Domain Knowledge-Infused Deep Learning for Automated Analog/Radio-Frequency Circuit Parameter Optimization

Weidong Cao[1,2], Mouhacine Benosman[1], Xuan Zhang[2], Rui Ma[1]
[1]Mitsubishi Electric Research Laboratories; [2]Department of ESE, Washington University in St. Louis

ABSTRACT

The design automation of analog circuits is a longstanding challenge. This paper presents a reinforcement learning method enhanced by graph learning to automate the analog circuit parameter optimization at the pre-layout stage, i.e., finding device parameters to fulfill desired circuit specifications. Unlike all prior methods, our approach is inspired by human experts who rely on domain knowledge of analog circuit design (e.g., circuit topology and couplings between circuit specifications) to tackle the problem. By originally incorporating such key domain knowledge into policy training with a multimodal network, the method best learns the complex relations between circuit parameters and design targets, enabling optimal decisions in the optimization process. Experimental results on exemplary circuits show it achieves human-level design accuracy (∼**99**%) with **1.5**× efficiency of existing best-performing methods. Our method also shows better generalization ability to unseen specifications and optimality in circuit performance optimization. Moreover, it applies to design radio-frequency circuits on emerging semiconductor technologies, breaking the limitations of prior learning methods in designing conventional analog circuits.

1 INTRODUCTION

Analog circuits play the fundamental role in processing analog signals and bridging the physical analog world and digital information world. Unlike digital circuits following standard automated design flows, analog circuit design relies on onerous human efforts and lacks effective design automation techniques at all stages. Prelayout design is one key stage in analog circuit design flow. It can be formulated as a parameter-to-specification (P2S) optimization problem, i.e., finding optimal device parameters (e.g., width and finger number of transistors) to meet desired circuit specifications (e.g., power and bandwidth) based on a pre-determined circuit topology. This problem is very challenging as it seeks optimum parameters of diverse devices in a huge design space without exact rules.

Various automated techniques have been proposed for the P2S problem, mainly falling into optimization/learning-based category. Optimization methods, e.g., Bayesian Optimization [8] and Genetic Algorithm [6], use corresponding algorithms to search for optimal device parameters. They often suffer from several key issues, such as divergence, and re-starting from scratch if any change is made on given specifications. Learning methods, i.e., supervised learning (SL) methods [5, 10] and reinforcement learning (RL) methods [13, 16], have emerged recently. They can achieve good convergence and cover a huge design space once well trained. Despite the promise, these learning methods are still unable to reach human-level design accuracy, i.e., ∼100%. SL methods learn the static mapping between device parameters and circuit specifications. Due to the inherent approximation errors, they cannot ensure high design accuracy and endure weak generalization abilities with one-step inference. RL methods learn a decision policy from state space of circuits to action space of device parameters and are often superior to SL methods via multi-step deployment. However, without incorporating sufficient key state observations from environments into training, they fail to accurately learn the complex relations between device parameters and circuit specifications, leading to sub-optimal policies. Moreover, existing learning methods cannot be applied to design more advanced analog circuits, e.g., radio-frequency (RF) circuits, which require sophisticated time-consuming characterizations. Without overcoming the issue, a much longer training time is needed by them before used for inference/deployment.

In this paper, we propose a domain knowledge-infused RL method to achieve human-level design accuracy and superior design efficiency for analog and RF circuits. We are inspired by experienced human designers who leverage the key domain knowledge, e.g., topologies of circuits and couplings of specifications, to derive device parameters. Particularly, they adopt a simplified circuit topology of a circuit, carefully consider design trade-offs between specifications, and use tens/hundreds of iterative fine tunings to seek the optimal circuit parameters. Our RL method infuses the key domain knowledge into policy learning with a tailored multimodal policy network composed of a graph neural network (GNN) and a fully connected neural network (FCNN). The GNN is built upon the topology of a given circuit. It can capture the underlying physics of the circuit, e.g., device's connections and interactions. The FCNN extracts the complex couplings of circuit specifications. With such a unique policy network, our RL agent learns the best policy and makes optimal sequential decisions like a human expert to find device parameters. Key contributions in the work are:

- This paper presents the first domain knowledge-infused RL method to automate the P2S optimization of analog/RF circuit at the pre-layout level.
- This work proposes a unique multimodal policy network made of a circuit topology-based GNN and an FCNN to infuse key domain knowledge of circuit design into policy learning.
- The work also leverages transfer learning to notably accelerate RF circuits' design in a sophisticated and time-consuming simulation environment with the learned experiences from a coarse but time-efficient simulation environment.
- Experiments show the method achieves 99% design accuracy, 1.5× design efficiency of existing best-performing methods, a stronger generalization ability to unseen specifications, and better optimality in maximizing circuit's figure-of-merit.

