Improved Algorithm of Dueling DQN for BSIM Parameter Extraction Task

Wenjun Chen, Yali Zhang, Zikang Zeng, Silin Chen, Kangjian Di, Guohao Wang, Chia-Yen Li, and Ningmu Zou

***Abstract*—Traditional Berkeley Short-channel IGFET Model (BSIM) parameter extraction methods are inefficient, time-consuming, and heavily dependent on the experience of specialists. To tackle these issues, this paper proposes a BSIM parameter extraction algorithm based on deep reinforcement learning, combining the Dueling Deep Q-Network (Dueling DQN) architecture with Prioritized Experience Replay (PER). The algorithm also enhances the traditional ε-greedy strategy by implementing optimal step exploration, significantly improving exploration efficiency. We achieve an average error of below 2.5%. Moreover, we have automated the parameter extraction process, offering a promising alternative to existing methods. This advancement aims to streamline and enhance the efficiency of BSIM parameter extraction in semiconductor modeling. Code is available at https://github.com/zouningmu/Dueling_DQN_for_BSIM.**

***Index Terms*—BSIM, parameter extraction, Dueling-DQN, deep reinforcement learning, compact model**

I. Introduction

Circuit simulation is a key part of the integrated circuit design process, which provides an accurate description of circuit behavior through compact models. With the continuous evolution of IC processes, the complexity of circuit simulation has increased exponentially, and the accuracy of compact models has become more demanding. Various specialized compact models have been developed for different device types, such as the BSIM-CMG model for FinFET devices and the BSIM4 model for conventional MOSFETs. However, such models often contain hundreds of physical parameters, the accuracy of which not only directly affects the predictive ability of the model but also determines the credibility of the overall circuit simulation. Therefore, achieving high-precision parameter extraction under the premise of ensuring efficiency has become a core technical challenge to overcome in semiconductor modeling.

Even though software like IC-CAP [4] and Meqlab [5] can optimize the modeling process to a certain extent, manual intervention is still required. It also relies heavily on the experience of engineers, which makes the parameter tuning process full of subjectivity. To enhance the efficiency and reliability of parameter extraction, researchers are exploring alternative methods to replace manual extraction processes, such as Genetic Algorithms (GA) [6], the Levenberg-Marquardt Algorithm [7], and Automatic Differentiation (AD) [8] have been proposed. However, these algorithms can easily fall into a dimensionality catastrophe and local optimal solutions with high computational complexity. Deep learning networks have been used for the extraction process in fixed gate-length structures [9], [10], which has inspired further research to develop optimized deep learning networks aimed at achieving global parameter extraction [11]–[14]. Leveraging a framework that combines physics-driven parameter initialization with data-driven deep learning has automated the parameter extraction process [15]. Lee et al. demonstrated the efficacy of integrating multiple structural parameters into real-time BSIM-CMG parameter extraction using multi-task learning, significantly improving model robustness across various technology nodes [16]. However, the black-box nature of neural networks masks the causal relationship between model parameters and device characteristics, and the process is unobservable; the accuracy of the model is closely related to the size and quality of the training dataset, and a large amount of accurately annotated measurements are required to achieve acceptable generalization across different technology nodes.

Deep Reinforcement Learning (DRL) methods utilize BSIM models to build environments that can effectively address the drawbacks of deep learning. Related studies use modular Q-learning for parameter extraction, which is divided into six groups according to the different effects of the nine parameters on the feature curves, which are extracted by agents [17], but only a limited number of parameters can be extracted at a time. By comparing the three algorithms, it is found that Soft Actor-Critic (SAC) is more efficient [18], but the accuracy is not satisfactory in the results.

This paper presents a parameter extraction method for the BSIM based on Dueling Deep Q-Network (Dueling DQN) [19]. By integrating the exploration and exploitation mechanisms of deep reinforcement learning, we can automate the parameter extraction process. For the first time, we utilize the deep reinforcement learning method on silicon data to achieve high-precision extraction results and migrate to different technology nodes to get rid of the dependence on labeled data through the interaction between the Agent and the HSPICE environment, and to provide the decision path tracing for parameter tuning to enhance the engineering utility.

Corresponding author: Ningmu Zou.

Wenjun Chen, Yali Zhang, Zikang Zeng, Silin Chen, and Kangjian Di are with the School of Integrated Circuits, Nanjing University. Suzhou, China. (e-mails:wenjunchen@smail.nju.edu.cn, yl_zhang@smail.nju.edu.cn,zikangzeng@smail.nju.edu.cn, silin.chen@smail.nju.edu.cn, kangjiandi@smail.k.edu.cn).

Guohao Wang is with ZetaTech Co., Ltd., Shanghai, China. (e-mail: guohao.wang@zetatech.com.cn).

Chia-Yen Li is with Nexchip Inc., Hefei, China. (e-mail: chiayenli@nexchip.com.c).

Ningmu Zou is with both the School of Integrated Circuits, Nanjing University, Suzhou, China, and the Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou, China. (e-mail: nzou@nju.edu.cn).

This work was partly supported by the Natural Science Foundation of China under Grant 62341408.

TABLE I
WAT DATA AT DIFFERENT PROCESS NODES WITH DIFFERENT GATE LENGTHS AND WIDTHS

150nm Data(shrink ratio=0.85)

W(um) \ L(um)	10	2	1	0.8	0.5	0.3	0.2	0.18
10	✓	✓	✓	✓	✓	✓	✓	✓
1	✓	✓	✓	✓	✓	✓	✓	✓
0.5	✓	-	-	✓	✓	✓	✓	✓
0.24	✓	✓	✓	✓	✓	✓	✓	✓

110nm Data

W(um) \ L(um)	10	1	0.2	0.13	0.12	0.11
10	✓	✓	✓	✓	✓	✓
1	✓	✓	✓	✓	-	✓
0.6	✓	✓	-	✓	-	✓
0.26	✓	-	-	-	-	✓
0.16	✓	✓	✓	✓	✓	✓

40nm Data(shrink ratio=0.9)

W(um) \ L(um)	10	0.04
10	✓	✓
0.3	-	✓
0.12	✓	✓

II. PROPOSED METHODOLOGY

A. WAT DATASETS

To validate our proposed algorithm, we utilized Wafer Acceptance Test (WAT) data obtained from a semiconductor foundry. WAT is a critical step in semiconductor manufacturing, where electrical tests are performed on test structures located in the scribe lines of each wafer. These tests measure key electrical parameters to ensure the wafer meets the required specifications before proceeding to packaging. In this study, we used WAT data from the 150nm, 110nm, and 40nm technology nodes. These nodes are not all standard technology nodes, but rather customized versions tailored to the specific requirements of the manufacturing process. Table I summarizes the range of device sizes covered by the WAT data, including variations in gate width (W) and gate length (L). Here, shrinkage refers to the proportional scaling of transistor dimensions. In EDA tools, the area is typically labeled as the pre-shrink area, corresponding to the original design dimensions before scaling.

