# Physics-Inspired Neural Networks (Pi-NN) for Effcient Device Compact Modeling

**Article** · December 2016

DOI: 10.1109/JXCDC.2016.2636161



**4 authors**, including:

Mingda Li
Tianjin University


Claire Cardie
Cornell University


Huili grace Xing
Cornell University

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JXCDC.2016.2636161, IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

# Physics-Inspired Neural Networks (Pi-NN) for Effcient Device Compact Modeling

Mingda Li[1], Ozan İrsoy[2], Claire Cardie[2] and Huili Grace Xing[1]
*1. School of Electrical and Computer Engineering, Cornell University, NY 14850, USA*
*2. Department of Computer Science, Cornell University, NY 14860, USA*
*Emails: ml888@cornell.edu, grace.xing@cornell.edu*

***Abstract*—**We present a novel Physics-Inspired Neural Network (Pi-NN) approach for compact modeling. Development of high-quality compact models for devices is key to connect device science with applications. One recent approach is to treat compact modeling as a regression problem in machine learning. The most common learning algorithm to develop compact models is the Multilayer Perceptron (MLP) neural network. However, device compact models derived using MLP neural networks often exhibit unphysical behavior, which is eliminated in the Pi-NN approach proposed in this work since the Pi-NN incorporates fundamental device physics. As a result, smooth, accurate and computationally efficient device models can be learnt from discrete data points by using Pi-NN. This work sheds new light on the future of the neural network compact modeling.

## I. Introduction

Device compact modeling bridges device science to applications, therefore it plays a very important role in device research. There are two extremes for device modeling, one is purely physical and the other is purely empirical. Looking at these two extremes, a purely physical modeling method, such as NEMO [1], is computational expensive for use in circuit simulations, and a purely empirical modeling method, such as table look-up model, has limited generalization (extrapolation) ability. Therefore, to find a middle ground between purely physical and purely empirical models, the Electron Design Automation industry, represented by the Compact Model Coalition, chooses to promote physics-based compact models. These use fundamental device physics as the building blocks, then add empirical fitting to modify and merge different analytical physical expressions into smooth functions. However, developing high-quality physics-based compact models is very time-consuming, and therefore often not available for emerging devices. As an alternative, regression with machine learning can be used to model relationships between different variables with certain generalization abilities. Among different regression algorithms, the neural network modeling method has raised a lot of interests [2-4] given the fact that it is theoretically capable of arbitrarily accurate approximation to any function and its derivatives [5].

Compared to another widely used data-driven model: table look-up model, the neural network model performs better on the following three aspects: 1) *Scalability:* in order to achieve certain level of accuracy, the table look-up model needs a large amount of data, and the space complexity increases exponentially with increasing dimensions. In contrast, the neural network model is lightweight and scalable; 2) *Generalization:* the table look-up model has poor

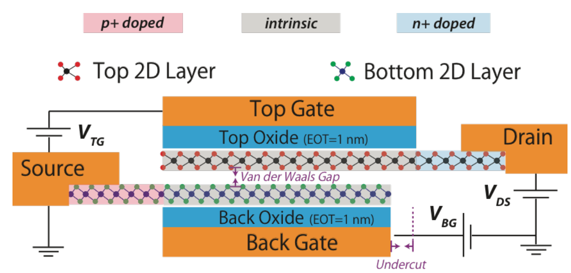



**Figure 1**: The schematic structure of the example emerging device modeled in this paper: an n-type Thin-TFET [7, 8]. Its I-V curves are obtained by sweeping the top gate ($V_{TG}$) with the back gate ($V_{BG}$) grounded.