2 BACKGROUND

Reinforcement Learning (RL): As shown in Figure 1(a), RL is an area of machine learning related to how an intelligent agent takes actions to maximize the cumulative return based on observed states

DAC '22, July 10–14, 2022, San Francisco, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9142-9/22/07.
https://doi.org/10.1145/3489517.3530501

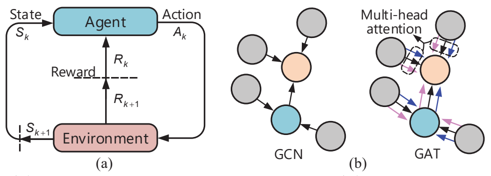

Figure 1: (a), A simplified illustration of RL. (b), A simplified illustration of a graph. Solid circles denote nodes and lines between circles represent edges. The graph is processed by two GNNs: GCN and GAT.

from an environment . In each episode, the agent starts from an initial state. It then observes the state S_k and takes an action A_k based on a policy. Meanwhile, the environment updates a reward R_{k+1} for that action and enters into a new state S_{k+1}. The agent iterates through the episode with multiple steps, accumulating the reward at each step to obtain the final return. With multiple episodes, the agent improves its decision quality and finally finds a well-learned policy to maximize the return. The policy would be deployed for practical tasks, i.e., the agent follows the policy to finish a given task. We apply RL to the P2S optimization of analog/RF circuits, which can best mimic the dynamic design process of human experts.

Learning with Graph Neural Networks (GNNs): GNNs [4, 15] directly learn the non-Euclidean data structure resembling a graph as shown in Figure 1(b). The graph is represented as $G = (V, E)$ with V the set of node and E the set of edge between connected nodes. Assuming each node $v_i \in V$ has an m-dimensional vector of features, all node features form an matrix $X \in \mathbb{R}^{n \times m}$, $n = |V|$. A GNN takes in X as inputs and uses the class of each node in a graph or the class of an entire graph as labels. Graph convolutional network (GCN) [4] and graph attention network (GAT) [15] are two representative GNNs. Compared to GCN, GAT has a multi-head attention mechanism on nodes as indicated in Figure 1(b) and can better learn high-dimensional complex relations between nodes.

Circuit topology is a graph and can be processed by GNNs. A prior RL method [16] uses GCN to process a circuit topology but has two key issues. First, only a partial circuit topology is adopted by excluding power supply and bias nodes which, however, are the indispensable parts of a circuit graph. Second, the GCN node features are all static technology information, such as threshold voltage and electron mobility. Without including the essential dynamic (variable) device parameters into node features, it is hard to learn the relations between device parameters and circuit specifications. There are also several SL methods applying GNN to physical design [7, 11] and electro-magnetic simulation [17] of analog circuits. In contrast, our work harnesses GCN/GAT as a key part of our RL policy network to capture the physics of a given circuit topology, e.g., device's parameters, connections, and interactions, at the pre-layout stage. We show that a GAT with the multi-head attention can better model a circuit topology than a GCN.

3 APPROACH

We target the P2S problem of analog/RF circuit design at the pre-layout stage and propose an RL approach for it. Figure 2 shows the proposed RL method with the following five key elements.

Reward Function: The reward is directly related to the design goal. We define the reward r_i at each time step i as

$$r_i = r, \text{ if } r < 0 \text{ or } r_i = R, \text{ if } r = 0, \tag{1}$$

where $r = \sum_{j=0}^{N-1} \min\{(g_i^j - g_*^j)/(g_i^j + g_*^j), 0\}$ is a normalized difference between the intermediate specifications g_i and the given specifications g_*. The upper bound of r is set to be 0 to avoid over-optimizing the parameters once the given specifications are reached. All N specifications are equally important. We also give a large reward (i.e., $R = 10$) to encourage the agent if the design goals are reached at some step. The episode return R_{s_0,g_*} of searching optimal device parameters for the given goals g_* starting from an initial state s_0, is the accumulated reward of all steps: $R_{s_0,g_*} = \sum_{i=0} r_i$. Our goal is to train a good policy to maximize R_{s_0,g_*}.

