Proceedings of Machine Learning Research vol 107:555–572, 2020

Data-driven Compact Models for Circuit Design and Analysis

K. Aadithya[1] KVAADIT@SANDIA.GOV
P. Kuberry[2] PAKUBER@SANDIA.GOV
B. Paskaleva[1] BSPASKA@SANDIA.GOV
P. Bochev[2] PBBOCHE@SANDIA.GOV
K. Leeson[1] KLEESON@SANDIA.GOV
A. Mar[3] AMAR@SANDIA.GOV
T. Mei[1] TMEI@SANDIA.GOV
E. Keiter[1] ERKEITE@SANDIA.GOV

[1]*Radiation and Electrical Sciences (Org. 1300), Sandia National Laboratories*

[2]*Center for Computing Research (Org. 1400), Sandia National Laboratories*

[3]*Integrated Military Systems Development (Org. 5400), Sandia National Laboratories*

Abstract

Compact semiconductor device models are essential for efficiently designing and analyzing large circuits. However, traditional compact model development requires a large amount of manual effort and can span many years. Moreover, inclusion of new physics (*e.g.*, radiation effects) into an existing model is not trivial and may require redevelopment from scratch. Machine Learning (ML) techniques have the potential to automate and significantly speed up the development of compact models. In addition, ML provides a range of modeling options that can be used to develop hierarchies of compact models tailored to specific circuit design stages. In this paper, we explore three such options: (1) table-based interpolation, (2) Generalized Moving Least-Squares, and (3) feed-forward Deep Neural Networks, to develop compact models for a p-n junction diode. We evaluate the performance of these "data-driven" compact models by (1) comparing their voltage-current characteristics against laboratory data, and (2) building a bridge rectifier circuit using these devices, predicting the circuit's behavior using SPICE-like circuit simulations, and then comparing these predictions against laboratory measurements of the same circuit.

Keywords: Compact model, p-n junction diode, 1N4148 switching diode, circuit simulation, cubic splines, generalized moving least-squares, deep neural networks, SPICE.

1. Introduction

Circuit simulation, sometimes referred to as SPICE simulation, is foundational to modern circuit design (Nagel, 1975). In circuit simulation, so-called "compact models" are used to capture the dynamics of voltages, currents, and charges in individual circuit components (*e.g.*, diodes). Given a circuit composed of many such components, a circuit simulator combines the compact models of the individual components to enforce Kirchhoff's voltage and current laws across the network. This is done by building a non-linear system of Differential-Algebraic Equations (DAEs); each equation in this system is of the form $a + b + c + \ldots = 0$, where the individual terms (a, b, c, $\ldots$) are provided by compact models. The circuit simulator then numerically solves the system of equations using time stepping algorithms and non-linear solvers; see, *e.g.*, (Keiter et al., 2019a,b).

As modern circuits can easily have many thousands of components (leading to DAE systems of similar size), it is important that each individual compact model be computationally inexpensive. In practice, typical compact models consist of only a handful of algebraic and differential equations – a combination of empirical formulas and simplified solutions to semiconductor transport equations.

Remark 1 *In addition to compact models, there also exist "first-principles", or TCAD (Technology Computer-Aided Design), semiconductor device models that offer much greater accuracy over a wide range of operating conditions. Such TCAD models work by predicting the electric field at every point in a 3D semiconductor device, and the resulting movement of charge carriers (electrons and holes) in the device. But doing so is computationally very expensive. Therefore, TCAD codes such as Charon (sandia.gov) are orders of magnitude slower than compact models. For this reason, TCAD is almost never used directly in a circuit simulator. Instead, circuit simulators use compact models that are much faster reduced-order approximations of TCAD models.*

Traditional methods for obtaining useful compact models involve two steps, often done independently of one another:

1. Developing a set of equations that capture the relationships between voltages, currents, and charges in a device, and

2. Calibrating model parameters.

The first step above requires expertise in semiconductor device physics and numerical analysis. It can take many months (or even years) of work to properly understand the physics of a device, and to derive the relevant equations and their parameterizations. For example, the BSIM family of compact models for Metal-Oxide-Semiconductor (MOS) transistors is the result of over 20 years of work by Prof. Chenming Hu and his large research group at UC Berkeley (Chauhan et al., 2015; Liu and Hu, 2011, 1998). Besides long development times, reliance on simplified solutions in traditional compact models may compromise their ability to generalize. As a result, adding new physics to a legacy compact model (*e.g.*, to scale the model down to a smaller CMOS technology, to take into account radiation effects in harsh environments, *etc.*) often requires extensive redevelopment.

As for the second step, the equations above typically contain several parameters whose values need to be "fitted" to measured electrical data. The number of parameters can grow significantly with device complexity; for example, the traditional SPICE model for a diode has about a dozen parameters, the model for a Bipolar Junction Transistor (BJT) has 42 parameters, and models for Metal Oxide Semiconductor (MOS) transistors in advanced technologies can have hundreds of parameters. Different companies/organizations take different approaches to model calibration. In general, as compact models are highly non-linear and as the number of parameters to be fitted can be large, model calibration is a long-standing non-trivial problem requiring large manual effort.

In contrast to the traditional method above, our data-driven approach offers a way to perform both steps at once, in an automated way that can be more accurate and also faster and cheaper by virtue of employing Machine Learning (ML) techniques to largely remove the need for human experts and manual effort. Moreover, ML techniques provide a wide range of regression methods that can be used to develop hierarchies of compact models tailored to specific circuit design stages, specific circuit simulation tasks, and even specific compute infrastructures. Indeed, in a world where compact model development is fully automated, where a variety of compact models capturing

different facets of a device's behaviour, with different computational efficiency and accuracy trade-offs, can all be generated at the push of a button, it is conceivable that a circuit designer would use a different compact model for initial exploration and a different one for late-stage design, one for timing analysis and another one for sensitivity analysis, one for CPU simulation and one for GPU simulation, and so on. The benefits would be immense – enabling rapid, cost-effective, and robust circuit design flows calibrated against real-world electrical data from day one. We believe we are the first to propose developing such a systematic, generally applicable, and mostly automated set of tools and techniques for data-driven compact model development.

In this paper, we investigate three markedly different ML approaches – Table-Based Interpolation (TBI), Generalized Moving Least-Squares (GMLS), and Deep Neural Networks (DNNs) – for developing data-driven compact device models. Specifically, we apply these approaches to develop compact models for a 1N4148 high-speed switching diode, a common mass-produced semiconductor device with well-documented electrical and thermal characteristics (Wikipedia; diodes.com).

The first approach, TBI (Section 3.1), is a local parametric regression technique that uses cubic splines to construct a piecewise polynomial approximation of available electrical data (de Boor, 1978; Gupta et al., 2017). TBI is used extensively in many modeling and simulation contexts, including compact semiconductor device modeling, where it offers simplicity, computational efficiency, and the ability to generate differentiable approximations. The drawbacks of table-based models include significant memory requirements and datasets restricted to rectangular grids. We refer to (Gupta, 2018; Gupta et al., 2017) for relevant recent work.

The second approach (Section 3.2), uses GMLS approximants (Wendland, 2004) to build compact device models; this method, unlike TBI, can be applied to scattered data as well.[1] GMLS is an example of non-parametric regression, which uses local kernels to build estimates from scattered data. Scientific computing applications of GMLS range from the design of meshfree discretizations for PDEs (Jiun-Shyan et al., 2017) to data transfers for coupled multiphysics simulations (Slattery, 2016; Bungartz et al., 2016). We believe we are the first to apply GMLS to compact modeling; we not only develop GMLS-based device models but also demonstrate them in circuit simulations.