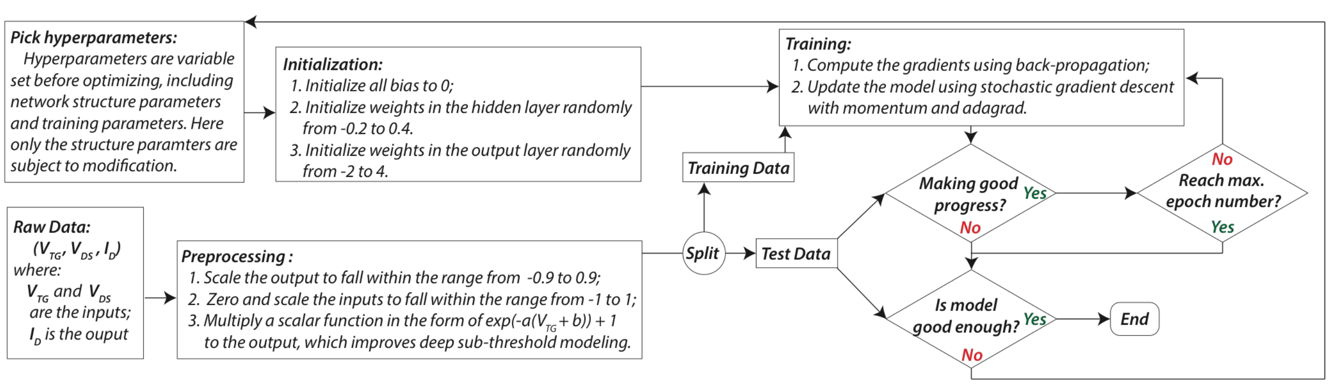



**Figure 2**: A training procedure for Artificial Neural Network (ANN) device compact modeling.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JXCDC.2016.2636161, IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

generalization performance. The polynomial fitting used in the table look-up model often has high out-of-sample errors. In contrast, by using correct learning algorithms, neural network model can be well generalized, which make it more robust against noises; 3) *Smoothness:* an ideal compact model needs to be infinitely differentiable. The table look-up model is not infinitely differentiable due to the nature of polynomial fitting. While using higher order polynomial fitting will improve the smoothness, it is at the expense of computation efficiency. Therefore, the table look-up model is not possible to be both smooth and computationally efficient. In contrast, the neural network model is guaranteed to be infinitely differentiable.