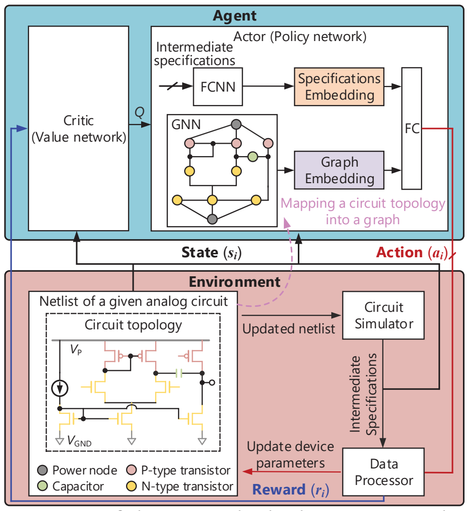

Figure 2: Overview of the RL method. The RL agent is based on an actor-critic method. Our multimodal policy network consists of a circuit-topology-based GNN and an FCNN. We use a two-stage Op-Amp to show how to map a circuit topology into a graph.

Action Representation: Inspired by human designers who iterate with fine-grained tuning steps to find optimal device parameters, we use discrete action space to tune device parameters. For each tunable parameter x of a device (e.g., width and finger number of transistors), there are three possible actions at each step: increasing $(x+\triangle x)$, keeping $(x+0)$, or decreasing $(x-\triangle x)$ the parameter, where "$\triangle x$" is the smallest unit to update the parameter within its bound $[x_{\min}, x_{\max}]$. Assuming total M device parameters, the output of the policy network is an $M \times 3$ probability distribution matrix with each row corresponding to a parameter. The action is taken based on the probability distribution.

Environment: A circuit design environment is used in this work. It consists of a given circuit netlist, an industrial circuit simulator, such as Cadence Spectre or Keysight Advanced Design system (ADS) (for high-frequency RF circuits), and a data processing module (DPM). As shown in Figure 2, the simulator obtains intermediate circuit specifications at each time step. The DPM then deals with the simulated results to feed back a reward to the agent using Eq. (1). Meanwhile, it updates the device parameters to rewrite the circuit netlist based on the actions from the agent.

State Representation: Capturing critical and adequate domain knowledge from the environment is key to training a good RL agent. In a circuit design environment, the circuit itself and the intermediate specifications are the main domain observations. In our work, we for the first time adopt these two key practical observations to represent each state s_i. We use a graph $G(V, E)$ to model the circuit based on its topology, where each node in set V is a device and the connections between devices form the edge set E. We also treat the power supply (V_P), ground (V_{GND}), and other DC bias voltages as extra nodes. Figure 2 takes a two-stage operational amplifier (Op-Amp) as an example to show the mapping between its topology and the graph. For a circuit with n nodes, the state for the k^{th} node is defined as its node feature $(t, \vec{p})$, where t is the binary representation of the node type and $\vec{p}$ is the parameter vector of the node. For transistors, the parameters are the width (x_W) and the finger number (x_F) while for capacitors, resistors, and inductors, the parameter is the scalar value of each device. The parameter for power supply (ground or DC bias) is a voltage of V_P (0 for V_{GND} or $V_{\text{bias},k}$ for bias node k). Zero padding is used to ensure that the length of $\vec{p}$ for each node is the same. For a circuit with five different types of devices, two power nodes, one bias, the state of an N-type transistor is $[0, 0, 1, x_W, x_F]$. We also create a vector to represent intermediate specifications. For example, to design the Op-Amp, the state vector of specifications is expressed as $[G, B, PM, P]$ which are gain (G), bandwidth (B), phase margin (PM), and power consumption (P).

Agent: To incorporate the key domain knowledge into agent training such that it can make human-level decisions, we propose a novel multimodal policy network for the agent based on actor-critic method [9] as shown in Figure 2. It consists of a circuit topology-based GNN and a fully connected neural network (FCNN), which is termed GNN-FC-based policy network. The GNN is to distill the underlying physics (e.g., device's types, parameters, and interactions) of a circuit graph into low-dimensional vector embedding. While the FCNN takes the design goals as inputs to extract their coupled relations, i.e., design trade-offs. The graph embedding and the FCNN embedding are then concatenated for further processing by the final fully-connected (FC) layers to update the actions.