Finally, in Section 3.3, we develop DNN (Goodfellow et al., 2016) device models. DNNs are compositions of non-linear activation functions and affine transformations, and represent global non-linear parametric regression. The success of DNNs in various classification tasks is well documented (LeCun et al., 2015). Their application to scientific computing is more recent (Bar-Sinai et al., 2019; Raissi et al., 2017) but is generating significant interest. It should be noted that DNN applications to circuit simulations (Zaabab et al., 1994, 1995; Meijer, 1996; Litovski et al., 1997; Andrejević and Litovski, 2003; Chen et al., 2006; Gorissen et al., 2009) predate these efforts, but have stayed fairly dormant over the years. It is likely though that this research direction will intensify and attract more attention, as evidenced by recent work (Chen et al., 2017). At the same time, compact DNN models of devices are few and far between in the literature. Early examples include (Meijer, 1996), (Zaabab et al., 1997), and (Hammouda et al., 2008), where DNNs were used to model various metal oxide and field effect transistors. More recent work includes a multi-layer perceptron (MLP) model of a transistor device (Lei et al., 2018), and a compact model for a thin TFET device using a hybrid MLP architecture with two different activation functions (Li et al., 2016).

Our interest in compact DNN device models is because they can capture complex nonlinear behaviors with relatively few parameters. As a result, DNNs have the potential to produce compact models that are both more efficient and have smaller memory footprints than either TBI or GMLS

1. Such datasets result from scattered electrical measurements of devices with more than two terminals, which will be considered in forthcoming work.

models. Indeed, unlike DNNs, TBI and GMLS models both require memory of the order of the size of the entire dataset. This could be a significant advantage for DNNs when modeling devices with more than one p-n junction. DNN models are also inherently more computationally efficient: being a global regression, DNNs do not require dataset searches as TBI and GMLS models do; their main cost is a few evaluations of the non-linear activation function. In this paper, we provide insights into the development of DNN compact models, informed by performing circuit simulations using them. In particular, our results strongly suggest that a "reasonable" Mean Square Error (MSE) fit of I–V characteristic curves alone may not be enough to ensure convergence of a data-driven device in circuit simulations and/or physically correct simulation results; for this, the compact model should also possess actual device physics properties, such as passivity, monotonicity, zero current at zero voltage, *etc*. A key contribution of this paper is the development of a DNN training strategy, based on transformed sets of electrical measurements, that consistently produces physically correct compact diode models across a range of DNN architectures; these models perform robustly in circuit simulations, and produce results that are in excellent agreement with laboratory measurements.

The rest of the paper is organized as follows. Section 2 provides some background information about the technical approach and the software tools used in this work. Section 3 describes the core techniques underlying the three regression methods above. Section 4 presents simulation results; to assess the performance of data-driven compact device models, we first compare their I–V characteristics with laboratory measurements using three different data views that expose different aspects of device operation. Then, we use these compact models to build a full-wave bridge rectifier circuit, simulate the circuit, and compare simulation results against laboratory measurements. In Section 5, we discuss our conclusions and outline directions for future research.

2. Preliminaries

2.1. Workflow

The main focus of this paper is the development and testing of data-driven compact models based on three different regression approaches, exemplified using a 1N4148 high-speed switching diode. Figure 1 shows the steps involved in our compact model development and testing workflow, as applied to this diode.

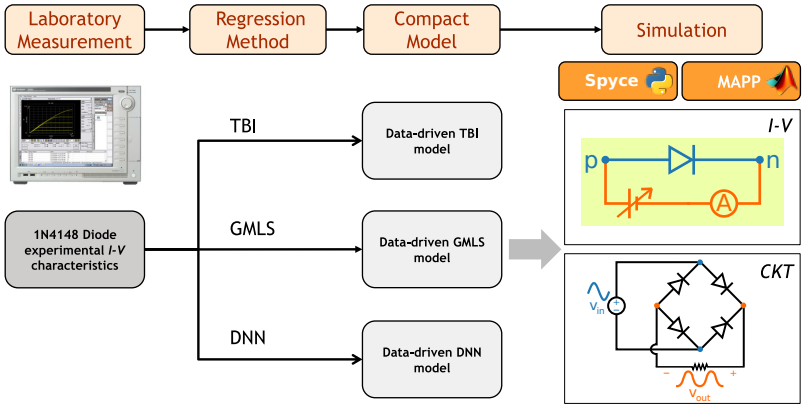

Figure 1: Our workflow to develop and test data-driven compact models, illustrated for the 1N4148 diode.

As shown in the figure, we first obtain I–V electrical measurements in the lab. Then, we apply our three different regression methods to this data, thereby generating three different sets of data-

driven compact models for the device. We then simulate these data-driven compact models to obtain their I–V characteristics, and we also simulate circuits where such models are deployed. Circuit simulations are then compared against laboratory measurements.

Software: We have implemented the entire workflow of Figure 1 both in MATLAB and in Python. For TBI, we use two tools (1) STEAM (Gupta et al., 2017; Gupta, 2018), an open-source MATLAB tool developed at UC Berkeley, and (2) a Python implementation of cubic splines developed for this work. For GMLS, we use the open-source Compadre toolkit (Kuberry et al., 2019), available as a Python package. For DNNs, we use TensorFlow (Abadi et al., 2015), an open-source tool available as a Python library. And for compact model and circuit simulations, we use: (1) the Berkeley Model and Algorithm Prototyping Platform (MAPP) (Wang and Roychowdhury, 2016), an open-source circuit simulator written in MATLAB, and (2) Spyce, a Python-based research circuit simulator developed at Sandia National Laboratories.

2.2. A compact p-n junction diode model

A p-n junction diode has two terminals, p and n, with n being the "reference" terminal. The voltage difference between p and n is denoted v_{pn}, and the current flowing into the diode at the non-reference terminal p is denoted i_{pn}. The current flowing into the reference terminal is $i_{np} = -i_{pn}$. A compact diode model is a mapping $f_{PN} : \mathbb{R} \to \mathbb{R}$ that gives the diode current as a function of the applied voltage, *i.e.*, $i_{pn} = f_{PN}(v_{pn})$. Constructing a compact diode model thus boils down to specifying the function f_{PN}. In addition, the derivative of this function with respect to v_{pn} is required by the non-linear solver in the circuit simulator.

Remark 2 *In traditional compact models, f_{PN} is usually given by an analytic expression involving k parameters $\vec{p} = (p_1, \ldots, p_k)$ that can be used to calibrate f_{PN} to data. As a result, the derivative of f_{PN} can be obtained by automatic differentiation (AD), and does not have to be provided as part of the model (Griewank and Walther, 2008). A typical example is the Shockley diode equation (Shockley, 1949),*

$$i_{pn} = f_{PN}(v_{pn}, \underbrace{(I_S, V_T, q)}_{\vec{p}}) = I_S \left(e^{\frac{v_{pn}}{qV_T}} - 1 \right). \tag{1}$$

Here, I_S is the reverse bias saturation current, V_T is the thermal voltage, and q is the "quality factor", a non-physical parameter used to account for imperfect p-n junctions in real diodes. AD may also be applied to some parametric regression models for f_{PN}, such as TBI. However, AD is not applicable to non-parametric regression such as GMLS, in which case the compact model must also provide its derivative. As a result, the accuracy of the regression fit for f_{PN} alone is not enough to ensure the quality of such models; testing them in actual circuits should be an integral part of the development and validation process.

In contrast to a traditional compact model such as (1), data-driven models estimate f_{PN} and its derivative by applying a regression technique $\mathcal{R}$ to a dataset $\mathcal{D}_{PN} = \{v_{pn}^k, i_{pn}^k\}_{k=1}^m$. The dataset, an $m \times 2$ real matrix, contains measurements of the device's I–V characteristic curve.