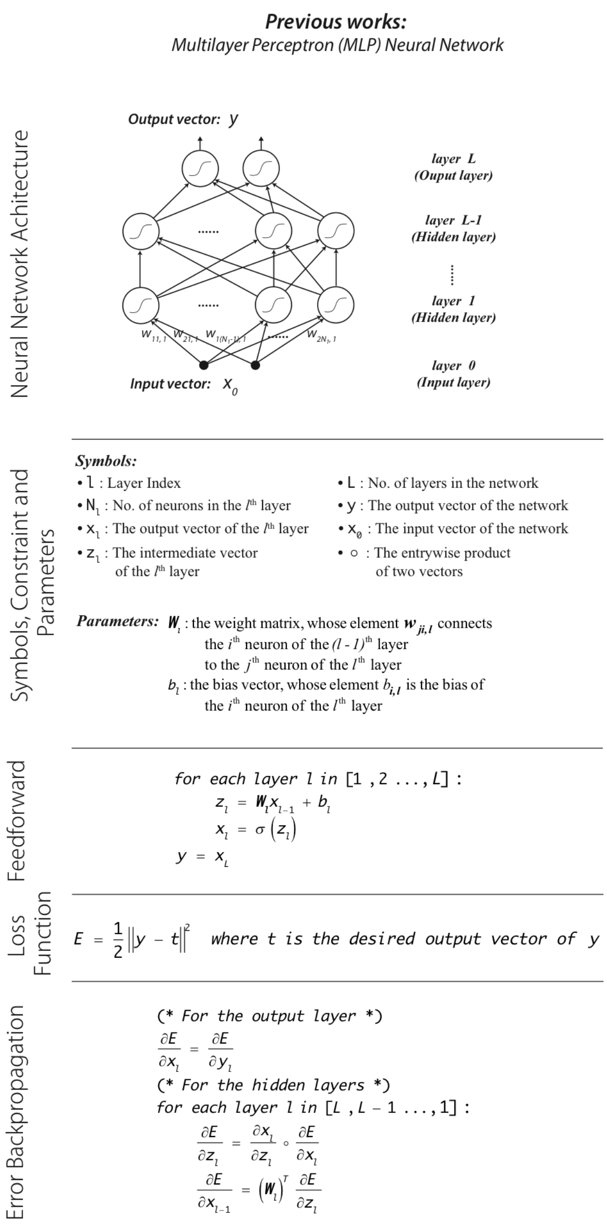



**Figure 3**: The Multiplayer Perception (MLP) neural network model.

Previous works [2-4] used Multilayer Perceptron (MLP) neural networks to develop compact models, which are prone to having unphysical behavior (see Fig. 4(e, f)). To eliminate the unphysical behavior, we have developed a novel neural network structure: Physics-Inspired Neural Network (Pi-NN), with fundamental device physics embedded. As a result, the Pi-NN can be trained to generate an accurate, smooth, and computational efficient device compact model.

## II. THIN-TFET AND TRAINING PROCEDURE

To illustrate the principles of Pi-NN, we develop compact models for the DC I-V curves of a transistor. Physics-based device modeling is typically challenging because the I-V curves are highly nonlinear and requires different analytical physical expressions in different bias windows. Therefore, it is usually difficult to handcraft an infinitely differentiable function from these physical expressions. Since high quality physics-based compact models are yet unavailable for emerging devices, such as Tunnel Field Effect Transistors (TFETs) [6], the neural network modeling approach has an added attraction. Here we used a novel device proposed in our group, a Thin-TFET [7] (**T**wo-dimensional **H**eterojunction **In**terlayer **T**unneling **F**ield **E**ffect **T**ransistor), as an example device for testing the neural network modeling techniques. The schematic device structure of an n-type Thin-TFET is shown in Fig. 1. The training data are simulated [7] for the top gate voltage ($V_{TG}$) from 0 to 0.4 V and the drain-source voltage ($V_{DS}$) from -0.1 to 0.4 V with a uniform step of 0.01 V, while the test data are for $V_{TG}$ from 0.005 to 0.405 V and $V_{DS}$ from -0.095 to 0.405 V with a uniform step of 0.01 V. The detailed training procedure is shown in Fig. 2. In the pre-processing step in Fig. 2, a scaler function in the form of $exp(-a(V_{TG}+b))+1$ is multiplied to the output, which helps improve deep sub-threshold modeling. The value of *a* and *b* are chosen by following the general rules described below: Since this scalar function is used to improve deep sub-threshold modeling, we should choose *a* and *b* such as:

$$exp\big(-a(V_{TG}+b)\big)+1\begin{cases}\cong 1 & when\ V_{TG}>V_{TH}\\ \gg 1 & when\ V_{TG}<V_{TH}\end{cases}$$

where $V_{TH}$ is the threshold voltage. Therefore $|b|$ should be smaller than the threshold voltage and *a* is approximately the slope of $I_D$–$V_{TG}$ curves in the deep sub-threshold region. The final values of *a* and *b* are fine-tuned by trial and error. For example, in this work, the threshold voltage of Thin-TFET is around 100 mV, so *b* is set to be -50 mV. As for *a*, the subthreshold swing for $V_{TG} < 50$ mV is around 17 mV/dec, therefore *a* is set to be $1/17 \times 2.3 = 0.135$ $mV^{-1}$ (where 2.3 comes from $ln(10) \approx 2.3$).

## III. MLP NEURAL NETWORK MODELING AND UNPHYSICAL BEHAVIOR

In this section, we use the MLP neural network to generate a compact model for the DC I-V curves of the Thin-TFET. The MLP neural network architecture and its well-established learning algorithms are shown in Fig. 3 [9]. After some initial training, we choose to use MLP neural networks with two

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JXCDC.2016.2636161, IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