We use GCN [4] and GAT [15] to learn the embedding of circuit-level physical features respectively from the circuit graph $G = (V, E)$. As an example, we show how to build the GCN below. GAT [15] can also be built similarly which is not elaborated here. The node features of the $(l+1)^{\text{th}}$ layer in the GCN are obtained as

$$H^{l+1} = f(H^l, A^*) = \sigma(A^* H^l W^l). \tag{2}$$

Here, $H^l \in \mathbb{R}^{n \times m_l}$ is the node feature matrix of the l^{th} layer (n: number of nodes, m_l: feature dimension per node in the layer). $H^0 = X$ is the initial input node feature matrix. W^l is a weight matrix which combines the aggregated node features and pass them into a learnable layer (i.e, the l^{th} layer) with a non-linear activation function σ (i.e., tanh in our work). A^* is the matrix used to aggregate the neighbourhood features for a node, which is defined as: $A^* = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$, $\hat{A} = A + I$. Here, A is the adjacent matrix of the circuit graph; I is an identity matrix; $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$. Using A^* for aggregation is straightforward, as a device in a circuit graph is directly affected by its neighbors. By stacking multiple GCN layers, one device can receive information from farther devices that do not have a direct connection with it.

Combining the GNN, FCNN, and FC forms our policy network $\pi_\theta(a|s)$ parameterized by $\theta = \{W_{\text{GNN}}, W_{\text{FCNN}}, W_{\text{FC}}\}$ with W_{GNN}, W_{FCNN}, and W_{FC} the learnable parameters of the GNN, FCNN and FC. The value network preserves the same structure as the policy network except of the last layer. It evaluates the actor's decision quality by yielding an estimation of the expected reward, Q, for the current policy execution. The objective function of the problem can be formally defined as $J(\theta, G) = 1/H \cdot \sum_{g \sim G} \mathbb{E}_{g, s \sim \pi_\theta}[R_{s,g}]$. Here, H is the the space size of all desired specifications G and $R_{s,g}$ is the episode reward. Our goal is to make the RL agent gain rich circuit design experiences by interacting with the environment. Given

Algorithm 1 Proximal Policy Optimization (PPO) Optimization

```
1: Input: initial policy parameters $\theta_0$ and initial value function parameters $\phi_0$
2: for $k = 0, 1, 2, \cdots$ do
3:     Collect a set of trajectories/episodes $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \theta_k$ in the circuit design environment.
4:     Compute rewards $\hat{R}_t$ for the trajectories/episodes.
5:     Compute advantage estimates, $\hat{A}_t$ based on the current value function $V_{\phi_k}$.
6:     Update the policy by maximizing the PPO-clip objective in Eq. (3), via stochastic gradient ascent with Adam [3].
7:     Fit value function by regression on mean-squared error:
           $\phi_{k+1} = \arg\min 1/(|\mathcal{D}_k|T) \cdot \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2$,
       via stochastic gradient ascent with Adam [3].
8: end for
```

the cumulative reward for each episode, we use Proximal Policy Optimization (PPO) [12] to update the parameters of the policy network as shown in Algorithm 1 with a clipped objective below:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_i[\min(b_i(\theta), \text{clip}(b_i(\theta), 1-\epsilon, 1+\epsilon))\hat{A}_i], \tag{3}$$

where $\hat{\mathbb{E}}_i$ represents the expected value at time step i; b_i is the probability ratio of the new policy and the old policy, and $\hat{A}_i$ is the estimated advantage at time step i.

Transfer Learning: We use transfer learning to speed up RF circuit design. Generally, AC and DC simulations are sufficient to obtain all intermediate specifications g_i at time step i for low-frequency analog circuits (e.g., two-stage Op-Amps). Such simulations are fast within tens of milliseconds in Cadence Spectre without delaying the learning of RL agents. However, RF circuits (e.g., RF power amplifiers) often need more sophisticated simulations to obtain accurate intermediate specifications which is timing-consuming. Typically, one uses Harmonic Balance (HB) simulation (~1 minute/round in ADS) to attain intermediate specifications. It significantly delays the reward calculation and training of RL agents. To tackle the issue, fast (~1 second) but rough DC simulation is used to replace HB simulation. It can obtain the not-very-accurate intermediate specifications for the quick approximation of the reward. Our analyses show that the approximated rewards are often in ±10% error range compared to the ones obtained from the HB simulation. Therefore, the learning process is remarkably speeded up. However, during the deployment stage for design automation, we still use HB simulation to guarantee the design quality and reliability. In this way, the learned experiences from a coarse simulation environment can be accurately transferred into a fine simulation environment as verified by our results. We think this may be due to the fact that a coarse design environment also provides sufficient information for the RL agent to learn the complicated relation between the device parameters and specifications. For other advanced analog circuits, similar approximated rewards can also be obtained correspondingly.

