abstraction not only significantly reduces computational overhead but also allows for the analysis and design of complex circuits with thousands to millions of components, a task that would be otherwise infeasible. This enables designers to predict behavior, optimize performance, and ensure reliability and robustness against variations in manufacturing and operating conditions [Gildenblat \(2010\)](#). In essence, compact models are the linchpin that enables the efficient and effective design and analysis of modern electronic circuits and systems.

While traditional physics-based compact models have served the field well, they increasingly face challenges in the nanoelectronic era. As devices shrink to nanometer scales, quantum mechanical effects become more pronounced, and the classical physics equations that underpin traditional models struggle to capture these phenomena accurately [Tsu \(2001\)](#); [Dasgupta et al. \(2020\)](#). Additionally, the sheer complexity and variety of new materials and device architectures outpace the development of corresponding physics-based models. These limitations demonstrate the need for alternative modeling approaches that can adapt more swiftly to the rapid pace of innovation in nanoelectronics and can capture the complex behavior of devices more accurately and with less computational cost.

1.1 Challenges with Lookup Table-Based Models

Lookup table (LUT)-based compact models are widely used in the simulation of electronic devices due to their simplicity and ability to implement without knowledge of the physics of a device. However, this approach often struggles with accurately capturing non-linear behaviors and the derivatives of those behaviors, which are crucial for circuit simulation and analysis. Electronic device characteristics are inherently non-linear, with their response curves often exhibiting complex behaviors that are difficult to model with simple equations. LUT-based compact models simplify these curves into piecewise linear segments, an approach that introduces approximation errors.

To illustrate the challenges of using LUT-based compact models, we consider the approximation of a quadratic function, $y = x^2$, representative of a non-linear device characteristic. The function and its piecewise linear approximation are plotted in Figure 1.1 (a) to demonstrate the model’s inability to accurately capture the curve’s true behavior, especially between widely spaced points in the lookup table.

Further complicating this issue is the introduction of noise, a common occurrence in real-world data due to measurement errors, device imperfections, or environmental

factors. Figure 1.1 (b) shows that even minor noise can significantly impact the accuracy of the piecewise linear approximation, leading to erroneous model outputs that deviate from the expected behavior.

In circuit simulation, the derivative of a device’s I-V characteristic, for example, is essential for simulation convergence [McAndrew et al. \(2015\)](#). Figure 1.2 shows the derivative of the lookup table-based model as compared to the actual derivative, which highlights the discrepancies that can arise. While the original function $y = x^2$ has a smooth and predictable derivative $y' = 2x$, the approximations introduce discontinuities and inaccuracies, particularly when noise is present. These inaccuracies can lead to significant errors in transient analysis, oscillatory behavior, and stability issues in the simulated circuits.

The derivative inaccuracies inherent in LUT-based compact models pose a significant challenge for their use in electronic device simulations, especially in applications requiring high fidelity and dynamic response analysis. The piecewise linear approximation, while computationally efficient, fails to capture the essential non-linear and dynamic characteristics of electronic devices. This limitation is exacerbated by real-world data imperfections, making it difficult to achieve the accuracy required for reliable circuit simulations.

While LUT-based compact models offer a practical solution for representing complex device behaviors in electronic simulations, their limitations, particularly in accurately capturing derivatives and handling noisy data, necessitate careful consideration. For applications where dynamic behavior and accurate derivatives are critical, alternative modeling approaches or enhanced interpolation methods may be required to ensure simulation accuracy and reliability.

1.2 Neural Network-Based Models

Neural networks present a compelling solution to the limitations of physics-based compact models. Unlike traditional models, neural networks do not require prior

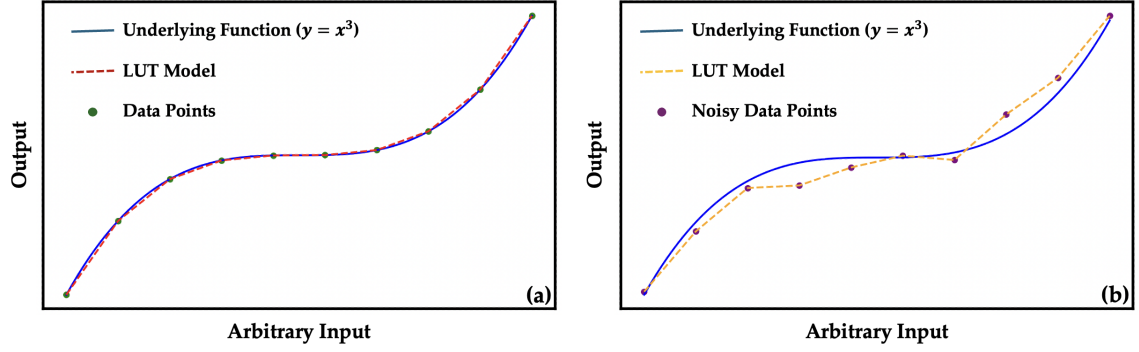


Figure 1.1: Lookup table model of a simple quadratic equation. (a) Shows a piecewise linear lookup table model, (b) shows the same model with noise in the data to represent real-world data.

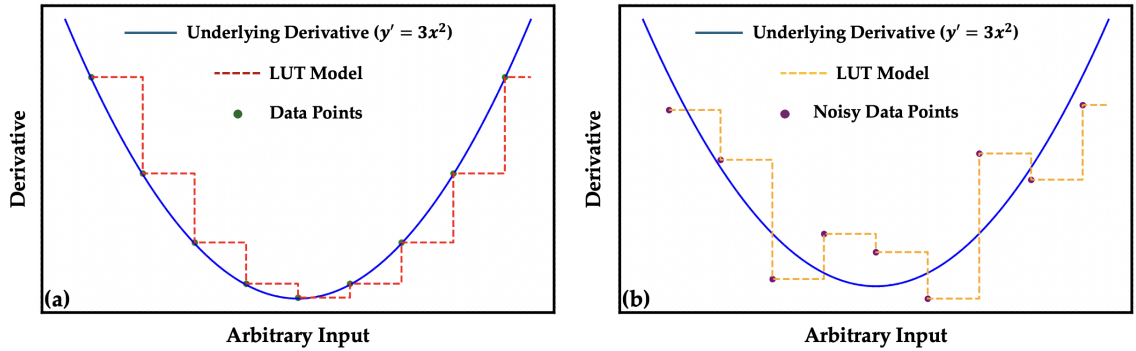


Figure 1.2: (a) Derivative of the piecewise linear lookup table mode vs ideal derivative with no noise, (b) derivative with added noise.

knowledge of the underlying physical processes. Instead, they learn the behavior of electronic devices directly from data. This data-driven approach enables neural networks to model complex nonlinearities and quantum mechanical effects inherent in nanoelectronic devices, which are often challenging for physics-based models to capture accurately.

Furthermore, neural networks offer several advantages over other alternatives, such as look-up tables. While look-up tables can also provide accurate modeling by interpolating between measured or simulated data points, they suffer from the curse of dimensionality, with exponential growth in size with each added variable, which makes them impractical for devices with multiple operational parameters [Bourenkov et al. \(2005\)](#). In contrast, neural networks can efficiently model complex interactions between multiple variables without an exponential increase in computational resources [Goh \(1995\)](#).

Additionally, neural networks can generalize from the data on which they are trained, allowing them to predict device behavior under conditions not explicitly included in the training set [Advani and Saxe \(2017\)](#). This capability, combined with their flexibility and adaptability, makes neural networks an ideal tool for developing compact models for the next generation of nanoelectronic devices.

1.3 Contributions of this Thesis

This thesis introduces three innovative methods for developing compact models through deep learning. The first method, the Generalized Multilayer Perception Compact Model, outlines a generalized framework for creating compact models using a multilayer perceptron (MLP). This technique aims to generate compact models powered by machine learning for both single-state and multistate devices, leveraging the neural networks' capability to approximate functions. To universalize this method further, hyperband tuning is employed to optimize the network's hyperparameters for any given device modeling task, with validation conducted on a HfOx memristor.

The second approach involves Mixture Density Networks (MDNs) for generating compact models that encapsulate the stochasticity of devices. MDNs enable the modeling of complex probability distributions for a set of input variables, thereby accurately reflecting the probabilistic behavior of stochastic devices. This method introduces a novel sampling technique based on inverse transform sampling, offering a more precise modeling of cycle-to-cycle variations in stochastic devices than traditional sampling methods. The efficacy of this approach is demonstrated through experimental data from a heater cryotron.

Lastly, the thesis presents a method using deep symbolic regression to create compact models. This method utilizes a transformer-based neural network architecture for deriving equations from a device dataset, facilitating the creation of compact models suitable for circuit simulation. The derived equations are more efficient than those generated by neural networks, resulting in quicker convergence during circuit simulation. This technique is validated by developing a compact model for a graphene nanoribbon field-effect transistor (GNRFET), showcasing its potential in simplifying and accelerating the simulation process for complex electronic devices.

The advent of nanoelectronics calls for a reevaluation of the traditional approaches to compact model development. This thesis posits that neural networks offer a superior alternative to physics-based models and other data-driven approaches like look-up tables. Through their ability to learn directly from data, neural networks can capture the complex behaviors of nanoelectronic devices with high accuracy and computational efficiency. The following chapters will delve deeper into the methodology, implementation, and validation of neural network-based compact models, aiming to chart a new course for the future of circuit simulation in the nanoelectronic era.

Chapter 2

Generalized Multilayer Perception Compact Model

2.1 Introduction

With the rapid evolution of emerging semiconductor devices, compact models have become crucial tools to facilitate device-circuit co-design [Aziz \(2019\)](#). Many of these compact models are derived from the physics of the device, but often, the underlying physics are not immediately understood or are too complicated to implement in a computationally efficient manner. This makes it difficult to create physics-based compact models for novel devices. These models become even more challenging for devices with multiple states such as memristors [Zeumault et al. \(2022\)](#), ferroelectrics [George et al. \(2016\)](#), antiferroelectrics [Hoffmann et al. \(2022\)](#), threshold switches [Shukla et al. \(2016\)](#), and so on. In recent years, machine learning (ML) has been adopted in compact modeling [Aadithya et al. \(2020\)](#); [Chiozzi et al. \(2018\)](#); [Kam \(2021\)](#); [Wang et al. \(2021\)](#); [Zhang and Chan \(2017\)](#); [Lin et al. \(2022\)](#); [Li et al. \(2016a\)](#); [Lin et al. \(2020\)](#). However, most of these approaches only focus on single-state devices. Many prior works use multilayer perceptron (MLP) neural networks as the basis for the compact model [Aadithya et al. \(2020\)](#); [Chiozzi et al. \(2018\)](#); [Kam \(2021\)](#);

Wang et al. (2021); Zhang and Chan (2017). This approach is very general, meaning that the MLP architecture itself will remain relatively constant between different devices. Another approach commonly used is physics-inspired neural network models Lin et al. (2022); Li et al. (2016a). These physics-inspired models modify the standard, sequential MLP architecture to match more closely to the known underlying physics of the device. While physics-inspired models typically achieve high accuracy, the physics-inspired modifications done to the network mean that the approach cannot be generalized to multiple devices.

One approach that has been proposed for creating machine learning-based compact models for multistate devices is to use long short-term memory (LSTM) neural networks Lin et al. (2020). As pointed out in a later work by the same group, LSTMs are far too computationally expensive for compact models Lin et al. (2022). As a result, they chose to use a physics-hybridization approach to create compact models for resistive random-access memory (RRAM) Lin et al. (2022). While this approach performed well, significant changes to the model are required for other types of multistate devices to work with it.

In this work, we propose a device agonistic framework to create machine learning-based compact models for multistate devices. For this, we propose using MLP networks to provide as much generalization as possible in our approach. Additionally, we use Hyperband tuning in our model to automatically select the hyperparameters (number of neurons, learning rate, etc.) for the MLP model. By doing this, we can further generalize the approach and decrease the amount of time required to make a new model using our framework. In addition, we add data augmentation to improve the performance of our model and decrease the amount of data required to train the model.

2.2 Methods

To capture the current-vs-voltage (I-V) characteristics of a semiconductor device, we choose to use MLP artificial neural networks (ANN). MLP models can learn any nonlinear function given that the network is large enough [Hornik et al. \(1989\)](#). This makes them ideal for compact modeling since there should be an underlying nonlinear function that describes the device. The first step to creating an ML-based compact model is to create a dataset that the model can train from. This dataset can either be compiled experimentally or with the use of tools such as technology computer-aided design (TCAD) simulation tools [Zeumault et al. \(2021\)](#). As with any neural network, data augmentation will drastically affect the performance of the model.

We propose duplicating the data and adding Gaussian noise to the data to improve the performance of the model. While intentionally adding noise to the data might be strange, we found that it provided a statistically significant improvement in the performance of the model (further discussion in section 3). Figure 2.1 shows a full flowchart of our framework.

To create and train our neural network, we use Python and TensorFlow [Abadi et al. \(2016\)](#) with Keras [Chollet et al. \(2015\)](#). Keras allows for the rapid development, training, and testing of neural networks, making it ideal for our application. We use a three hidden layer MLP network to create our compact model to provide a good balance of computational complexity and accuracy. As shown in Figure 2.2, our model takes the voltage, state, and device parameters as inputs. The state is one-hot encoded and switched by thresholds defined by the user. Device Parameters are additional parameters (such as device size, operating temperature, etc.) that affect the model.