2.3. I–V measurements of a 1N4148 diode

The I–V characteristic of a device is a basic set of electrical measurements and a fundamental way to understand the performance of various materials and devices under test (DUT). I–V measurements obtain the current *vs.* voltage characteristic (denoted $\mathcal{D}_{PN}$ above) of a device by applying a

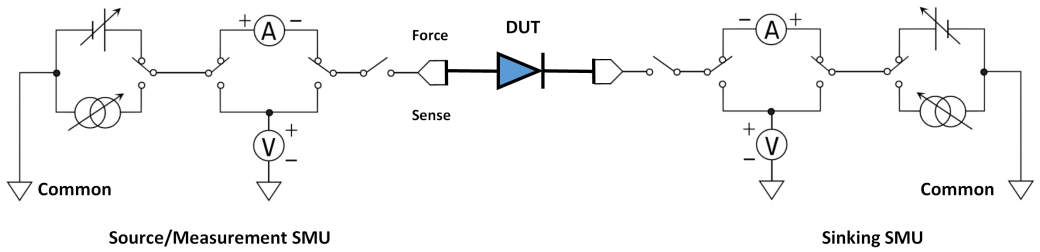

Figure 2: Schematic of our laboratory measurement setup for the 1N4148 diode.

series of voltage stimuli to the device and measuring the resulting current responses. For this work, we used a Keysight B1505A Parametric Analyzer on a 1N4148 diode specimen to obtain $\mathcal{D}_{PN}$. This parametric analyzer uses Source Measurement Units (SMUs) that combine a current source, a voltage source, an ammeter, and a voltmeter into a single unit; see Figure 2 for a schematic. We chose the B1505A's HPSMU (High Power Source Measurement Unit) for its ability to supply ample current over an extended voltage range while maintaining adequate measurement resolution.

For this work we sampled the $I{-}V$ curve at $m = 9682$ points, resulting in a dataset $\mathcal{D}_{PN}$ given by a 9682×2 matrix. The voltage stimuli ranged from $v_{pn}^{\min} = -125\text{V}$ to $v_{pn}^{\max} = 0.8\text{V}$, in increments Δv_{pn} ranging between 10mV and 20mV. The non-uniformity of the voltage increments is due to rounding errors and the fact that the measurements have inherent noise and stability issues, and are at the limit of forced voltage step resolution.

3. Core techniques for developing data-driven compact models

In this section, we apply three different regression techniques $\mathcal{R}$ to the dataset $\mathcal{D}_{PN}$ above, to develop three different sets of data-driven compact device models for the 1N4148 diode.

3.1. Table-Based Interpolation (TBI) devices

A TBI diode compact model is a smooth function $i_{pn} = f_{PN}(v_{pn})$ that interpolates the data points defined by the rows of $\mathcal{D}_{PN}$. There are many ways to construct such a function – including cubic splines, Chebyshev polynomials, and Barycentric Lagrange interpolation (de Boor, 1978; Trefethen, 2013). In this paper, we use cubic splines; they are simple to describe and construct, they produce robust, C^2-regular compact models that converge well in circuit simulations, and they offer inexpensive compact model evaluation as well as derivative computation as they only require cheap polynomial evaluations (with pre-computed coefficients that can be looked up efficiently). Examples of such compact models can be found in (Gupta, 2018) and (Gupta et al., 2017).

Remark 3 *A table-based model for devices with more than two terminals involves multivariate cubic spline interpolation. The main drawback of such models is that they require electrical data over a rectangular grid of voltages. Such models do not straightforwardly extend to scattered data.*

Below, we briefly review the construction of the univariate cubic splines used in this work. To improve convergence of the resulting compact models in circuit simulations, our development differs in important ways from standard splines found in the literature such as natural splines.

To declutter notation, just for this subsection we switch to labeling the data points in $\mathcal{D}_{PN}$ as (x_k, y_k), $k = 1, \ldots, m$. Thus, we have that $x_1 = v_{pn}^{\min}$, $x_m = v_{pn}^{\max}$, and $x_k = v_{pn}^{k}$ for $1 < k < m$. Likewise, $y_k = i_{pn}^{k}$. Without loss of generality, we assume that $x_1 < x_2 < \ldots < x_m$.

The voltage measurements x_k induce a partition of the x-axis into $m+1$ intervals $(-\infty, x_1)$, $[x_1, x_2)$, $[x_2, x_3)$, …, $[x_{m-1}, x_m)$, $[x_m, +\infty)$. We refer to the first and last interval as "boundary" intervals and the rest as "interior" intervals. A cubic spline defined with respect to this partition is a piecewise cubic polynomial $f(x)$ which interpolates the data (x_k, y_k), $k = 1, \ldots, m$, and has continuous first and second derivatives, *i.e.*, it is of class $C^2(\mathbb{R})$. We denote the restriction of $f(x)$ to the i^{th} interval above as $C_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$, for $1 \leq i \leq m+1$. Succinctly,

$$f(x) = \begin{cases} C_1(x), \text{ if } x < x_1, \\ C_{i+1}(x), \text{ if } x \in [x_i, x_{i+1}) \text{ for each } 1 \leq i \leq m-1, \text{ and} \\ C_{m+1}(x), \text{ if } x \geq x_m. \end{cases}$$

To determine the $4(m+1)$ polynomial coefficients $\{a_i, b_i, c_i, d_i\}_{i=1}^{m+1}$ defining the cubic spline segment on each interval, we enforce the following $4(m+1)$ constraints:

- $C_i(x_i) = y_i$ and $C_{i+1}(x_i) = y_i$; $1 \leq i \leq m$: interpolation ($2m$ constraints).
- $C_i'(x_i) = C_{i+1}'(x_i)$; $1 \leq i \leq m$: continuity of first derivatives (m constraints).
- $C_i''(x_i) = C_{i+1}''(x_i)$; $1 \leq i \leq m$: continuity of second derivatives (m constraints).
- $a_1 = b_1 = 0$ and $a_{m+1} = b_{m+1} = 0$: linearity at boundary intervals (4 constraints).

These conditions are sufficient to determine a unique, globally C^2 piecewise cubic interpolant $f(x)$ of the data in $\mathcal{D}_{PN}$. In practice, given the electrical measurements $\mathcal{D}_{PN}$, the constraints above are used to pre-compute and store the coefficient set $\{a_i, b_i, c_i, d_i\}_{i=1}^{m+1}$ in memory. To evaluate the compact model, *i.e.*, to compute f_{PN} and f'_{PN} at a query point v^*_{pn}, one first locates the interval containing v^*_{pn} and retrieves the four coefficients corresponding to the restriction of f_{PN} to that interval. Then the corresponding cubic polynomial and its derivative are calculated and returned.

3.2. GMLS devices

GMLS is a non-parametric regression approach for approximating linear functionals from scattered data (Wendland, 2004). Here, we use GMLS to estimate f_{PN} and f'_{PN} from the data $\mathcal{D}_{PN}$. Below, we describe the basics necessary for this task and refer to (Wendland, 2004) for further details.

Consider a C^1 function $f : R \to R$ with domain $\mathcal{D} \subseteq \mathbb{R}$, a point cloud $X = \{x_1, \ldots, x_m\} \subset \mathcal{D}$ with a fill distance h_X, and a collection of point samples $\mathbf{f} = \{f(x_i)\}_{i=1}^m$ on the cloud. Let $P_k(\mathbb{R})$ denote the set of all real polynomials of degree less than or equal to $k > 0$ with dimension $\dim P_k(\mathbb{R}) = k+1$ and basis $\boldsymbol{\phi} = \{\phi_q\}_{q=1}^{k+1}$. Finally, $\rho_\epsilon(r)$ will denote a radially symmetric kernel function that is at least $C^1(\mathbb{R})$ and whose support is contained in $(-\epsilon, \epsilon)$ for some real $\epsilon > 0$.

