This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3183156

A Deep Reinforcement Learning Framework for High-Dimensional Circuit Linearization

Chao Rong, Jeyanandh Paramesh and L. Richard Carley†
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, USA
chaor @ andrew.cmu.edu, jeyanandh.paramesh @ ieee.org, lrc @ andrew.cmu.edu

***Abstract*—Despite the successes of Reinforcement Learning (RL) in recent years, tasks that require exploring over long trajectories with limited feedback and searching in high-dimensional space remain challenging. This paper proposes a deep RL framework for high-dimensional circuit linearization with an efficient exploration strategy leveraging a scaled dot-product attention scheme and search on the replay technique. As a proof of concept, a 5-bit digital-to-time converter (DTC) is built as the environment, and an RL agent learns to tune the calibration words of the delay stages to minimize the integral nonlinearity (INL) with only scalar feedback. The policy network which selects calibration words is trained by the Soft Actor-Critic (SAC) algorithm. Our results show that the proposed RL framework can reduce the INL to less than 0.5 LSB within 60, 000 trials, which is much smaller than the size of searching space.**

***Keywords*—Deep Reinforcement Learning, Circuits Calibration, High-Dimensional Searching, Attention Scheme**

I. Introduction

Linearity is one of the most challenging specifications in analog-to-digital and time-to-digital converters. In phase-locked loops (PLLs), nonlinearity in the phase comparison path leads to potential folding of out-of-band noise or to aliasing, resulting in in-band spurs, especially at near-integer channels [1]. Recently, PLL's based on digital-to-time converters (DTCs) have become popular due to the fundamentally superior resolution of DTC's over the time-to-digital converters (TDCs), which are the limiting components in TDC-based PLL's [2]-[3]. Although the DTC's topology is simple, it is not trivial to achieve an INL of less than ~1 LSB. Two major INL contributors are code-dependent supply settling error and static distortion due to the mismatches between the delay units. To mitigate supply induced INL, a high-speed regulator [2] and a replica DTC [3] can be attached to the supply.

The mismatch problem is more challenging due to its random nature. An intuitive solution to the mismatch-induced INL is to make each delay unit tunable and calibrate the DTC after fabrication. However, the high-dimensional search space hinders the use of this method. For example, a 5-bit DTC would have 32 delay stages, assuming each delay stage has a 3-bit calibration word, leading to a searching space of 7.9×10^{28}. Most of the prior techniques for nonlinearity calibration are based on least-mean squares (LMS) algorithms. In [4], an LMS-based pre-distortion technique is proposed to mitigate the DTC nonlinearity in Bang-Bang PLLs. Injecting dither into the loop through a DTC to scramble the spectrum then cancel it with an adaptive filter is another way of reducing fractional spurs [5]. Although these calibration engines are usually power efficient, they require substantial engineering effort and expertise. In addition, such methods are highly specialized thus can't be applied to any other kinds of circuits.

Deep reinforcement learning methods have produced stunning results in game playing [6]. Recently, reinforcement learning algorithms have been applied in the integrated circuit area. These RL applications can be categorized into three broad classes: circuit design, circuit response tuning, and circuit calibration. Circuit design automation focuses on optimizing circuit parameters for given target design specifications [7]-[8]. Circuit tuning problems usually consider how to tune the digital control words to track environmental changes such as supply voltage and temperature drift [9]. Similar to circuit tuning, circuit calibration methods also tune the control words of the circuits. However, in calibration problems, usually good initial words, the control words which can make the circuit meet design specifications, are not given. This is because it is hard to simulate those control words deterministically due to process variations and inaccuracies of the system model.

Besides gradient-based methods, evolutionary algorithms [10]-[11] are also widely used to solve non-linear, non-convex optimization or searching problems. The key difference between evolutionary methods and deep RL methods is that the former methods randomly sample the next action from a distribution while the latter methods select the next action from a policy network.

In this paper, we present an RL framework (Fig. 1) that can search target calibration words in high-dimensional space. A 5-bit DTC model is built to demonstrate the concept. Our goal is to calibrate each delay unit such that the output delay can be a linear curve against the delay code. We consider the linearity is good enough when the $|INL|_{max}$ is ≤ 0.5 LSB.