To further generalize the ML model for other devices, we employ Hyperband tuning [Li et al. \(2018\)](#) to select the optimal hyperparameters for the neural network (flow chart is shown in Figure 2.3). Hyperparameter selection can drastically affect the performance of a model, and different data require different hyperparameters [18].

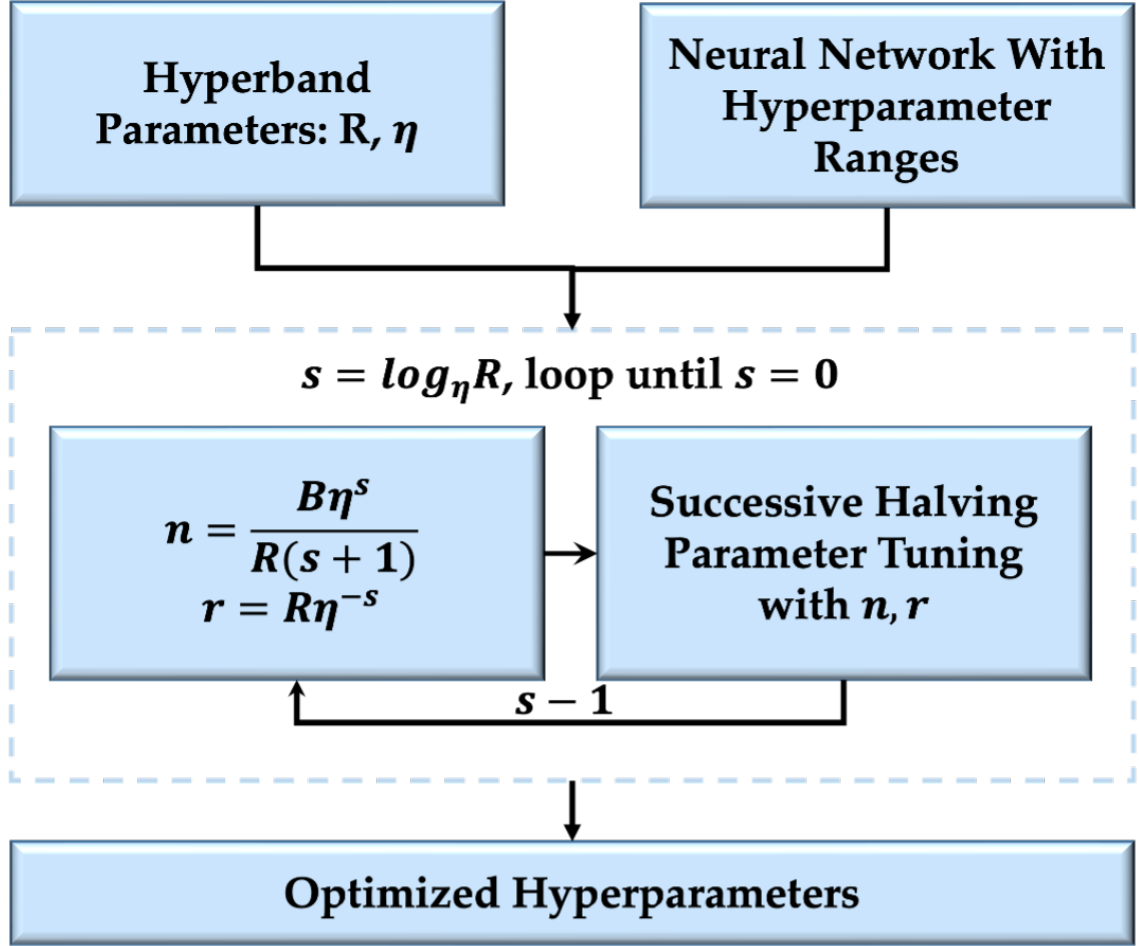


Figure 2.3: Hyperband algorithm for automatically tuning hyperparameters with successive halving.

Hyperband tuning automatically selects the hyperparameters from a user-provided range by expanding upon the Successive Halving approach [Jamieson and Talwalkar \(2016\)](#). As shown in Figure 2.3, Hyperband runs the Successive Halving parameter tuning algorithm in several configurations of its parameters. Hyperband itself takes two parameters, R and η . R represents the maximum time the algorithm can spend on any configuration and η controls the number of configurations discarded after each run of Successive Halving. We implement Hyperband using the KerasTuner [O’Malley et al. \(2019\)](#) library, which provides a natively supported implementation of Hyperband for Keras models. For this work, we use Hyperband to select the learning rate, the learning rate decay, and the number of neurons in each of the hidden layers.

After a model is trained in Python, a Verilog-A [McAndrew et al. \(2015\)](#) file is created via a conversion script to make it usable in major circuit simulators such as SPICE, SPECTRE, etc. Our custom-built conversion script works by extracting the weights and biases from our tuned MLP model and converting them into multidimensional arrays in Verilog-A. Next, the appropriate matrix operations for forward propagation of the neural network are added to the file so the neural network can perform inference on inputted data. Finally, additional code to facilitate state switching is included according to the thresholds provided by a user. From here, the Verilog-A file can be used like any other compact model in circuit simulations. Figure 2.4 shows the overall process of creating the Verilog-A file.

2.3 Results and Analysis

To test our modeling framework, we chose to use a hafnium oxide (HfOx) memristor [Zeumault et al. \(2021\)](#) (Figure 2.5). This device has four distinct states: unformed, formed, post-forming low resistive state (LRS), and post-forming high resistive state (HRS). Initially, the device remains in an unformed state since there is no conducting path between the electrodes. Then, the device is formed with a high voltage, known as forming. Forming is a one-time operation [Zeumault et al. \(2021\)](#). After forming,

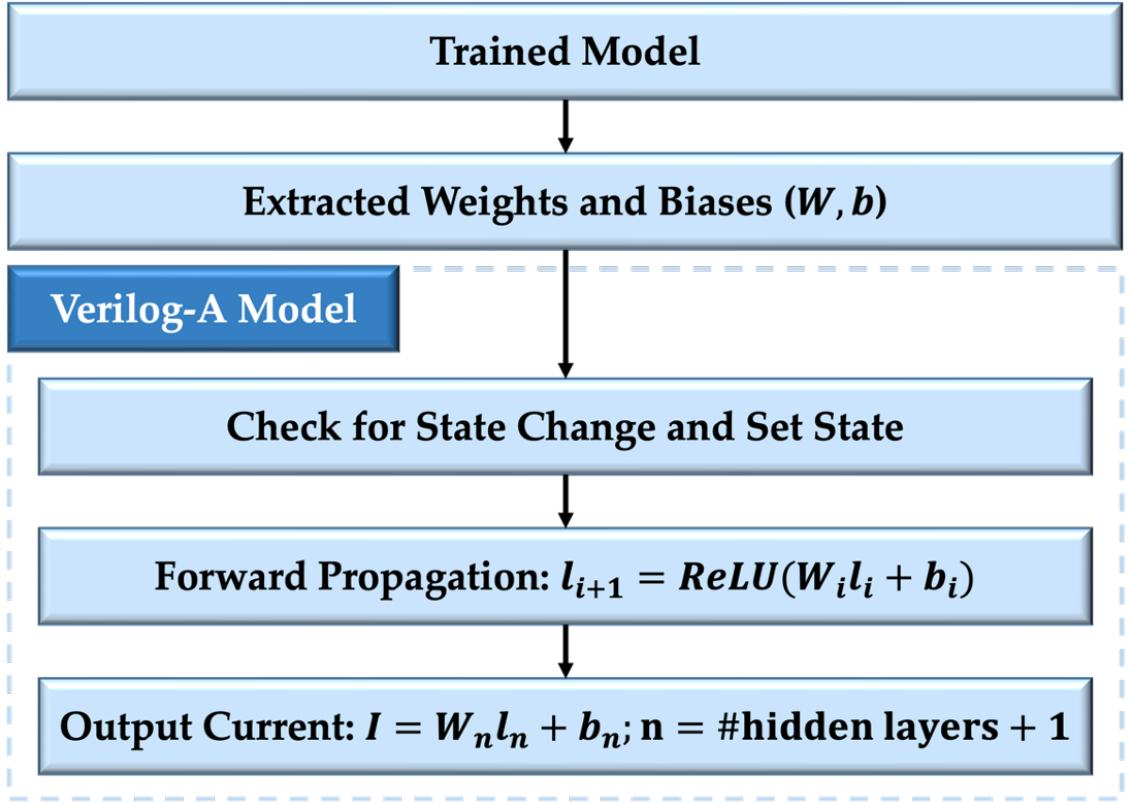


Figure 2.4: Flowchart of the steps to prepare the Verilog-A model and its execution.

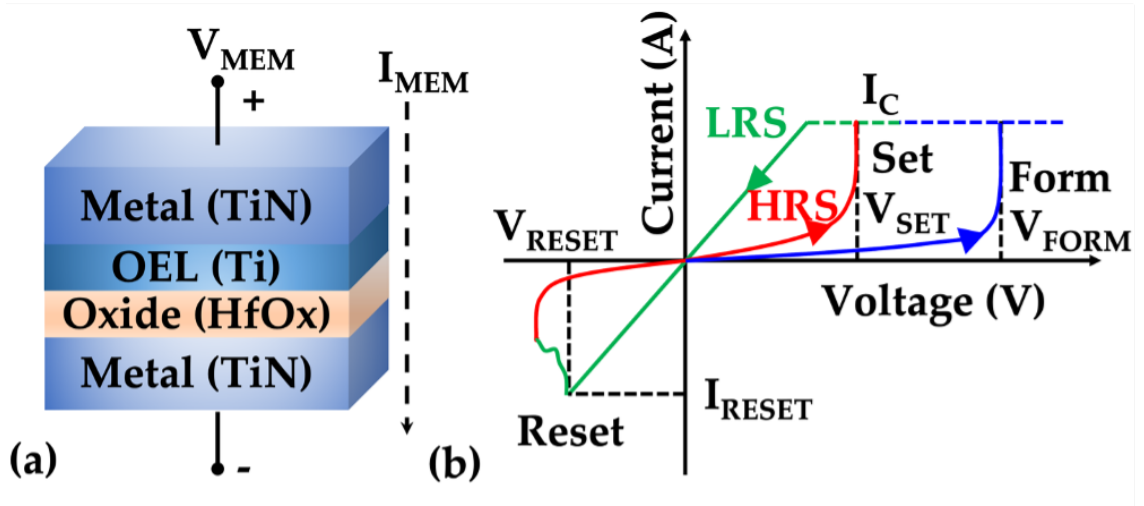


Figure 2.5: (a) Device structure and (b) I-V characteristics of an oxide memristor.

the device can show two resistance states (HRS and LRS) based on the voltage across it. We first generate several sets of I-V data for the memristor using a TCAD model interfaced with a Kinetic Monte Carlo (KMC) framework [Zeumault et al. \(2021\)](#). Here, the KMC framework is utilized to model the formation and rupture of the filament. Then, TCAD takes the filament dynamics as the input and performs the quasi-static simulations to estimate the switching I-V characteristics. We use this model to generate a single sweep for each of the states in intervals of 0.1 V. The data is between -1.0 V and 2.0 V for set and reset, and between 0 and 2.5 V for forming. We repeat these sweeps for 5 different values of capture cross-section [Zeumault et al. \(2022\)](#), a crucial device parameter. Finally, we add the state to the dataset using one-hot encoding. Since there are four states, we add four additional variables to the data where each represents a state. When the device is in a specific state, its corresponding variable is equal to '1' and the other three are set to '0'. After generating the data, we added 125 duplicates of the data with Gaussian noise as described in section 2. This is required since we only have one sweep for each configuration of the memristors. Note, this step may not be necessary for larger datasets with multiple sweeps.

For evaluating models, we use two statistics: root mean squared error (RMSE) and R^2 . RMSE shows the typical error in the model for a particular dataset, while R^2 shows the fraction of variance explained by the model (1 is a perfect score). There are two ways in which we will use the RMSE and R^2 , with training data and testing data. Before we train the model, we remove 20% of the dataset randomly to use as testing data so that the model is never trained on this data. This way, we can use this data to evaluate the model and prevent overfitting [Dietterich \(1995\)](#) on the training data. We use TensorFlow’s early stopping function to stop the model from training early when the RMSE stops improving.

Figure 2.6 shows the fit of the model to the data using a base configuration of hyperparameters and noise added. This model provides an RMSE of 0.000915 and an R^2 of 0.9737 on the testing data. Now that we have a baseline model, we can compare this to other configurations of our model. First, we can see how using Hyperband

tuning affects the performance of our model. Table 2.1 shows the hyperparameters that Hyperband selected compared to the ones we selected. For the parameters of Hyperband, we use $R=10$ and $\eta=3$. Since we are using the KerasTuner package [O'Malley et al. \(2019\)](#), in this case, R means the maximum that a certain configuration can run, and η is set to its default of 3. Hyperband was able to select the number of neurons in each layer (between 20 and 256), the learning rate (between 0 and 0.1), and the learning rate decay (between 0.8 and 0.99). The hyperparameters that Hyperband selected seem reasonable, so now we can compare the configuration it chose with the one we chose. Table 2.2 shows the RMSE and R^2 for both the training and testing data, while the human-selected parameters perform slightly better with the training data, the Hyperband-selected parameters perform much better with the testing data. The testing data is much more valuable for us since it more accurately represents the performance of the model in approximating the underlying function of the I-V characteristics. Beyond the performance in this case, Hyperband also helps us generalize this approach to other devices. Since Hyperband will select the model's parameters given a reasonable range to pick from, tuning hyperparameters in this model for any new device will be relatively simple.