B. ENVIRONMENT SETUP FOR DRL

Due to the large number of parameters to be extracted, we adopted a parameter grouping method, assigning each set of parameters corresponding to different curve groups to an independent agent. Since each parameter operates within its specific range and characteristics, it is essential to use distinct parameters to fit different types of curves, such as current-voltage (I-V) and capacitance-voltage (C-V) curves. Each agent is responsible for iteratively predicting and optimizing the values of multiple related parameters until the discrepancy between the simulation results and actual silicon data converges to an acceptable level. To improve the efficiency and reduce the parameter coupling effect, we group the parameters according to their sensitivity to the characteristics of C-V, I-V, etc. based on the physical characteristics, and the total reward of the agent is the weighted sum of the errors of each characteristic, forcing the agent to balance the optimization of different characteristics.

We formulate the BSIM parameter extraction task as a Markov Decision Process (MDP), which is a mathematical framework for modeling decision-making in situations. In this framework, an agent interacts with an environment over a sequence of discrete time steps. At each time step, the agent observes the current state s_t, takes an action a_t, receives a reward r_t, and transitions to a new state s_{t+1}. The agent's goal is to learn a policy that maximizes the cumulative reward over time.

Each parameter to be extracted has two discrete actions: increase and decrease. The step size for parameter changes is set to be 1 divided by the specified step length of the variation range. The parameter ranges refer to the accumulated experience of the foundry and conform to the physical meaning.

The state is represented as a vector containing information about the current parameter values, for example, [DTOX, ACDE, PHIN, K1, MOIN, VOFFCV, NOFF, NGATE], which can be used to fit capacitance characteristic curves.

Changes in fitting errors determine the reward function. Specifically, if an action effectively reduces the fitting error with the target curve, it receives a positive reward. Conversely, a negative reward is given if the error increases. For ease of differentiation, we denote the WAT data as the target curve and the simulated curve as the extracted curve. For a given target curve Tar_i and extracted curve Ext_i, the fitting error is measured using the relative root mean square error (RRMSE), and the reward value can be determined as follows:

$$RRMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{Tar_i - Ext_i}{Ext_i})^2}, \quad (1)$$

$$reward = (\frac{1}{newstate\_rrmse} - \frac{1}{oldstate\_rrmse}), \quad (2)$$

where newstate_rrmse represents the new RRMSE after taking the action, and oldstate_rrmse is the RRMSE obtained after executing the previous action. The interaction between the agent and the environment is implemented via HSPICE simulation, with all BSIM parameters recorded in the model card. Simulation results are modified by altering the values in the model card to change the simulation outcomes. By comparing the simulation results with actual silicon data each time, we can quantitatively assess the degree of curve fitting. The applied voltage and other relevant information are specified in the netlist file, and the netlist and model card serve as input files for the HSPICE simulation.

C. TRAINING ALGORITHM OF DUELING DQN

Based on the above MDP environment framework, we further designed an efficient deep reinforcement learning algorithm to automatically optimize BSIM parameters. Q-learning [22] is a model-free algorithm that seeks to learn the value of

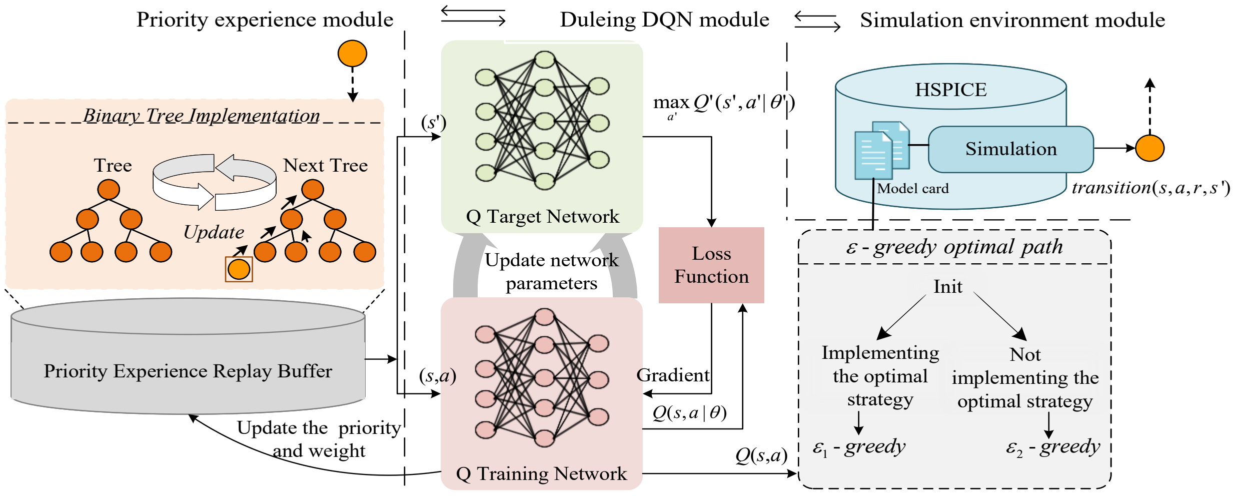
Priority experience module
Duleing DQN module
Simulation environment module
Binary Tree Implementation
Tree
Next Tree
Update
Priority Experience Replay Buffer
Update the priority and weight
(s')
(s,a)
Q Target Network
Update network parameters
Q Training Network
$\max_{a'} Q'(s', a' | \theta')$
Loss Function
Gradient
$Q(s,a|\theta)$
$Q(s,a)$
HSPICE
Simulation
Model card
$transition(s,a,r,s')$
ε - greedy optimal path
Init
Implementing the optimal strategy
Not implementing the optimal strategy
ε_1 - greedy
ε_2 - greedy

Fig. 1. Framework Overview. Our algorithm integrates Dueling DQN to learn and adjust the parameters of the strategy. For sample storage and utilization, we selected priority experience replay, prioritizing experiences with greater impact to optimize the fitting process. The environment was built using HSPICE for interaction with the Dueling module. For action selection, we used past experiences to optimize the traditional greedy strategy.

an action in a particular state. The Q-value $Q(s_t, a_t)$ represents the expected cumulative reward of taking action a_t in state s_t and following the optimal policy thereafter. The goal of Q-learning is to approximate the optimal Q-function $Q^*(s_t, a_t)$, which satisfies the Bellman Optimality Equation:

$$Q^*(s_t, a_t) = \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t\right], \quad (3)$$

where $Q^*(s_t, a_t)$ is the optimal action-value function, representing the maximum expected cumulative reward of taking action a_t in the state s_t and thereafter following the optimal policy. γ is the discount factor($0 \leq \gamma \leq 1$), which balances the importance of immediate rewards versus future rewards. As shown in Equation 3, a value of γ close to 0 emphasizes short-term rewards, while a value close to 1 encourages long-term planning.