The key features of our framework are as follows:

- To boost the gradient required by backpropagation training, we adopt the scaled dot-product attention

†Corresponding author: L. Richard Carley.
This work was supported in part by the National Science Foundation under grant CRI1823235.

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3183156

scheme from Transformer [12] to quantify how new the recently visited states are.

- The training can easily diverge due to the high-dimensional search space, therefore it is necessary to reset the calibration words after a fixed number of trials. Inspired by the well-known upper confidence bound (UCB) algorithm [13], we choose the reset calibration words according to the product of their Q-values provided by the SAC and the number of steps in which that word is visited.

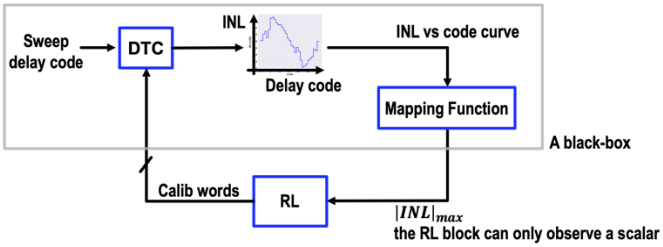

Fig. 1. Fig. 1. RL framework for DTC linearity calibration. No prior knowledge or internal signals are provided to the RL block. The goal of the RL agent is to find calibration words such that the $|INL|_{max}$ is less than 0.5 LSB.

The rest of the paper is organized as follows. Section II formulates the linearity calibration problem. We then introduce our framework in Section III and demonstrate the performance in Section IV. Section V draws conclusions.

II. Formulating The DTC Calibration Problem

As a proof of concept, we built all blocks in Python. To achieve background calibration, besides implementing the RL block on-chip, an internal feedback signal for RL is required. We suggest selecting the average phase error as the objective, which is widely used in traditional calibration methods [4]-[5].

A. DTC Model for RL

In general, there are two types of DTCs. The variable-slope DTC changes the delay by tuning the capacitive load [2]-[3], while the constant-slope DTC keeps the RC time constant fixed and uses a digital-to-analog converter (DAC) to pre-charge internal nodes [14]. Thus the delay is controlled by tuning the pre-charging voltage. The additional noise from the DAC limits the phase noise performance. Currently, the effort of constant-slope DTCs focuses on ultralow-power applications [15].

We choose the variable-slope topology in order to minimize the overall phase noise of the synthesizer. To ensure monotonicity and improve the linearity, we designed a 5-bit thermometric-coded DTC (Fig. 2) in TSMC 65nm CMOS process. The number of bits is usually limited by routing complexity. Further increasing the bits necessitates a segmented design comprising a coarse and a fine DTC [4]. It is reported that a TDC in 65nm process can achieve 10 ps resolution without calibration [16], therefore we targeted at 1ps delay resolution. The DTC has 32 delay units, each unit has 32 identical calibration cells. We size the R_{poly} to 50 Ω and C_L to 2 fF so that the calibration step is around 0.12 ps. By setting the default calibration word to 8 we can achieve a nominal delay of ~0.900 ps.

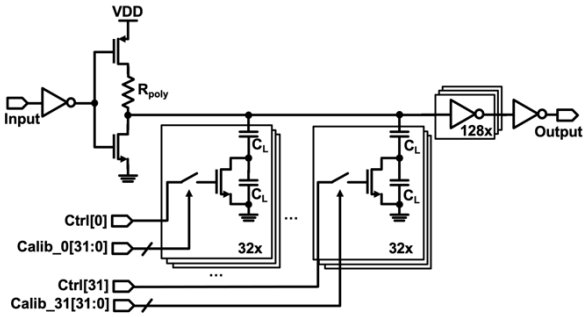

Fig. 2. A 5-bit DTC. Ctrl bits represent the input delay code, Calib_x bits are calibration words.

A 2000-point Monte Carlo simulation is run to extract the variation of the calibration step. The mean and standard deviation values from the Monte Carlo simulation are put into a multivariate Gaussian distribution to generate the DTC model (Fig. 3). The output delay is modeled as

$$Delay_{out}(i) = \sum_{j=1}^{i} delay_{unit}\left(j; Calib_j\right) \qquad (1)$$

The output delay at delay code i is equal to the summation of all the previous delay units and the i^{th} delay unit, where $I \in \{1, 2, \ldots, 32\}$. Each delay unit has a separate calibration word $Calib_j$, whose valid values are $\{1, 2, \ldots, 32\}$.