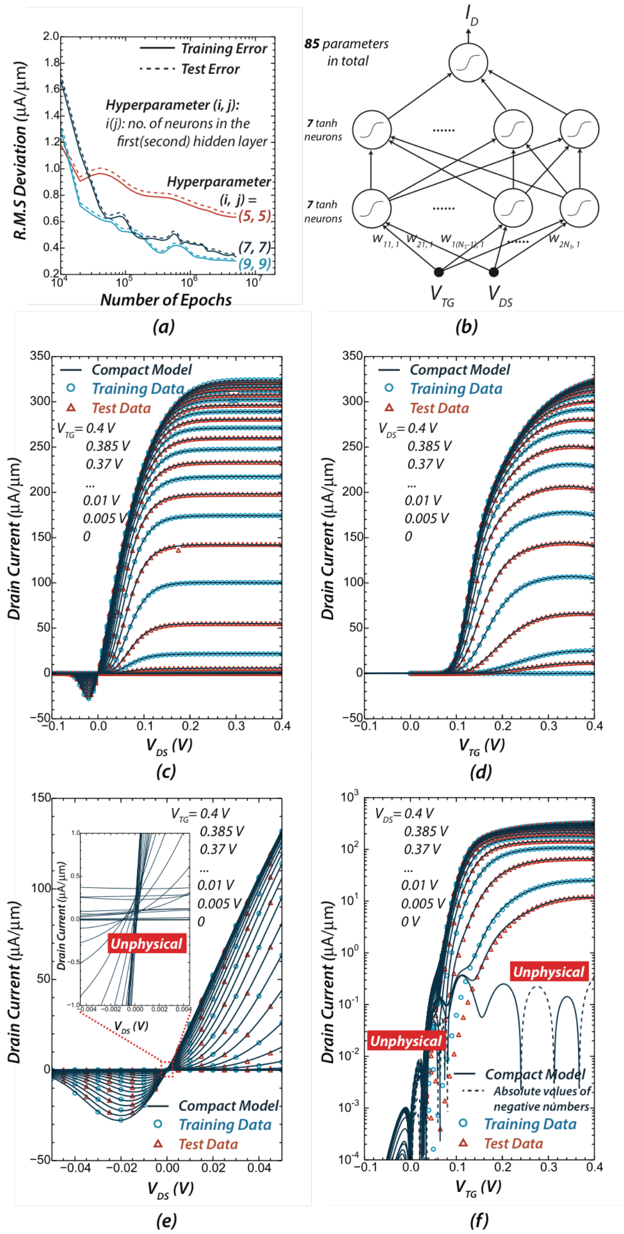



**Figure 4**: The compact model of the n-type Thin-TFET derived based on the MLP neural network widely used in previous works [2-4], (a) the training errors and test errors for a variety of hyper-parameters; (b) the MLP neural network with 7 *tanh* neurons in the first and second hidden layers. From (c) to (f), the I-V curves generated by the MLP neural network shown in (b) are plotted along with the training data and the test data: (c) $I_D$ versus $V_{DS}$ at different $V_{TG}$; (d) $I_D$ vs. $V_{TG}$ at different $V_{DS}$ in linear scale; (e) $I_D$ vs. $V_{DS}$ at different $V_{TG}$ around $V_{DS} = 0$, the embedded plot shows unphysical $I_D$-$V_{DS}$ relationships around $V_{DS}$ equals 0; (f) $I_D$ vs. $V_{TG}$ at different $V_{DS}$ in semi-log scale, unphysical oscillation of $I_D$ around zero appears in the sub-threshold region and when $V_{DS} = 0$.