Next, we evaluate how adding the Gaussian noise affects the model. For this, we use the model without Hyperband. To appropriately test this, we need to make sure the same number of training steps are used. The fairest way to do this is to still duplicate the data, but not add the Gaussian noise to it. This way, we can maintain the same number of epochs and the same batch size. Table 2.3 shows that the model with Gaussian noise is better in terms of RMSE and R^2 with both the training and testing data. Using a two-sample t-test [Cressie and Whitford \(1986\)](#), we can see that the difference in RMSE between these models is statistically significant (likelihood to happen by chance, $p=0.048$). This result shows that to some extent, we can artificially increase the amount of data we have using this technique. This is important since generating additional data with physical devices takes a significant

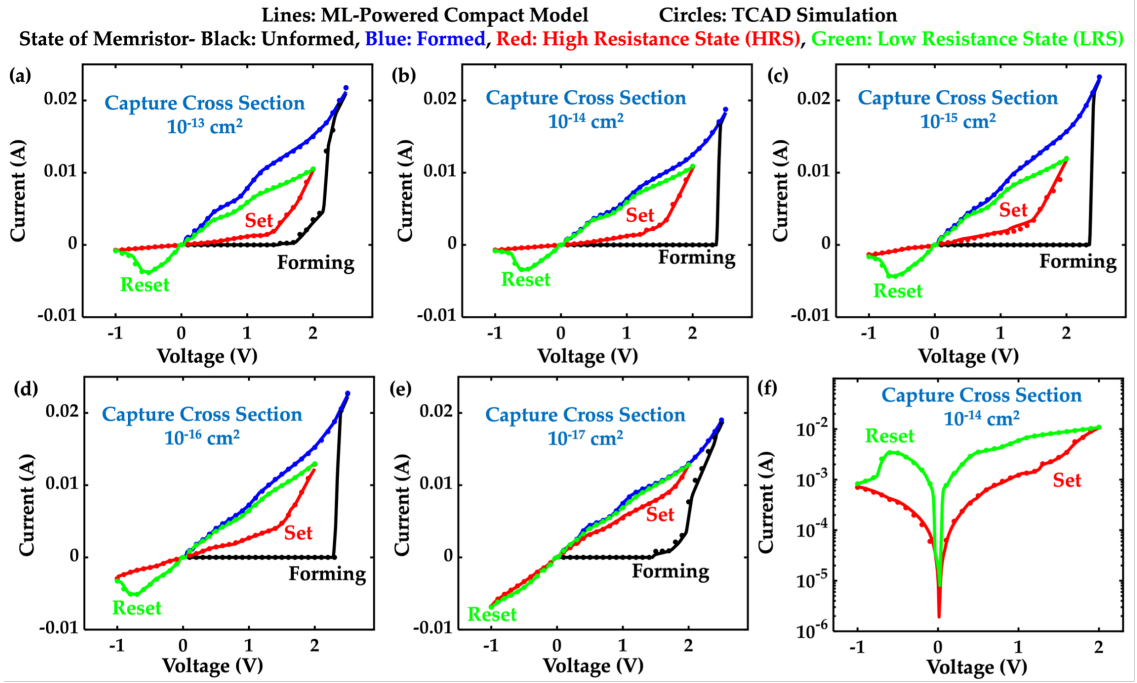


Figure 2.6: Fit of our baseline model on several values of capture cross-section (a)-(e) in linear scale and (f) in logarithmic scale.

Table 2.1: Parameter selection: Human vs. Hyperband.

Selection Method	Parameter Name	Selected Parameter
Human-Selected Parameters	Layer 1 Neurons	256
	Layer 2 Neurons	128
	Layer 3 Neurons	64
	Learning Rate	0.02
	Learning Rate Decay	0.9
Hyperband-Selected Parameters	Layer 1 Neurons	182
	Layer 2 Neurons	188
	Layer 3 Neurons	148
	Learning Rate	0.0119
	Learning Rate Decay	0.8318

amount of time and resources. On the other hand, generating additional data with TCAD is computationally expensive.

Next, we can evaluate how well our model interpolates for data points it has not been trained on. Our dataset contains 5 values of capture cross-section: 10^{-17} , 10^{-16} , 10^{-15} , 10^{-14} , and 10^{-13} cm^2 (Figure 2.6). To test how well the model interpolates, we exclude a specific capture cross-section (10^{-15} cm^2) from our training data and then evaluate the performance of the model at that value of the capture cross-section. To rationalize the performance of our approach, we compare it against a lookup table model. Figure 2.7 shows the interpolation of our model to the data for an untrained value of capture cross-section, and Table 2.4 shows the RMSE and R^2 for each of the models (ML and LUT models). Our model drastically outperforms the lookup table-based model in all our metrics from both LRS and HRS. A single-sided t-test shows the performance increase is statistically significant ($p < 0.001$ for all metrics).

Finally, we evaluate our model using experimental data from fabricated memristors. Here, we use I-V characteristics of HfOx memristor samples, fabricated in the 65 nm CMOS/memristor process developed by SUNY Polytechnic Institute. The experimentally obtained data is unique from the simulated data since we have a collection of multiple device samples, each with 12 sweeps of the same device. Thus, we include both sweep-to-sweep and device-to-device variations in the I-V characteristics. Figure 2.8 shows the performance of our model with this data, with the model achieving an RMSE of 8.5977×10^{-6} and an R^2 of 0.99248. Figure 2.8(a) shows that our model will naturally converge to the average of the multi-sweep data. Additionally, we test our modeling approach with data from multiple devices to test how the model handles device-to-device variations. As shown in Figure 2.8 (b), the model again converges to the average device I-V characteristics. As opposed to the sweep-to-sweep variation, the multi-device data exhibit high variability in the switching voltage. When the model is trained on this type of data, the model converges to the average switching voltage.

Table 2.2: Performance of ‘Human’ and ‘Hyperband’ selected parameters.

Method	Training		Testing	
	RMSE	R^2	RMSE	R^2
Human Selected Parameters	0.000271	0.99718	0.000915	0.973700
Hyperband Selected Parameters	0.000274	0.997031	0.000863	0.977371

Table 2.3: Model Performance with and without added noise.

Method	Training		Testing	
	RMSE	R^2	RMSE	R^2
Gaussian Noise	0.000271	0.99718	0.000915	0.973700
Without Noise	0.000281	0.997062	0.001128	0.962922

Table 2.4: Interpolation performance of our model and the Lookup Table-based model.

Method	LRS		HRS	
	RMSE	R^2	RMSE	R^2
Our Model	0.000484	0.990896	0.000323	0.988363
Lookup Table	0.000764	0.975318	0.000957	0.937633

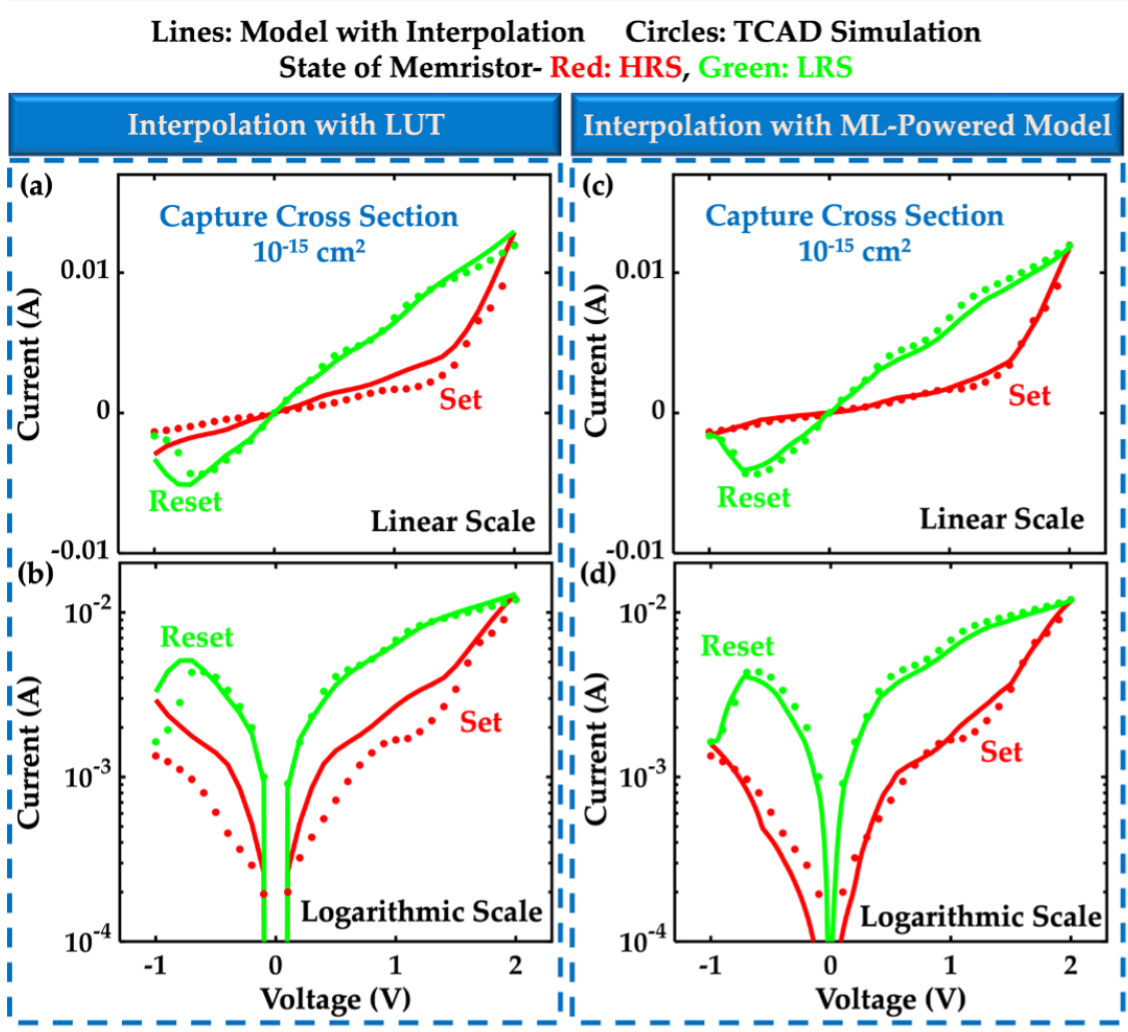


Figure 2.7: Interpolation performance of our model and a LUT-based model for an untrained value of capture cross-section.

2.4 Discussion

With our approach, we can generalize the process of creating ML-based compact models for multistate devices while reducing the amount of training data required. Our process of adding gaussian noise to the training data simulates having multiple devices in the dataset, reducing the amount of data required to create an effective compact model. Duplicating and adding noise reduced the RMSE of the testing data from 0.001128 to 0.000915, an 18.8% decrease in the model’s error on testing data. Additionally, our use of Hyperband parameter tuning reduces the RMSE on the testing data by an additional 5.7%, while also generalizing the approach to different devices that require different hyperparameters. This leads to a total decrease in testing RMSE of 23.5% when compared to a neural network on its own for the tested memristor. Lastly, we have shown that our model is capable of being trained from a variety of data sources, including TCAD model derived data and experimentally measured data. In this work, we also developed a script to convert the ML-based model into Verilog-A. Verilog-A models are versatile and cross-platform compatible. They can be used to simulate different varieties of memristive circuits, such as memory arrays [Aziz et al. \(2016\)](#), neurons and synapses [Xia and Yang \(2019\)](#), and oscillators [Fouda et al. \(2013\)](#). The possibility to enable such a wide spectrum of simulations was the prime motivation behind converting our ML model to the Verilog-A platform.

Validation for Experimental Data from 65 nm CMOS/memristor Process

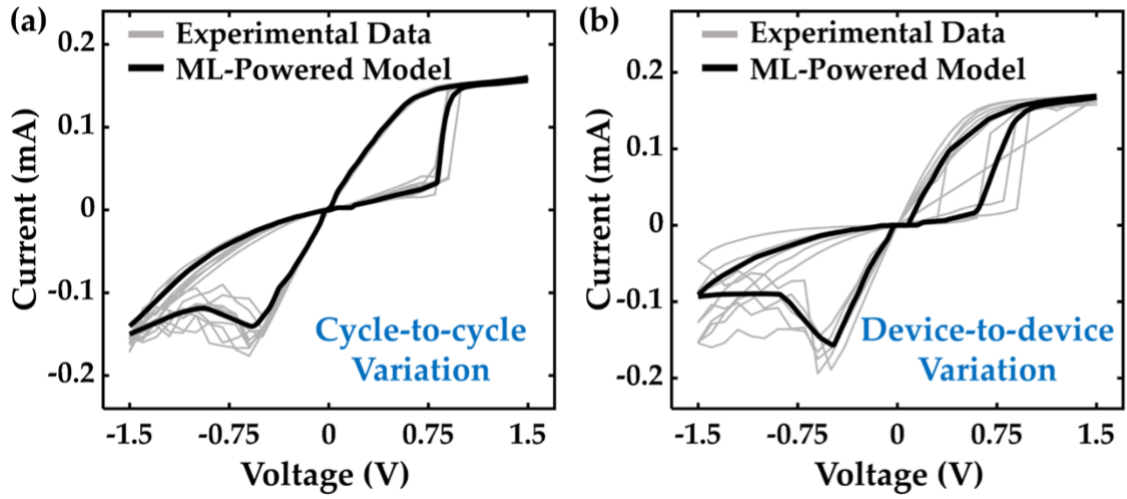


Figure 2.8: Performance evaluation of the model using experimental data with (a) sweep-to-sweep and (b) device-to-device variations.