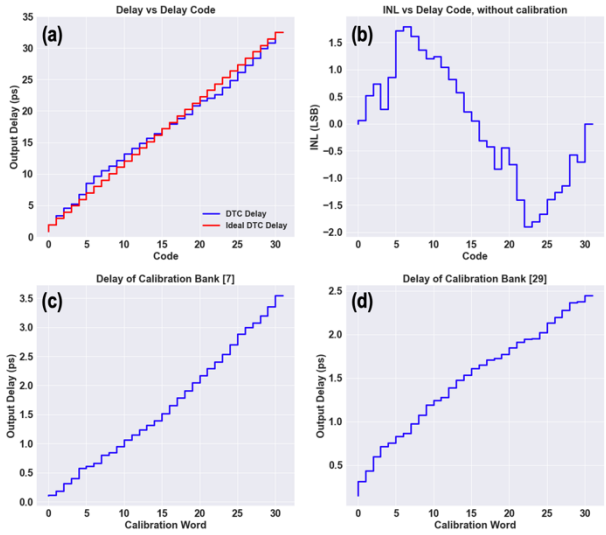

Fig. 3. Generated DTC model. (a) Output delay vs. delay code. (b) INL vs. delay code. (c) (d) Calibration steps. Notice that the calibration steps are highly nonlinear.

B. Definition of RL Environment

In general, the goal of RL is to maximize the accumulated numerical reward signal over time through its choice of actions. The action-taker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment [17]. At each discrete-time steps t, the agent receives an observation of the environment, state s_t. Given s_t, the agent selects an action a_t and executes the action. The environment then outputs a reward r_{t+1} and moves to the next state s_{t+1}.

During the training, it is common to reset the state to a pre-defined starting after a fixed number of trials. The collection of interaction tuples (s_t, a_t, s_{t+1}, r_{t+1}) between two resets is called an episode.

The 5-bit DTC environment is defined as follows:

State: the state s_t is a vector of size 32 where the j^{th} element corresponds to the calibration word $Calib_j \in \{1, 2, \ldots, 32\}$ of the j^{th} delay unit.

Action: the action a_t is a vector of size 32 where each element corresponds to the changing of calibration words. The valid values of each element are {-2, -1, 0, +1, +2}.

Transition: the next state $s_{t+1} = s_t + a_t$.

Feedback: We choose $|INL|_{max}$ as the scalar feedback. Firstly, it is very hard to measure picosecond level delay in practice, so we cannot assume that the RL agent can observe the whole delay vs. input code curve. Instead, one measurable quantity is the spectrum at the output of the PLL, and it is reported that the fractional spurs are related to the maximum INL of DTC [18], therefore, we assume that $|INL|_{max}$ is observable. Secondly, using $|INL|_{max}$ can make the calibration problem more challenging because now the sensitivity (Eq. (2)) changes with the states.

$$S_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{\partial(|INL|_{max})}{\partial(Calib_j)} \tag{2}$$

To validate this claim, let's say the $|INL|_{max}$ is located in the middle (j=16), then the sensitivity of the later stages (j = 17, …, 32) equals zero. After taking a few actions, assume that the $|INL|_{max}$ moves to a later stage (j=30), now only the last two stages have zero sensitivity. Therefore, performing sensitivity analysis only at the beginning of the training [8] might not work. However, the cost of performing sensitivity analysis is not trivial. In [8], the folded-cascode amplifier design example has 10 design specifications and 20 variables, performing one round of sensitivity analysis needs 200 circuit simulations, this number already exceeds the reported number of RL iterations. Therefore, even though the sensitivity analysis can prune some high-dimensional problems and speed up the convergence, we argue that the sensitivity analysis is not applicable to linearity calibration problems.

