Japanese Journal of Applied Physics

The Japan Society of Applied Physics

ACCEPTED MANUSCRIPT

Comprehensive studies on deep learning applicable to TCAD

To cite this article before publication: Sanghoon Myung *et al* 2023 *Jpn. J. Appl. Phys.* in press https://doi.org/10.35848/1347-4065/acbaa6

Comprehensive Studies on Deep Learning Applicable to TCAD

Sanghoon Myung, Byungseon Choi, Wonik Jang, Jinwoo Kim, In Huh, Jae Myung Choe, Young-Gu Kim, and Dae Sin Kim

Computational Science and Engineering Team, Innovation Center, Samsung Electronics.

E-mail: shoon.myung@samsung.com

TCAD simulation has incessantly solved many complex problems, but it becomes demanding that alternatives be found because TCAD simulation cannot provide the precise and fast prediction in the nano-scale era. With the success story on deep learning in research area, many big data companies have attempted to introduce deep learning to support or replace TCAD simulation. The reason is deep learning models has great potential that solves the problems of the TCAD simulation in terms of execution time, coverage. This paper aims to describe various scenarios of deep learning applicable to TCAD. We firstly describe an application that supplies TCAD data to the deep learning model although TCAD simulation is not calibrated. We then review various approaches that mimic TCAD simulation itself. We finally introduce an application that deep learning model automatically calibrates TCAD models to the measurement without experts. In each scenario, we review the related papers and compare pros and cons.

1. Introduction

The success of semiconductor process development has long relied on technology computer-aided design (TCAD) simulators, which can calculate physical phenomena described as complex partial differential equations (PDEs). Since the advent of the 3-D device, however, recent process development has gone through tough times. There are two reasons; one is the TCAD simulator requires high computational costs on predicting the 3-D transistors and the other is the TCAD models cannot fully describe the physical phenomena of modern technologies.

Recently, data-driven approaches (e.g., deep learning) received great attention. The reason is the research has reported that deep learning has exceeded human performance in terms of accuracy across various tasks such as vision recognition[1] or natural language processing[2]. Thus, with such successful stories, data-driven approaches have initiated expanding their application into many scientific and industrial fields.

Most data in the semiconductor fields, however, is not available due to high measurement costs. Thus, data-driven approaches have been applied to limited applications such as chip design automation[3], test vector generation[4], process optimization[5], and defect detection[6-7]. Such studies have something in common they can gather massive data, readily. The reason is the data that they used is easily generated by SPICE simulation or necessarily measured in the semiconductor field (e.g., defect).

In that sense, at a low cost, TCAD simulation can generate the data that could not be measured due to the costs. Thus, many studies[8-11] have applied data-driven approaches to their problem with TCAD data. This paper categorizes the previous studies into three main folds.

The first is to leverage TCAD data in terms of data augmentation. Deep learning often underperforms due to scarce measurement data. In such cases, we can utilize TCAD data for training. Many previous studies[8-11] have demonstrated that deep learning trained with TCAD data presents superior results. The second[12-14] is to imitate TCAD simulation per se. Once the inputs and outputs of TCAD are well-defined, deep learning can learn massive TCAD data. Many previous studies have proven that such trained deep learning models can emulate TCAD simulation well. The last is to automatically calibrate the TCAD models to the measurement. Typically TCAD calibration has been performed in a heuristic manner, which widely varies the execution time depending on the ability of experts. To alleviate this problem, some works[15] have introduced deep learning to learn the relationship between fitting parameters and measurement data and then have made optimization with deep

learning to obtain fitting parameters to fit the TCAD simulation to the measurement data. This paper aims to review the comprehensive studies of deep learning models utilizing TCAD simulations. In particular, this paper will compare the previous works on a common problem to analyze their pros and cons. To our best knowledge, such analysis is the first attempt in the semiconductor field. The rest of this paper scrutinizes various deep learning applications to TCAD simulation; Section II explores an application that *augments* the training data with TCAD simulation when there is a lack of the measurements, Section III examines an application that *mimics* TCAD simulation itself, Section IV investigates an application that *automatically calibrates* the TCAD models to the measurements. Finally, we summarize our conclusion in Section V.

2. Data Augmentation

This section discusses the application to augment the training data with TCAD to improve the performance of deep learning. Typically, it is hard to gather the massive data due to a heavy cost though deep learning needs the massive data. In this case, one of the proper solutions is to leverage TCAD data because it facilitates to generating as much TCAD data as deep learning needs. Many previous studies[9-11] have demonstrated that deep learning training with TCAD-augmented data can present the better results than training with only measurements. The reason is we can use the many TCAD data consistent with the measurements thanks to well-calibrated TCAD simulation. However, often it is difficult to calibrate the TCAD models to the measurement in case that TCAD uses the complex models that have the many fitting parameters.

Thus, this paper[8] will present the method that can utilize the TCAD data to train deep learning although TCAD simulation is not calibrated with the measurements. As one of the cases, this paper introduces the example to predict the Deep Trench Isolation (DTI) profile[16] from an input recipe. Figs. 1 (a) and (b) depicts the examples of input recipes to etch the DTI profile and the outputs measured by destructive inspection, respectively. To predict the etching profile, Fig. 1 (c) shows that TCAD simulation has to solve the complex phenomena based on physical equations. Since such the TCAD models have many calibration parameters, it has struggled to predict the accurate etching profile. In addition to this problem, we should calibrate TCAD models with equipment by equipment, because TCAD models cannot consider the equipment dispersion. Such the problems motivates us to introduce deep learning as data-driven approach. Fortunately, there are the massive data measured by

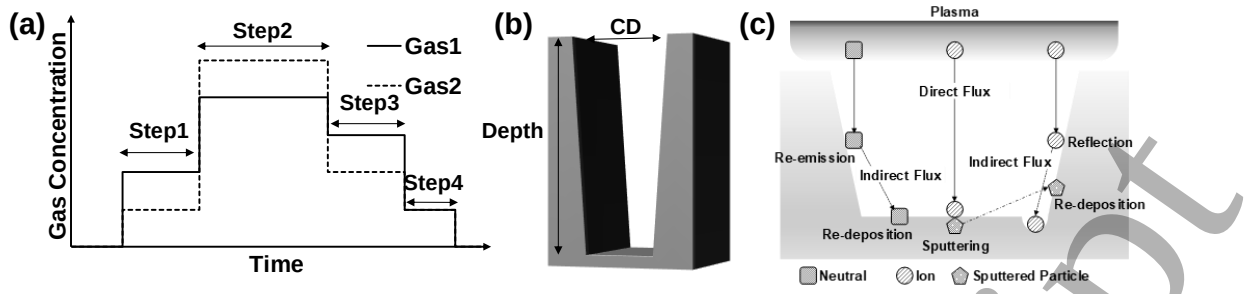

Fig. 1. The examples of (a) input recipe and (b) DTI profile. CD is critical dimension. (c) Illustration of physical effects on etching processes.