Chapter 3

Mixture Density Networks for Stochastic Compact Models

3.1 Introduction

In the ever-evolving landscape of electronic devices, the pursuit of smaller, faster, and more efficient technology has led to groundbreaking innovations, unlocking new possibilities for various applications across industries [Salahuddin et al. \(2018\)](#). However, as devices continue to shrink and push the boundaries of Moore’s law [Lundstrom \(2003\)](#), the inherent stochastic nature of certain electronic components has become increasingly significant [Alawad and Lin \(2016\)](#). These stochastic electronic devices exhibit inherent variability in their behavior, posing a formidable challenge for conventional deterministic modeling approaches [Hamilton et al. \(2014\)](#).

In the pursuit of greater accuracy and ease of implementation in electronic circuit simulations, a significant leap has been taken by incorporating machine learning methodologies into compact modeling [Li et al. \(2016b\)](#); [Zhang and Chan \(2017\)](#); [Wang et al. \(2021\)](#); [Abouelyazid et al. \(2022\)](#); [Lin et al. \(2022\)](#); [Hutchins et al. \(2022\)](#). While various machine learning-based compact models have emerged, these models, thus far, have failed to address the implications of stochastic behavior in electronic

devices. And while traditional physics-based compact models have been created with stochastic properties [Wang et al. \(2014\)](#); [Beckle et al. \(2021\)](#), these models lack the benefits of machine learning-based methods such as decreased turnaround time and little required device knowledge. The dynamics of electronic devices, traditionally considered deterministic, are increasingly proving to be intrinsically stochastic at the nanoscale. Consequently, conventional compact models that ignore these inherent stochastic processes fail to capture the true behavior of the devices, leading to inaccuracies in circuit simulation.

This paper presents a new approach that addresses this gap in the realm of compact modeling. We introduce the use of Mixture Density Networks (MDNs) [Bishop \(1994\)](#) to capture the stochastic nature of electronic devices accurately. In addition to this neural network approach, we introduce a novel sampling methodology that uses inverse transform sampling [Devroye \(1986\)](#) to make our model more accurate to the stochastic nature of devices. By doing these, we create compact models that account for the inherent stochasticity of certain devices, allowing the development of new circuits that are both reliable and innovative. Our approach represents a paradigm shift in electronic device modeling, as it for the first time encompasses the full spectrum of electronic device behavior, from deterministic to stochastic.

To demonstrate the effectiveness and versatility of our approach, we focus on the modeling of heater cryotron [McCaughan and Berggren \(2014\)](#), a three-terminal device that shows gate-current controlled switching between superconducting and resistive states. Heater cryotrons, with their stochastic switching behavior, are a particularly compelling case study. Our methodology accurately captures the nuances of this stochasticity, providing a foundation for the development of novel circuits and the optimization of existing designs.

The contributions of this paper are threefold:

- **Mixture Density Networks:** By employing MDNs, we equip our modeling framework with the capacity to capture complex probability distributions,

thereby enabling a more faithful representation of device variability. This empowers designers to explore device behavior beyond traditional deterministic bounds.

- **Inverse Transform Sampling:** By leveraging inverse transform sampling, we can use our MDN to its full potential. This approach allows the model to output smooth outputs in transient simulations while maintaining stochasticity. This in turn makes our model more realistic to device-to-device or cycle-to-cycle variation.
- **Demonstration with Heater Cryotrons:** To showcase the effectiveness of our approach, we focus on modeling heater cryotrons—a class of devices known for their stochastic switching behavior. Through this demonstration, we illustrate the practical applicability of our methodology in real-world electronic systems.

As we delve into the details of our approach, we will elucidate the intricacies of MDN-based modeling for stochastic electronic devices, providing a comprehensive understanding of the benefits it offers to circuit simulation and electronic design.

3.2 Results

The following section will describe our neural network architecture, sampling methodology, and the results of our approach using experimentally derived heater cryotron data.

3.2.1 Mixture Density Network

Mixture density networks provide the perfect approach for modeling stochastic devices. Mixture density networks work similarly to standard multilayer perceptions (MLP) [Hornik et al. \(1989\)](#), but with three output layers connected to the last hidden

layer instead of just one. Each output layer has the same number of neurons N , and we label the output layers as μ , σ , and α . In our model, we use 2 hidden layers with 512 neurons each using the ReLU activation function, and we set $N = 20$. The output of these layers is used as the parameters of a Gaussian mixture distribution probability density function (PDF) as defined in equation (1).

$$p(x) = \sum_{k=1}^N \alpha_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right) \quad (3.1)$$

Using this approach allows the model to learn the unique output distribution for any given input. The PDF is a combination of N different Gaussian distributions, which are multiplied by a scaling factor α . In order to maintain a valid probability distribution, we need to ensure that the sum of α is always equal to 1. To do that, we apply a softmax function to the output of the α layer, as defined in equation (2).

$$s(\alpha_i) = \frac{e^{\alpha_i}}{\sum_{j=1}^N e^{\alpha_j}} \quad (3.2)$$

Additionally, we need to ensure that the output of the σ layer needs to be positive and non-zero since they represent the standard deviation of the Gaussian distributions in the mixture density PDF. In order to do this, we use the exponential linear unit (ELU) activation function plus 1 plus ϵ , where ϵ is the smallest value before loss of precision in floating point calculations. The activation function can be seen in equation (3).

$$ELU(\sigma_i) = \begin{cases} \sigma_i + 1 + \epsilon, & \text{if } \sigma \geq 0 \\ 0.5(e^{\sigma_i} - 1) + 1 + \epsilon, & \text{if } \sigma < 0 \end{cases} \quad (3.3)$$

Finally, we allow μ to be any value, so there is no need to use an activation function to restrict its output. For the hidden layers, we chose to use the rectified linear unit activation function, which is defined in equation (4).

$$ReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \quad (3.4)$$

In order to train the model, we need an appropriate loss function to measure the performance of the model. We choose to use Gaussian Negative Log-Likelihood (GNLL), since it effectively captures the performance with respect to probability. The GNLL for a single data point x is the negative logarithm of this PDF:

$$NLL(x) = -\log \left(\sum_{k=1}^N \alpha_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp \left(-\frac{(x - \mu_k)^2}{2\sigma_k^2} \right) \right) \quad (3.5)$$

The goal during training is to minimize the GNLL across all training data points. Therefore, the overall GNLL loss is the average GNLL over all data points:

$$L = -\frac{1}{N} \sum_{i=1}^N \log \left(\sum_{k=1}^K \alpha_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp \left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2} \right) \right) \quad (3.6)$$

During the training process, the network undergoes parameter adjustments aimed at minimizing the GNLL loss. To achieve this, we employ the backpropagation algorithm coupled with the AdamW optimizer, as introduced by Loshchilov and Hutter in 2019 [Loshchilov and Hutter \(2019\)](#). This optimization technique facilitates the iterative refinement of the network's weights and biases. The gradients of GNLL with respect to key parameters, namely μ , σ , and α , are computed using the depicted framework in Figure 3.1.

3.2.2 Model Sampling Methodology

In order to properly utilize our model, it is critical that we use an appropriate sampling methodology for the output distributions. The obvious approach for this would be to randomly sample from the distribution, however, this approach presents several issues. In order to sample in this way, we can use the approach outlined in Algorithm 1, where we randomly select a distribution k for the mixture

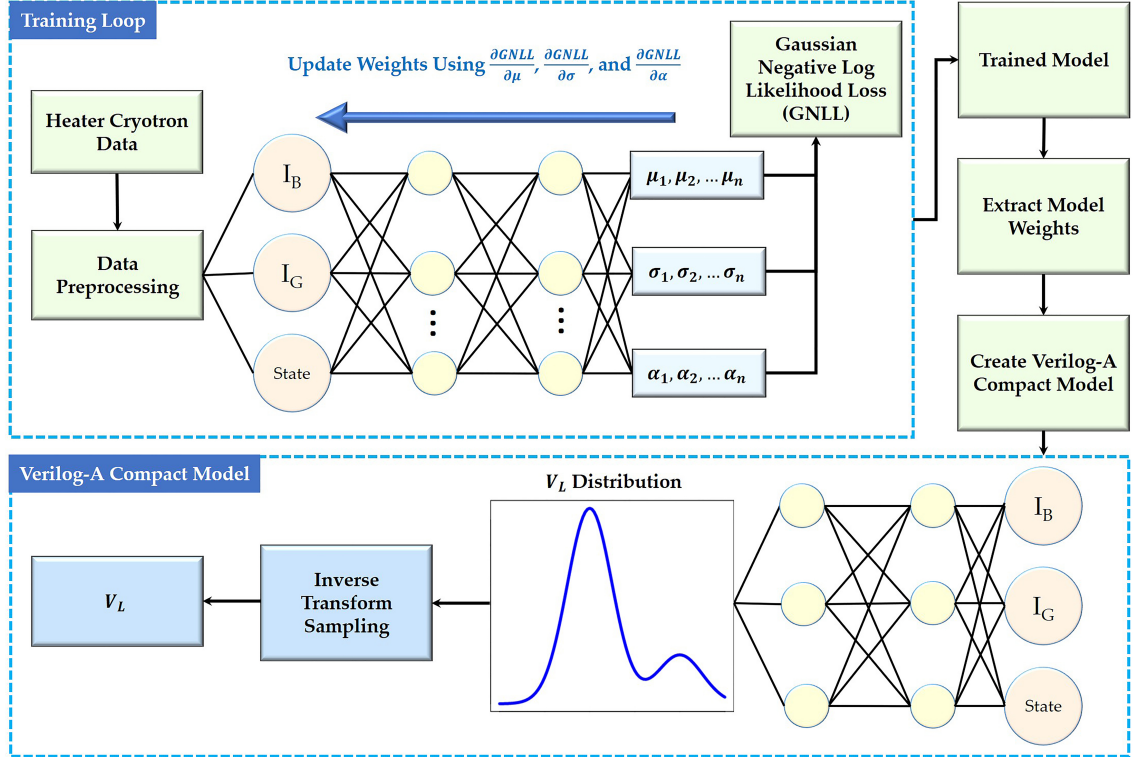


Figure 3.1: Overview of the machine learning-powered modeling approach for the stochastic behavior of heater cryotron. After training the mixture density network with the device characteristics obtained from experiments, the weights are extracted which are eventually used to create the Verilog-A-based compact model for circuit/system-level simulations.

density distribution and sample from a standard normal distribution $\mathcal{N}(\mu_k, \sigma_k)$. This approach allows us to utilize uniform and standard Gaussian sampling, which is built into most modeling languages including Verilog-A. The issue with this approach is that it results in sporadic currents in transient simulations since the current will jump around the distribution every time step. In addition, this approach will cause multi-state devices to switch sporadically between states near the critical value if the switching of the device is stochastic. While this approach may be valuable for some devices, for most devices we will need another sampling methodology that more accurately reflects the variation we see in devices. We propose using inverse transform sampling [Devroye \(1986\)](#) to accomplish this.

Inverse transform sampling involves obtaining the cumulative distribution function (CDF) of the target distribution and finding its inverse. The CDF represents the probability that a random variable is less than or equal to a specific value. In this method, a uniform random number between 0 and 1 serves as the input, representing a quantile of the distribution. The inverse CDF maps this input to the corresponding value in the distribution.

As the distribution evolves over time, the CDF is updated to reflect these changes. Inverse transform sampling allows for the generation of samples that consistently align with a predetermined quantile (q) for each sweep of the device. This ensures that the model accurately captures the device's behavior at specific points in the distribution, providing a constant sampling point across different timesteps of the same sweep. This approach allows the model to provide a smooth curve while still demonstrating the stochasticity that we are trying to model and offering a coherent representation of the device's performance under dynamic conditions. To initiate this process, we can begin by examining the probability density function in equation (7).

$$p(x) = \sum_{k=1}^K \alpha_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left(-\frac{(x - \mu_k)^2}{2\sigma_k^2}\right) \quad (3.7)$$

Our CDF will be equal to the integral from $-\infty$ to x as shown in equation (8).

$$F(x) = \int_{-\infty}^x \sum_{k=1}^K \alpha_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left(-\frac{(t - \mu_k)^2}{2\sigma_k^2}\right) dt \quad (3.8)$$

While this is the CDF, it would be beneficial for us to manipulate this into a more calculable form. By bringing the integral inside the summation and leaving the scaling factor α outside the integral, we are left with equation 9.

$$F(x) = \sum_{k=1}^K \alpha_k \int_{-\infty}^x \frac{1}{\sqrt{2\pi\sigma_k^2}} \cdot \exp\left(-\frac{(t - \mu_k)^2}{2\sigma_k^2}\right) dt \quad (3.9)$$

This leaves us with the weighted sum of the CDF of a standard normal distribution, which can be defined in terms of the error function as shown in equation (10).

$$F(x) = \sum_{k=1}^K \frac{\alpha_k}{2} \left[1 + \operatorname{erf}\left(\frac{x - \mu_k}{\sigma_k \sqrt{2}}\right) \right] \quad (3.10)$$

Defining this in terms of the error function makes calculation much easier since the error function is well-defined and easy to calculate numerically. The error function is defined in equation (11).

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (3.11)$$

The issue is that there is no closed-form solution to the inverse of this CDF. As such, we use a numerical root finder to find what value of x leads to $F(x) - q = 0$ where q is the percentile we sample at. We choose to use Brent's method [Brent \(1973\)](#), however, most numerical root finders should work here. For derivative-based root finders, the derivative of the CDF is equal to the PDF.