Environment reward function: To make this method applicable to other kinds of circuits, we simply use a straight line to avoid embedding any prior knowledge. The environment reward is given by Eq. (3).

$$r_t^{Env}(|INL|_{max}) = \begin{cases} +100, & if\,|INL|_{max} \le 0.5 \\ -10, & if\,|INL|_{max} > 6 \\ -1.8 \cdot |INL|_{max} + 0.9, & elsewhere \end{cases} \tag{3}$$

When the $|INL|_{max}$ is less than 0.5 LSB, the agent receives a high positive reward, otherwise, the reward should stay negative because we want to encourage the agent to find the target calibration words as fast as possible. The positive reward +100 is selected so that the environment reward is much larger than the exploration reward provided by the attention and random network distillation block. The negative reward part is a straight line that connects reward = 0 and reward = -10. Using a reward lower than -10, such as -20, can lead the training to diverge, thus we bound the minimum negative reward with -10. The algorithm is robust to the threshold $|INL|_{max}$ = 6, we tested with 5, 6, 7 and no significant differences were observed.

III. The Proposed RL Framework

The overall deep RL framework is shown in Fig. 4. There are four main parts: Soft Actor-Critic (SAC) [19], memory buffers, the attention block and the random network distillation (RND) [20].

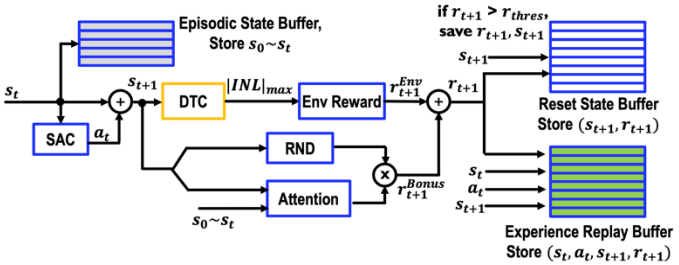

Fig. 4. The proposed deep RL framework. There are four main parts: The SAC selects action based on current state; The three memory buffers store experience tuples to train the critic and actor networks in the SAC block; The RND and attention modules provides bonus reward to encourage exploration.

The agent is trained by the Soft Actor-Critic algorithm. The actor network that maps state s_t to action a_t has 4 fully-connected layers with 128 hidden units, followed by the ReLU activation. The output of the actor network is first bounded by $2 \cdot tanh(x)$, then rounded to the nearest integer. The critic network Q_θ that estimates the *action-value function* $Q_\pi(s_t, a_t)$, has similar architecture but with 256 hidden units in each layer. The second part is the replay buffers. Experience tuples (s_t, a_t, s_{t+1}, r_{t+1}) are saved in the experience replay. To train the SAC agent, mini-batches from experience replay are sampled by using the proportional prioritization sampling technique [21]. The episodic state buffer stores all states visited in one episode and this buffer will be emptied at the beginning of the next episode. The states saved in the episodic buffer will be used by the attention block. The reset state buffer stores reset state candidates. To select the reset state for next episode, we proposed to search the reset state with a probability that is proportional to their $Q_\pi(s_t, a_t)$ and the number of steps in which that state is visited N_{count}.

$$Softmax\left(\min_{i=1,2} Q'_{\theta_i'}\left(S, \pi_\emptyset(A|S)\right) \cdot \sqrt{\frac{\mu}{N_{count}+1}}\right) \tag{4}$$

Another important part in Fig. 4 is the bonus reward block that quantifies the novelty of the state. Inspired by the episodic reward concept [22] and the Transformer [12], we modified the dot-product attention scheme to quantify the novelty of states within one episode, while the RND captures long-term novelty by keeping updating its weights during the training. The details of the attention and RND are presented in Fig. 5. Unlike the k-nearest neighbors algorithm (KNN) [22], the attention scheme adjusts the summation weights adaptively thus we can compare the new state s_{t+1} with all previously visited states, thus eliminating the need of considering how many neighbors should be included. The other portion of the

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3183156

bonus block, RND, is shown at the bottom of Fig. 5. The RND calculates the MSE between a frozen network and a trainable network. The prediction error is expected to be higher for states dissimilar to the ones the predictor has been trained on.