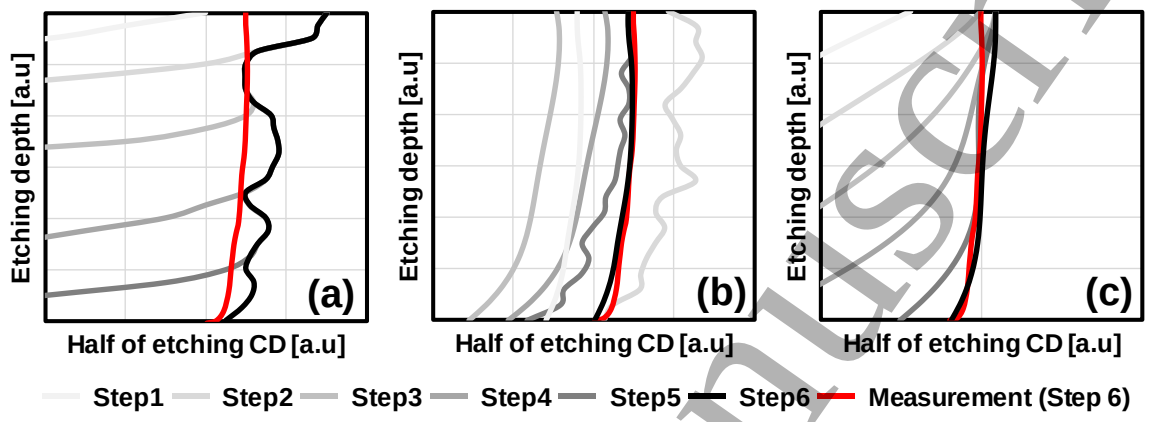

Fig. 2. The prediction results of (a) TCAD simulation, (b) deep learning model trained with only measurements, and (c) deep learning model with combined data trained with TCAD simulation.

destructive inspection. With the massive data, we trained recurrent neural network (RNN)-based model that can interpret sequential data. As a result, deep learning model that learned the measurement data can present the final DTI profile from an arbitrary recipe while TCAD cannot, as depicted in Fig. 2. Such a result demonstrates deep learning model can be a good alternative to TCAD simulation if there are lots of data.

As aforementioned, however, we found that deep learning may over-fit albeit training with the massive data. To examine how much the model learned the etching physics well, we conducted an additional simulation that makes the model infer the DTI profile as time advances. Interestingly, as illustrated in Fig. 2, this simulation reveals that deep learning model fails to estimate the intermediate DTI profile. The reason is such the profiles are never available when training. To tackle this problem, there are several typical solutions. The first is to gather the intermediate DTI profile per each step, then training the model with them. Although this method is the most accurate, it imposes us to the heavy cost. The second is to apply the proper inductive biases to the model, which helps the model learn the intended relation between steps. For the etching processes, our previous work[8] has presented the one of the proper inductive biases. The last is to leverage TCAD data, which this paper will focus on. Different from the previous works[9-11] that used the well-calibrated TCAD data, we utilize

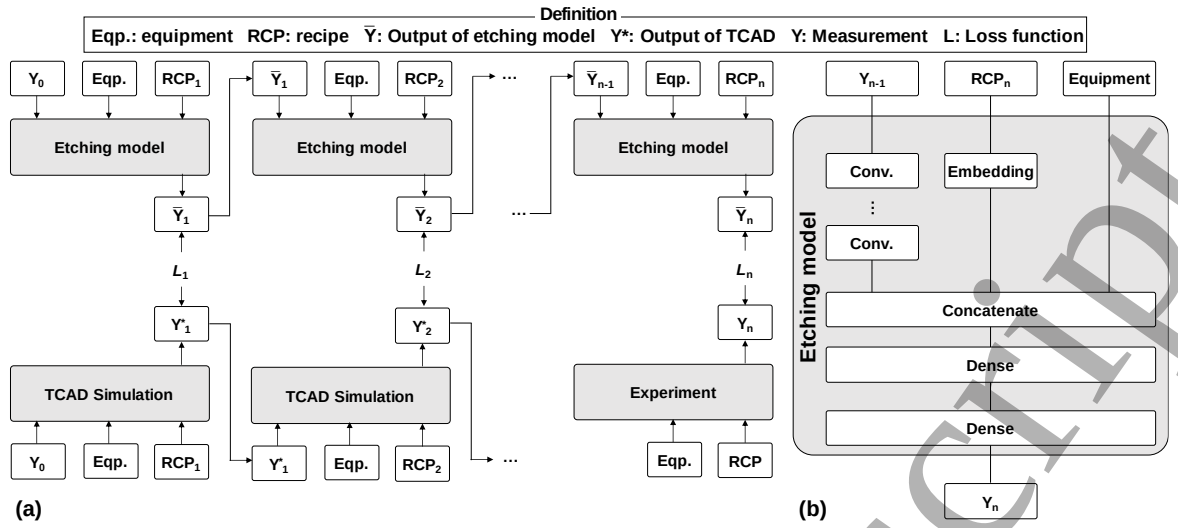

Fig. 3. (a) Illustration of the etching model with multitask learning framework. Each subscript is the step sequence. (b) Illustration of the etching model architecture.

the uncalibrated TCAD data to train the deep learning model. Figure 3 illustrates the multi-task learning framework[17] that enables the model to learn both TCAD and measurement data, simultaneously. It uses the measurement data for the final DTI profile, but utilizes the TCAD data for the intermediate DTI profile owing to no data. In this case, because data imbalance problem occurs, we design the weighted loss function as following:

$$L = \frac{\alpha}{N-1}\sum_{n=1}^{N-1}(y_n^* - \bar{y}_n)^2 + \beta(y_N - \bar{y}_N)^2,$$

where y^* denotes the output of TCAD simulation, $\bar{y}$ the output of the etching model, y the measurement (i.e., final etching profile), N the length of step sequence, and α, β the fitting parameters to determine which data to focus on. Since the model has to more fit the measurement data rather than TCAD simulation data, typically β is much greater than α (i.e., $\beta \gg \alpha$). Thus, we choose $\beta = 10$ and $\alpha = 1$. As a result, Fig. 2 (c) shows leveraging TCAD data (i.e., multitask learning with the weighted loss function) can estimate the intermediate DTI profile well matched with TCAD results as well as predict the final profile consistent with the measurement. This result, however, has limitations in verifying that they are accurate modeling because the finally measured intermediate profiles were not available due to high cost.