Given a point $x^\star \in \mathcal{D}$, GMLS computes approximations $\widetilde{f(x^\star)} \approx f(x^\star)$ and $\widetilde{d_x f}(x^\star) \approx d_x f(x^\star)$ that are exact for all $f \in P_k(\mathbb{R})$ (polynomial reproduction) in two steps. To describe these steps, let $W(x^\star) \in \mathbb{R}^{m \times m}$ be a diagonal matrix with element $W_{ii}(x^\star) = \rho_\epsilon(|x^\star - x_i|)$. Let $B \in \mathbb{R}^{m \times (k+1)}$, the matrix with element $B_{ij} = \phi_j(x_i)$; $i = 1, \ldots, m$; $j = 1, \ldots, k+1$. And let $\|\cdot\|_W$ be the Euclidean ℓ^2 norm on $\mathbb{R}^m$ weighted by $W(x^\star)$. The two GMLS steps are:

Step 1. Computing the GMLS coefficient vector. Solve the weighted least-squares problem

$$\mathbf{c}(\mathbf{f}) = \underset{\mathbf{c} \in \mathbb{R}^{k+1}}{\operatorname{argmin}} \frac{1}{2} \|B\mathbf{c} - \mathbf{f}\|_W^2 \,. \tag{2}$$

Step 2. Computing the GMLS approximants. Set[2]

$$\widetilde{f}(x^\star) := \boldsymbol{c}(\mathbf{f}) \cdot \boldsymbol{\phi}(x^\star) \quad \text{and} \quad \widetilde{d_x f}(x^\star) := \boldsymbol{c}(\mathbf{f}) \cdot d_x\boldsymbol{\phi}(x^\star)\,. \tag{3}$$

To evaluate the GMLS compact diode model at a given query point v_{pn}^*, we proceed as follows. If $v_{pn}^* \in [v_{pn}^{\min}, v_{pn}^{\max}]$, we associate the first and the second columns of $\mathcal{D}_{PN}$ with a point cloud $X \subset \mathbb{R}$ and a sample set $\mathbf{f} \in \mathbb{R}^m$, respectively, solve (2) and define $f_{PN}(v_{pn}^*)$ and $d_x f_{PN}(v_{pn}^*)$ according to (3). If $v_{pn}^* \notin [v_{pn}^{\min}, v_{pn}^{\max}]$, we evaluate $f_{PN}(v_{pn}^*)$ and $d_x f_{PN}(v_{pn}^*)$ as follows. Let v_{pn}^{clo} be the point from $\mathcal{D}_{PN}$ that is closest to v_{pn}^*. Note that v_{pn}^{clo} is either $v_{pn}^{\min}$ or $v_{pn}^{\max}$. We then set

$$f_{PN}(v_{pn}^*) := f_{PN}(v_{pn}^{\text{clo}}) + (v_{pn}^* - v_{pn}^{\text{clo}}) d_x f_{PN}(v_{pn}^{\text{clo}}) \quad \text{and} \quad d_x f_{PN}(v_{pn}^*) := d_x f_{PN}(v_{pn}^{\text{clo}}),$$

respectively, where $f_{PN}(v_{pn}^{\text{clo}})$ and $d_x f_{PN}(v_{pn}^{\text{clo}})$ are the GMLS estimates at v_{pn}^{clo}.

In this work, we use the Compadre toolkit (Kuberry et al., 2019) for performant implementation of GMLS, with polynomial orders $k = 1, 2, 3$, and kernel $\rho_\epsilon(r) = (1 - r/\varepsilon)_+^p$, with $p = 4$ and $\varepsilon = h_X$. Compadre uses an adaptive procedure to adjust ε until $\text{supp}(\rho_\epsilon(|x^\star - x_i|) \cap X$ is guaranteed to contain enough points to ensure the desired degree of polynomial reproduction. For real polynomials and quasi-uniform point clouds, the number of points selected by this procedure does not exceed $2(k+1)$.

Compadre solves the weighted least-squares problem (2) using QR factorization, which formally requires $2(k+1)^2(m-(k+1)/3)$ flops (Golub and Loan, 1996, p.240). However, the actual number of non-zero rows in B equals the number of points selected by Compadre's adaptive procedure and is bounded by $2(k+1)$, $k = 1, 2, 3$, *i.e.*, it is orders of magnitude less than the size m of the dataset $\mathcal{D}_{PN}$. As a result, the actual cost per model evaluation is $O((k+1)^3)$ rather than $O((k+1)^2 m)$. In practice, we did not observe noticeable differences in the performance of GMLS compact models as we increased the polynomial degree from 1 to 3.

3.3. DNN devices

In this work, we use a simplified DNN definition from (Opschoor et al., 2019). Let d and D be two natural numbers defining the input dimension and the depth of the network, respectively. Consider a set of natural numbers $\{n_0, \ldots, n_D\}$ such that $n_0 = d$, and a set of matrix-vector tuples $\{A_i, \mathbf{b}_i\}$, $i = 1, \ldots, D$ such that $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and $\mathbf{b}_i \in \mathbb{R}^{n_i}$. The elements of A_i and $\mathbf{b}_i$ are usually called the weights and the biases of the DNN respectively. Finally, let $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-linear "activation" function. The action of the resulting DNN on an input vector $\mathbf{x} \in \mathbb{R}^d$ is defined as

$$\begin{cases} \mathbf{y}_0 & := & \mathbf{x} \\ \mathbf{y}_k & := & \sigma(A_k \mathbf{y}_{k-1} + \mathbf{b}_k) \quad \text{for } k = 1, \ldots, D-1 \\ \mathbf{y}_D & := & A_D \mathbf{y}_{D-1} + \mathbf{b}_D \end{cases}, \tag{4}$$

where σ is applied component-wise. We denote the transformation of the input vector by the DNN as $\mathcal{N}(\mathbf{x})$, that is, $\mathbf{y}_D = \mathcal{N}(\mathbf{x})$. The mapping $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^{n_D}$ defines a global non-linear function

2. The GMLS derivative approximation in (3) *appears* to violate the product rule and for this reason it was often referred to as the "diffuse derivative approximation" in the literature; see (Mirzaei et al., 2012). This confusion stems from misconstruing how the GMLS approximation works and assuming (erroneously) that $\widetilde{d_x f}(x^\star)$ is defined by differentiating $\widetilde{f}(x^\star)$. In fact, the GMLS derivative approximation is derived independently of the GMLS field approximation and does not involve differentiation of the latter; see (Wendland, 2004).

parameterized by the unknown weights and biases $\{A_i, \mathbf{b}_i\}$. To determine these parameters one "trains" the network by solving a constrained optimization problem,

$$\{A_i, \mathbf{b}_i\} = \arg\min \mathcal{L}(\mathcal{N}(\mathbf{X}_T), \mathbf{Y}_T) \quad \text{subject to} \quad \mathcal{C}\left(\{A_i, \mathbf{b}_i\}\right) \tag{5}$$

where $\mathcal{L}$ is a "loss" function measuring the mismatch between the network's output and the training output, $\mathcal{C}$ is a non-linear constraint operator, $\mathbf{X}_T = \{\mathbf{x}_{T,1}, \ldots, \mathbf{x}_{T,m}\}$ is a set of training inputs, and $\mathbf{Y}_T = \{\mathbf{y}_{T,1}, \ldots, \mathbf{y}_{T,m}\}$ are the corresponding outputs. We refer to the pair $\{\mathbf{X}_T, \mathbf{Y}_T\}$ as the training set and denote it by $\mathcal{T}$. In this work, we train the neural network model using Adam, a variant of stochastic gradient descent, with a maximum of E epochs.

For classification tasks, the non-linearity of $\mathcal{N}$ is used to transform input datasets representing different classes into linearly separable sets. In contrast, here we shall use (4) as a *regression* tool to build a compact diode model from $I-V$ measurements. To that end, we set the input dimension $n_0 = 1$, the output dimension $n_D = 1$ and define $\mathcal{L}$ as the mean square (MSE) error, a common choice for regression tasks. Furthermore, since circuit simulations require differentiable compact models, we only consider smooth activation functions σ and forego the widely used piecewise linear ReLU activation. As a constraint operator $\mathcal{C}$, we use non-negativity of the weights to ensure monotonicity of the function f_{PN}, as required by the physics of p-n junction diodes.