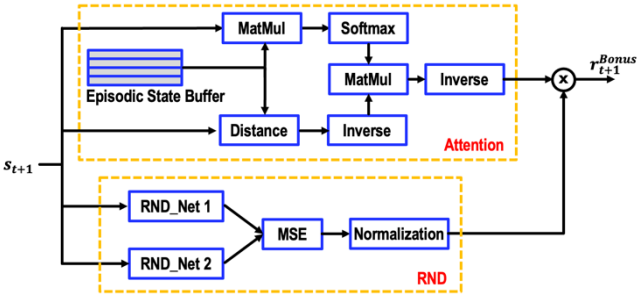

Fig. 5. The bonus reward block. To encourage exploration, states that are very different from previous states should be assigned a high bonus. The attention contains no trainable parameters, and the episodic buffer is emptied at the end of each episode, thus this module only captures short-term information. The long-range novelty is captured by the RND networks.

IV. Simulation Results

We generated four types of environments (Fig. 6). The first row and the second row are single-peak examples. The third and fourth rows are double-peak cases. We refer to them as Env_1, Env_2, Env_3 and Env_4 respectively. The last column in Fig. 6 shows that the INL lies between -0.5 LSB and 0.5 LSB, this proves that our agent can calibrate all four types of environments successfully.

To demonstrate the effectiveness of our method, we choose the Never Give Up (NGU) agent as the baseline method, which uses KNN episodic reward and RND to achieve good performance in hard exploration games [22]. Note that the distributed training procedure is not included. The other two RL methods for high-dimensional circuit design are AutoCkt [7] and DNN-Opt [8]. The AutoCkt samples a subset of design specifications and shows more efficiency over random agents. Since we assume that we can only observe one objective, the AutoCkt is not implemented, instead, we only compare the size of searching space and the number of iterations needed to reach the target. The DNN-Opt [8] is not compared because the sensitivity analysis is not applicable when the sensitivity changes with states.

We also compare our RL algorithm with the covariance matrix adaptation evolution strategy (CMA-ES), which is one of the most popular evolutionary algorithms with many successful applications [23]. We first sample N calibration word vectors (state s_t) from a random initialized multivariate Gaussian distribution, then N_{elite} samples with highest fitness scores are selected to update the mean vector μ_{t+1} and covariance matrix C_{t+1}. We choose N_{elite} = 256 which is the same as the size of mini-batch used in our RL method. The fitness function and update rule are given by Eq. (5).

$$\begin{aligned} fitness &= -|INL|_{max} \\ \mu_{t+1} &= \frac{1}{N_{elite}} \sum_{i=1}^{N_{elite}} s_t^i \\ C_{t+1} &= Cov\left(s_t^1, \ldots, s_t^{N_{elite}}\right) \end{aligned} \tag{5}$$

More complicated update rules [23] can also be used. However, as suggested in [11], the choice of evolutionary operations is somewhat arbitrary as long as they converge, given large enough time and number of samples. Therefore we select a simple update.

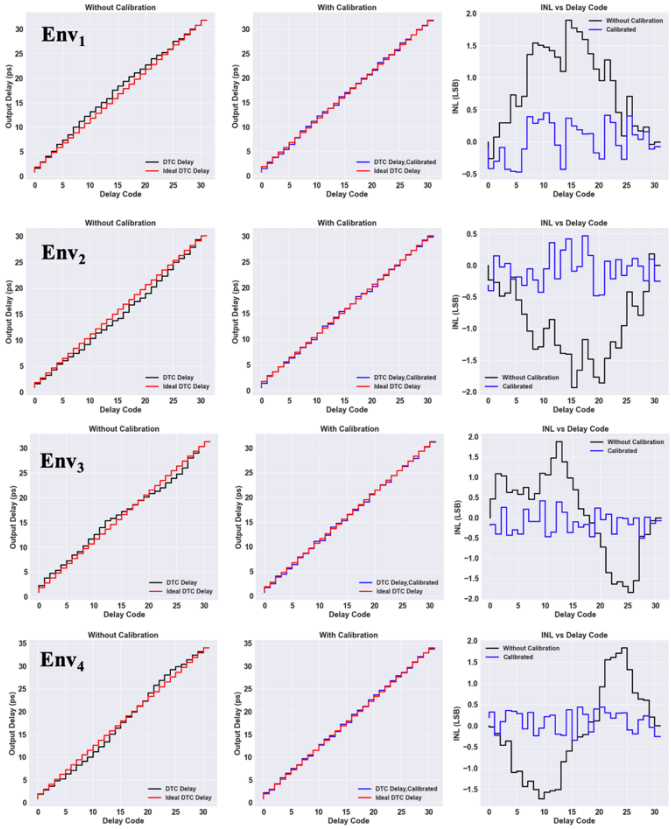

Fig. 6. Before / after calibration. i^{th} row represents Env_i. Red stairs are ideal linear curves. Black and blues curves are generated DTC, before and after calibration, respectively.