While Q-learning is effective for problems with small state and action spaces, it struggles to scale to high-dimensional problems due to the curse of dimensionality. To improve the efficiency and accuracy of parameter tuning, we employ a Dueling DQN architecture as shown in Fig. 1. The input is the s_t and the output is $Q(s_t, a_t)$, essentially replacing the Bellman equation with a neural network. The target network Q' is a lagged copy of the primary network, and its parameters θ' are periodically updated from the primary network θ. It also mitigates Q-value overestimation, improving learning accuracy in high-dimensional parameter extraction tasks. Dueling DQN enhances convergence speed and stability by splitting the Q-network into two components: one for estimating state values and another for evaluating action advantages. This design allows the algorithm to identify important states better and reduce reliance on ineffective actions.

Samples generated from each interaction between the agent and the environment are collected into an experience pool and sampled with equal probability for training. To address this, we introduce prioritized experience replay (PER) to optimize training efficiency. PER is implemented using a binary tree structure for priority sorting. Each leaf node stores the priority of the sample, each parent node maintains the sum of its child nodes' priorities, and the root node records the total priority of all samples. During sampling, the algorithm starts from the root node, selects a path based on node priorities, and ultimately selects samples from leaf nodes. Each sample records transition information (s_t, a_t, r_t, s_{t+1}) and is stored in PER. The priority of each sample is associated with the absolute value of its TD error δ_t, ensuring that more important samples are prioritized during training. δ_t reflects the deviation between the estimated value and the true value for a given state-action pair. The larger the error, the more significant the impact of that experience on model optimization. Therefore, in PER, δ_t is used as a priority indicator for samples, with higher-priority experiences being used more frequently in training to enhance learning efficiency. The relationship between each sample's priority and its δ_t is expressed by the following equation:

$$p_t = |\delta_t| + 0.01, \quad (4)$$

$$\delta_t = r_t + \gamma \cdot \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}) - Q(s_t, a_t), \quad (5)$$

$$P(t) = \frac{p_t^\alpha}{\sum_k p_k^\alpha}, \quad (6)$$

where p_t denotes the priority of the t-th experience sample in the replay buffer and alpha controls the degree of prioritization. The higher the priority, the higher the probability that the sample will be used for training. Naturally, the probability of each sample being adopted can be obtained, P_t is the probability of prioritizing an empirical sample by normalizing it to determine the likelihood that that sample will be drawn at training time as shown in Equation 6. To prevent excessive bias towards specific samples, we implement importance sampling weights to balance the training bias. These weights are calculated based on the total number of samples, N_i, and a parameter β that ranges from 0 to 1, which controls the strength of the importance sampling correction.

$$w_i = (\frac{1}{N_\mathrm{i}} \cdot \frac{1}{Pi})^{\beta}, \tag{7}$$

$$a_t = \begin{cases} \text{Randomly select an action,} & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a), & \text{with probability } 1-\varepsilon \end{cases} \tag{8}$$

In Equation 8, it is illustrated that during each interaction between the agent and the environment, the agent selects an action based on the current state using the ε-greedy strategy. When the agent needs to decide the current state s_t, it queries the Q-network to obtain the Q-values of all possible actions a. The agent chooses an action randomly with probability ε and the action with the largest Q value with probability 1-ε. In the early training phases, a higher value of ε ensures that the agent thoroughly explores the state space, helping to prevent it from getting trapped in a local optimum. As training progresses, a gradually smaller value of ε allows the agent to concentrate more on the optimal strategy. In this method, the value of ε decreases as the number of executions increases. The rate at which ε is reduced is defined as follows:

$$\varepsilon = (\varepsilon_{initial} - \varepsilon_{final})/n, \tag{9}$$

where n denotes the number of iterations required to reach the predetermined value. The agent executes the action and obtains a new state and reward. A random mini-batch of samples is drawn from the experience replay buffer to train the network. The target Q-values and losses are computed to update the training network parameters for dueling DQN. The algorithm records the current state, executed operations, rewards, next state, and RRMSE in each training iteration. As the intelligence continues to interact with the environment, more and more training data are obtained, and the network iterates towards obtaining an accurate estimate of the value of the action. The training process terminates when the extraction error for the BSIM parameters reaches a predetermined convergence criterion. The method for computing the loss function for the network is:

$$Loss = \frac{1}{N} \sum_t (\delta_t)^2, \tag{10}$$

There are two termination conditions for every training epoch: one is reaching the specified maximum training steps, and the other is when the extraction error falls below a set threshold. Training will conclude when either condition is satisfied. In the final state, since there is no subsequent state, the Q-value directly equals the immediate reward r. The final Q-value combines the global value of the state with the corresponding action's advantage, yielding the optimal action choice for the current state. This Q-value guides the agent's subsequent action selections, ensuring convergence to the optimal tuning scheme.

Algorithm 1 Dueling DQN with PER and optimal step exploration

```
1: Initialize training network Q and target network $Q'$
2: Initialize PER and set total training episodes M
3: Initialize optimal tuning steps list L
4: for episode $i \in [1, M]$ do
5:    Randomly choose between normal initialization or executing optimal fitting steps L with probability $p$
6:    With probability $\varepsilon$ select a random action $a_t$
7:    Execute action $a_t$ in the environment, obtaining next state $s_{t+1}$, reward $r_t$, and current epoch's minimum RRMSE as $RRMSE_{\text{epoch}}$
8:    Store the obtained tuple $(s_t, a_t, r_t, s_{t+1})$ in PER
9:    if enough $R$ data then
10:       Sample batch_size data points from $PER$
11:       Calculate target using $Q'$ target network:
              $y = r + \gamma \max_a Q\prime(s_{t+1}, a \mid \theta)$
12:       Calculate Temporal Difference (TD) error:
              $\delta = y - Q(s, a \mid \theta)$
13:       Update priorities in PER based on TD error
14:       Update target network parameters:
              $Q' \leftarrow Q$
15:       if $RRMSE_{\text{epoch}} < RRMSE_{\text{min}}$ then
16:          $RRMSE_{\text{min}} \leftarrow RRMSE_{\text{epoch}}$
17:          Update optimal tuning steps list $L$
18:       end if
19:    end if
20: end for
```

D. OPTIMIZING GREEDY STRATEGIES IN DUELING DQN

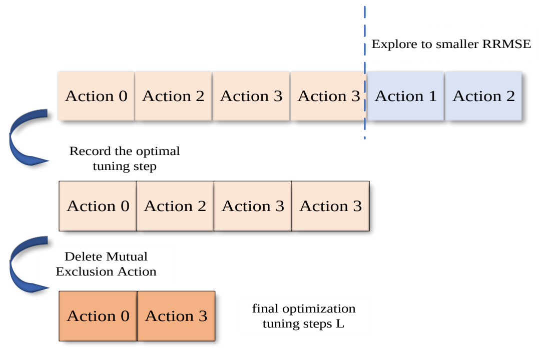

Fig. 2. Flow chart of optimization steps.