3. Real Time TCAD

This section reviews the application to mimic TCAD simulation itself. Generally, TCAD simulation composes of process and device simulation. The process simulation plays a role

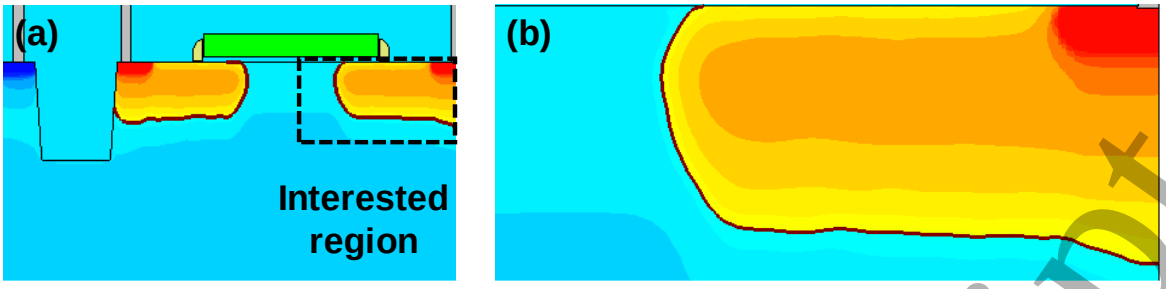

Fig. 4. (a) The examples of full region. (b) Doping profile for the enlarged region of interest.

in estimating the doping profile of a transistor from the process condition. The device simulation is responsible for calculating the current-*versus*-voltage curve (*I-V* curve) from the output of the process simulation (i.e., the doping profile). This section will discuss the comprehensive deep learning models to mimic each simulation.

3.1 Process simulation

TCAD process simulation predicts the doping profile of a transistor. The predicted doping profile serves two purposes. The first is to insert it into the TCAD device simulation, which calculates the *I-V* characteristics. The other is to analyze the effects of the process condition that we design, visually. In this case, we often convert the result of the TCAD process simulation to an image that focuses on the interesting region such as channel, source, and drain region. From the deep learning model perspective, the data type of the former and latter are regarded as the real value (unstructured mesh data) and the 2-dimensional (2-D) image (meshless data), respectively. Hence, the previous works[12, 18] have suggested the convolutional neural network (CNN)-based deep learning models to predict the doping profile in accordance with those types. As a review paper, we will review and analyze the previous studies. To this end, we will compare and evaluate the architectures and performance of all models on a common problem.

As a common problem, we introduce a 130-nm process for display driver integrated circuits (DDI), which also has been described in this work[12]. The input variables of the TCAD simulation composes of two types: ten variables related to structure and twenty-four variables associated with ion implantation condition. Figures 4 (a) and (b) illustrate the examples of the total and the interested region, respectively. Note that Fig. 4 (a) consists of the real value of the doping profile with an unstructured mesh although Fig. 4 (a) seems to be the image. For training data, we generated total of 1,000 samples using an in-house TCAD simulator (Polaris).

Now, we review the related papers and compare the performance of each algorithm. Figure

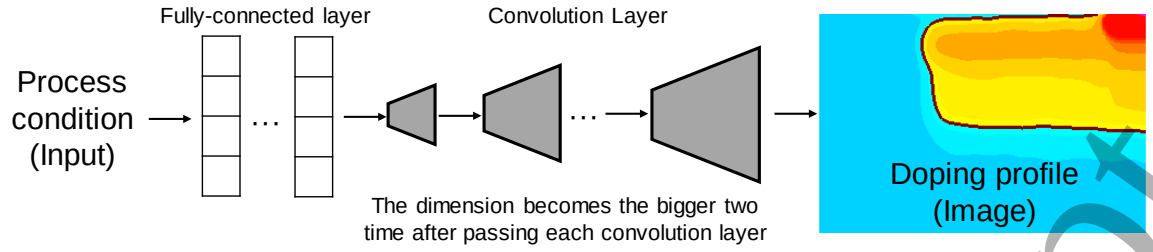

Fig. 5. Illustration of the CNN-based architecture as a baseline.

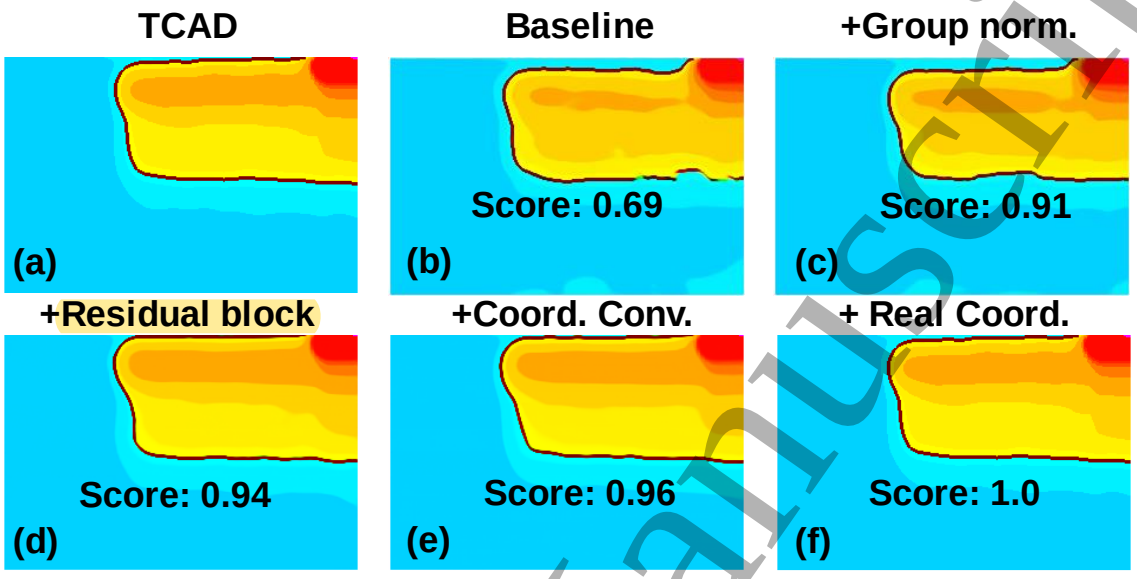

Fig. 6. The examples that each algorithm presents. + indicates that we added the new algorithm to the accumulated algorithm so far. Score is an average of SSIM and IOU.