We run the experiments 10 times on all four generated DTC models and the average number of trials and success rate are shown in Fig. 7. The maximum allowed number of iterations is 10^5, the calibration is considered failed if the maximum allowed iterations number is reached. The random agent and CMA-ES agent fail to find any target calibration words for the 5-bit DTC. Since their curves completely overlap with each other, only random agent is drawn in Fig. 7. If we reduce the size of searching space (e.g. testing on a 3-bit DTC model), those two methods start to work. Those results indicate that purely random or sampling-based methods have limited exploration capability, which consist with the results reported in [7]-[8].

For the machine learning methods, the NGU can only find the target states on the single-peak cases, yet the success rates are less than 100%. But it fails to solve the double-peak examples.

We summarize the performance comparison in Table I. We list the average number of trials for each DTC model. Thanks to the attention reward and the reset buffer, our method can explore a larger space with fewer steps.

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3183156

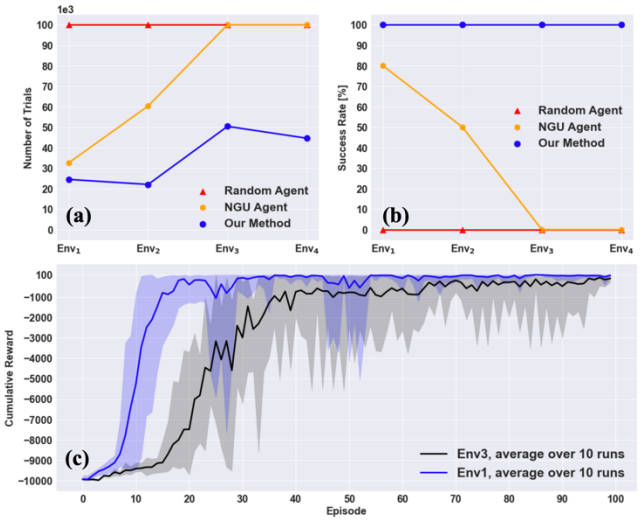

Fig. 7. Performance on 5-bit DTC environment. (a) shows the average number of trials over 10 runs, while (b) plots the success rate. The search is terminated once the agent finds the target calibration word. In (c), we keep the program running for 100 episodes and the learning curves of our method are plotted. Each mean cumulated reward (solid line) is surrounded by a shaded area bounded by the maximum and the minimum over 10 runs. In (a)(b), we notice that the two double-peak DTC models (Env_3, Env_4) need significantly more trials, the learning curve (Env_3) is also much noisier, as shown in (c).

TABLE I. **PERFORMANCE COMPARISON OF STATE-OF-THE-ART**

	Iterations	Size of Search Space
Random Agent	Failed	10^{19}
CMA-ES	Failed	10^{19}
AutoCkt [7]	170000	10^{14}
BagNet [11]	55102	10^{14}
NGU [22]	32618, 60354, Failed, Failed	10^{19}
Our Method	24517, 22065, 50425, 44620	10^{19}**

**: Since most of the design variables in AutoCkt only have 3~4 valid values, to compare fairly we calculate the state space by assuming that each calibration word only has 4 values (2 bits).

V. CONCLUSION

This paper presents a deep reinforcement learning framework that is capable of searching effectively in high-dimensional space. Attention episodic novelty and Q-value-based reset buffer are proposed to improve the exploration. Our algorithm's effectiveness has been successfully demonstrated on a linearity calibration problem and it shows superior sample efficiency compared to the prior state-of-the-art. Compared to prior work, our framework only requires scalar feedback thus the effort of human designers can be minimized. The proposed RL framework is also applicable when there are multiple optimization objectives. This is because multiple objects can always be combined into a single objective by a function, as simple as the product, together with some constraints.

REFERENCES

[1] Lacaita, Andrea Leonardo, Salvatore Levantino, and Carlo Samori. *Integrated frequency synthesizers for wireless systems*. Cambridge University Press, 2007.