4 EXPERIMENTS

Two circuits are used for evaluations. One is the CMOS two-stage Op-Amp as shown in Figure 2, which is a standard benchmark taken by prior methods [6, 8, 13, 16]. The other one is a gallium nitride (GaN) RF power amplifier (PA) [2] whose schematic is shown in Figure 4. GaN is a promising alternative for conventional CMOS technology and for high-frequency power electronic applications [14].

Table 1: Design space of device parameters and sampling space of desired specifications of two circuit benchmarks.

Circuit types	Two-stage Op-Amp				RF PA	
Implementation technology	45 nm CMOS				150 nm GaN	
# of device parameters	$2 \cdot 7 + 1 = 15$				$2 \cdot 7 = 14$	
Parameter constraints (Design space)	Width (μm) [1, 100]	# of fingers [2, 32]	capacitance (pF) [0.1, 10]		Width (μm) [16, 100]	# of fingers 1, 2, ..., 16
Desired specifications (Sampling space)	Gain (G) [300, 500]	Bandwidth (B) $[10^6, 2.5 \cdot 10^7]$ Hz	Phase margin (PM) $[55^\circ, 60^\circ]$	Power consumption (P) $[10^{-4}, 10^{-2}]$ W	Power efficiency (E) [50%, 60%]	Output power (P) [2, 3] W

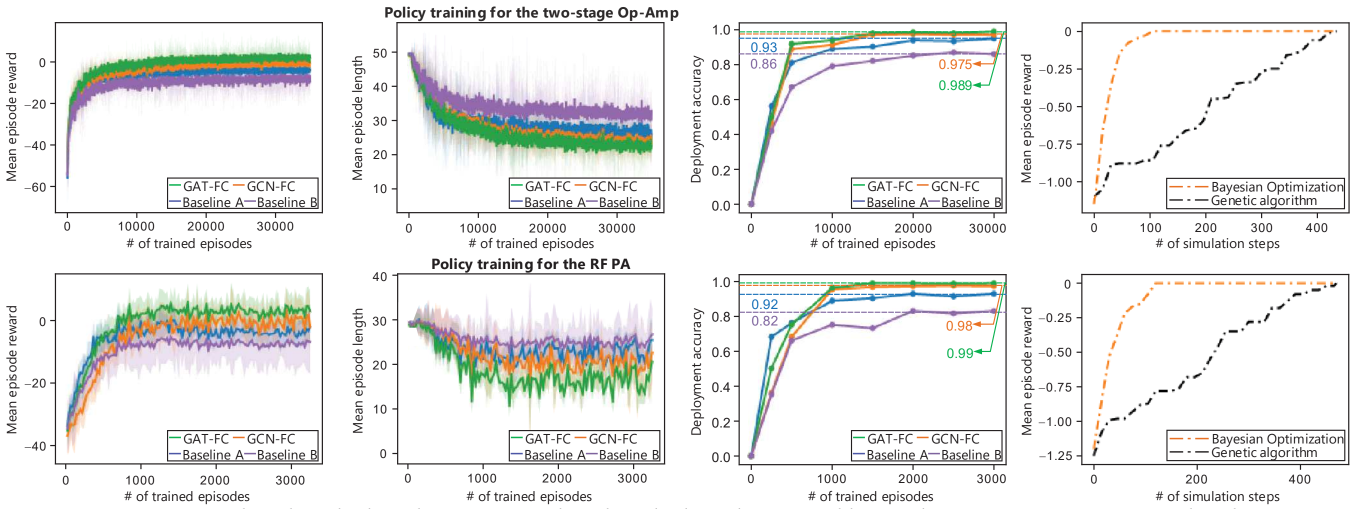

Figure 3: Comparing RL-based methods and optimization-based methods in the P2S problem with ours. Two rows correspond to the two-stage Op-Amp and the RF PA. All results of RL methods are based on 6 random seeds.