Now that we have established a way to perform inverse transform sampling, we can take full advantage of its properties to improve our model. To solve our issue with transient simulations, we can keep the same value of q for the entirety of a sweep. This will result in a continuous output that is more in line with the device's

properties of cycle-to-cycle variation than traditional sampling. In this case, we can generate a new q every time the derivative of an input with respect to time crosses 0, which can easily be done in Verilog-A. Another benefit of this sampling approach is the ability to clip the probability distribution. For example, we could generate q only between 0.05 and 0.95 so that the model doesn't predict values more than 2 standard deviations away (or in any range). Another use could be modeling device-to-device variation by weighting the distribution when sampling for q , though this work doesn't explore this option.

3.2.3 Device Characteristics of Heater Cryotron

The heater cryotron is a superconducting device designed to exploit the unique properties of superconducting nanowires [McCaughan and Berggren \(2014\)](#). This device consists of two superconducting nanowires separated by a dielectric, forming the gate and the channel of the device. Figure 3.2 (a) shows a false-colored scanning electron microscope (SEM) image of the fabricated heater cryotron device, with a channel width of $1\mu m$ and a gate width $450nm$. The gate and channel nanowires are separated by a SiO_2 dielectric spacer of $25nm$ thickness. Initially, for a given channel current (I_B), both nanowires (gate and channel), when maintained at cryogenic temperatures, remain superconducting exhibiting zero resistance, and remain so until the gate nanowire becomes resistive (Figure 3.2 (b)). However, when a sufficient amount of current (I_G) is passed through the gate nanowire to make it resistive, it starts generating thermal phonons which transport to the channel nanowire with the help of the spacer. These thermal phonons suppress the superconductivity and reduce the critical current of the channel nanowire (I_{Ch}^C). Eventually, when the increase of gate current, reduces the channel critical current below the applied channel current, the channel transitions from its superconducting state to a resistive state as shown in Figure 3.2 (c). Conversely, removing the current from the heater enables the channel nanowire to revert to its superconducting state. The specific

Algorithm 1 Standard Sampling from a Gaussian Mixture Density Distribution

Input:

K - Number of Gaussian components
 $\{\mu_1, \mu_2, \dots, \mu_K\}$ - Means of the Gaussians
 $\{\sigma_1, \sigma_2, \dots, \sigma_K\}$ - Stand deviation of the Gaussians
 $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ - Mixing coefficients, $\sum_{i=1}^K \pi_i = 1$

Output:

Sample x from the mixture distribution

Sample Function:

```

sum = 0
Sample  $q$  from a uniform distribution  $[0, 1]$ 
for  $k \in \{1, K\}$  do
    sum = sum +  $\alpha_k$ 
    if sum  $\geq q$  then
        break
end
Sample  $x$  from the Gaussian distribution  $\mathcal{N}(\mu_k, \sigma_k)$ 
  
```

Return:

x - Sample from the Gaussian mixture density

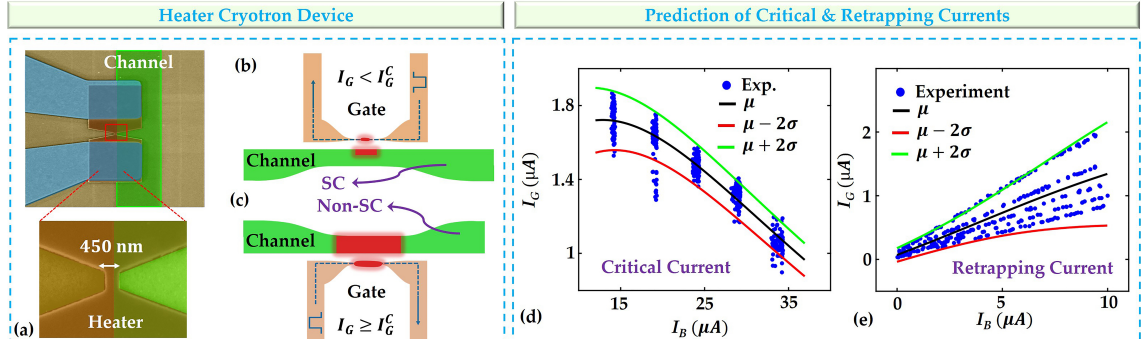


Figure 3.2: Prediction of Switching characteristics of heater cryotron.

(a) A false-colored scanning electron microscope (SEM) image of heater cryotron device consisting two superconducting nanowires of tungsten silicide (WSi) and a dielectric spacer of SiO_2 . (b)-(c) Illustration of the gate-current-controlled switching mechanism. An enough gate current (I_G) switches the gate nanowire to its resistive state and generates thermal phonons which eventually causes the switching of the channel from its superconducting to resistive state. (d) Switching model for critical current evaluated at the mean (μ) and plus or minus 2 standard deviations (σ) of the resultant distribution and (e) retrapping current of heater cryotron switching model evaluated at the same points.

current levels at which superconducting to resistive and resistive to superconducting transitions occur are referred to as the critical current and retrapping current, respectively. The gate current-controlled switching of heater cryotrons has been used in a multitude of applications, including designing logic circuits [McCaughan and Berggren \(2014\)](#); [Alam et al. \(2022a, 2023d,c\)](#), memory systems [Alam et al. \(2023a, 2021, 2022c,b, 2023b\)](#), and neuromorphic systems [Islam et al. \(2023a, 2022a, 2023b, 2022b, 2023c\)](#) for cryogenic environment, and in interfacing superconducting circuits with semiconducting technology [McCaughan et al. \(2019\)](#).

Notably, the point at which the device switches between its superconducting and resistive states is characterized by stochastic behavior, which introduces an inherent element of randomness into its operation. This stochasticity makes this device ideal for testing our MDN-based compact modeling approach. The critical current value is not a fixed constant but instead depends on the applied channel current. For example, a higher bias current leads to a lower critical current for the gate, thus influencing the switching behavior of the device. Conversely, a higher bias current leads to a lower retrapping current. Additionally, a much lower bias current is required to switch from a resistive to a superconducting state since the nanowire produces its own heat when resistive among other factors. Understanding this dependence on the channel current is pivotal in characterizing and modeling the behavior of the heater cryotron.

To sample the characteristics of the device, we sweep the gate at different bias currents. By performing each of the sweeps multiple times and recording the resulting current-voltage characteristics of the device, we can have enough data for the MDN to learn the stochastic switching behavior of the device. This experimental approach allows us to explore the device’s response under varying conditions and analyze the interplay between the gate and channel currents.

3.2.4 Heater Cryotron Model

For our first iteration of the model, we chose to use our MDN to model the critical gate current for a given bias current. This approach allows the use of a simpler neural network structure for our MDN when compared to learning the I-V characteristics directly. To accomplish this, we use bias current and current state of the device as input with the MDN trained to predict gate critical current (if the heater cryotron is in its superconducting state) and retrapping current if the heater cryotron is in its resistive state. The results of this model can be seen in Figure 3.2 (d) and (e), where we show the experimental data as well as our model’s output distribution at the mean (μ), and with a variation of twice the standard deviation (σ) in either side ($\mu - 2\sigma$ and $\mu + 2\sigma$) of the mean. We can see that the model is able to capture the critical and recapture current at different bias currents as well as the variance in these switching points. Another benefit to this approach is that in addition to reduced model complexity, we don’t have to run the model every time step as long as the bias current is constant, making the model even more efficient in certain applications.

While the switching model could work well for some devices, it comes with some major drawbacks. Most notably, it requires separate models for the I-V characteristics of the device in its different states in addition to the switching model. This means that this approach is unable to capture the variance in the device’s I-V characteristics outside of switching. Going forward, we are going to use the MDN to directly learn the I-V characteristics of the heater cryotron. This means that we will be using I_G , I_B , and state as input to the model, and directly predicting V_L as the output.

One challenge with MDNs is that there is no good way to provide easily interpretable numerical metrics for performance. Log-likelihood can be used to compare different models, but it is not a useful metric on its own. Since there are not any existing approaches to compare against, this metric will not be useful for us. Because of this, we will need to rely on graphical comparisons between the model and the experimental data. First, we can look at a comparison between the

distribution from the model vs a histogram of the experimental data for different values of I_G and I_B . This is shown in Figure 3.3, where we are using I_B of $23.5\mu A$ and values of I_G from $1.35\mu A$ to $1.6\mu A$. Using this range of I_G allows us to see how the switching probability distributions change around the critical current. Figure 3.3 shows that our probability distributions closely match the distributions obtained from the experiment.

Next, we can compare the switching probability of the model. To do this, we use a sweep of I_G from $0\mu A$ to $3\mu A$ for I_B values of $14\mu A$, 16.5 , $23.5\mu A$, $28\mu A$, and $33\mu A$. To evaluate the switching probability of the model, we can use the CDF evaluated at the voltage midway between the load voltage when the H-Tron is in superconducting state and when it is in resistive state. Figure 3.4 (left) shows the results of switching probability observed in the experimental data compared to the MDN model. Framing the model in this way allows us to use traditional regression performance metrics, which can be seen in Table 1. On the entire dataset, our model achieves a mean absolute error (MAE) of 0.82% for switching probability with an R^2 of 0.9891. Here, the R^2 value represents the percentage of variance in the switching probability explained by the model. We can compare this to a deterministic neural network such as a multilayer perceptron (MLP), which has been proposed for compact modeling by several previous works [Zhang and Chan \(2017\)](#); [Wang et al. \(2021\)](#); [Hutchins et al. \(2022\)](#). To do this, we can train the model to predict the threshold gate current for a given bias voltage (I_B). Since this model isn't stochastic, we consider the switching probability to be 0% for any value below the threshold current and 100% above. After testing in this way, the model's MAE is 2.61%, a significant drop in performance compared to our probabilistic approach.

Finally, we can evaluate the performance of the model in transient simulation. So that we can match the output of the model with experimental results, we mirror I_G and I_B from the experimental data as shown in Figure 3.4 (right). We use the inverse transform sampling technique as described in section II to achieve the smooth, continuous curves as seen in Figure 3.4 (right). To show the distribution of the model

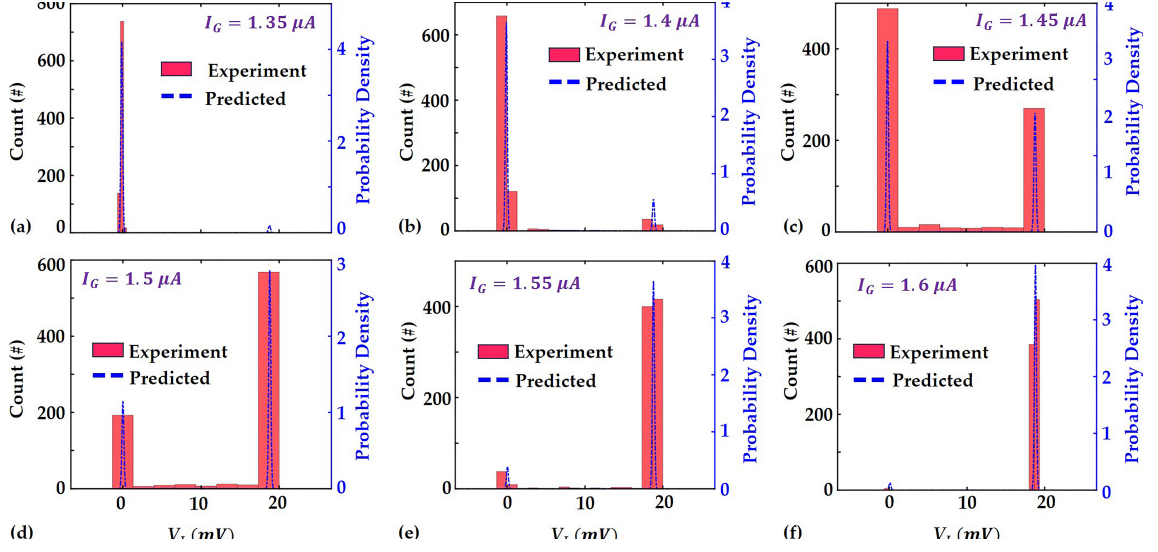


Figure 3.3: Model validation for load voltage (V_L) against the experiment. The predicted vs experimental distributions of V_L (dropped across a $1\text{ k}\Omega$ load resistor) at a bias current of $23.5\text{ }\mu\text{A}$ and gate current of (a) $1.35\text{ }\mu\text{A}$, (b) $1.4\text{ }\mu\text{A}$, (c) $1.45\text{ }\mu\text{A}$, (d) $1.5\text{ }\mu\text{A}$, (e) $1.55\text{ }\mu\text{A}$, and (f) $1.6\text{ }\mu\text{A}$.