For the BSIM parameter extraction task, finding a better solution in the vast state space is crucial. However, the ε-greedy strategy used in Dueling DQN struggles to efficiently explore the high-dimensional state space in BSIM parameter extraction due to its random exploration mechanism. To address this, we propose an optimal step exploration strategy that guides the agent by leveraging historical tuning experiences, combining heuristic search with dynamic path optimization.

The core idea of the optimization algorithm is to improve subsequent tuning processes based on previously explored tuning steps. During training, action sequences leading to reduced RRMSE are dynamically recorded. If the current RRMSE improves over the historical minimum, the corresponding action sequence is stored in the action list L, while outdated sequences are purged.

As shown in Fig.2, further elimination of mutually exclusive action pairs is performed. The action list is assumed to be numbered based on the number of parameters being extracted. For n parameters, actions are encoded as integers $[0, 2n-1]$, where even/odd indices represent $+/-$ adjustments respectively (e.g., Action 0: +Param1, Action 1: -Param1, Action 2: +Param2). The recorded action list is traversed, and for each action processed, the following checks are performed. If the current action is even, subsequent actions in the list are checked to see if an odd action corresponding to the current action exists. If such a subsequent action is found, both the current action and this subsequent action are removed from the list. If not found, the algorithm proceeds to the next action, repeating the check. If the current action is odd, a similar process is followed to find any corresponding exclusive even actions and perform the necessary removal operations.

After processing the action list, each action's impact is further evaluated by calculating the difference between the current RRMSE and the next RRMSE. The action list is sorted in descending order based on the computed RRMSE differences, ensuring that actions significantly improving the RRMSE value are prioritized. Finally, only the top P best-performing actions are retained, forming a new action list for subsequent training. To prevent overtraining, the recorded optimal steps are randomly shuffled before being stored in the action list L. This shuffling process ensures that the model is exposed to a diverse set of experiences, breaking the sequential correlation between consecutive actions and states.

Overall, we integrate the Dueling DQN algorithm with prioritized experience replay, improving it by replacing the epsilon-greedy strategy with optimal step exploration (see Algorithm 1).

III. Results and Discussion

A. SIMULATION RESULTS AND DISCUSSION

The Dueling DQN-based reinforcement learning model is employed to extract BSIM4 parameters for NMOS devices. The primary objective is to minimize the RRMSE between the simulated and measured I-V and C-V characteristics. The training data are derived from WAT measurements. The experiments are performed on devices using a 150nm technology node, with varying L while the W remains fixed at $10\ \mu m$. The initial parameters are randomly selected within a predefined range. The results demonstrate the effectiveness of the proposed DRL-based model across both I-V and C-V characteristics. The extracted parameters are summarized in Table II, with the selection of the model and parameters being based on long-term experience from the wafer fabrication industry. All parameters were carefully chosen based on the length expansion capabilities required by the I-V and C-V models and can be found in the BSIM4.6.2 manual.

The Dueling DQN employs a single hidden layer with 128 nodes. The input layer receives the BSIM parameter values, and the output layer provides Q-values for each action. Key hyperparameters include a learning rate of 5×10^{-3}, a discount factor $\gamma = 0.99$, and a target network update frequency of every 10 episodes. The replay buffer size is set to 10,000 experiences, with a batch size of 128 for training. The exploration rate ϵ decays from an initial value of 0.8 to 0.01 over 2,000 actions. The total training consists of 2000 episodes, and training begins only when the replay buffer contains at least 128 samples. For prioritized experience replay, the priority exponent α is set to 0.6, and the importance sampling exponent β is set to 0.4. These parameters control the degree of prioritization and the bias correction strength, respectively.

TABLE II
A Part of the Parameters We Used

Region	Model Parameters
C_{gg}–V_{gs}	ACDE, DTOX, PHIN, K1, MOIN and NOFF
C_{gc}–V_{gs}	CGSO, CGDO, CGSL, CGDL, DLC and DWC
I_d–V_{gs} Linear	VOFF, NFACTOR, CIT, MINV, U0, UA and UB
I_d–V_{gs} Saturation	AGS, VSAT, A1, A2, ETA0, ETAB and DSUB
V_{tlin}(threshold voltage)	LPE0,DVT0,DVT1,VOFFL, LVTH0, and lvoff

The results of C-V characteristics are illustrated in Fig. 3, showing the fitting accuracy of the extracted capacitance parameters. The RRMSE values for $C_{gg}-V_{gs}$ and $C_{gc}-V_{gs}$ are summarized in Table III. The extracted I_d–V_{gs} characteristics are shown in Fig. 4, where the fitting results are presented for both linear and logarithmic scales. Excellent agreement is observed between the simulated and measured data, with RRMSE values summarized in Table II. For long-channel devices ($L = 10\ \mu m$), the RRMSE stabilizes around 1.37%. As the gate length decreases to $L = 0.2\ \mu m$ and $0.18\ \mu m$, the RRMSE remains below 1.6%, accurately capturing short-channel effects. This indicates that the DRL-based method can effectively model the device behavior across a wide range of gate lengths. To further validate the accuracy of the extracted parameters, the first derivative of the I_d–V_{gs} characteristics concerning V_{gs} is evaluated, as shown in Fig. 5. It should be noted that, due to the measurement limit, the part of the current that is too small has been smoothed.The first derivative highlights the transitions between different operational regions, emphasizing changes in I_{ds} concerning V_{gs}.