We train the network using three different options for the training set $\mathcal{T}$. The first one is $\mathcal{T} := \mathcal{D}_{PN}$, *i.e.*, the original set of $I-V$ measurements with bounding box $[v_{pn}^{\min}, v_{pn}^{\max}] \times [i_{pn}^{\min}, i_{pn}^{\max}]$. The second option is the *transformed* dataset $\mathcal{D}_{PN}^{VI} := \{T_V(v_{pn}^k), T_I(i_{pn}^k)\}$, where

$$T_V(v_{pn}) = \begin{cases} v_{pn}/V^+ & \text{if } v_{pn} \geq 0, \\ v_{pn}/V^- & \text{if } v_{pn} < 0 \end{cases} \qquad \text{and} \tag{6}$$

$$T_I(i_{pn}) = \begin{cases} -(\log_{10}(-i_{pn}) - B^-)/A^- & \text{if } i_{pn} \in (-\infty, -P_{\min}^-] \\ -1 + S(i_{pn} + P_{\min}^-) & \text{if } i_{pn} \in (-P_{\min}^-, P_{\min}^+) \\ (\log_{10}(i_{pn}) - B^+)/A^+ & \text{if } i_{pn} \in [P_{\min}^+, \infty) \end{cases} ; \tag{7}$$

with $P_{\min}^{\pm} = 10^{p_{\min}^{\pm}}$, $A^{\pm} = (p_{\max}^{\pm} - p_{\min}^{\pm})/7$, $B^{\pm} = p_{\min}^{\pm} - A^{\pm}$, and $S = 2/(P_{\min}^+ + P_{\min}^-)$. The last option is the *partially transformed* set $\mathcal{D}_{PN}^I := \{v_{pn}^k, T_I(i_{pn}^k)\}$, *i.e.*, only i_{pn} is transformed, not v_{pn}. For the numerical examples, we set $V^+ = 0.1$, $V^- = 15.625$, $p_{\min}^{\pm} = -10$, $p_{\max}^+ = -1$ and $p_{\max}^- = -5$. Given a training set $\mathcal{T}_{PN}^*$ the training process yields an instance of (4) which we label as $\mathcal{N}_{PN}^*$. The data-driven compact device model is then defined as

$$f_{PN}(v_{pn}) = \begin{cases} \mathcal{N}_{PN}(v_{pn}) & \text{if } \mathcal{T} = \mathcal{D}_{PN} \\ T_I^{-1} \circ \mathcal{N}_{PN}^{VI} \circ T_V(v_{pn}) & \text{if } \mathcal{T} = \mathcal{D}_{PN}^{VI} \\ T_I^{-1} \circ \mathcal{N}_{PN}^{I}(v_{pn}) & \text{if } \mathcal{T} = \mathcal{D}_{PN}^{I} \end{cases} .$$

Training DNNs on appropriately transformed electrical measurement data is one of the key ideas in this paper. Our results in Section 4 show that this strategy consistently produces DNN compact models that provide accurate data fit and perform well in circuit simulations. These transformations are motivated by the wide range of currents a p-n junction is capable of exhibiting; such currents can vary over several orders of magnitude, which undermines the effectiveness of standard ML loss functions such as MSE.

For example, when $v_{pn} < 0$, i_{pn} is a very small, negative current (in our dataset, this current ranged from about -1.5 micro-Amperes to -0.5 nano-Amperes). To mitigate the inability of the

MSE loss function to differentiate between such small values, we transform the current in the reverse bias regime as $i_{pn} \to -log_{10}(-i_{pn})$ (with appropriate shifting and scaling to preserve its negativity). In so doing, a negative micro-Ampere current transforms into 6 and a negative nano-Ampere becomes a 9. Such large values are easily resolvable by the MSE loss function.

Similarly, when $v_{pn} > 0$, i_{pn} is a positive current that exponentially rises over several orders of magnitude. In our dataset, this current ranged from 0.5 pico-Amperes to 35 milli-Amperes. To resolve these values with the MSE loss function, we apply a log transformation, followed by a shift so that the positivity of i_{pn} is preserved. For instance, with a shift of 14, a pico-Ampere becomes a 2 and a milli-Ampere transforms into 11.

However, since we have used different log transformations for $i_{pn} < 0$ and $i_{pn} > 0$, we will have a discontinuity at $i_{pn} = 0$, which can cause convergence problems in circuit simulations. To get around this, we choose a small interval around $i_{pn} = 0$, apply the respective log transformations only outside this interval, and apply a continuity-preserving linear transformation within this interval. This approach ensures that an MSE loss function in conjunction with the transformed set $\mathcal{D}_{PN}^{VI}$ resolves both positive and negative p-n junction currents very well, leading to a much better fit.

Also, we would like to point out that the parameters used in the transformations above can be automatically calculated from the diode dataset; they need not be manually derived. This is the approach we took.

Configuring and training a DNN involves choosing several parameters, such as the number of hidden layers $D-1$, the number of neurons n per hidden layer, the activation function, whether or not to use a kernel constraint, whether or not to transform v_{pn}, and whether or not to transform i_{pn}. In this work, we experimented with DNNs containing 1 and 2 hidden layers, with 5, 10, 25, 50, or 100 neurons per hidden layer, and 3 different activation functions (eLU, sigmoid, and tanh). Taken together with the 3 binary choices of whether to have a kernel constraint, transform v_{pn}, and/or transform i_{pn}, this corresponds to a parameter space of size $2 \times 5 \times 3 \times 2 \times 2 \times 2 = 240$. We systematically explored this space, generating all 240 DNN compact models. Then we simulated them all using Spyce, to determine their I–V characteristics as well as their performance in a bridge rectifier circuit. We do not have the space to discuss all the DNNs here; so we discuss only a small sample of DNNs, as summarized in Table 1.

Model	σ	$D-1$	n_i	$\mathcal{T}$	$\mathcal{C}$
M1-50-E	eLU	1	50	$\mathcal{D}_{PN}$	Non-neg
M1-50-S	sigmoid	1	50	$\mathcal{D}_{PN}$	Non-neg
M1-50-S-neg	sigmoid	1	50	$\mathcal{D}_{PN}$	None
M1-10-E-VI	eLU	1	10	$\mathcal{D}_{PN}^{VI}$	Non-neg
M1-10-E-VI-neg	eLU	1	10	$\mathcal{D}_{PN}^{VI}$	None
M1-10-T-VI	tanh	1	10	$\mathcal{D}_{PN}^{VI}$	Non-neg

Table 1: Summary of DNN architectures and training configurations.

4. Results

In this section, we carry out two types of simulations for each data-driven diode compact model: we (1) produce its I–V characteristics, and (2) simulate a bridge rectifier circuit (schematic shown in the bottom right of Figure 1) that uses 4 instances of the model. The input signal to the rectifier is

a sine wave with frequency 10Hz and phase shift approximately $\pi/1.63$. This simulation provides device model validation at a circuit level. We compare the results of both simulations against laboratory data. To visualize how well a data-driven diode compact model matches measured data, we

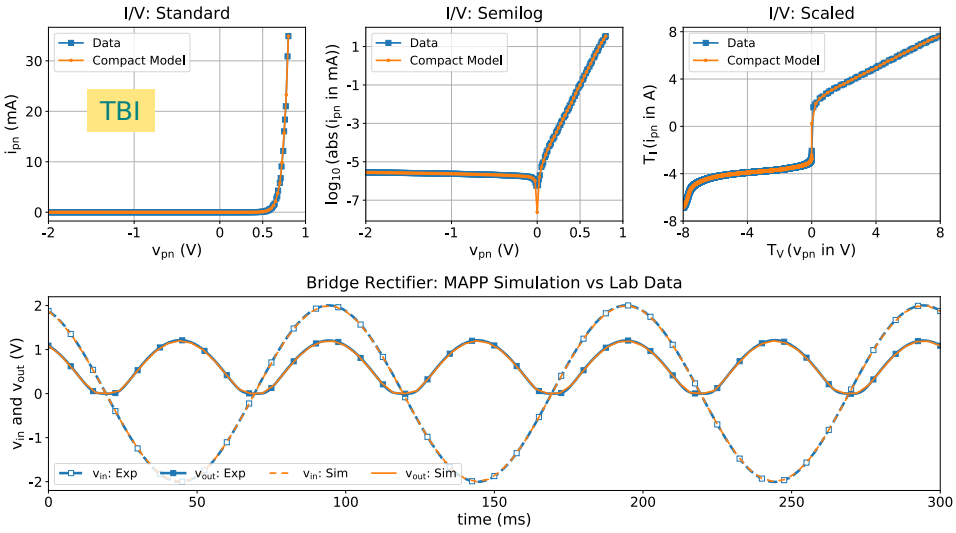

Figure 3: Simulations comparing a cubic spline TBI diode compact model against laboratory data

arrange results in plots containing 3 top sub-plots and 1 bottom sub-plot. The top sub-plots show the I–V characteristics of the compact model, overlaid on top of laboratory data, using three different "data views" defined as follows:

- Standard (top-left sub-plot): i_{pn} is plotted as a function of v_{pn}.
- Semilog (top-middle sub-plot): $\log_{10}(abs(i_{pn}))$ is plotted as a function of v_{pn}.
- Scaled (top-right sub-plot): $T_I(i_{pn})$ is plotted as a function of $T_V(v_{pn})$, where T_I and T_V are the transformations defined in (7) and (6), respectively.