5 (a) shows the schematic of these works[13, 19-21] that used CNN based model. The architecture originated in generator of DCGAN[22], a well-known model to generate images in the computer vision field. DCGAN for the process model, however, presents a somewhat inaccurate image compared to the ground truth (Figs. 6 (a) and (b)). Based on the DCGAN, here, we compare the doping profile image by adding the several key algorithms that this work[12] has proposed to the baseline model, one by one.

The first one that we introduce is the normalization method, which makes the loss landscape of the model to be smoother[23]. For stable model training regardless of batch size, group normalization technique[24] shows better result than batch normalization[25], as depicted in Figs. 7 (a) and (b). Figure 6 (c) shows it enhances the quality of the junction profile.

The second method that we review is the residual block[1] with the swish activation[26]. Figure 7 (c) illustrates the residual block that can prevent vanishing gradient when training and Fig. 7 (d) shows the swish activation that allows the networks to deeper. Figure 6 (d) indicates that applying all the algorithms we describe so far improves the quality of the doping profile.

The third method that we describe is to concatenate the relative coordinate into each representation on every convolution operation. The literature, so-called Coord. Conv.[27], has

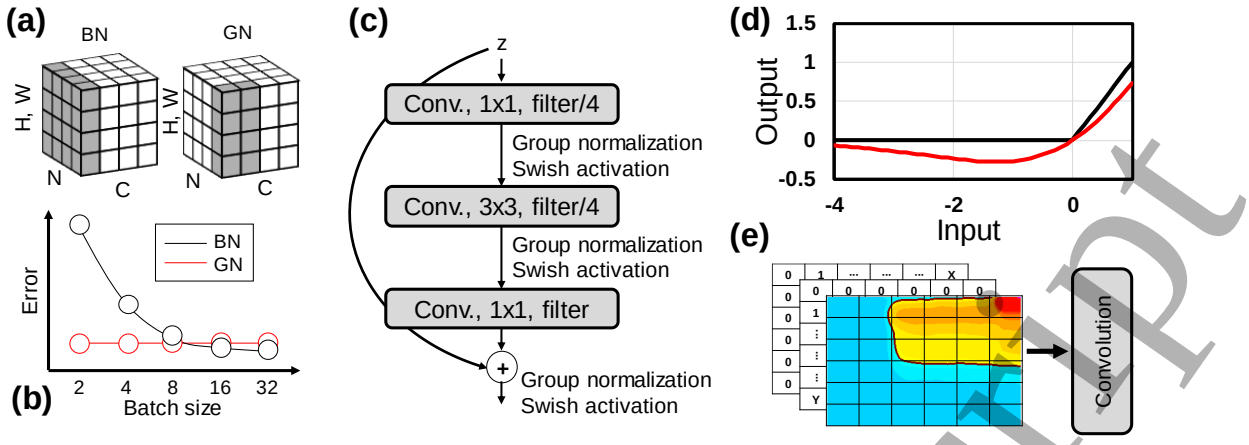

Fig. 7. (a) Illustration of discrepancy between batch normalization (BN) and group normalization. (b) Error comparison between GN and BN. (c) Illustration of residual blocks that this work[12] has proposed (d) Comparison of Relu and Swish activation. (e) Illustration of Coord. Conv. operation.

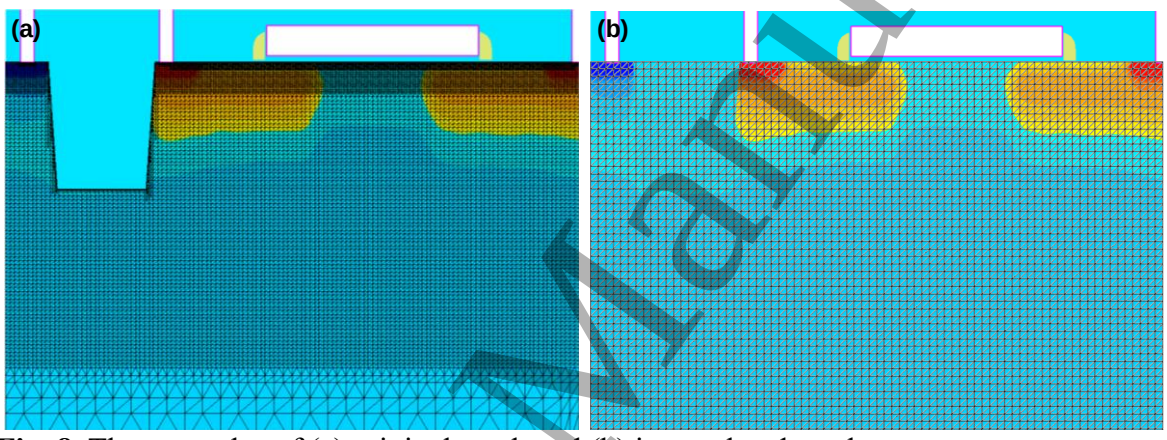

Fig. 8. The examples of (a) original mesh and (b) interpolated mesh.

revealed that CNN structure cannot capture the positional information of images. However, it is crucial for the process model to capture the important geometric information (e.g., oxide thickness, junction depth, or gate length). Hence, to inform the geometric information to the mode l, it is natural to put Cartesian coordinates on every convolution, as shown in Fig. 7 (e). Figure 6 (e) shows it can generate an image consistent with the ground truth.

Hitherto, we reviewed all algorithms for generating the image for the process model. Although such models to present the images are simple yet effective, the TCAD process model indeed should estimate the doping profile to transfer it into the TCAD device model. Figure 8 shows the doping profile on the unstructured mesh that the deep learning model typically cannot deal with. Thus, many previous works[13, 18-21] have suggested converting such profiles into a structured mesh, as shown in Fig. 8 (b). Note that the size of all profiles regardless of the simulation dimension is the same to make the model interpret it. In other words, the size of grids can vary corresponding to the structure variable of inputs.