The DRL-based model achieves excellent fitting accuracy, with RRMSE values below 2% across all tested gate lengths. For instance, the RRMSE for $C_{gg}-V_{gs}$ at $L = 10\ \mu m$ is as low as 0.937%, while for shorter gate lengths ($L = 0.18\ \mu m$), the

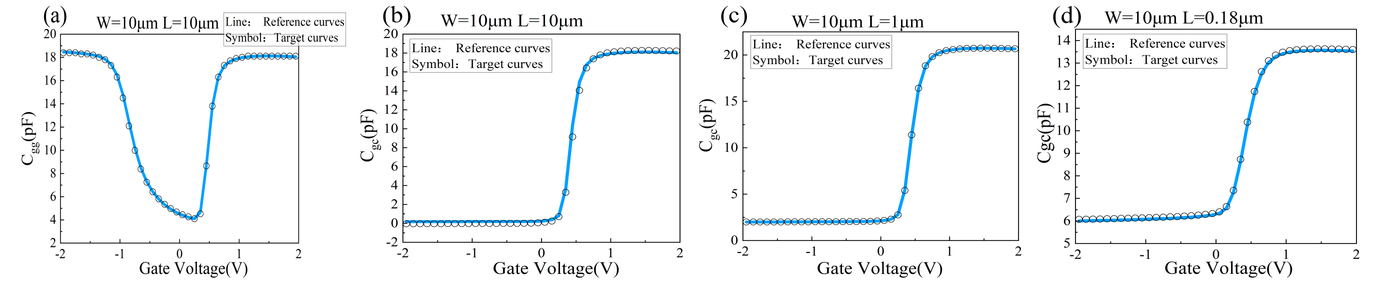

Fig. 3. Fitting results of the C-V characteristic curve for different gate lengths. All curves correspond to a gate length of 10 μm. (a) $C_{gg} - V_{gs}$ for gate length 10 μm, (b) $C_{gc} - V_{gs}$ for gate length 10 μm (with width 10 μm), (c) $C_{gc} - V_{gs}$ for gate length 1 μm, (d) $C_{gc} - V_{gs}$ for gate length 0.18 μm.

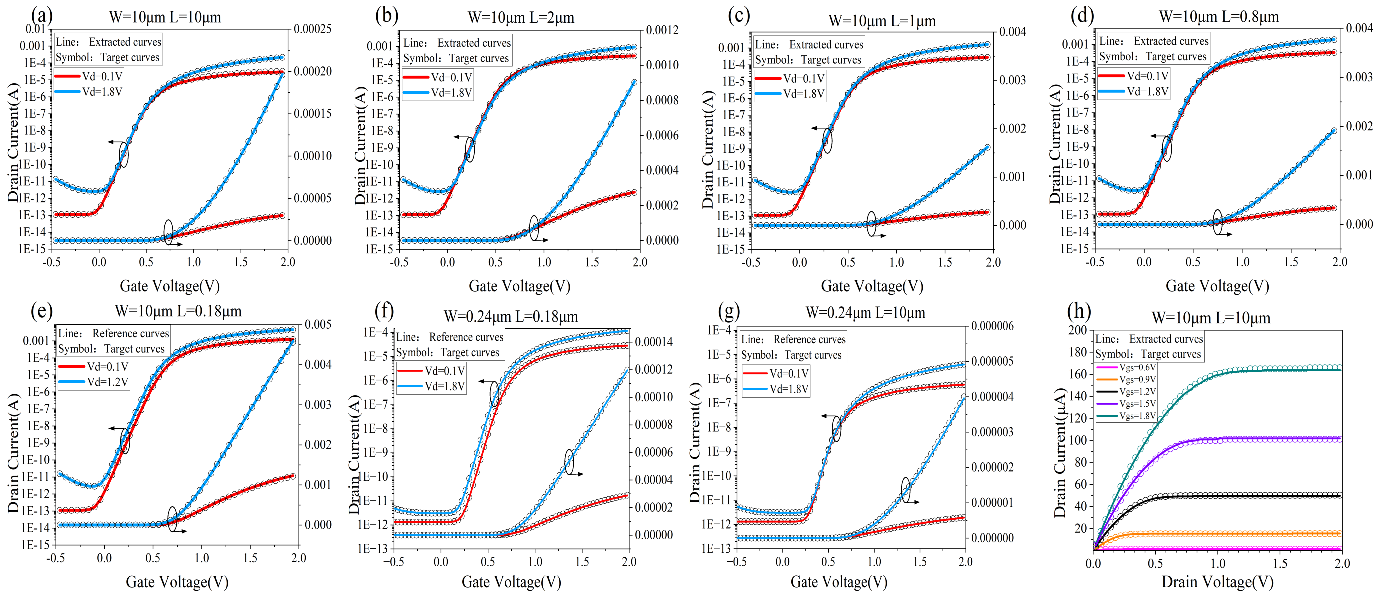

Fig. 4. DRL-based parameter extraction fitting results for $I_d - V_{gs}$ and $I_d - V_{ds}$ characteristics. (a)-(g) show $I_d - V_{gs}$ curves (linear and log scales) with varying gate width and varying gate lengths: (a) W/L = 10μm/10μm, (b) W/L = 10μm/2μm, (c) W/L = 10μm/1μm, (d) W/L = 10μm/0.8μm, (e) W/L = 10μm/0.5μm, (f) W/L = 0.24μm/0.18μm, (g) W/L = 0.24μm/10μm. (h) shows $I_d - V_{ds}$ characteristics

RRMSE for $C_{gc} - V_{gs}$ remains within 0.832%, demonstrating the model's robustness in capturing capacitance behavior under short-channel effects.

The findings indicate that the Dueling DQN-based parameter extraction method not only rapidly fits the I-V and C-V curves but also achieves superior fitting accuracy compared to traditional methods. The method also enables the use of arbitrary I_d–V_{gs} data points for fitting, without strict bias conditions. The proposed method enhances accuracy and generalization for 150 nm NMOS devices by leveraging WAT measurements. Moreover, the algorithm demonstrates excellent exploration efficiency in large state spaces, significantly reducing the extraction time.

B. ALGORITHM EXTENSION TO 110nm AND 40nm TECHNOLOGY NODES

To further validate the robustness and generalization capability of the proposed algorithm, we extended its application to the 110nm and 40nm technology nodes. Unlike the experiments conducted for the 150nm node, where the W was fixed at 10 μm and the L was varied, the devices for 110nm and 40nm nodes utilized smaller dimensions. Specifically, for the 110nm node, the gate width and length were set to $W = 0.16$ μm and $L = 0.11$ μm, respectively. For the 40nm node, the dimensions were $W = 0.12$ μm and $L = 0.04$ μm.