hidden layers and defined its hyper-parameter as *(i, j)*, where *i* is the number of neurons in the first hidden layer and *j* is the number of neurons in the second hidden layer. Each neuron uses the hyperbolic tangent function $tanh(x)=(e^x-e^{-x})/(e^x+e^{-x})$ as the activation function. By choosing the hyper-parameter *(i, j)* to be (5, 5), (7, 7) and (9, 9), these three MLP neural networks were trained for 5 million epochs. Using the loss function defined in Fig. 3, the root-mean-squared (R.M.S) deviations for training data and test data are plotted in Fig. 4(a). The test errors are used to evaluate the generalization ability of the model, namely how the model fit the unseen data. As shown in Fig. 4(a), the test errors stay close to the training errors, which indicated a good generalization. We choose to plot the I-V curves modeled by the MLP neural network with 7 *tanh* neurons in the first and second hidden layers as shown in Fig. 4(b), which gives a neural network with 15 neurons and 85 parameters in total. Figure 4(c-f) show the I-V curves generated by the MLP neural network compact model along with the training data and the test data. Good fitting in the linear scale is achieved for both the $I_D$-$V_{DS}$ and the $I_D$-$V_{TG}$ curves. However, if we zoom in the region near $V_{DS} = 0$, $I_D$ is not zero when $V_{DS}$ is zero, indicating the $I_D$-$V_{DS}$ relationship is unphysical around $V_{DS} = 0$ (see Fig. 4(e) and the inset). Moreover, the $I_D$-$V_{TG}$ relationship is also unphysical in the sub-threshold region (shown in Fig. 4(f)). The fundamental reason of these unphysical behaviors is that the MLP neural network has no knowledge of the device physics; therefore, the fitting is no longer physical when $I_D$ is very small. In order to eliminate these unphysical behaviors, we have to design a neural network with *a priori* knowledge of the fundamental device physics.

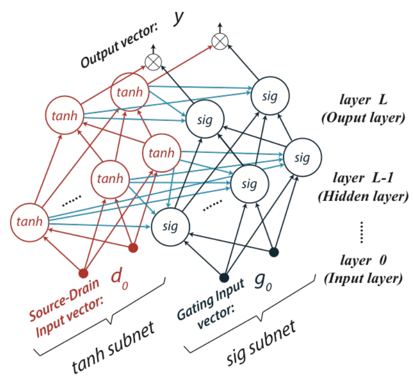
## IV. A Physics-Inpired Neural Network Design

First, we note that the inputs $V_{DS}$ and $V_{TG}$ are related to two different physical effects: $V_{DS}$ drives the current through the device while $V_{TG}$ controls the channel potential profile to change the magnitude of the current. Therefore, $V_{DS}$ and $V_{TG}$ should be fed to two different neural networks. According to the fundamental device physics, we know $I_D$-$V_{DS}$ curves have a linear region at small $V_{DS}$ and a saturation region at large $V_{DS}$. This behavior is similar to a *tanh* function. This indicates $V_{DS}$ should be fed into a neural network with *tanh* activation functions (*tanh subnet*). To ensure $I_D$ equals zero when $V_{DS}$ equals zero, all the *tanh* neurons in the *tanh subnet* must have no bias terms. On the other hand, the $I_D$-$V_{TG}$ curves have an exponential turn-on in the sub-threshold region and then become a polynomial in the ON region. This is best simulated as a sigmoid function $sig(x)=1/(1+e^{-x})$. Therefore, $V_{TG}$ is fed into a neural network with sigmoid activation functions (*sig subnet*). It should be noted that we assumed gate leakage current is negligible, so $V_{TG}$ would not change the sign of $I_D$. The final drain current is the entrywise product of the outputs of the *tanh subnet* and the *sig subnet*. This entrywise product reflects the control of $V_{TG}$ on the drain current driven by $V_{DS}$. In addition, $V_{DS}$ can affect the channel potential profile controlled by $V_{TG}$ due to various non-ideal effects such as the short channel effects. A simple but effective remedy for this is to add weighted connections from each layer in the *tanh subnet* to its corresponding layer in the *sig subnet*. By embedding the above device physics in a neural network structure, we arrive at the Physics-Inspired Neural Network (Pi-NN). The Pi-NN architecture and its pseudo-codes for the feed-forward and error back-propagation algorithms are shown in Fig. 5. This novel neural network is reminiscent of the peephole Long-Short Term

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JXCDC.2016.2636161, IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