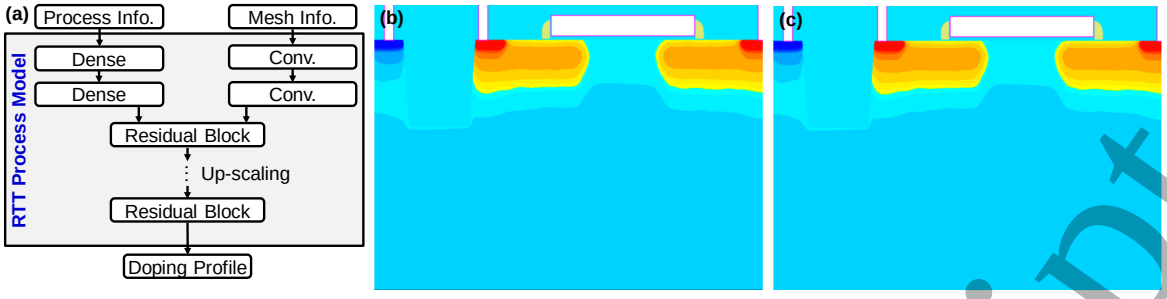

Fig. 9. (a) The model architecture of RTT process model. (b) and (c) are doping profiles that TCAD and RTT process model predict, respectively.

Consequently, the deep learning model cannot employ Coord. Conv. algorithm (i.e., putting relative coordinate) described before because it cannot distinguish the size of grids. Hence, this work[18] has proposed a method, so-called Real-Time TCAD (RTT) process model, which samples the real coordinates from the structured mesh. Figure 9 (a) illustrates the final architecture of the process model that applies the method. Figs. 6 (a) and (f) show that this method clearly generates the doping profile as an image. Figs. 9 (b) and (c) indicates it also clearly predict the doping profile as a real value.

3.2 Device simulation

TCAD device simulation calculates *I-V* characteristics from the doping profile that TCAD process simulation estimates. However, the doping profile generated by the process simulation may be inefficient to use as an input of the deep learning models. Therefore, many existing studies[12-16, 19-21], have used simple scalar values that can represent the doping profile (i.e., the input value of the process simulation) as the input of deep learning. These studies have applied shallow learning (i.e., artificial neural network), which often struggles to learn the data. However, such models using shallow learning cannot consider the relation between outputs[12]. Although *I-V* curves depend on previous voltage, such properties often require more data than expected to train artificial neural networks. To address this problem, previous work has proposed an RNN structure that can capture the relationship between neighboring voltages instead of ANNs, as shown in Figs. 10 (a) and (b). The RNN-based model shares the parameters across various voltage steps, which results in both fewer parameters to train and data required. Figure 10 (c) shows that the RNN-based model produces better *I-V* curves than ANNs. Next, to examine how precise the predicted *I-V* curve is, we calculate derivatives of current with respect to the gate voltage (i.e., G_{DS}-V_{GS} curve). Figure 10 (d) shows that even the RNNs failed to precisely predict the *I-V* curve. To tackle this problem, several studies have suggested derivative losses[12, 28] as follows:

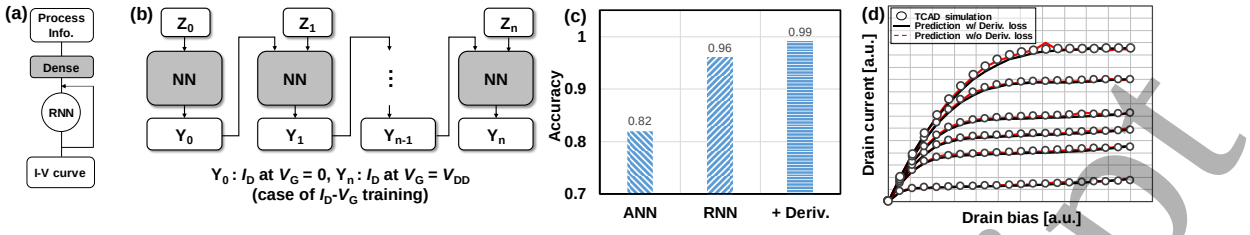

Fig. 10. (a) Illustration of RNN-based *I-V* model. (b) Illustration of unrolled RNN structure. (c) Scores on the architectures of device models (d) I_D-*versus*-V_D curve whether applying derivative loss.

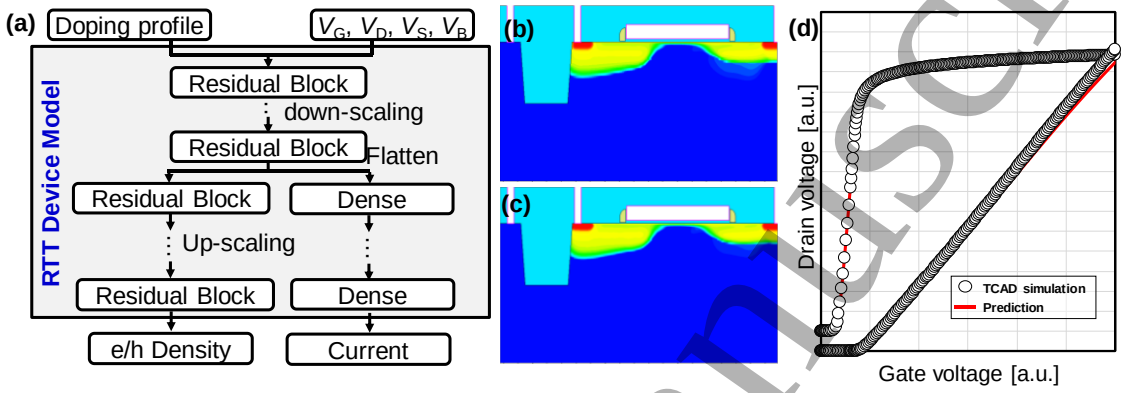

Fig. 11. (a) The model architecture of RTT device model. (b) and (c) are profiles of electron density that TCAD and RTT device model predict, respectively. (d) An example of I_D-V_G curves. *I-V* curve that RTT device model presents is good in agreement with that of TCAD.

$$Loss = \frac{1}{N}\sum_{i=1}^{N}\left[(Y_i - \bar{Y}_i)^2 + \sum_{j=1}^{K}\left(\frac{\partial^j Y_i}{\partial {V_d}^j} - \frac{\partial^j \bar{Y}_i}{\partial {V_d}^j}\right)^2\right],$$

where Y and $\bar{Y}$ are true and prediction values, respectively, K the order of derivatives, N the number of samples. W ith derivative losses, the RNN-based model can predict the accurate *I-V* curve, as shown in Fig. 10 (d). So far, we have reviewed simple deep learning models used for *I-V* prediction. However, device simulation actually solves the Poisson equation and the continuity equation to predict *I-V* curves, which calculates the electrostatic potential and carrier density across the transistor. Henceforth, we review some studies that have attempted to predict these characteristics.