Together, the three data views provide a comprehensive picture: the standard view highlights the *forward bias* regime where the diode conducts positive, exponentially-growing current for positive v_{pn}. The semilog view highlights the "zero crossing point", where the diode transitions from *reverse bias* (where it conducts very little current) to *forward bias*. The scaled view exposes the entire region of operation of the diode, from *avalanche breakdown* (where i_{pn} starts to become more and more negative at very low v_{pn}, risking irreversible damage to the device) all the way to *forward bias*.

The bottom sub-plot in each figure shows a bridge rectifier circuit simulation (see schematic at the bottom right of Figure 1), overlaid on top of laboratory data.

Figure 3 shows simulation results for cubic spline TBI diode compact model, generated via STEAM and MAPP. Figure 4 shows the same for 3 GMLS compact models (with polynomial orders 1, 2, and 3). In both cases, we see that the data-driven models are in an excellent agreement with laboratory data. Due to space constraints, we do not show results for TBI and GMLS models generated via Spyce, but they are virtually identical to the MAPP/STEAM results shown here.

We now turn to compact DNN models. Using the original dataset $\mathcal{D}_{pn}$ as the training set failed to produce even a single accurate DNN compact model. For all parameter choices described in Section 3.3, the learned DNN models were both inaccurate and unphysical. Figure 5 illustrates these

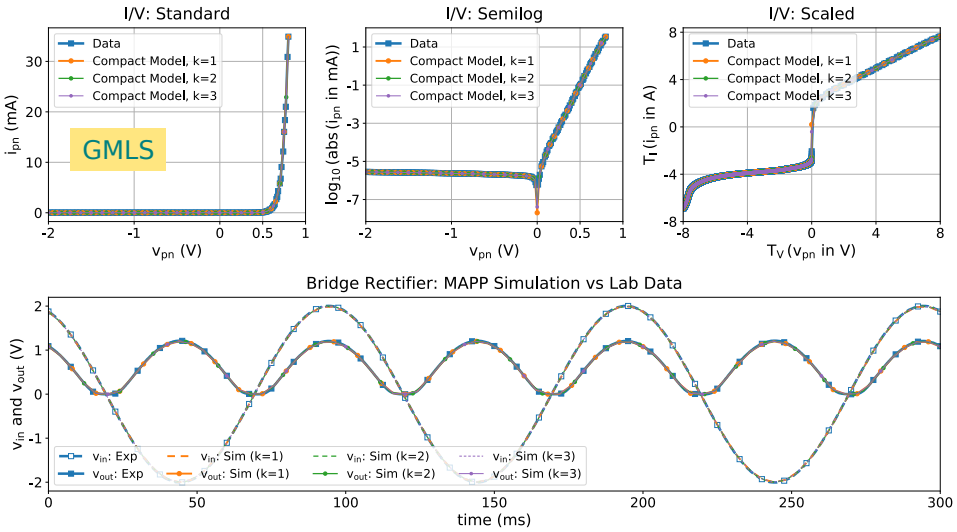

Figure 4: Simulations comparing GMLS diode compact models against laboratory data

failures using 3 examples drawn from the 240 combinations examined in this work, all based on a DNN with 1 hidden 50-neuron layer. The top plot uses eLU, the rest use sigmoid activation. The middle plot enforces monotonicity, while the bottom does not. These examples highlight the following key points: (1) data transformations are critical to obtaining accurate and physically consistent DNN compact models, (2) the positive weight constraint enforcing monotonicity of i_{pn} with respect to v_{pn} is crucial; without this, the resulting DNN is often highly unphysical, and (3) even though the standard I–V characteristic of a data-driven compact model may appear to match laboratory data satisfactorily, the characteristic needs to be viewed on a semilog plot as well as a (T_V, T_I) transformed plot before one can be sure that a data-driven compact model is accurate enough.

Our results indicate that DNN performance markedly improves by switching the training set to the transformed dataset $\mathcal{D}_{pn}^{VI}$. However, an important takeaway is that not all activation functions lead to satisfactory compact models. In particular, we found that eLU activation performed much worse than sigmoid or tanh. For example, Figure 6 shows simulations of the M1-10-E-VI compact model; from the standard I–V plot, it is apparent that the model greatly overestimates the current in the forward bias region. Still, the transformations of the training set ensure that the model behaves well and has good convergence properties during circuit simulation; the bottom plot shows that the model even gets reasonably close to matching laboratory data when it comes to circuit behaviour.

Figure 7 shows an interesting corner case: with eLU activation and data transformations, it looks like removing the non-negative weight constraint actually helps the model become more accurate, both in its I–V characteristic and in circuit simulations. Recall that eLU is the identity function for positive inputs and an exponential converging to a negative value for negative inputs. Thus, if the DNN weights are constrained to be positive, the DNN's behavior is constrained to be close to linear – a bad fit for a diode. Allowing negative weights helps the DNN become more non-linear, and hence mimic the data more closely.

By far, our most positive DNN finding is that combining sigmoid or tanh activation with data transformations can produce extremely accurate, physically consistent, and efficient DNN models. Figure 8 shows this for the M1-10-T-VI case; with just a single hidden layer and 10 neurons, this tanh activated DNN is able to model all aspects of the diode's I–V characteristic very well, and perform very accurately in circuit simulations as well. We found that similar results held true

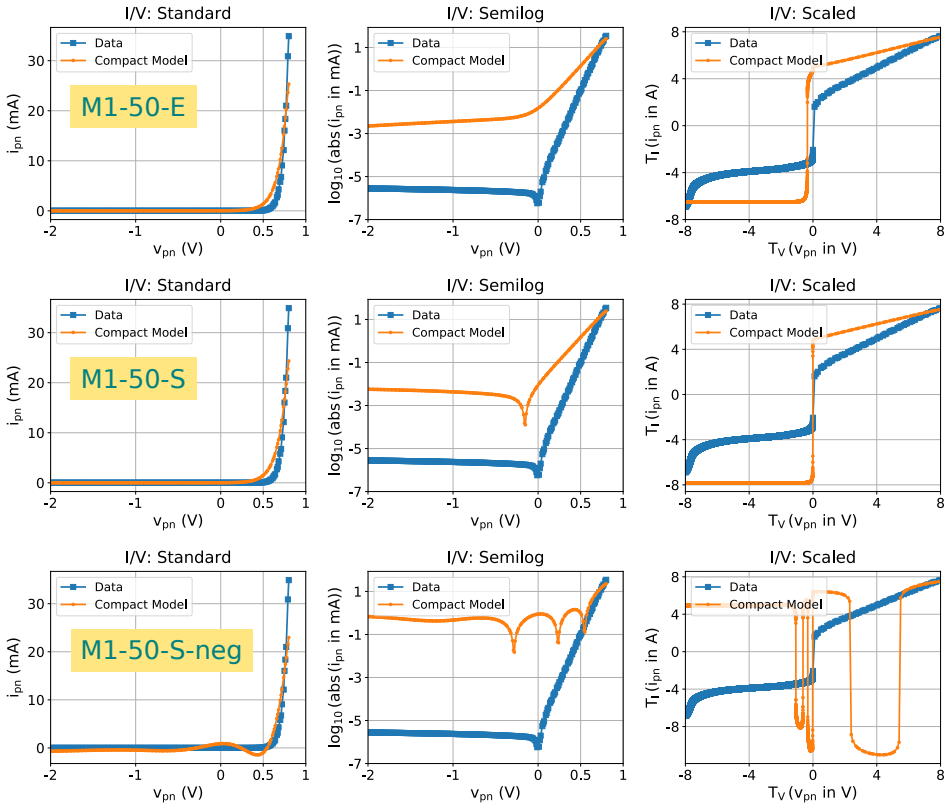

Figure 5: $I-V$ characteristics of DNN compact models learned without data transformations.