***This work:***
*Physics-Inspired Neural Network (Pi-NN)*

Neural Network Achitecture




Symbols, Constraint and Parameters

***Symbols:***

- $l$ : Layer Index • $L$ : No. of layers in the network
- $N_l^{T(S)}$ : No. of neurons in the $l^{th}$ layer of the ***tanh(sig) subnet***
- $y$ : The output vector of the network
- $d_0(g_0)$ : The input vector of the ***tanh(sig) subnet***
- $d_l(g_l)$: The output vector of the $l^{th}$ layer in the ***tanh(sig) subnet***
- $z_l$ : The intermediate vector of the $l^{th}$ layer in the ***tanh(sig) subnet***
- $\circ$ : The entrywise product of two vectors

***Constraint:*** $N_l^T = N_l^S$ (The output layers of the two subnet have same no. of neurons)

***Parameters:***

$W_l^{T(S)}$ : the weight matrix for the *tanh(sig) subnet, whose element* $w_{ji,l}^{T(S)}$ connects the $i^{th}$ neuron of the $(l - 1)^{th}$ layer to the $j^{th}$ neuron of the $l^{th}$ layer

$W_l^P$ : the weight matrix for the peephole connections from the *tanh subnet* to the *tanh subnet, whose element* $w_{ji,l}^P$ connects the $i^{th}$ neuron of the $l^{th}$ layer in the *tanh subnet* to the $j^{th}$ neuron of the $l^{th}$ layer in the *sig subnet*

$b_l^S$ : the bias vector, whose element $b_{i,l}^S$ is the bias of the $i^{th}$ neuron of the $l^{th}$ layer in the *sig subnet*

*(**Note**: There is **no** bias in the tanh neurons of the tanh subnet)*

Feedforward

for each layer $l$ in $[1, 2 \ldots, L]$ :

$z_l^T = W_l^T d_{l-1}$ (* tanh subnet *)

$d_l = tanh\left(z_l^T\right)$ (* tanh subnet *)

$z_l^S = W_l^P d_l + W_l^S g_{l-1} + b_l^S$

(* sig subnet & peephole connections *)

$g_l = sig\left(z_l^S\right)$ (* tanh subnet *)

$y = d_L \circ g_L$

Loss Function

$E = \frac{1}{2}\left\|y - t\right\|^2$ where $t$ is the desired output vector of $y$

Error Backpropagation

(* For the output layer *)

$\frac{\partial E}{\partial g_L} = d_L \circ \frac{\partial E}{\partial y}, \quad \frac{\partial E}{\partial d_L} = g_L \circ \frac{\partial E}{\partial y}$

(* For the hidden layers *)

for each layer $l$ in $[L, L-1 \ldots, 1]$ :

$\frac{\partial E}{\partial z_l^S} = g_l \circ \left(\vec{1} - g_l\right) \circ \frac{\partial E}{\partial g_l}$ (* sig subnet *)

$\frac{\partial E}{\partial g_{l-1}} = \left(W_l^S\right)^T \frac{\partial E}{\partial z_l^S}$ (* sig subnet *)

$\frac{\partial E}{\partial z_{j,l}^T} = \left(\vec{1} - d_l \circ d_l\right) \circ \left(\frac{\partial E}{\partial d_l} + \left(W_l^P\right)^T \frac{\partial E}{\partial z_l^S}\right)$

(* tanh subnet & peephole connections *)

$\frac{\partial E}{\partial d_{l-1}} = \left(W_l^T\right)^T \frac{\partial E}{\partial z_l^T}$ (* tanh subnet *)

**Figure 5**: The Physics-Inspired Neural Network (Pi-NN) model. *(Source code available at https://github.com/Oscarlight/Pi-NN)*

Memory (LSTM) [10], with the notable difference that the Pi-NN does not propagate through time. After all, the Pi-NN architecture can model the I-V curves of any transistor if two conditions are satisfied: 1) $I_D$ equals zeros if and only if $V_{DS}$ equals zero; 2) $V_G$ doesn't change the sign of $I_D$ (i.e. the gate leakage current is negligible).