TABLE III
TRAINING RESULTS

Characteristic curves	RRMSE after training	RRMSE before training	Num. of episodes
C_{gg}–V_{gs}	0.94%	29.57%	2000
C_{gc}–V_{gs}	0.83%	40.56%	2000
I_d–V_{gs}	1.31%	79.34%	2000
g_m–V_{gs}	1.61%	18.6%	2000
I_d–V_{gs} at log scale	0.10%	2.26%	2000

Similar to the 150nm experiments, the primary focus was on evaluating the fitting accuracy of the I_d–V_{gs} characteristics, including both the logarithmic scale and the first derivative of the curve concerning V_{gs}. Training data were derived from WAT measurements to ensure comprehensive coverage of device characteristics.

The extracted I_d–V_{gs} characteristics for the 110nm node demonstrated excellent agreement between the simulated and

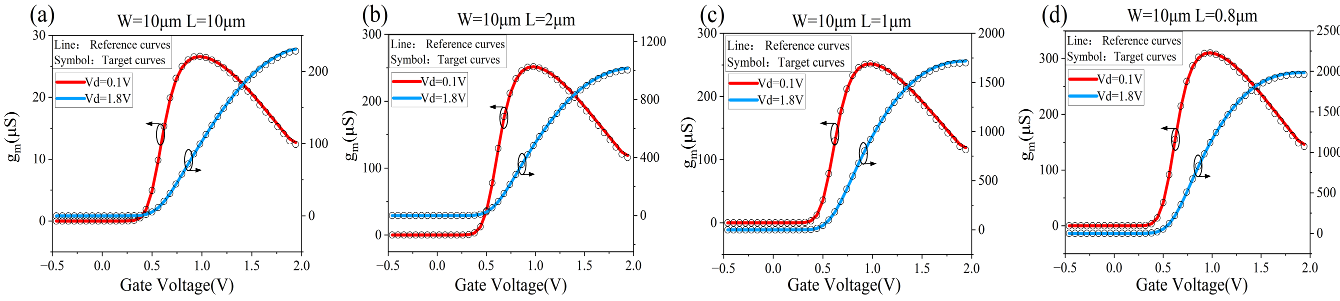

Fig. 5. DRL-based parameter extraction fitting results for the first derivative of I_d–V_{gs} characteristic curves.

measured data. The RRMSE for the logarithmic-scale fitting stabilized at 1.71%, while the RRMSE for the first derivative fitting remained below 1.75%. This indicates that the algorithm effectively captures both linear and nonlinear behaviors, as well as transitions between different operational regions for the 110nm node.

For the 40nm node, despite the more pronounced short-channel effects, the algorithm successfully fitted the I_d–V_{gs} characteristics with a final RRMSE of 2.08% for the logarithmic-scale fitting. For the first derivative fitting, the RRMSE stabilized at 2.12%, demonstrating the algorithm's ability to accurately capture the subtle changes in device behavior under aggressive scaling. The fitting results for the logarithmic scale and first derivative of the I_d–V_{gs} curves at 110nm and 40nm nodes are summarized in Table IV. Figure 6 presents the logarithmic scale and first derivative fitting results for both nodes.

TABLE IV
FITTING ACCURACY FOR DIFFERENT TECHNOLOGY NODES

Technology Node	RRMSE for Log Scale I_d–V_{gs} (%)	RRMSE for First Derivative (%)
150nm	0.10%	1.61%
110nm	1.71%	1.75%
40nm	2.08%	2.12%

The extension of the algorithm to 110nm and 40nm nodes further validates its effectiveness and adaptability. The results indicate that the proposed method can maintain high fitting accuracy for both I-V and derivative characteristics, even under aggressive scaling conditions. This demonstrates the algorithm's potential for widespread application across a range of semiconductor technologies, enabling efficient and accurate BSIM parameter extraction for advanced technology nodes. In addition, Fig.7 shows the 13-stage ring oscillator, simulated using Hspice, at the end of the algorithm run after extracting the 110-nm device parameters.

The fitting performance across the 150nm, 110nm, and 40nm nodes demonstrates the scalability and robustness of the proposed algorithm. By maintaining RRMSE values below 2.5% for both logarithmic and derivative fittings across all nodes, the algorithm is a reliable and scalable solution for BSIM parameter extraction in modern semiconductor technologies.

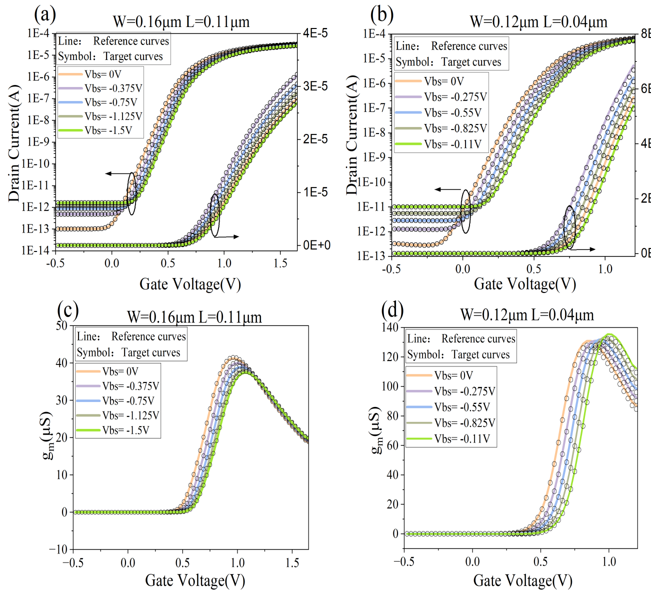

Fig. 6. Logarithmic-scale and first derivative fitting results for I_d–V_{gs} characteristics at 110nm and 40nm CMOS nodes. (a) Logarithmic-scale fitting for 110nm ($W = 0.16$ μm, $L = 0.11$ μm). (b) Logarithmic-scale fitting for 40nm ($W = 0.12$ μm, $L = 0.04$ μm). (c) First derivative fitting for 110nm node. (d) First derivative fitting for 40nm node.

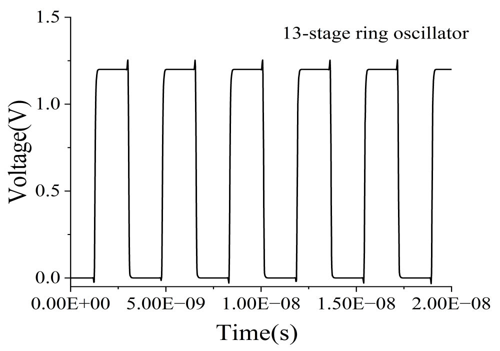

Fig. 7. 13-stage ring oscillator simulations.