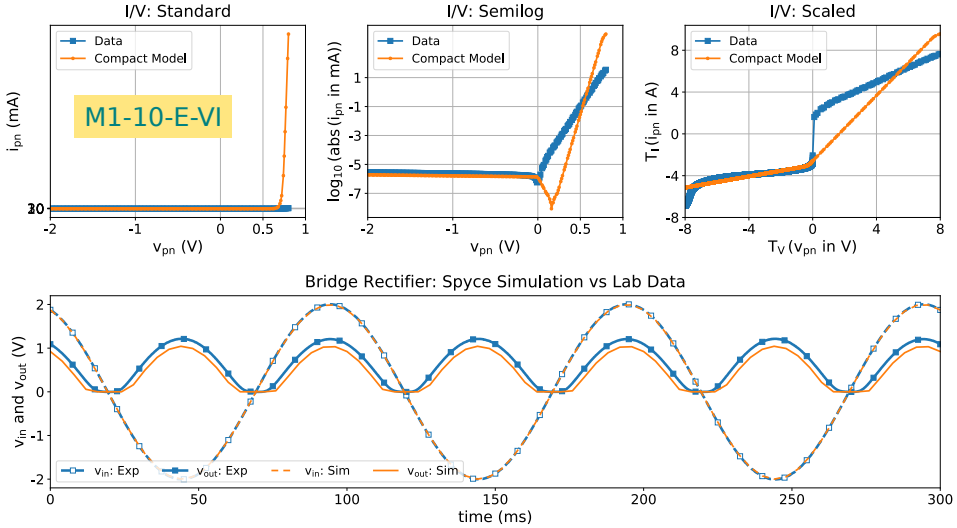

Figure 6: Simulations comparing the M1-10-E-VI DNN diode compact model against laboratory data

with sigmoid activation. Moreover, combining tanh/sigmoid with data transformations proved very resilient to other training parameters; no matter what we chose for the other parameters, the resulting DNN models were always accurate and converged robustly.

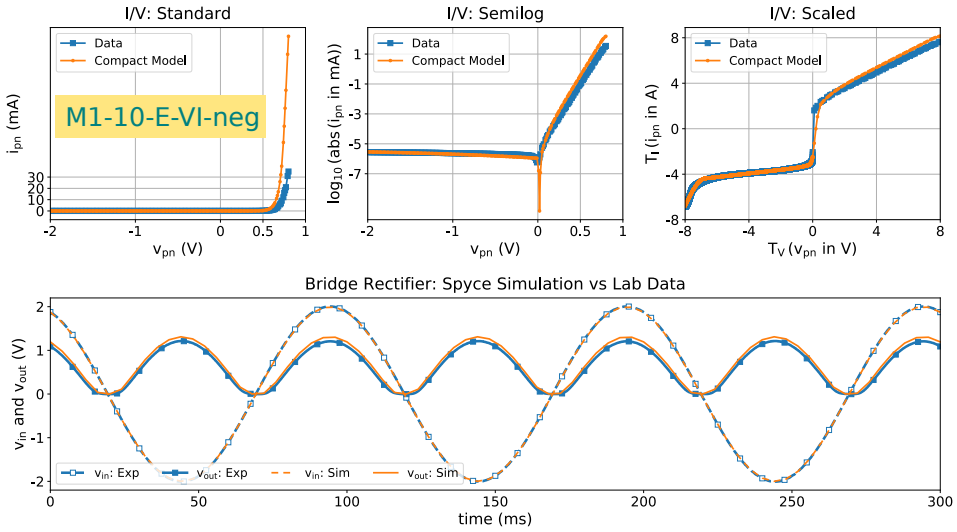

Figure 7: Simulations comparing the M1-10-E-VI-neg DNN diode compact model against laboratory data

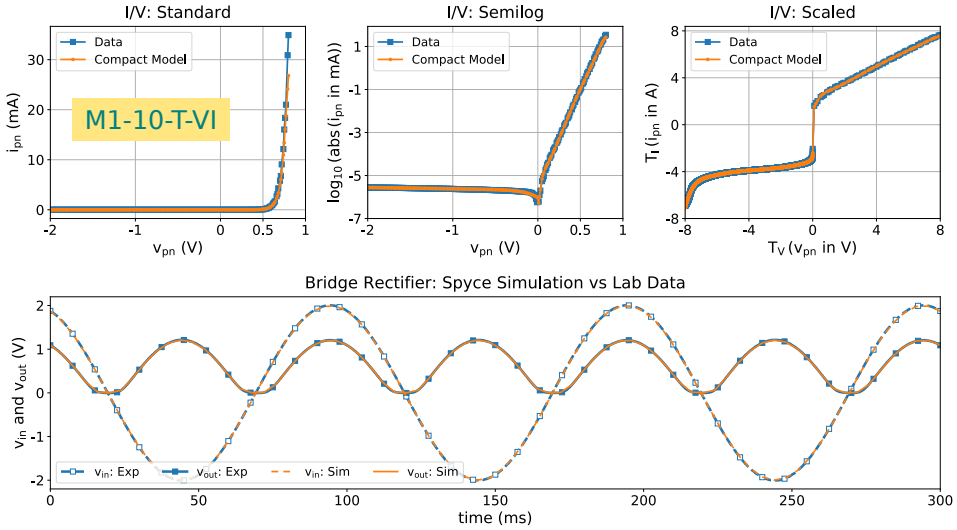

Figure 8: Simulations comparing the M1-10-T-VI DNN diode compact model against laboratory data

Finally, we found that applying the data transformation to i_{pn} was far more critical than applying it to v_{pn}. In fact, the performance of the DNN compact model showed only minor degradation (often invisible in plots) when we applied T_I but omitted T_V.

5. Conclusions

In this paper, we investigated three different regression approaches to develop data-driven compact models for a 1N4148 diode from laboratory measurements. The first two (TBI via cubic splines and GMLS) are examples of local parametric and non-parametric regression models. Simulation results demonstrate that both of these approaches deliver accurate and physically consistent compact device models that show excellent agreement with the laboratory $I-V$ measurements in all device operational regimes. Furthermore, both compact models performed robustly in circuit sim-

ulations where the simulated output of the bridge rectifier circuit was in an excellent agreement with laboratory measurements. The TBI model has higher memory requirements than GMLS but faster model evaluation. The use of the performant Compadre toolkit however enables highly efficient GMLS computation. As a result, both models can be deemed appropriate for the one-junction device considered in this work.

For two-junction devices that will be the subject of a forthcoming paper, TBI via multivariate splines will require data on a rectangular grid, whereas GMLS will not have such a restriction. For such devices, GMLS may be more appropriate for scattered electrical measurements.

On the one hand, our experiences with DNNs highlight the potential of this regression technique for compact model development. In particular, using data transformations and sigmoid/tanh activation, we were able to accurately regress a complex dataset spanning multiple scales and comprising more than 9000 data points by a shallow network with just 10 neurons. This makes our DNN compact model by far the most memory efficient of all three kinds of data-driven models considered here, and corroborates our conjecture that DNNs can provide computationally efficient compact models with small memory footprints.