First, with the architecture in Fig. 5, these studies[13, 19-21] predicted the electrostatic potential across the transistors. After that, to obtain the carrier density, they proposed a strategy that solves the continuous equation with the fixed potential after the TCAD device simulation loads the electrostatic potential predicted by the deep learning model. This strategy is the first study that TCAD simulator receives results of the deep learning model as an initial value of it. However, it has two critical problems; i) the input of architecture that they have

proposed is still a scalar value as aforementioned before, not an original doping profile. ii) This strategy cannot consider external resistance (e.g., Schottky contact[29]).

To tackle these problems, this study[18] has suggested an architecture, so-called Real-Time TCAD (RTT) device model, to deal with the original doping profile, as shown in Fig. 11 (a). The residual blocks are the same as the process model. This device model is designed to calculate the current and carrier density, simultaneously. To extract the most important features, the inputs share the residual blocks with down-sampling until they reach the diverging point. From thenceforth, there are two sub-networks; the network for career density consists of the residual blocks with the up-scaling method, which converts the output into device characteristics and the network for the current employs the fully-connected layers to predict the current as a scalar value. Figures 11 (b) - (d) shows that this model predicts the *I-V* curve and carrier density well.

4. Auto-calibration

This section explores the application to automatically calibrate the TCAD models to the measurement data. Typically, it is essential to calibrate TCAD models to the measurement data because TCAD uses simplified models to describe physical phenomena due to high computational costs. TCAD calibration, however, often charges a heavy cost to require skillful engineers to perform TCAD simulation iteratively until fitting on the measurement. To address this problem, many researchers have initiated Bayesian optimization[30] (BO), which can facilitate calibrating TCAD models without experts (i.e., auto-calibration).

Generally, BO requires defining the environments and designing the surrogate model that can represent predictive uncertainty. And then BO optimizes the TCAD model parameter. This section explores such three functions in detail.

4.1 Environments setting

First, we introduce what models we calibrate with which data. TCAD model calibration to the I-V curve is the best exemplar of TCAD use. What is important at this step is to determine how much data we generate as initial data and the range of the fitting parameters. Note that this section does not need massive data at once when training in contrast to the previous section because BO aims to collect the data that it needs, automatically. As the initial design of experiments, there are several strategies; random sampling, Sobol sequence[31], and Latin hypercube sampling (LHS)[32]. Regardless of any strategy, the BO framework can work well. In this problem, we generated fifty data as initial data that the surrogate model learns under

Table I. Description of the calibration parameters.

Category	Model	Min	Max
Process	Segregation	0.1	10
	⋮	⋮	⋮
	Diffusivity	0.2	5
Device	Low Field Mobility	0.1	10
	High Field Saturation	0.1	10
	⋮	⋮	⋮
	Tunneling	0.1	10

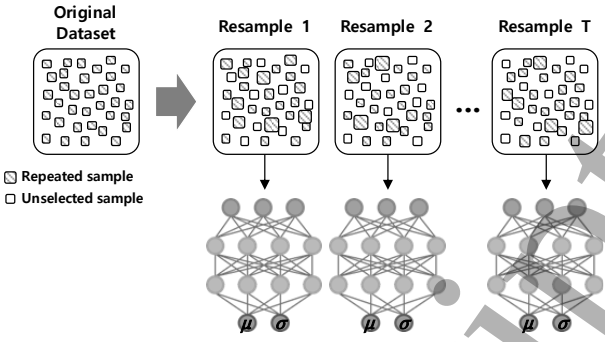

Fig. 12. Illustration of ensemble method. Each model is independently trained using bootstrap method

the LHS strategy. Table I shows an example of the TCAD model parameters and their ranges used for *I-V* calibration.

4.2 Surrogate model

Second, we elucidate what role the surrogate model plays under the BO framework. In this problem (i.e., *I-V* calibration), the surrogate model aims to learn the relationship between the fitting parameters and the *I-V* curve that TCAD will present. As aforementioned, there are few data that the surrogate model can learn in this problem. Therefore, the surrogate model has a low possibility to infer the accurate *I-V* curves that TCAD will produce. To alleviate this problem, there are essential properties that the surrogate model should have. That is the surrogate model should represent the confidence interval that it thinks of itself. However, it is typically hard to make the surrogate model tell whether the prediction is trustworthy or not. Gaussian process (GP), which is a probabilistic model, is one of the well-known surrogate models to estimate uncertainty. However, GP assumes that the inputs are paired with only a single output. Thus, to treat the multiple outputs, we have to employ the multiple GPs. This treats the multiple outputs for each input as separate tasks. For this reason, GP cannot consider the relation between outputs. Therefore, the RNN-based model that Sec. III explained is the proper solution if it can present uncertainty. To this end, we trained the RNN-based model with a particular loss[33], called negative log-likelihood (NLL) as follows:

$$NLL = \frac{(y-u(x))^2}{2\sigma^2(x)} + \frac{\log\sigma^2(x)}{2},$$

where $u(x)$ and $\sigma^2(x)$ are mean and variance of model outputs. When training, the surrogate model learns in the direction where NLL loss decreases. There are two ways; one is to reduce the numerator of the first term (the same as the mean-squared error), and the

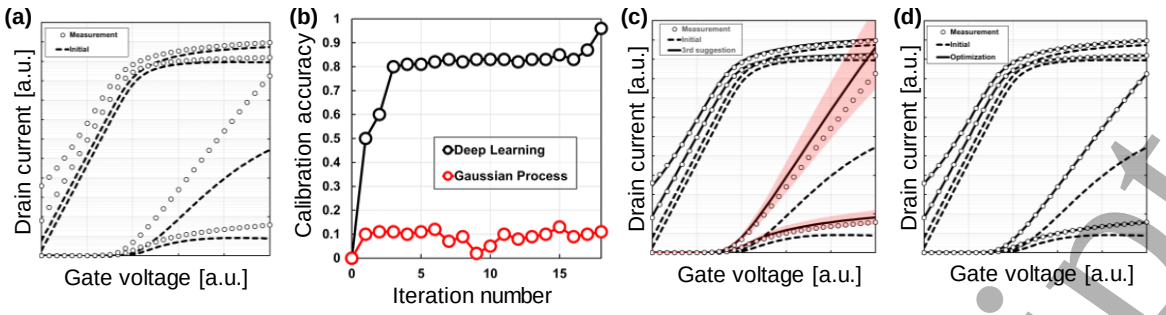

Fig. 13. (a) *I-V* curves of initial TCAD and the measurement (b) Accuracy plot as BO progresses. (c) *I-V* curve when the surrogate model suggests the candidate at second iteration. (d) I curve after finishing BO.

other increases the variance of the first term as long as the second term is not too large. In addition to NLL loss, it is necessary to introduce the ensemble method to avoid the over fitting issue, as depicted in Fig. 12 In the ensemble method, the mean and variance of the ensemble method are given by the mean and variance of the mixture as follows[34]:

$$u_{Ens.}(x) = \frac{1}{M}\sum_{i=1}^{M} u_i(x), \sigma^2{}_{Ens.}(x) = \frac{1}{M}\sum_{i=1}^{M}\left(\sigma^2{}_i(x) + u_i{}^2(x)\right) - u_{Ens.}{}^2(x),$$

where M is the number of the surrogate models $u_{Ens.}(x)$ and $\sigma^2{}_{Ens.}(x)$ are the mean and variance of the ensemble model, respectively.