Table 3.1: MDN and MLP Performance of Switching Probability Mean Absolute Error (MAE) and R^2

$I_B(\mu\text{A})$	MDN (Ours)		MLP	
	MAE	R^2	MAE	R^2
14	1.149%	0.9941	2.793%	0.9611
16.5	0.534%	0.9986	2.876%	0.9334
23.5	1.344%	0.9572	2.874%	0.9340
28	0.611%	0.9986	2.054%	0.9734
33	0.662%	0.9971	2.457%	0.9538

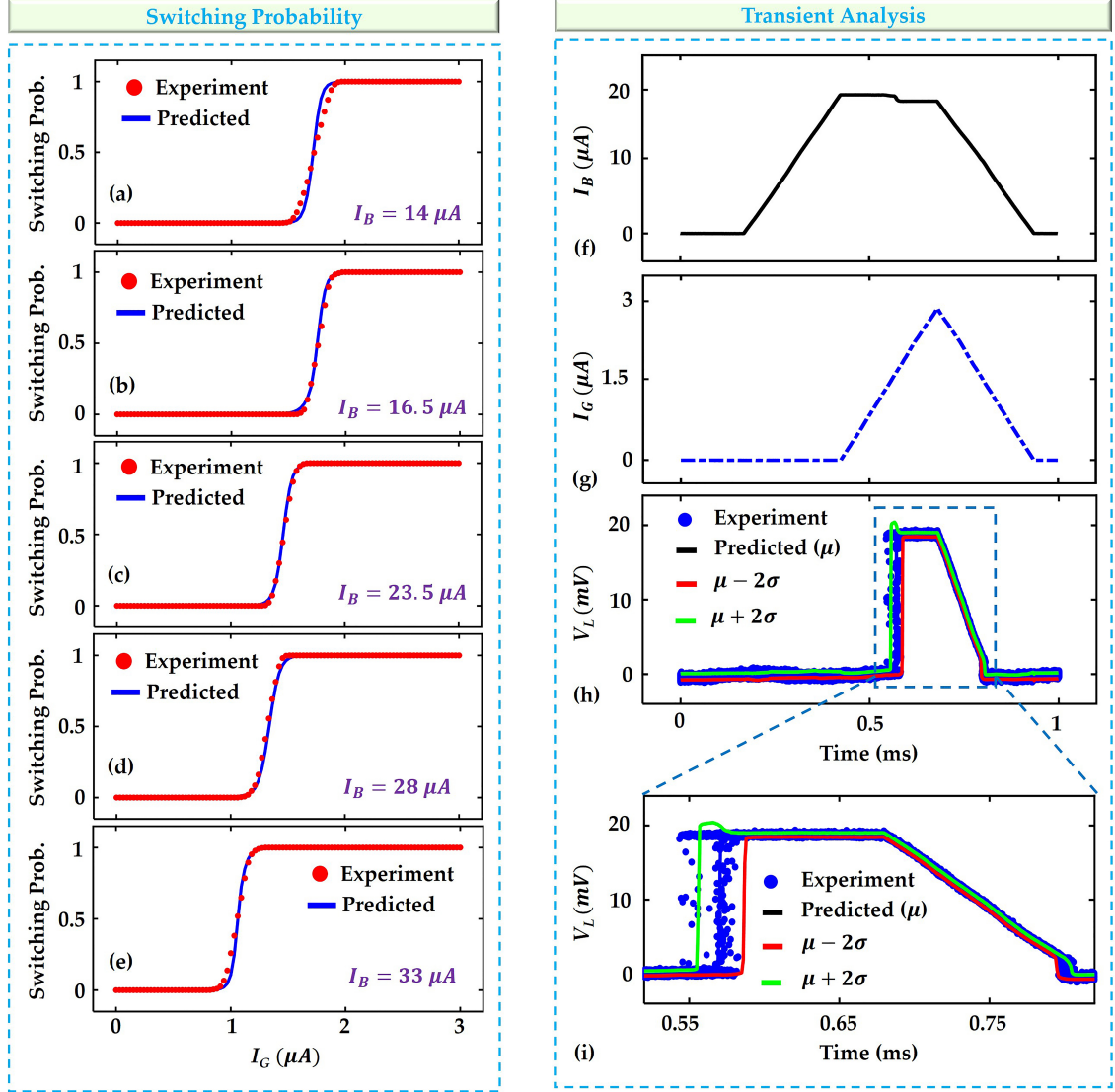


Figure 3.4: Switching probabilities at varying bias currents and transient simulation results. The predicted vs experimental switching probability for gate current (I_G) between $0 \mu A$ and $3 \mu A$ at a bias current (I_B) of (a) $14 \mu A$, (b) $16.5 \mu A$, (c) $23.5 \mu A$, (d) $28 \mu A$, and (e) $33 \mu A$. The increase in I_B leads to a reduction in the gate switching current. Transient dynamics of (f) bias current and (g) gate current which cause switching of the channel after exceeding the corresponding thresholds. (h) Switching is shown and validated with the experiment by the time dynamics of the load voltage (V_L) at mean (μ) and plus or minus 2 standard deviations (σ). (i) A zoomed-in view of V_L shown in (h).

in the transient simulation, we report μ , $\mu - 2\sigma$, and $\mu + 2\sigma$ conditions. We can do this by setting the quantile value (q) to 0.5 for the mean and 0.05 and 0.95 for the inverse transform sampling. Figure 3.4 (right) shows that our MDN model accurately captures the variance in the I-V characteristics of the devices, including the switching dynamics.

3.3 Discussion

Our approach utilizing mixture density networks has demonstrated its remarkable ability to accurately model the variance of I-V characteristics and stochastic switching behavior of heater cryotrons. With a 0.82% mean absolute error in the switching probability and an R^2 value of 0.9891, our method has proven its efficacy in capturing the true variance of device behavior. While we’ve only shown the model using heater cryotrons, the generalizing power of neural networks means this approach will be easily adaptable to other devices. This achievement marks a significant stride towards more precise and reliable electronic circuit simulations, offering unprecedented opportunities for the development of cutting-edge technologies in an era where stochasticity plays an increasingly pivotal role.

3.4 Methods

3.4.1 Heater Cryotron Fabrication

The fabrication process began with the deposition of a 4 *nm* layer of superconducting tungsten silicide (WSi) onto a silicon wafer through magnetron sputtering. Following this, contact pads consisting of 5 *nm* Ti and 30 *nm* Au were applied using optical lithography and a liftoff procedure. Subsequently, electron beam lithography was employed to define and etch the WSi layer in an RIE process with SF6 chemistry to create the device’s channel. A 25 *nm* layer of SiO₂ was then sputter-deposited

across the entire wafer to serve as the dielectric spacer separating the heater from the channel. Another 4 nm layer of WSi was subsequently sputter-deposited, patterned, and etched to form the heater input gate, with contact pads added to this upper layer through the same liftoff process as the initial layer. Finally, openings were etched through the SiO₂ dielectric layer using an RIE process with CHF₃:O₂ chemistry to establish electrical contact with the underlying channel layer.

3.4.2 Heater Cryotron Characterization

The measurements in this study employed an arbitrary waveform generator (AWG) equipped with two channels and incorporated 10 k Ω resistors in series with each channel. In the data acquisition process, one channel of the AWG was configured to maintain a constant voltage, thereby delivering a fixed bias current, while the other channel was programmed to incrementally ramp up from zero current to 3 μA . Simultaneously, the voltage of the device channel was meticulously recorded on an oscilloscope. The experimental conditions covered six distinct bias current levels, spanning from 14 μA to 33 μA . Following the ramp-up of gate current, a subsequent phase involved the concurrent reduction of gate and bias currents to get a larger breadth of data. This entire procedure was repeated 1000 times for each bias current setting, ensuring the capture of the stochastic nature of the device.

3.4.3 Model Creation

We use Python with TensorFlow to create the mixture density network. We train the model using the data derived in the previous section. We implement a custom loss function to optimize the model for minimizing Gaussian negative log-likelihood. In addition, we implement a custom activation function for exponential linear unit plus 1 plus epsilon as described in equation (3). To integrate this model into circuit simulation, we translate the neural network architecture and the inverse transform sampling methodology into Verilog-A. Subsequently, the trained model weights are

extracted using a Python script and seamlessly integrated into the Verilog-A model. This finalized Verilog-A model is compatible with various circuit simulators that support Verilog-A models, such as HSPICE or Spectre. This process is derived from Hutchins et al. [Hutchins et al. \(2022\)](#).

Chapter 4

Compact Modeling using Deep Symbolic Regression

4.1 Introduction

In recent years, the relentless march of technological advancement has continuously pressed the boundaries of electronic device complexity and functionality. This inexorable progress demands equally sophisticated tools for modeling and simulation, which are essential for the design, optimization, and analysis of electronic devices. Traditional neural network models, particularly those based on complex neural networks, encounter significant limitations due to their computational complexity and lack of interpretability [Lin et al. \(2020\)](#). This is particularly problematic when integrating with circuit simulation environments via Verilog-A, which is not well-suited to handle the computational and memory overheads of large neural networks [McAndrew et al. \(2015\)](#).

In this section, we introduce a novel approach that utilizes transformer-based deep symbolic regression (DSR) to generate compact, equation-based models of electronic devices [Makke and Chawla \(2024\)](#). Unlike conventional neural network modeling that results in a "black box" model, deep symbolic regression aims to discover the

underlying mathematical expressions that best describe the device characteristics. This method utilizes the power of transformer architectures, popular for their success in large language models such as ChatGPT, to extract meaningful patterns and relationships from data Vaswani et al. (2023). The result is a simple, interpretable equation that captures the essential physics of the device’s operation.

The adoption of DSR for electronic device modeling offers significant advantages over traditional neural network approaches. Firstly, the resultant mathematical models are inherently compact, reducing the computational burden on simulation environments. This is especially critical for circuit simulation, where the inclusion of complex models can severely degrade simulation performance. By distilling the device behavior down to a simple equation, we circumvent the limitations imposed by Verilog-A’s inability to efficiently handle large neural networks. Furthermore, the transformer-based DSR models retain high accuracy to the device characteristics, ensuring that the simulations remain both fast and reliable. Moreover, the interpretability of the models generated through deep symbolic regression provides additional value. Unlike opaque neural network models, the equations produced allow designers and engineers to gain insights into the fundamental behaviors and dependencies within the device. This can facilitate a deeper understanding of the device’s physics, potentially unveiling opportunities for optimization and innovation that were not apparent before.

In the following sections, we detail the methodology of transformer-based deep symbolic regression and its integration into electronic device modeling. Through extensive simulations and case studies, we demonstrate the ability of DSR to create efficient, accurate, and interpretable models for a wide range of electronic devices. This work not only presents a significant leap forward in modeling techniques but also underscores the potential of advanced machine learning methods to revolutionize electronic device design and simulation.

4.2 Deep Symbolic Regression

To create our compact model for circuit simulation, we choose to use symbolic regression. This is in contrast to regression with neural networks, where we train a large system of neurons to learn the underlying equation present in the data. While neural networks can learn any non-linear function [Hornik et al. \(1989\)](#), they do come with some serious disadvantages. To create an accurate model, often large networks with thousands or tens of thousands of weights are required. For a compact model that needs to be run repeatedly, the large size of the model means that circuit simulations will take much longer than with traditional compact models, and for large simulations, they may not be viable. For this reason, it would be much better if we could use traditional equations for our model while also harnessing the function-learning capability of deep learning. To achieve this, we chose to use symbolic regression. Symbolic regression is a form of regression analysis where the goal is to identify a mathematical expression that best fits a given set of data. Traditionally, symbolic regression relied heavily on genetic programming and other evolutionary algorithms. However, this type of modeling has been revolutionized by leveraging the same powerful model behind popular large language models like ChatGPT—transformers.

The key advantage of transformers in symbolic regression is their ability to handle complex patterns and relationships in data, thanks to their attention mechanisms. This allows them to better capture the underlying structure of the mathematical expressions that describe the data. For example, SymFormer predicts formulas by simultaneously outputting individual symbols and their corresponding constants, offering more accurate fits to the data [Vastl et al. \(2022\)](#).

Furthermore, transformers can be pre-trained on a vast array of equations, enabling them to generalize better to unseen data and discover well-known physical equations more reliably than previous methods. This pre-training aspect significantly reduces the time and computational resources required for symbolic regression tasks,

making transformers a more practical choice for scientific discovery and industrial applications.

For creating compact models, these advantages mean that this type of approach is perfect. We chose to use the model from Kamienny et al., which utilizes the transformer architecture to create an end-to-end framework for symbolic regression [Kamienny et al. \(2022\)](#). By using this work, we can leverage their pre-trained model to create our compact models and avoid the time-consuming and computationally expensive training process. Figure 4.1 shows an overview of the inference process.

To understand how this model works, we first have to discuss the preprocessing of the data. Once we have obtained the data from atomistic simulation, the data is normalized and tokenized. The tokenization process reduces the values to 4 significant digits and then converts them to floating point notation. Here, each number in the input converts to 3 tokens: one for the sign, one for the mantissa, and one for the exponent. Once tokenization is complete, a feedforward network is used to create unique vector representations for each token.