## V. Physics-Inspired Neural Network Modeling

After initial training, we choose to use Pi-NNs with one hidden layer and define the hyper-parameter as (*m, n*), where *m* is the number of the *tanh* neurons in the hidden layer and *n* is the number of the *sigmoid* neurons in the same hidden layer. The test errors stay close to the training errors as shown in Fig. 6(a), which indicates good generalization. The model complexity is gradually increased from the hyper-parameter (2, 2) to (3, 4). From Fig. 6(a), the model with the hyper-parameter (2, 3) is the simplest model with converging training and test error. More complex models can achieve smaller training and test error but the improvement is not significant enough to justify the increased complexity. Balancing between model complexity and accuracy, we choose the model with the hyper-parameter (2, 3) as shown in Fig. 6(b), which give a small Pi-NN model with only 7 neurons and 20 parameters in total. Excellent modeling is demonstrated in both the ON region (shown in Fig. 6(c, d)) and the sub-threshold region (shown in Fig. 6(f)). The $I_D$-$V_{DS}$ relationship around $V_{DS}$ equals zero is shown in Fig. 6(e). All the unphysical behaviors that appeared in the MLP neural network model have been eliminated. Moreover, thanks to the embedded device physics, the Pi-NN requires much less parameters than the MLP neural network, which results in a smaller, more efficient compact model.

## VI. Conclusions

Motivated by the need of high-quality compact models for emerging devices, we have proposed a novel neural network: Pi-NN, for compact modeling. With fundamental device physics incorporated, the Pi-NN method can produce accurate, smooth and computational efficient transistor models with good generalization ability. Thin-TFET is presented as an example to illustrate the capabilities of Pi-NN: a relatively small compact model is achieved with excellent fitting in both the ON and the sub-threshold region of the Thin-TFET. The charge-voltage Q-V relationships in a device are highly desirable for circuit design. It is possible to construct Q-V relations from the device C-V data (not shown here). However, since the sign of the terminal charge density is dependent on both $V_{TG}$ and $V_{DS}$, the Pi-NN architecture cannot be directly applied for modeling Q-V relations. The walk-around is to connect $V_{TG}$ and $V_{DS}$ to both the *tanh* subnet and the *sig* subnet in the Pi-NN, and add the bias terms in the *tanh* neurons. This modified Pi-NN is compatible with the adjoint neural network method for constructing Q-V relation from C-V measurements [2, 11]. However, this modified Pi-NN architecture has no apparent advantage over the MLP architecture for Q-V modeling. Future work will focus on how to better integrate Q-V modeling into the Pi-NN framework.

Finally, the Pi-NN approach is readily implementable in commercial measurement and modeling systems.