4.3 Optimization

Now, let us discuss what the surrogate model does. At the initial step, the surrogate model learns the initial data. Next, the model has to find the proper fitting parameters based on the initial data. To this end, one should design the acquisition function that the surrogate model can find the next candidates. There are several well-known acquisition functions: i) probability of improvement (PI), ii) expected improvement (EI), and iii) Upper Confidence Bound (UCB). Each acquisition function varies how to distribute the weights between exploitation and exploration. According to this work[30], EI is the best acquisition function to balance the weights. In this work, we also used EI defined as follows:

$$\gamma(x) = \frac{u(x) - y_{best}}{\sigma(x)},$$

$$EI(x) = (\gamma(x)\Phi(u) + \phi(u))\sigma(x),$$

where Φ denotes the cumulative distribution and ϕ density function. Note that the acquisition function is described as maximizing problem.

Now, we examine BO framework with two surrogate models: RNN-based model and GP. As aforementioned, we aims to find the parameters within the pre-defined range to close the gap between *I-V* curve that TCAD calculates and the measurement. Figure 13 (a) shows the initial *I-V* curves that TCAD predicts are too far from the measurements. The more data accumulates as conducting BO in the iterative manner, the more the RNN-based surrogate model is accurate while GP does not (Fig. 13 (b)). These results are consistent with Fig. 10 (c), as described in Sec. III. Figure 13 (c) shows the results that the surrogate model guesses *I-V* curves that TCAD will predict. Considering the uncertainty, the surrogate model suggests the next candidates to meet the TCAD models to the measurement. Figure 13 (d) shows the results after conducting BO framework. These results demonstrate this BO framework with deep learning can calibrate the TCAD models without experts. These results are well consistent with our previous study[15]. The whole procedure described here about auto-calibration, has been realized in in-house python library, dubbed generic optimization algorithm tank (GOAT).

5. Discussion

Until now, we have explored the various application of state-of-the-art deep learning technologies applicable to TCAD simulation. Hereafter, we will explain how to utilize the trained deep learning model and then discuss the current limitations of deep learning models.

5.1 Utilization

This section explains the ways to make more valuable utilization of trained deep learning models. One is to produce the results in real-time. Figure 14 (a) shows that trained deep learning models can accelerate TCAD simulation. For problems in Sec. III, RTT models are at least 470 times faster than TCAD simulation, quantitatively. In addition, RTT models can resolve the convergence problem that TCAD typically suffers from (Fig. 14 (b)) without solving PDEs. Next, we discuss RTT models as the interchangeable model mimicking TCAD simulation. As described before, RTT models can treat the outputs of TCAD simulation as a real value (Fig. 8). It means the results that RTT models can be compatible with TCAD simulation, or vice versa. Figure 14 (c) demonstrates RTT and TCAD can be compatible with each other. In particular, such properties are useful when we use the inputs out of training range. In that case, we can utilize the RTT model as much as possible within the training range and then insert the results that RTT model predicts to TCAD simulation to solve it out of training range.

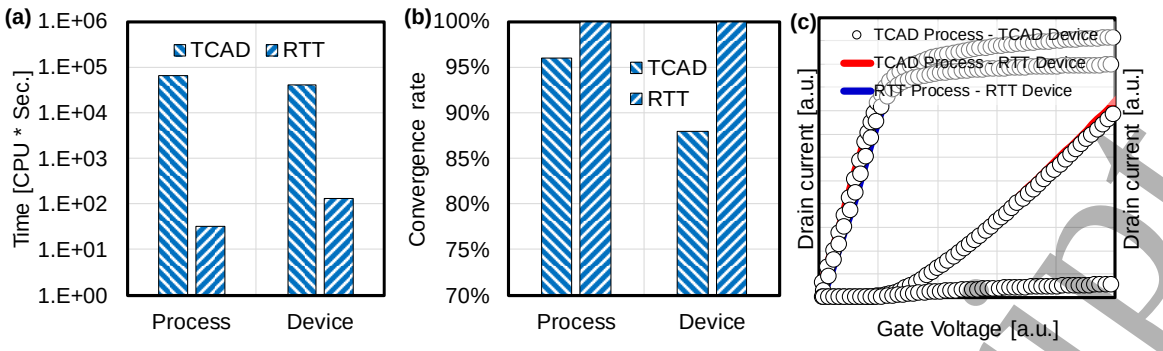

Fig. 14. (a) The comparison of prediction time between TCAD and RTT. (b) Convergence rate of each simulation. (c) I_D-V_G curves at $V_D = 0.1$ or V_{DD} [V], estimated by varying the multiple simulator (TCAD – TCAD, TCAD – RTT, and RTT – RTT).

The last is to optimize the process condition to meet the target values. TCAD engineers typically optimize the process condition with TCAD simulation in the heuristic manners. However, with the trained deep learning models, it can be optimized automatically. These works[12, 35] have demonstrated that deep learning models can find the best solution. In this paper, we discuss the optimization method. There are some optimization algorithms that can be applied to the deep learning models: genetic algorithm (GA)[36], simulation annealing (SA)[37], and gradient descent (GD). Typically, GA and SA can find the global optimum point compared to GD. However, they need to consume long times for optimization. To alleviate this problem, we propose GD algorithm with multi-starting points. The reason is GD is very faster than other algorithms and it can applied to deep learning models, which are differentiable. This work has proven that GD also can find the global optimum from a number of starting points while conducting 280 times faster than GA.