[2] Wu, Wanghua, et al. "A 28-nm 75-fs rms Analog Fractional-N Sampling PLL With a Highly Linear DTC Incorporating Background DTC Gain Calibration and Reference Clock Duty Cycle Correction." in *IEEE Journal of Solid-State Circuits* 54.5 (2019): 1254-1265.

[3] Santiccioli et al., "A 66-fs-rms Jitter 12.8-to-15.2-GHz Fractional-N Bang–Bang PLL With Digital Frequency-Error Recovery for Fast Locking," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 12, pp. 3349-3361, Dec. 2020, doi: 10.1109/JSSC.2020.3019344.

[4] S. Levantino, G. Marzin and C. Samori, "An Adaptive Pre-Distortion Technique to Mitigate the DTC Nonlinearity in Digital PLLs," in *IEEE Journal of Solid-State Circuits*, vol. 49, no. 8, pp. 1762-1772, Aug. 2014, doi: 10.1109/JSSC.2014.2314436.

[5] C. Ho and M. S. Chen, "A fractional-N digital PLL with background-dither-noise-cancellation loop achieving <-62.5dBc worst-case near-carrier fractional spurs in 65nm CMOS," *2018 IEEE International Solid -State Circuits Conference-(ISSCC)*, 2018, pp. 394-396, doi: 10.1109/ISSCC.2018.8310350.

[6] Mnih, V., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[7] Settaluri, Keertana, et al. "AutoCkt: deep reinforcement learning of analog circuit designs." *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020.

[8] Budak, Ahmet F., et al. "DNN-Opt: An RL Inspired Optimization for Analog Circuit Sizing using Deep Neural Networks." *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021.

[9] Wang, Fei, et al. "An artificial-intelligence (AI) assisted mm-wave Doherty power amplifier with rapid mixed-mode in-field performance optimization." *2019 IEEE MTT-S International Microwave Conference on Hardware and Systems for 5G and Beyond (IMC-5G)*. IEEE, 2019.

[10] Eiben, Agoston E., and James E. Smith. *Introduction to evolutionary computing*. Vol. 53. Berlin: springer, 2003.

[11] Hakhamaneshi, Kourosh, et al. "BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks." *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019.

[12] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

[13] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." *Machine learning* 47.2 (2002): 235-256.

[14] H. Liu et al., "A 0.98mW fractional-N ADPLL using 10b isolated constant-slope DTC with FOM of −246dB for IoT applications in 65nm CMOS," *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 246-248, doi: 10.1109/ISSCC.2018.8310276.

[15] P. Chen, F. Zhang, Z. Zong, S. Hu, T. Siriburanon and R. B. Staszewski, "A 31-μW, 148-fs Step, 9-bit Capacitor-DAC-Based Constant-Slope Digital-to-Time Converter in 28-nm CMOS," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 3075-3085, Nov. 2019, doi: 10.1109/JSSC.2019.2939663.

[16] W. Wu, R. B. Staszewski and J. R. Long, "A 56.4-to-63.4 GHz Multi-Rate All-Digital Fractional-N PLL for FMCW Radar Applications in 65 nm CMOS," in *IEEE Journal of Solid-State Circuits*, vol. 49, no. 5, pp. 1081-1096, May 2014, doi: 10.1109/JSSC.2014.2301764.

[17] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[18] P. Chen, J. Yin, F. Zhang, P. -I. Mak, R. P. Martins and R. B. Staszewski, "Mismatch Analysis of DTCs With an Improved BIST-TDC in 28-nm CMOS," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 1, pp. 196-206, Jan. 2022, doi: 10.1109/TCSI.2021.3105451.

[19] Haarnoja, Tuomas, et al. "Soft actor-critic algorithms and applications." *arXiv preprint arXiv:1812.05905* (2018).

[20] Burda, Yuri, et al. "Exploration by random network distillation." *arXiv preprint arXiv:1810.12894* (2018).

[21] Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).

[22] Badia, Adrià Puigdomènech, et al. "Never Give Up: Learning Directed Exploration Strategies." *International Conference on Learning Representations*. 2020.

[23] Hansen, Nikolaus. "The CMA evolution strategy: a comparing review." *Towards a new evolutionary computation* (2006): 75-102.