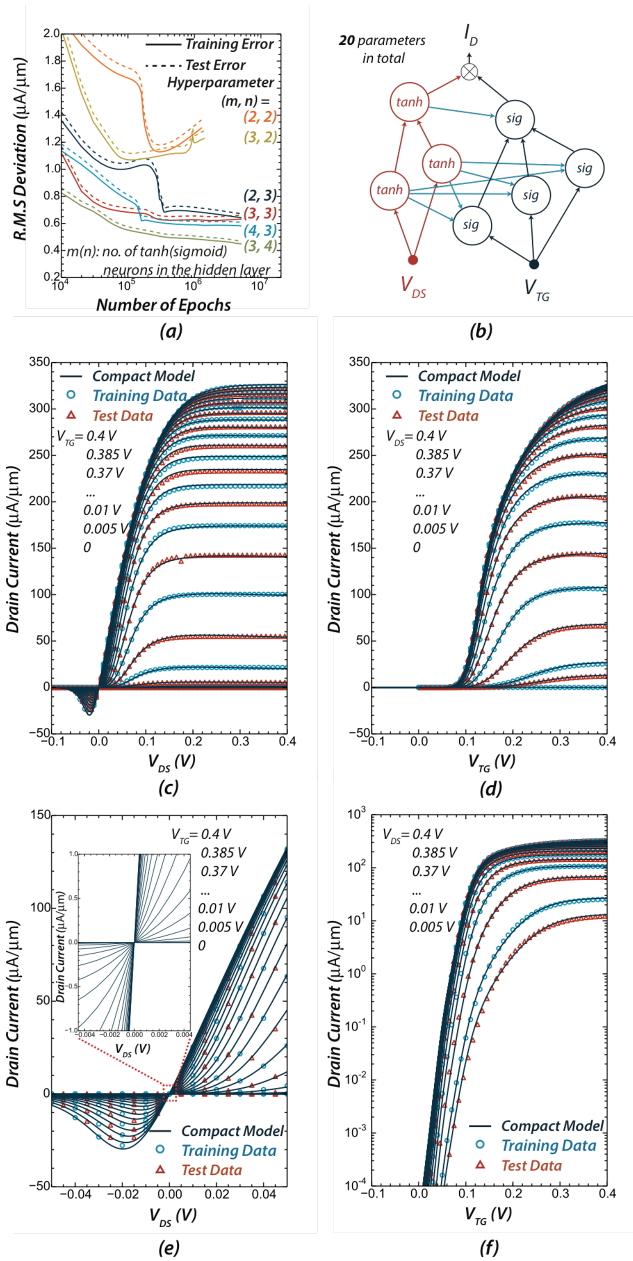



**Figure 6**: The compact model of the n-type Thin-TFET derived based on the Pi-NN developed in this work, (a) the training errors and test errors for a variety of hyper-parameters. (b) the Pi-NN model with 2 *tanh* neurons and 3 *sigmoid* neurons in the hidden layer. From (c) to (f), the I-V curves generated by the Pi-NN model shown in (b) are plotted along with the training data and the test data: (c) $I_D$ versus $V_{DS}$ at different $V_{TG}$; (d) $I_D$ vs. $V_{TG}$ at different $V_{DS}$ in linear scale; (e) $I_D$ vs. $V_{DS}$ at different $V_{TG}$ around $V_{DS}$ = 0, the embedded plot shows well-behaved $I_D$-$V_{DS}$ relationship around $V_{DS}$ = 0; (f) $I_D$ vs. $V_{TG}$ at different $V_{DS}$ in semi-log scale, good fitting is achieved in the sub-threshold region. All the unphysical behaviors of the MLP neural network are eliminated, and the size of the neural network is largely reduced.

## Acknowledgment

This work was supported in part by the Center for Low Energy Systems Technology (LEAST), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA, and the National Science Foundation and Air Force Office of Scientific Research under Grant EFRI 2-DARE (1433490).

## References

[1] Steiger, Sebastian, et al. "NEMO5: a parallel multiscale nanoelectronics modeling tool." (2011).

[2] Xu, Jianjun, and David E. Root. "Advances in artificial neural network models of active devices." *2015 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*. IEEE, 2015.

[3] Hammouda, H. Ben, et al. "Neural based models of semiconductor devices for SPICE simulator." American Journal of Applied Sciences 5.4 (2008): 385-391.

[4] Wang, Fang, and Qi-Jun Zhang. "Knowledge-based neural models for microwave design." IEEE Transactions on Microwave Theory and Techniques 45.12 (1997): 2333-2343.

[5] Hornik, Kurt. "Approximation capabilities of multilayer feedforward networks." *Neural networks* 4.2 (1991): 251-257.

[6] Seabaugh, Alan C., and Qin Zhang. "Low-voltage tunnel transistors for beyond CMOS logic." *Proceedings of the IEEE* 98.12 (2010): 2095-2110.

[7] Li, Mingda Oscar, et al. "Two-dimensional heterojunction interlayer tunneling field effect transistors (Thin-TFETs)." *IEEE Journal of the Electron Devices Society* 3.3 (2015): 200-207.

[8] Li, Mingda Oscar, et al. "Single particle transport in two-dimensional heterojunction interlayer tunneling field effect transistor." *Journal of Applied Physics* 115.7 (2014): 074508.

[9] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Cognitive modeling* 5.3 (1988): 1.

[10] Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." *Journal of machine learning research* 3. Aug (2002): 115-143.

[11] Xu, Jianjun, et al. "Exact adjoint sensitivity analysis for neural-based microwave modeling and design." IEEE Transactions on Microwave Theory and Techniques 51.1 (2003): 226-237.