Next, the embedded data is taken to the encoder. The encoder’s primary role is to process the input sequence and map it into a continuous representation that captures the contextual relationships between the input values. Each encoder layer within the transformer model performs two main operations. First is the self-attention layer, which allows the encoder to weigh the importance of different data points (or tokens) in the input sequence relative to each other. The self-attention mechanism calculates attention scores to determine how much focus should be given to other parts of the input sequence when encoding a particular data point. This process ensures that the encoder captures the context of each data point within the entire input sequence. After the self-attention mechanism, the encoder passes its output through a feed-forward neural network. This network is applied identically to each token separately and independently. The purpose here is to transform the attention-augmented representations further, allowing for complex relationships between different parts of the input sequence to be captured. Each encoder layer outputs a high-dimensional

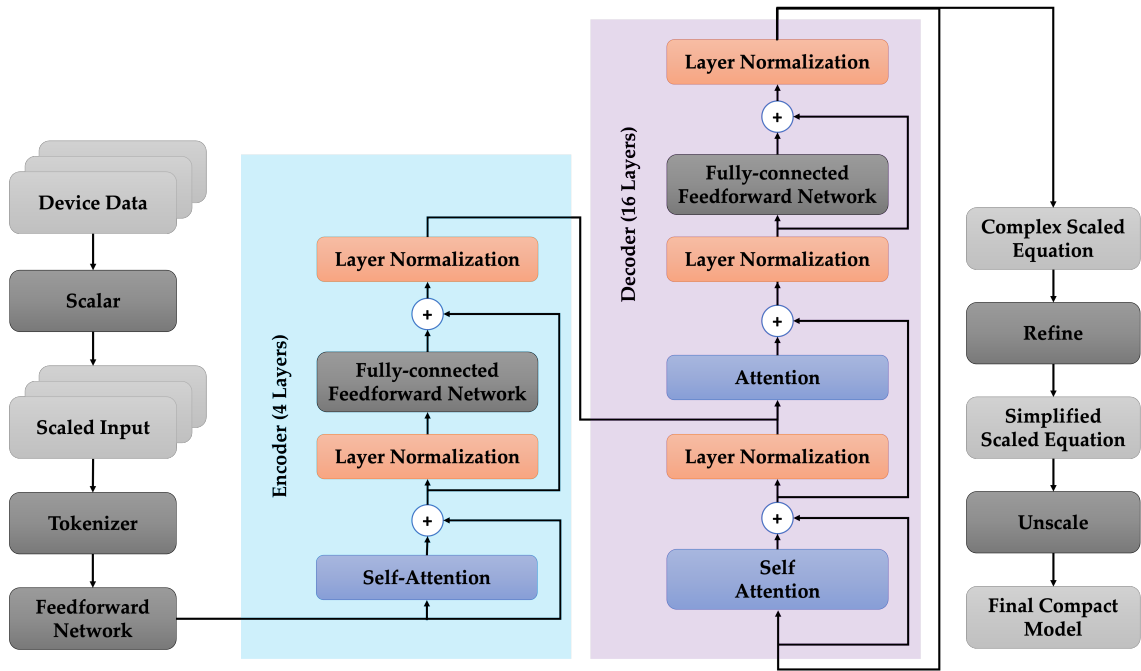


Figure 4.1: Overview of our modeling approach.

vector for each word in the input sequence. These vectors carry the contextual information necessary for the decoder to generate the target equation.

The decoder is designed to generate the equation based on the encoded input. The decoder generates the equation one token at a time, leveraging both the contextual information provided by the encoder and its own previously generated tokens to inform each subsequent token’s generation. It also consists of multiple layers, each of which has three core components. The first is the self-attention mechanism, similar to the one in the encoder. The difference this time is that instead of handling the data, it will make connections between the tokens it has generated for the equation so far. After the self-attention layer, the decoder includes an encoder-decoder attention layer. This layer allows the decoder to focus on different parts of the input sequence, as represented by the encoder’s output. It’s crucial for aligning the decoder’s generated equation with the input data, enabling the model to consider the entire input context when generating each token of the equation. Similar to the encoder, each decoder layer also includes a feed-forward neural network. This network processes the output from the encoder-decoder attention layer, further refining the generated sequence.

Once the entire equation is generated by the decoder, it is then further refined via a BFGS [Liu and Nocedal \(1989\)](#). Finally, constants can be rescaled to match the input before scaling. Once we have this equation, we can then implement it in a Verilog-A model to use it in circuit simulation.

4.3 Results

To test our approach, we choose to use a graphene nanoribbon field effect transistor (GRNFET) [Lone et al. \(2021\)](#). These are a type of transistor that utilizes graphene nanoribbons as the channel material through which current flows. Graphene, a single layer of carbon atoms arranged in a hexagonal lattice, is renowned for its exceptional electrical and thermal properties, making it a highly attractive material for electronic devices.

A GNRFET operates similarly to other field-effect transistors but with some key advantages due to the properties of graphene. These advantages include higher electron mobility, which can lead to faster switching speeds, and the potential for scaling down to very small dimensions while maintaining performance, which is critical for the continued advancement of electronic devices according to Moore’s Law.

After fitting the model, we test its performance compared to the data as shown in Figure 4.2. Here, we see that the equation accurately models the output characteristics of the GRNFET. We also tested the R^2 for our equation compared with the data, which was 0.9978. This means that 99.78% of the variance in the output is explained by the model, which means it is highly accurate. As a point of comparison, we also tested how the approach described in Chapter 2 would perform. Upon running this model, we achieved an R^2 of 0.9988. While the neural network approach performed very slightly better, it also requires several thousand times more operations per inference of the model.

Finally, we can test how our model performs in circuit simulation. To do this, we implement our model in Verilog-A so that it can be used in a circuit simulator, Next, we created an inverter using GNRFETs as shown in Figure 4.3 (b) using HSPICE. We can see in Figure 4.3 (a) that our model effectively captures the behavior of the GRNFET, and allows us to get accurate simulation results for the inverter. However, the inverter is unable to achieve full rail-to-rail swing like a CMOS inverter due the limitations of GRNFETs. Regardless, this simulation result demonstrates the efficacy of our compact modeling approach.

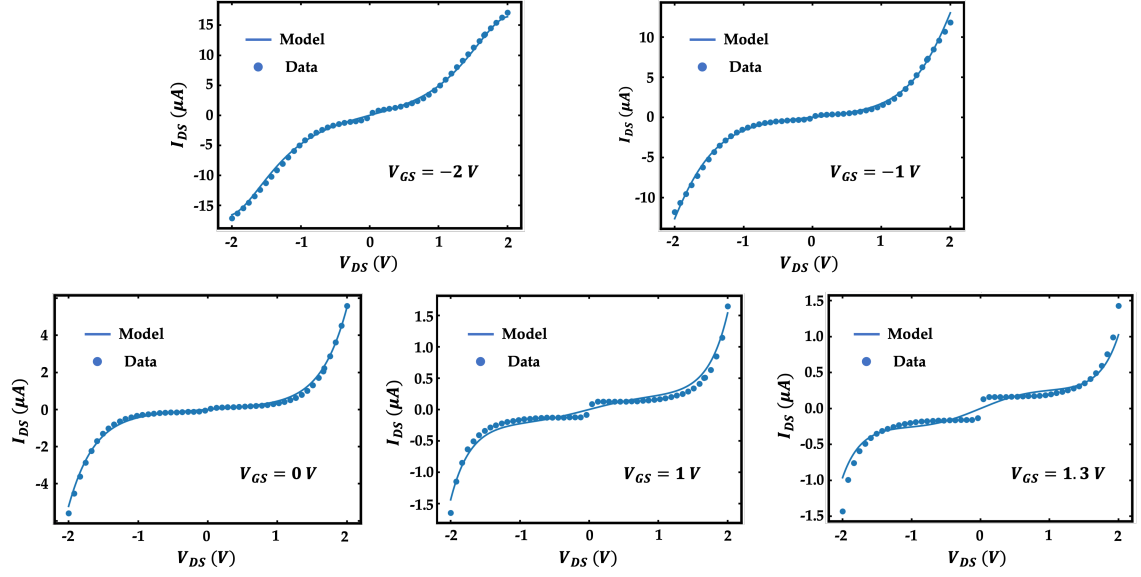


Figure 4.2: Performance of model compared to data for output characteristics.

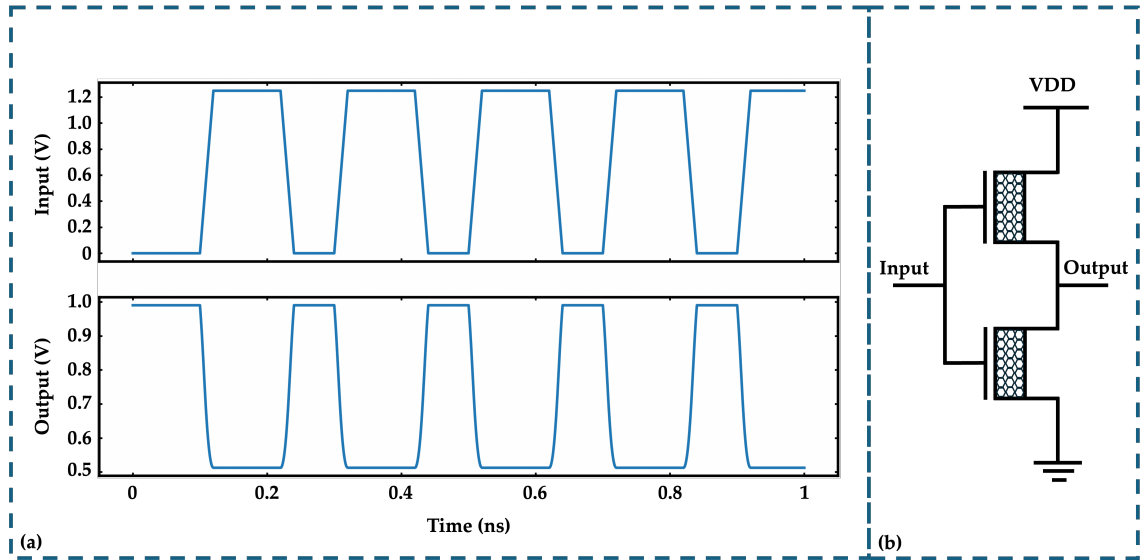


Figure 4.3: Inverter simulation using our GRNFET model (a) and GNRFET inverter diagram (b).

Chapter 5

Conclusions

This thesis has introduced several innovative methods of compact modeling utilizing several deep learning techniques. Our first approach was a generalized framework to create deep learning-powered compact models using multilayer perceptrons. This approach allows us to create compact models for devices without having an understanding of the underlying device physics. Unlike previous approaches that use machine learning for compact modeling, this approach is designed to be as general as possible and work for both single-state and multi-state devices. To facilitate this, we implemented standardized data processing techniques and automatic hyperparameter optimization. Each of these shows significant improvement over previous work while also making the framework more accepting of a broad range of data without the need for manual tuning.

Furthermore, we have explored the potential of mixture density networks for compact modeling, showcasing our ability to capture the stochastic behavior of electronic devices at an unprecedented level of detail. This deep learning approach opens new avenues for circuit design by accurately modeling cycle-to-cycle and device-to-device variations, thereby facilitating more efficient and reliable testing. The stochastic models we developed pave the way for innovative circuit designs that

exploit the inherent stochastic nature of electronic devices, such as probabilistic, demonstrating the transformative potential of deep learning in this field.

Finally, we created a framework that leverages symbolic regression to create compact models. Specifically, we utilized a deep symbolic regression model that uses a transformer architecture to learn an in-depth map of the relationships in the data. This method represents a novel application of generative AI, distinguishing itself from conventional regression-based approaches by producing compact models in the form of simple equations. This approach not only simplifies integration into circuit simulation software but also enhances explainability and operational efficiency. Symbolic regression models require orders of magnitude fewer computational operations for inference compared to traditional deep learning models, offering a more viable solution for large-scale simulations.

5.1 Challenges and Limitations

While deep learning-based compact modeling presents several advantages over traditional physics-based models and lookup table-based approaches, it is important to recognize that they have some limitations. The requirement for large datasets for training and the computational demands for training are notable considerations. However, the models proposed in this work are designed to be relatively compact, allowing for efficient training on modest hardware.

Finally, the biggest limitation of deep learning-powered compact modeling is the complexity of the models. Even though the neural networks these models use are relatively small, they still have thousands, if not tens of thousands of parameters. This means that each time we inference the model it has to do a massive amount of operations, especially compared to traditional compact modeling approaches. This problem is compounded by the limitations of existing circuit simulation software, which is not designed to run models this large. Specifically, Verilog-A automatically calculates the derivative of a model, which is fine for a simple equation, but calculating

the derivative of a neural network is much more of a complex task than Verilog-A is designed to handle. This was the motivation for the symbolic regression approach described in Chapter 4, but that approach has its own challenges, such as reduced accuracy. This being said it is possible to implement derivatives manually in most circuit simulation software, so with more research it should be possible to mitigate this issue for any neural network-based modeling approach.

5.2 Future Work

While this thesis tries to offer a wide range of approaches for deep learning-powered compact modeling, there is still much work to be done in this field. For example, implementing models with explicitly defined derivatives as discussed in the previous section could go a long way to making neural network approaches more viable for large-scale circuit simulation. In the same vein as creating more efficient models, there is much more work to be done in symbolic regression approaches. For our approach, we used a pre-trained symbolic regression model trained on a wide variety of different equations. Since this model is so similar to a large language model, we can borrow techniques from that domain, such as fine-tuning. This is the process of taking existing models trained on a wide variety of problems, and training them further on a smaller subset of problems, in our case compact models. This would undoubtedly give us better results, but it would be a massive project that would require an equally massive dataset.

Additionally, other neural network architectures would offer benefits over traditional approaches or the architectures proposed here. For example, this thesis has not explored sequence modeling-based approaches to compact modeling that could be implemented with RNNs, LSTMs, or transformers. These types of networks could be useful for devices that rely on timing and the sequence of previous inputs, such as ferroelectric devices. An approach like this could model partial polarization in a ferroelectric capacitor, which no model we have discussed here is capable of doing.

This is just one example, but several other neural network architectures may offer advantages for compact modeling that are worth exploring in the future.

5.3 Final Thoughts

This thesis introduces significant advancements in compact modeling by applying deep learning techniques. We developed a generalized framework for deep learning-powered compact models and explored various approaches, including multilayer perceptrons, mixture density networks, and symbolic regression. These methods improve the efficiency and accuracy of modeling a wide range of electronic devices without relying on traditional modeling constraints.