At the same time, applying DNNs to approximate physics-based models requires deeper understanding of their properties as regression tools. Although DNNs potentially have the best generalizability of all the models considered in this work, their regression accuracy depends on a complex interplay between depth, width, activation functions, loss functions, constraint operators, *etc*. Although we have gained some insights on training DNNs for compact model development (as described in Section 4), we believe that our understanding is far from complete, and that devices with more than two terminals will pose significant additional challenges in identifying the best combinations of data transformations, architectures, constraint operators, *etc*.

Furthermore, circuit simulations underscore the importance of incorporating correct physical behavior in the data fit. For example, a good fit in the forward bias region is not enough to ensure robust and physically correct circuit simulations; this requires accurate representation of the zero crossing and the leakage current in the reverse bias regime as well. Without these features, the circuit simulation is unphysical at best, and often completely wrong.

The results in this paper suggest that *data transformations* are currently the most effective heuristics for achieving acceptable accuracy in data-driven DNN compact diode models. The transformations developed in this work aim to reduce the vast difference in scales present in the $I-V$ diode characteristic curve, which enables good data fits using a standard MSE loss function. A custom loss function that adapts to multiscale data is another potential option to improve DNN fit. We plan to pursue this work in the future.

Acknowledgments

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

The work of P. Bochev has also been supported by the U.S. Department of Energy, the Office of Science, and the Office of Advanced Scientific Computing Research under Award Number DE-SC-0000230927 as well as the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs) project.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, K. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, W. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale Machine Learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

M. Andrejević and V. B. Litovski. Electronic circuits modeling using artificial neural networks. *Journal of Automatic Control, University of Belgrade*, 13(1):31–37, 2003.

Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann. PRECICE: A fully parallel library for multi-physics surface coupling. *Computers & Fluids*, 141:250–258, 2016. Advances in Fluid-Structure Interaction.

Y. S. Chauhan, D. D. Lu, S. Venugopalan, S. Khandelwal, J. P. Duarte, N. Payvadosi, A. Niknejad, and C. Hu. *FinFET modeling for IC simulation and design: Using the BSIM-CMG standard*. Academic Press, 2015.

X. Chen, G. F. Wang, W. Zhou, Q. L. Zhang, and J. F. Xu. Application of neural networks for integrated circuit modeling. In *Advances in Neural Networks*, volume 3973, pages 1304–1312, 2006.

Z. Chen, M. Raginsky, and E. Rosenbaum. Verilog-A compatible recurrent neural network model for transient circuit simulation. In *EPEPS '17: The* 26^{th} *IEEE Conference on Electrical Performance of Electronic Packaging and Systems*, pages 1–3, 2017.

C. de Boor. *A Practical Guide to Splines*. Springer-Verlag New York, 1978.

diodes.com. https://www.diodes.com/assets/Datasheets/ds12019.pdf.

G. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3 edition, 1996.

I. Goodfellow, J. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

D. Gorissen, L. D. Tommasi, and K. Crombecq. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing & Applications*, 18:485–494, 2009.

A. Griewank and A. Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM, 2 edition, 2008.

A. Gupta. Table-based device modeling: Methods and applications. Master's thesis, EECS Department, University of California, Berkeley, 2018. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-66.html.

A. Gupta, T. Wang, A. G. Mahmutoglu, and J. Roychowdhury. STEAM: Spline-based tables for efficient and accurate device modelling. In *ASPDAC '17: The* 22^{nd} *Asia and South Pacific Design Automation Conference*, pages 463–468, 2017.

H. B. Hammouda, M. Mhiri, Z. Gafsi, and K. Besbes. Neural-based models of semiconductor devices for SPICE simulator. *American Journal of Applied Sciences*, pages 385–391, 2008.

C. Jiun-Shyan, H. Michael, and C. Sheng-Wei. Meshfree methods: Progress made after 20 years. 143(4): 04017001, 2017.

E. R. Keiter, K. V. Aadithya, T. Mei, T. V. Russo, R. L. Schiek, P. E. Sholander, H. K. Thornquist, and J. C. Verley. Xyce parallel electronic simulator: Users' guide, version 6.11. Technical Report SAND2019-5949, Sandia National Laboratories, Albuquerque, NM, 2019a.

E. R. Keiter, K. V. Aadithya, T. Mei, T. V. Russo, R. L. Schiek, P. E. Sholander, H. K. Thornquist, and J. C. Verley. Xyce parallel electronic simulator: Reference guide, version 6.11. Technical Report SAND2019-5950, Sandia National Laboratories, Albuquerque, NM, 2019b.

P. Kuberry, P. Bosler, and N. Trask. Compadre toolkit, February 2019. URL https://doi.org/10.5281/zenodo.2560287.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

Y. Lei, X. Huo, and B. Yan. Deep neural network for device modeling. In *EDTM '18: The* 2^{nd} *IEEE Electron Devices Technology and Manufacturing Conference*, pages 154–156, 2018.

M. Li, O. İrsoy, C. Cardie, and H. G. Xing. Physics-inspired neural networks for efficient device compact modeling. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 2:44–49, Dec 2016.

V. B. Litovski, Ž. Mrčarica, and T. Ilić. Simulation of non-linear magnetic circuits modelled using artificial neural network. *Simulation Practice and Theory*, 5(6):553–570, 1997.

W. Liu and C. Hu. BSIM3v3 MOSFET model. 9(03):671–701, 1998.

W. Liu and C. Hu. *BSIM4 and MOSFET modeling for IC simulation*. World Scientific, 2011.

P. B. L. Meijer. *Neural network applications in device and sub-circuit modelling for circuit simulation*. PhD thesis, Department of Chemical Engineering and Chemistry, Technische Universiteit Eindhoven, 1996.

D. Mirzaei, R. Schaback, and M. Dehghan. On generalized moving least squares and diffuse derivatives. *IMA Journal of Numerical Analysis*, 32(3):983–1000, 2012.

L. W. Nagel. *SPICE2: A computer program to simulate semiconductor circuits*. PhD thesis, EECS Department, University of California, Berkeley, 1975. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html.

J. A. A. Opschoor, P. C. Petersen, and C. Schwab. Deep ReLU networks and high-order finite element methods. Research Report 2019-07, ETH Zurich, January 2019. URL https://www.sam.math.ethz.ch/sam_reports/reports_final/reports2019/2019-07.pdf.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part 1): Data-driven solutions of non-linear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017. URL https://arxiv.org/abs/1711.10561.

sandia.gov. https://charon.sandia.gov/index.html.

W. Shockley. The theory of p-n junctions in semiconductors and p-n junction transistors. *Bell System Technical Journal*, 28(3):435–489, 1949.

S. R. Slattery. Mesh-free data transfer algorithms for partitioned multiphysics problems: Conservation, accuracy, and parallelism. *Journal of Computational Physics*, 307:164–188, 2016.

L. N. Trefethen. *Approximation theory and approximation practice*, volume 128. SIAM, 2013.

T. Wang and J. Roychowdhury. Multiphysics modelling and simulation in Berkeley MAPP. In *NEMO '16: The IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization*, pages 1–3, July 2016.

H. Wendland. *Scattered data approximation*. Cambridge University Press, 2004.

Wikipedia. https://en.wikipedia.org/wiki/1N4148_signal_diode.

A. H. Zaabab, Q.-J. Zhang, and M. S. Nakhla. Analysis and optimization of microwave circuits and devices using neural network models. In *IEEE MTT-S International Microwave Symposium Digest*, volume 1, pages 393–396, May 1994.

A. H. Zaabab, Q.-J. Zhang, and M. S. Nakhla. A neural network modeling approach to circuit optimization and statistical design. *IEEE Transactions on Microwave Theory and Techniques*, 43(6):1349–1358, June 1995.

A. H. Zaabab, Q.-J. Zhang, and M. S. Nakhla. Device and circuit-level modeling using neural networks with faster training based on network sparsity. *IEEE Transactions on Microwave Theory and Techniques*, 45 (10):1696–1704, Oct 1997.