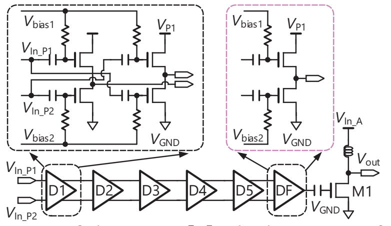

Figure 4: Schematic of the RF PA [2] which consists of a driver stage (D1~D5 and DF) and a power amplifying transistor (M1).

The design space of device parameters and the sampling space of desired specifications for the two circuits are listed in Table 1.

We adopt prior methods, i.e., Genetic Algorithm [6], Bayesian Optimization [8], and RL methods [13, 16] as our baselines. These prior arts focus on two problems, i.e., P2S optimization [6, 13] and figure-of-merit (FoM) optimization [8, 16]. The prior RL methods exclude the key domain knowledge into policy learning and are not capable of designing RF circuits. Baseline A (i.e., prior work [13]) simply observes intermediate and given specifications from the environment and vectorizes them to train a feedforward policy network. Baseline B (i.e., prior work [16]) uses all static semiconductor technology information as observations to train a partial circuit topology-based GCN as the policy. Such a method is often found to be divergent during training. For conservative comparisons, we interpret and implement these RL arts [13, 16] with our method. First, we use the GCN design in our policy network as a more advanced implementation for the Baseline B. Note that our GCN design is not only built upon a full circuit topology but also uses the essential dynamic (variable) device parameters as node features to better learn the relations between device parameters and circuit specifications. Second, we build these RL baselines with the PPO technique [12] and discrete action space as done in our work as well as the transfer leaning technique to enable them to design RF circuits. Our methods have two versions: GCN-FC policy and GAT-FC policy as GCN and GAT are respectively used as the GNN to capture the underlying physics of a full circuit topology. We build circuit graphs using Deep Graph Library [1] and implement all methods with PyTorch. We use equal amount of network parameters and the same set-ups for each baseline. All our experiments are performed on an 8-core Intel CPU. Moreover, surprised learning method [10] and a GAT-based implementation of Baseline B are also used as auxiliary comparisons with our method, whose training results are not shown in the paper but summarized in Table 2.

P2S Optimization: Figure 3 shows the training curves (i.e., mean episode reward, mean episode length, and deployment accuracy) of different RL methods for the P2S optimization. The maximum episode length for each Op-Amp agent (RF PA agent) is set to be 50 (30). The total episodes used to train the two RL agents are chosen to be $3.5 \cdot 10^4$ and $3.5 \cdot 10^3$, respectively. As observed, our method achieves higher reward (first column), shorter mean episode length (second column), and higher deployment accuracy (third column) than all RL baselines. Policy deployment applies a trained policy to automatically find the device parameters for given specifications. Each point in Figure 3 (third column) comes from deploying each RL agent for 200 groups of randomly sampled specifications in Table 1.

Given the desired circuit specifications, Genetic algorithm [6] and Bayesian Optimization [8] use algorithms to guide the searching process by maximizing r in Eq. (1) without training. The last column in Figure 3 shows the optimization curves. However, they cannot leverage transfer learning and have to use HB simulation to ensure design quality, which is time-consuming. We observe that Genetic Algorithm (Bayesian Optimization) often requires ~400 (~100) steps/simulations to find optimal device parameters, incurring long run-time delay. Moreover, due to the limitations, such as being stuck at a local optimum and even divergence, the algorithm cannot guarantee the correctness of each design. Based on 30-group random experiments, the design accuracy is 76.7% (83.7%) for the Genetic Algorithm (Bayesian Optimization).

