Analog Circuit Design with Dyna-Style Reinforcement Learning

Wook Lee, Frans A. Oliehoek

Delft University of Technology

NEURAL INFORMATION PROCESSING SYSTEMS

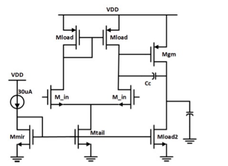
Introduction: Analog Circuit Design

The objective of analog circuit design is to search for **optimal circuit parameters (Xs)** satisfying a set of design constraints (Ys), given a circuit topology.

- **Xs: geometric size of transistors, resistance, capacitance, etc.**
- Ys: power consumption < 1mW, voltage gain > 100 V/V, etc.

Two key aspects in circuit optimization are mainly considered.

- Sample efficiency: minimize costly simulations (e.g., post-layout design)
- Solution generalization: learn the general structure of diverse solution space



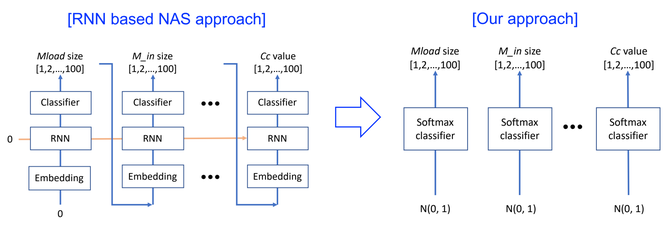
Circuit schematic of two-stage op-amp.

Motivation: Benchmarking NAS

The circuit optimization can be formulated as a **RL (reinforcement learning) problem** like NAS (neural architecture search) [1].

- Action (<-> generation of *discretized* circuit parameters), reward (<-> estimate of circuit performance), etc.

However, the RNN (recurrent neural net.) connection in the policy network does not help for the problem according to our experiments. Thus it can be greatly simplified as **a set of softmax classifiers** acting as independent learning agents.

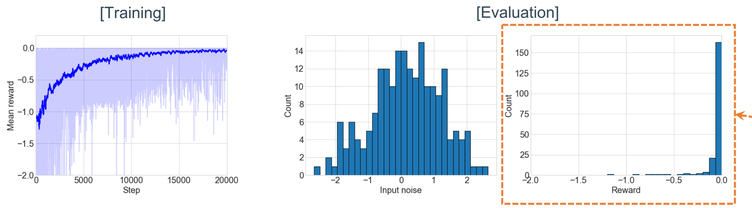


Policy network used in RL problem

Methods

1) Policy Generator

Inspired from the GAN (generative adversarial net.) [2], it is trained such that it **learns to map the random noise to a distribution of on-policy actions**, which approaches the feasible solution space as the policy improves.



Policy training curve (REINFORCE) | Input Gaussian noise distribution | Output reward distribution of generated actions (200 samples)

2) Reward Model

Reward evaluation based on circuit simulation is modeled using **learnable neural nets**., which enables to leverage sample-efficient model-based RL.

3) DynaOpt: RL based Optimization Framework

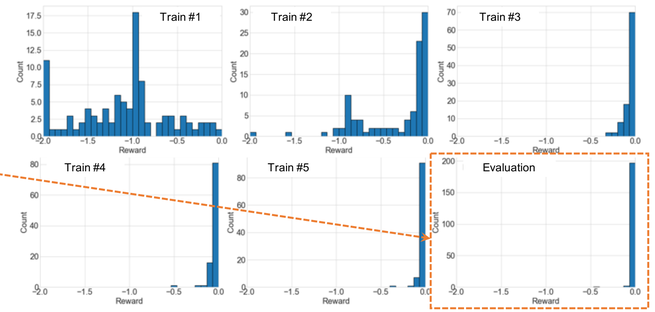
To maximize the sample efficiency, **both the mode-based and model-free RL methods are integrated** to a unified Dyna-style [3] algorithm to learn from scratch.

Algorithm 1 Proposed DynOpt method.

Initialize policy generator $\pi_\theta(\boldsymbol{a}|\boldsymbol{z})$, reward model $\rho_\phi(\boldsymbol{a})$ and sample buffer B
For number of training cycles **do**
. **For** $i = 1 : N_{direct}$ **do** // Loop for model-free RL
. . Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
. . Sample a vector of T actions $\boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{z})$ // T: number of circuit parameters
. . Estimate reward R *using circuit simulation with* $\boldsymbol{a}$
. . Update policy parameters θ based on REINFORCE rule
. . Store action-reward pair $(\boldsymbol{a}, R)$ in B
. **End**
. Update reward model parameters ϕ by regression on B
. **For** $i = 1 : N_{model}$ **do** // Loop for model-based RL
. . Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
. . Sample $\boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{z})$
. . Estimate R *using reward model* $\rho_\phi(\boldsymbol{a})$
. . Update θ based on REINFORCE rule
. **End**
End

Experiments

DynaOpt is applied to the design of a two-stage op-amp circuit over 5 training cycles with **500 simulations** in total. → It outperforms the model-free RL approach trained with **20,000 simulations**.



Evolution of reward distributions with the policy improvement
(100 simulation samples per training cycle & 200 samples in final evaluation)

Summary

- We propose DynaOpt (i.e., Dyna-style RL based optimization framework) for analog circuit design, by intermixing the model-free and model-based methods with two key components - **the stochastic policy generator** and **the reward model**.
- The policy generator learns to map the random noise input to a distribution of optimal actions, and the reward model allows to leverage sample the efficient model-based RL.
- The proposed methodology is expected to be readily applicable to a broad range of simulation-based optimization problems in engineering.

References

[1] Barret Zoph and Quoc V. Le (2017). "Neural architecture search with reinforcement learning." In: In International Conference on Learning Representations.

[2] Ian J. Goodfellow et al. (2014). "Generative adversarial networks." In: Advances in Neural Information Processing Systems.

[3] Richard S. Sutton (1991). "Dyna, an integrated architecture for learning, planning, and reacting." In: ACM Sigart Bulletin.