However, our work also highlights several challenges, such as the high data and computational requirements of deep learning models and the difficulty of integrating these models with current simulation tools. Addressing these challenges will be crucial for future research, which should focus on optimizing model efficiency and exploring new neural network architectures to cover more complex device behaviors.

The integration of deep learning with compact modeling offers a promising path forward for enhancing circuit design and simulation. There's a lot more to explore and improve in this area, and our work lays the groundwork for future advancements in the field.

Bibliography

- Aadithya, K. V., Kuberry, P. A., Paskaleva, B. S., Bochev, P. B., Leeson, K. M., Mar, A., Mei, T., and Keiter, E. R. (2020). Data-driven compact models for circuit design and analysis. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States). [7](#)
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*. [9](#)
- Abouelyazid, M. S., Hammouda, S., and Ismail, Y. (2022). Fast and accurate machine learning compact models for interconnect parasitic capacitances considering systematic process variations. *IEEE Access*, 10:7533–7553. [22](#)
- Advani, M. S. and Saxe, A. M. (2017). High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428 – 446. [5](#)
- Alam, S., Hossain, M. S., and Aziz, A. (2021). A cryogenic memory array based on superconducting memristors. *Applied Physics Letters*, 119(8). [32](#)
- Alam, S., Hossain, M. S., Ni, K., Narayanan, V., and Aziz, A. (2022a). Voltage-controlled cryogenic boolean logic family based on ferroelectric squid. *arXiv preprint arXiv:2212.08202*. [32](#)

- Alam, S., Hossain, M. S., Srinivasa, S. R., and Aziz, A. (2023a). Cryogenic memory technologies. *Nature Electronics*, 6(3):185–198. [32](#)
- Alam, S., Hutchins, J., Hossain, M. S., Ni, K., Narayanan, V., and Aziz, A. (2023b). Cryogenic in-memory matrix-vector multiplication using ferroelectric superconducting quantum interference device (fe-squid). In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE. [32](#)
- Alam, S., Islam, M. M., Hossain, M. S., and Aziz, A. (2022b). Superconducting josephson junction fet-based cryogenic voltage sense amplifier. In *2022 Device Research Conference (DRC)*, pages 1–2. IEEE. [32](#)
- Alam, S., Islam, M. M., Hossain, M. S., Ni, K., Narayanan, V., and Aziz, A. (2022c). Cryogenic memory array based on ferroelectric squid and heater cryotron. In *2022 Device Research Conference (DRC)*, pages 1–2. IEEE. [32](#)
- Alam, S., McCaughan, A. N., and Aziz, A. (2023c). Reconfigurable superconducting logic using multi-gate switching of a nano-cryotron. In *2023 Device Research Conference (DRC)*, pages 1–2. IEEE. [32](#)
- Alam, S., Rampini, D. S., Oripov, B. G., McCaughan, A. N., and Aziz, A. (2023d). Cryogenic reconfigurable logic with superconducting heater cryotron: Enhancing area efficiency and enabling camouflaged processors. *Applied Physics Letters*, 123(15). [32](#)
- Alawad, M. and Lin, M. (2016). Survey of stochastic-based computation paradigms. *IEEE Transactions on Emerging Topics in Computing*, 7(1):98–114. [22](#)
- Aziz, A. (2019). *Device-circuit co-design employing phase transition materials for low power electronics*. PhD thesis, Purdue University. [7](#)
- Aziz, A., Jao, N., Datta, S., and Gupta, S. K. (2016). Analysis of functional oxide based selectors for cross-point memories. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(12):2222–2235. [20](#)

- Becele, E., Talatchian, P., Prenat, G., Anghel, L., and Prejbeanu, I.-L. (2021). Fast behavioral veriloga compact model for stochastic mtj. pages 259–262. IEEE. 23
- Bishop, C. (1994). Mixture density networks. Workingpaper, Aston University. 23
- Bourenkov, V., McCarthy, K., and Mathewson, A. (2005). Mos table models for circuit simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24:352–362. 5
- Brent, R. P. (1973). *Chapter 4: An Algorithm with Guaranteed Convergence for Finding a Zero of a Function*. Prentice-Hall. 29
- Chiozzi, D., Bernardoni, M., Delmonte, N., and Cova, P. (2018). A neural network based approach to simulate electrothermal device interaction in spice environment. *IEEE Transactions on Power Electronics*, 34(5):4703–4710. 7
- Chollet, F. et al. (2015). Keras. <https://keras.io>. 9
- Cressie, N. and Whitford, H. (1986). How to use the two sample t-test. *Biometrical Journal*, 28(2):131–148. 15
- Dasgupta, A., Parihar, S. S., Kushwaha, P., Agarwal, H., Kao, M., Salahuddin, S., Chauhan, Y., and Hu, C. (2020). Bsim compact model of quantum confinement in advanced nanosheet fets. *IEEE Transactions on Electron Devices*, 67:730–737. 2
- Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer New York. 23, 28
- Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327. 14
- Fouda, M. E., Khatib, M. A., Mosad, A. G., and Radwan, A. G. (2013). Generalized analysis of symmetric and asymmetric memristive two-gate relaxation oscillators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(10):2701–2708. 20

- George, S., Aziz, A., Li, X., Kim, M. S., Datta, S., Sampson, J., Gupta, S., and Narayanan, V. (2016). Device circuit co design of fefet based logic for low voltage processors. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 649–654. IEEE. [7](#)
- Gildenblat, G. (2010). Compact modeling. *Netherlands: Springer*. [1](#)
- Goh, A. (1995). Back-propagation neural networks for modeling complex systems. *Artif. Intell. Eng.*, 9:143–151. [5](#)
- Hamilton, T. J., Afshar, S., van Schaik, A., and Tapson, J. (2014). Stochastic electronics: A neuro-inspired design paradigm for integrated circuits. *Proceedings of the IEEE*, 102(5):843–859. [22](#)
- Hoffmann, M., Wang, Z., Tasneem, N., Zubair, A., Ravindran, P. V., Tian, M., Gaskell, A. A., Triyoso, D., Consiglio, S., Tapily, K., et al. (2022). Antiferroelectric negative capacitance from a structural phase transition in zirconia. *Nature communications*, 13(1):1228. [7](#)
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366. [9](#), [24](#), [42](#)
- Hutchins, J., Alam, S., Zeumault, A., Beckmann, K., Cady, N., Rose, G., and Aziz, A. (2022). A generalized workflow for creating machine learning-powered compact models for multi-state devices. *IEEE Access*. [22](#), [34](#), [39](#)
- Islam, M. M., Alam, S., Hossain, M. S., and Aziz, A. (2022a). Dynamically reconfigurable cryogenic spiking neuron based on superconducting memristor. In *2022 IEEE 22nd International Conference on Nanotechnology (NANO)*, pages 307–310. IEEE. [32](#)
- Islam, M. M., Alam, S., Hossain, M. S., Roy, K., and Aziz, A. (2023a). A review of cryogenic neuromorphic hardware. *Journal of Applied Physics*, 133(7). [32](#)

- Islam, M. M., Alam, S., Schuman, C. D., Hossain, M. S., and Aziz, A. (2023b). A deep dive into the design space of a dynamically reconfigurable cryogenic spiking neuron. *IEEE Transactions on Nanotechnology*. [32](#)
- Islam, M. M., Alam, S., Shukla, N., and Aziz, A. (2022b). Design space analysis of superconducting nanowire-based cryogenic oscillators. In *2022 Device Research Conference (DRC)*, pages 1–2. IEEE. [32](#)
- Islam, M. M., Alam, S., Udo, M. R. I., Hossain, M. S., and Aziz, A. (2023c). A cryogenic artificial synapse based on superconducting memristor. In *Proceedings of the Great Lakes Symposium on VLSI 2023*, pages 143–148. [32](#)
- Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pages 240–248. PMLR. [12](#)
- Kam, H. (2021). Deep learning assisted compact modeling of nanoscale transistor. *arXiv preprint arXiv:2107.06167*. [7](#)
- Kamienny, P.-A., d’Ascoli, S., Lample, G., and Charton, F. (2022). End-to-end symbolic regression with transformers. [43](#)
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52. [9](#)
- Li, M., Irsoy, O., Cardie, C., and Xing, H. G. (2016a). Physics-inspired neural networks for efficient device compact modeling. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2:44–49. [7](#), [8](#)
- Li, M., Irsoy, O., Cardie, C., and Xing, H. G. (2016b). Physics-inspired neural networks for efficient device compact modeling. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2:44–49. [22](#)

- Lin, A. S., Liu, P.-N., Pratik, S., Yang, Z.-K., Rawat, T., and Tseng, T.-Y. (2022). Rram compact modeling using physics and machine learning hybridization. *IEEE Transactions on Electron Devices*, 69:1835–1841. [7](#), [8](#), [22](#)
- Lin, A. S., Pratik, S., Ota, J., Rawat, T. S., Huang, T.-H., Hsu, C.-L., Su, W.-M., and Tseng, T.-Y. (2020). A process-aware memory compact-device model using long-short term memory. *IEEE Access*, 9:3126–3139. [7](#), [8](#), [40](#)
- Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528. [45](#)
- Lone, S., Bhardwaj, A., Pandit, A. K., Gupta, S., and Mahajan, S. (2021). A review of graphene nanoribbon field-effect transistor structures. *Journal of Electronic Materials*, 50:3169–3186. [45](#)
- Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. [26](#)
- Lundstrom, M. (2003). Moore’s law forever? *Science*, 299(5604):210–211. [22](#)
- Makke, N. and Chawla, S. (2024). Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2. [40](#)
- McAndrew, C. C., Coram, G. J., Gullapalli, K. K., Jones, J. R., Nagel, L. W., Roy, A. S., Roychowdhury, J., Scholten, A. J., Smit, G. D., Wang, X., et al. (2015). Best practices for compact modeling in verilog-a. *IEEE Journal of the Electron Devices Society*, 3(5):383–396. [3](#), [12](#), [40](#)
- McCaughan, A. N. and Berggren, K. K. (2014). A superconducting-nanowire three-terminal electrothermal device. *Nano letters*, 14(10):5748–5753. [23](#), [30](#), [32](#)
- McCaughan, A. N., Verma, V. B., Buckley, S. M., Allmaras, J., Kozorezov, A., Tait, A., Nam, S., and Shainline, J. (2019). A superconducting thermal switch with ultrahigh impedance for interfacing superconductors to semiconductors. *Nature electronics*, 2(10):451–456. [32](#)

- O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). Kerastuner. <https://github.com/keras-team/keras-tuner>. 12, 15
- Salahuddin, S., Ni, K., and Datta, S. (2018). The era of hyper-scaling in electronics. *Nature Electronics*, 1(8):442–450. 22
- Shukla, N., Grisafe, B., Ghosh, R., Jao, N., Aziz, A., Frougier, J., Jerry, M., Sonde, S., Rouvimov, S., Orlova, T., et al. (2016). Ag/hfo 2 based threshold switch with extreme non-linearity for unipolar cross-point memory and steep-slope phase-fets. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 34–6. IEEE. 7
- Tsu, R. (2001). Challenges in nanoelectronics. *Nanotechnology*, 12:625–628. 2
- Vastl, M., Kulhánek, J., Kubalík, J., Derner, E., and Babuška, R. (2022). Symformer: End-to-end symbolic regression using transformer-based architecture. 42
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need. 41
- Wang, J., Kim, Y.-H., Ryu, J., Jeong, C., Choi, W., and Kim, D. (2021). Artificial neural network-based compact modeling methodology for advanced transistors. *IEEE Transactions on Electron Devices*, 68:1318–1325. 7, 8, 22, 34
- Wang, Y., Zhang, Y., Deng, E., Klein, J., Naviner, L., and Zhao, W. (2014). Compact model of magnetic tunnel junction with stochastic spin transfer torque switching for reliability analyses. *Microelectronics Reliability*, 54:1774–1778. 23
- Xia, Q. and Yang, J. J. (2019). Memristive crossbar arrays for brain-inspired computing. *Nature materials*, 18(4):309–323. 20
- Zeumault, A., Alam, S., Omar Faruk, M., and Aziz, A. (2022). Memristor compact model with oxygen vacancy concentrations as state variables. *Journal of Applied Physics*, 131(12). 7, 14

- Zeumault, A., Alam, S., Wood, Z., Weiss, R. J., Aziz, A., and Rose, G. S. (2021). Tcad modeling of resistive-switching of hfo2 memristors: Efficient device-circuit co-design for neuromorphic systems. *Frontiers in Nanotechnology*, 3:734121. [9](#), [12](#), [14](#)
- Zhang, L. and Chan, M. (2017). Artificial neural network design for compact modeling of generic transistors. *Journal of Computational Electronics*, 16:825–832. [7](#), [8](#), [22](#), [34](#)

Vita

Jack Hutchins is a Master's candidate in the department Electrical Engineering and Computer Science at the University of Tennessee. He received his B.S. degree in computer science from the University of Tennessee, Knoxville, in 2022. During his undergraduate studies, he interned at Oak Ridge National Laboratory (ORNL), where he worked on machine learning techniques to detect power-related issues with the Titan Supercomputer. In 2021, he began working as an Undergraduate Research Assistant with the Nanoelectronic Devices and Integrated Circuits (NorDIC) Laboratory, where he continued to work during his graduate studies. He has been received several awards for his research, including being honored as a Design Automation Conference (DAC) Young Fellow (2023) and received the University of Tennessee Student Faculty Research Award along with his advisor Dr. Ahmedullah Aziz (2023). He was also honored as a University of Tennessee Milton Scholar during his undergrad, which allowed him to participate in research utilizing refinement learning for high frequency stock trading and dynamic price modeling.