5.2 Limitation

The assumption of the deep learning models is that TCAD was well calibrated to measurements data. Thus, the application of deep learning models are limited to mass production phase, not development phase because novel process technologies are introduced or process changes, frequently in development phase. To resolve this problem, transfer learning approach[38] is one of the proper solutions. The strategy of the transfer learning progresses three steps. The first is we train the deep learning models with TCAD data, not calibrated to the measurement. Then, we calibrate the TCAD simulation to the measurement applying any method (expert calibration or auto-calibration) if TCAD or the deep learning models is not consistent with the measurement. Finally, transfer learning can be applied to close the gap between deep learning model and the measurements. It is currently under consideration as a future work.

6. Conclusions

This article covered the various way to utilize deep learning technologies to TCAD simulation. In case TCAD models cannot fully describe the physical behaviors, we showed the deep learning models could be a good alternative to TCAD simulation if there are massive data generated by TCAD simulation. We have also reviewed pros and cons of the deep learning models that mimics TCAD simulation with the comprehensive investigation. From the analysis, we showed that Real-Time TCAD models would be the best solution that can solve the problems on the high computational costs. Last, we have demonstrated deep learning models can automatically calibrate the TCAD models to the measurement when applying Bayesian optimization. Albeit several drawbacks, we believe that deep learning models may be suitable approach to tackle the problems that TCAD suffers from. We hope that this article will lead many researchers to introduce the deep learning models and also trigger many studies to apply deep learning technologies to TCAD simulation.

Acknowledgments

Sanghoon Myung would like to thank Prof. Kobayashi for inviting me to write this review paper.

References

1) K. He, X. Zhang, S. Ren, and J. Sun. IEEE conference on computer vision and pattern recognition, pp. 770-778, (2016).
2) P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Conference on Empirical Methods in Natural Language Processing, (2016).
3) A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, and Y.-J. Lee. arXiv preprint axXiv:2004,10746 (2020).
4) H. Choi, I. Huh, S. Kim, J. Ko, C. Jeong, H. son, K. Kwon, J. Chae, Y. Park, J. Jeong, D. S. Kim, and J. Y. Choi. ACM/IEEE Design Automation Conference (DAC), (2021).
5) S. Shim, S. Choi, and Y. Shin. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), (2016).
6) Y. Yuan-Fu. Annual SEMI Advanced Semiconductor Manufacturing Conference, (2019)
7) S. Cheon, H. Lee, C. O. Kim, and S. H. Lee. IEEE Transaction on Semiconductor Manufacturing 32.2: 163-170, (2019).
8) S. Myung, H. Jang, B. Choi, J. Ryu, H. Kim, S. W. Park, C. Jeong, and D. S. Kim. Neural Information Processing Systems (NeurIPS) Workshop on Interpretable Inductive Biases and Physically Structured Learning (2020).
9) H. Dhillon, K. Mehta, M. Xiao, B. Wang, Y. Zhang, and H. Y. Wong. IEEE Transactions on Electron Devices, 68(11), 5498-5503 (2021).
10) K. Mehta, S. S. Raju, M. Xiao, B. Wang, Y. Zhang, and H. Y. Wong. IEEE Access, 8, 143519-143529 (2020).
11) Y. S. Bankapalli and H. Y. Wong. IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 1-4 (2019).
12) S. Myung, J. Kim, Y. Jeon, W. Jang, I. Huh, J. Kim, S. Han, K.-H. Baek, J. Ryu, Y.-S. Kim, J. Doh, J. Kim, C. Jeong, and D. S. Kim. IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 18-2 (2020).
13) S. Han, S. C. Choi, and S. M. Hong. IEEE Transactions on Electron Devices 68.11 (2021): 5483-5489.
14) J. Chen, M. B. Alawieh, Y. Lin, M. Zhang, J. Zhang, Y. Guo, and D. Z. Pan. IEEE Access 8 (2020): 25372-25382.
15) C. Jeong, S. Myung, I. Huh, B. Choi, J. Kim, H. Jang, H. Lee, D. Park, K. Lee, W. Jang, J. Ryu, M.-H. Cha, J. M. Choe, M. Shim, and Dae Sin Kim. IEEE Transactions on Electron Devices, 68(11), 5364-5371 (2021).
16) R. D. Rung, H. Momose, and Y. Nagakubo. IEEE International Electron Devices

Meeting, pp. 237-240 (1982).
17) R. Caruana. Machine Learning, 28(1), 41-75 (1997).
18) S. Myung, W. Jang, S. Jin, J. M. Choe, C. Jeong, and D. S. Kim. IEEE International Electron Devices Meeting, pp. 18.2.1-18.2.4 (2021).
19) S. Han, S. C. Choi, and S. M. Hong. IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 52-55 (2021).
20) S. Han, S. C. Choi, and S. M. Hong. IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 297-300 (2020).
21) S. Han, S. C. Choi, and S. M. Hong. IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 1-4 (2019).
22) A. Radford, L. Metz, and S. Chintala. arXiv preprint arXiv:1511.06434 (2015).
23) S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. Neural Information Processing Systems (NeurIPS), 31 (2018).
24) Y. wu and K. He. European conference on computer vision (ECCV), pp. 3-19, (2018).
25) S. Ioffe and C. Szegedy. International conference on machine learning (pp. 448-456), (2015).
26) P. Ramachandran, B. Zoph, and Q. V. Le. arXiv preprent arXiv:1710.05941, (2017).
27) R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. Neural Information Processing Systems, 31 (2018).
28) W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Neural Information Processing Systems, 30 (2017).
29) K. Varahramyan and E. J. Verret. Solid-State Electronics, 39(11), 1601-1607, (1996).
30) J. Snoek, H. Larochelle, and R. P. Adams. Neural Information Processing Systems, 25 (2012).
31) I. A. Antonov and V. M. Saleev. USSR Computational Mathematics and Mathematical Physics, 19(1), 252-256, (1979).
32) M. D. McKay, R. J. Beckman, and W. J. Conover. Technometrics, 42(1), 55-61, (2000).
33) D. A. Nix and A. S. Weigend. IEEE International conference on neural networks, Vol. 1, pp. 55-60, (1994).
34) B. Lakshminarayanan, A. Pritzel, and C. Blundell. Neural Information Processing Systems, 30, (2017).
35) H. Xu, W. Gan, L. Cao, C. Yang, J. Wu, M. Zhou, H, Qu, S, Zhang, H. Yin, and Z. Wu. IEEE transactions on electron devices (2022).
36) M. Mitchell, MIT press (1998).

37) S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi. Science, 220(4598), 671-680, (1983)
38) S. Myung, I. Huh, W. Jang, J. M. Choe, J. Ryu, D. S. Kim, K.-E. Kim, and C. Jeong. International Conference on Machine Learning, pp. 16240-16252, (2022).