C. Ablation Study

To demonstrate the effectiveness of our improvements, we conducted comparative experiments on the algorithm. We trained the model with 10 random seeds and recorded the average episodic rewards. To ensure fairness in the comparison, all

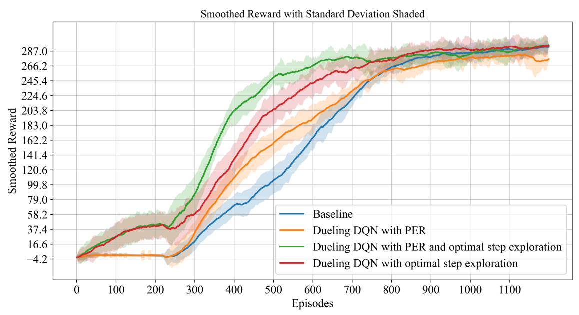

Fig. 8. Ablation Study: Smoothed Reward and Convergence of Different Strategies. The smoothed reward reflects the overall performance trend of each strategy and the shading indicates the standard deviation.

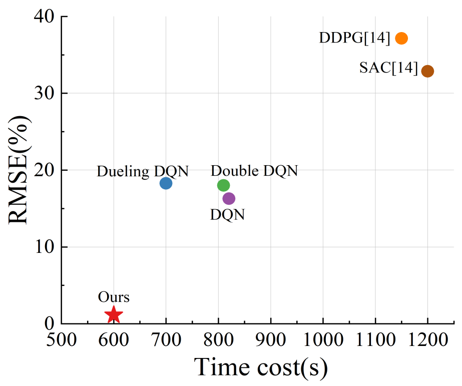

Fig. 9. The comparison of DQN and other reinforcement algorithms.Compare the time required to find the minimum RRMSE.

experiments used the ϵ value starting at 0.8, which decayed to 0.01 over 200 training steps and remained constant thereafter. The total training period consisted of 1200 episodes, with the exploration strategy parameter ζ set to 0.5. The results show that the improved algorithm helps the agent during training and aids in finding the optimal solution. In Fig. 8, the solid lines represent the mean episodic rewards across the 10 runs, while the shaded regions correspond to one standard deviation above and below the mean, providing an indication of the variability and stability of the training process under each strategy.

In this ablation study, we systematically enhanced the Duelling DQN algorithm by introducing PER and optimal step exploration (optimizing the next exploration based on effective action sequences previously discovered). The results show that adding PER alone (orange line) provides a modest improvement in convergence speed and final reward compared to the baseline (blue line), but the enhancement is relatively limited. In contrast, optimal step exploration (red and green lines) has a much more significant impact, greatly accelerating convergence and achieving higher rewards. Notably, the red curve, which only includes optimal step exploration, achieves nearly the same performance as the green curve (combining PER and optimal step exploration), indicating that optimal step exploration plays a more crucial role in performance improvement, while the contribution of PER is comparatively minor. Both the red and green curves converge to the same reward level.

We conducted a comparative analysis of DQN and its various variants, with a primary focus on convergence speed and average reward. The results of this comparison are presented in Fig. 9. It is important to note that the random seed influences convergence speed. Therefore, we report the results based on the majority of trials. For the average reward, we computed the mean from the results obtained across 10 random seed experiments. As observed, DQN suffers from the problem of overestimating Q-values, an issue that both Dueling DQN and Double DQN [21] aim to address. Based on Dueling DQN, our algorithm finds better solutions more quickly and improves convergence speed. The architectural design of Dueling DQN makes it particularly well-suited for handling high-dimensional state spaces in parameter extraction. We also referenced two continuous-action deep reinforcement learning algorithms. However, due to the poor performance of [18] when extracting more than one parameter, we replicated the [18] method and iterated multiple times. Overall, these methods have not shown promising results at this stage.

D. DRL METHODS vs DL METHODS

To illustrate the advantages of our proposed algorithm, we perform a comparative analysis with existing research results focusing on the RRMSE of the linear region fitted by I_d-V_{gs}. as shown in Table 5, highlighting the robustness of the deep reinforcement learning approach. In the table, ”Amount” represents the total number of extracted device parameters, while ”Model” indicates the AI architecture type - either Single ANN (individually trained per device) or Global ANN/DRL (generalized for multi-device scaling).

TABLE V
COMPARISON OF PARAMETER EXTRACTION METHODS FOR MOSFET MODELING.,

Method	Amount	RRMSE (%)	Model
RFIT’2022 [10]	12	8 (max error)	Single ANN
Solid-State Electron’2023 [12]	16	7.41	Global ANN
TED’2022 [9]	12	6.1	Single ANN
TED’2023 [14]	21	3.4	Global ANN
IEEE Access’2024 [16]	15	4.26	Global ANN
Solid-State Electron’2024 [11]	28	2.735	Global ANN
Solid-State Electron’2025 [13]	28	1.5	Global ANN
TED’2024 [23]	16	-	Global BO
Ours	98	**1.31**	Global DRL

Alongside the existing studies, we employed the ANN method for parameter extraction within our dataset and compared it with the proposed DRL method. Due to the limited availability of WAT data from semiconductor foundries, we creatively applied data enhancement techniques to thoroughly validate our proposed algorithm. The device parameters, specifically Width and Length, were uniformly sampled from their minimum to maximum value ranges. These sampled parameters were then used in HSPICE simulations to generate large-scale synthetic datasets that remain physically consistent with actual device characteristics.

For the ANN architecture, we utilized three hidden layers with 128, 64, and 32 neurons, respectively. The activation function for the hidden layer is ReLU, and the learning rate is set to 1e-5. The input consists of data from the $I_d - V_{gs}$ curve, ranging from -0.5V to 1.98V in increments of 0.02V, while the output corresponds to the extracted parameters:[VOFF, NFACTOR, CIT, MINV, K2, U0, UA, UB, A0, AGS, VSAT, A1, A2, ETA0, ETAB, DSUB]. All these parameters are used in the DRL. The output parameter values represent those extracted manually using Meqlab software, which was used as the ground truth. To ensure robust validation, the dataset was randomly split into training and testing sets in a 9:1 ratio, with a fixed random seed of 0 for reproducibility. ANN achieved the best performance on the test set with RRMSE of 2.28%, which is still not as effective as the DRL performance.

We try to analyze the above results. In BSIM parameter extraction, DL methods rely heavily on multilayer perceptual machines that model the relationship between inputs and outputs through nonlinear activation functions to minimize prediction errors. However, DL methods are highly dependent on data quality and require large-scale data generation to ensure accuracy across process nodes.In addition, we also compare an algorithm based on Bayesian optimization(BO) [23]. Our algorithm does not need complex objective function and can extract more parameters.

In contrast, DRL does not directly predict the parameters but instead trains an agent to optimize a strategy for adjusting them. The ability to record, track, and visualize decision steps or intermediate states during a model run allows engineers to see which parameters were modified when and how the error varied, all of which help engineers understand the tuning process.DRL agents demonstrate significant benefits in BSIM parameter extraction.DRL improves generalization to different process nodes and equipment models by dynamically adjusting the search strategy. In addition, DRL provides a clearer understanding of parameter-output relationships, revealing the impact of key parameters through policy-guided parameter tuning, in contrast to the fuzzy relationships in deep learning models.

IV. Conclusion

This paper presents a method for BSIM parameter extraction based on Dueling DQN, designed to address the challenges of high-dimensional optimization and precision in the BSIM4 model. The proposed approach effectively overcomes key difficulties in parameter extraction, such as managing high-dimensional state spaces, minimizing fitting errors, and efficiently identifying optimal solutions. Furthermore, after the model is trained, it can output results in just a few seconds, offering a promising alternative for automating the BSIM parameter extraction process. This reduces reliance on manual intervention and significantly improves both accuracy and efficiency. The final fitting results consistently achieve less than a 2.5% error margin, meeting the acceptance standards of semiconductor fabrication.

References

[1] D. M, “The role of TCAD in compact modeling,” *TechConnect Briefs*, vol. 1, no. 2002, pp. 719–721, Apr. 2002.

[2] *BSIM-CMG 111.2.1 Technical mannul.* Accessed on: Jun. 6, 2022. [Online]. Available:https://www.bsim.berkeley.edu/models/bsimcmg/.

[3] *BSIM4 4.8.2 Technical mannul.* Accessed on: Jan. 1, 2020. [Online]. Available:https://www.bsim.berkeley.edu/models/bsim4/.

[4] Keysight Technologies. *Keysight Technologies Official Website*. Available: https://www.keysight.com.cn/. [Accessed: Mar. 10, 2023].

[5] Primarius Technologies. *MeQLab - Manufacturing EDA*. Available: https://www.primarius-tech.com/products/manufacturing_eda/MeQLab.html. [Accessed: Mar. 10, 2023].

[6] Y. Li, “An automatic parameter extraction technique for advanced CMOS device modeling using genetic algorithm,” *Microelectron. Eng.*, vol. 84, no. 2, pp. 260–272, Feb. 2007.

[7] W. Jouha, A. E. Oualkadi, P. Dherbecourt, E. Joubert, and M. Masmoudi, “Silicon carbide power MOSFET model: An accurate parameter extraction method based on Levenberg-Marquardt algorithm,” *IEEE Trans. Power Electron.*, vol. 33, no. 11, pp. 9130–9133, Nov. 2018.

[8] M. Shintani, A. Ueda, and T. Sato, “Accelerating parameter extraction of power MOSFET models using automatic differentiation,” *IEEE Trans. Power Electron.*, vol. 37, no. 3, pp. 2970–2982, Mar. 2022.

[9] M.-Y. Kao, F. Chavez, S. Khandelwal, and C. Hu, “Deep learning-based BSIM-CMG parameter extraction for 10-nm Finfet,” *IEEE Trans. Electron Devices*, vol. 69, no. 8, pp. 4765–4768, Aug. 2022.

[10] F. Chavez, M.-Y. Kao, C. Hu, and S. Khandelwal, “Optimization of deep learning-based BSIM-CMG I-V parameter extraction in seconds,” in *2022 IEEE International Symposium on Radio-Frequency Integration Technology (RFIT)*, 2022, pp. 124–126.

[11] J.-H. Chen, F. Chavez, C.-T. Tung, S. Khandelwal, and C. Hu, “A single neural network global I-V and C-V parameter extractor for BSIM-CMG compact model,” *Solid-State Electron.*, vol. 216, p. 108898, Jun. 2024.

[12] F. Chavez, C.-T. Tung, M.-Y. Kao, C. Hu, J.-H. Chen, and S. Khandelwal, “Deep learning-based I-V global parameter extraction for BSIM-CMG,” *Solid-State Electronics*, vol. 209, p. 108766, 2023. doi:10.1016/j.sse.2023.108766.

[13] G. Guo, Z. Tang, Z. Cui, C. Li, and H. You, “GatedNN: An accurate deep learning-based parameter extraction for BSIM-CMG,” *Solid-State Electronics*, vol. 224, p. 109044, 2025.

[14] A. Ashai, A. Jadhav, A. K. Behera, S. Roy, and B. Sarkar, “Deep learning-based fast BSIM-CMG parameter extraction for general input dataset,” *IEEE Trans. Electron Devices*, vol. 70, no. 7, 2023.

[15] A. Singhal, G. Pahwa, and H. Agarwal, “A novel physics aware ANN-based framework for BSIM-CMG model parameter extraction,” *IEEE Trans. Electron Devices*, vol. 71, no. 5, pp. 3307–3314, May 2024.

[16] S. Lee, S. Eom, J. Jeong, J. Lee, S. Lee, H. Yun, Y. Ahn, and R.-H. Baek, “Multi-Task Learning for Real-Time BSIM-CMG Parameter Extraction of NSFETs With Multiple Structural Variations,” *IEEE Access*, vol. 12, pp. 184619–184628, 2024. doi: 10.1109/ACCESS.2024.3512612.

[17] A. Dutta, D. Rajasekharan, and Y. S. Chauhan, “Compact model parameter extraction using modular Q learning for nano-scale transistors,” in *2020 5th IEEE International Conference on Emerging Electronics (ICEE)*. New Delhi, India: IEEE, Nov. 2020, pp. 1–4.

[18] E. Papageorgiou, G. Alia, A. Buzo, G. Pelz, and T. Noulis, “MOSFET model parameter extraction using reinforcement learning,” in *2024 Panhellenic Conference on Electronics &amp; Telecommunications (PACET)*. Thessaloniki, Greece: IEEE, Mar. 2024, pp. 1–5.

[19] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, “Dueling network architectures for deep reinforcement learning,” in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” 2016. [Online]. Available:https://arxiv.org/abs/1511.05952

[21] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, AAAI Press, 2016, pp. 2094–2100.

[22] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, “Q-Learning Algorithms: A Comprehensive Classification and Applications,” *IEEE Access*, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[23] O. Maheshwari, A. Singh, and N. R. Mohapatra, ”Training-Free Parameter Extraction for Compact Device Models Using Sequential Bayesian Optimization With Adaptive Sampling,” *IEEE Trans. Electron Devices*, vol. 71, no. 12, pp. 7889-7895, Dec. 2024.