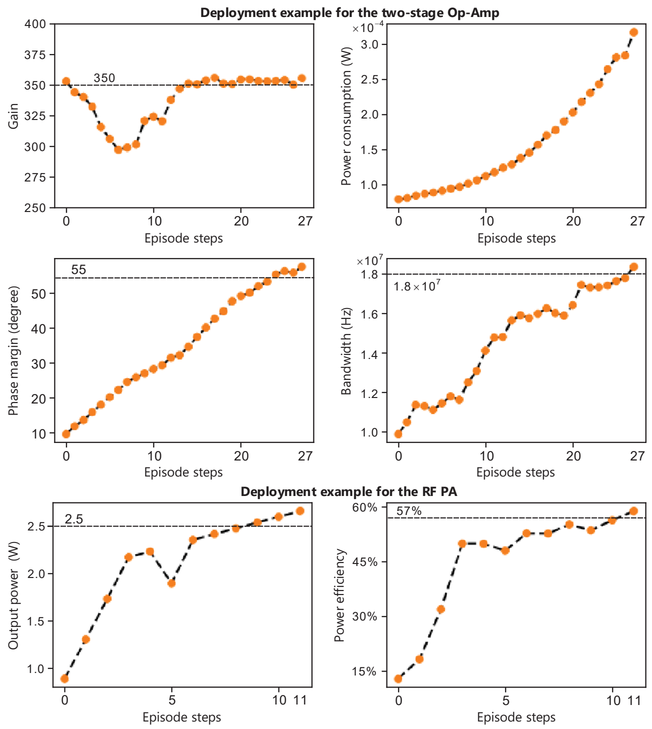

Figure 5: Deployment examples of the trained RL agent attempting to reach one group of the target specifications for each circuit.

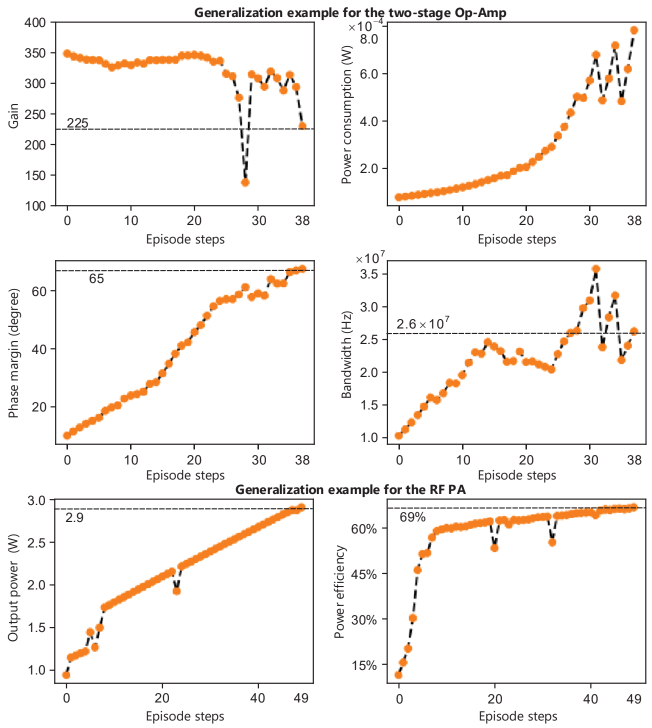

Figure 6: Generalization examples of the trained RL agent attempting to reach one group of the unseen new specifications for each circuit.

The comparison shows that the our methods achieve the highest design efficiency (with fewer deployment steps per episode, ∼20 steps for Op-Amp and ∼15 steps for RF-PA) and human-level design accuracy (higher policy deployment accuracy, 99%) for both circuits design. Particularly, we also note that the GAT-FC-based policy is superior to the GCN-FC-based policy. Such a comparison shows that circuit topology is an important ingredient in RL-based policy learning. And a better circuit topology modeling method, that is using GAT with the multi-head attention mechanism to learn higher-dimensional interactions among circuitry nodes, can further improve the performance of a policy.

Automated Design with Policy Deployment: We take our GCN-FC-based policy as an example to show the deployment process. Figure 5 illustrates the deployment where RL agents automatically find optimal device parameters for a group of randomly sampled specifications (the horizontal dashed lines in each sub figure). The sampled desired specifications for the two-stage Op-Amp are gain ($G = 350$), bandwidth ($B = 1.8 \cdot 10^7$ Hz), phase margin ($PM = 55°$), and power consumption ($P = 4 \cdot 10^{-3}$ W). And for the RF PA, they are output power ($P = 2.5$ W), and power efficiency ($E = 57\%$). Note that the smaller the power consumption is, the better the performance is. At the initial state, the intermediate specifications (y-axis of each sub-figure) often deviate a lot from the desired ones. As the deployment continues, they get closer to the desired ones by following the trained policy. An interesting phenomenon here is that when some specification is first achieved, the RL agent will not over-optimize it too much but instead try to optimize the remaining ones. For example, the gain of the two-stage Op-Amp is first attained at the 14^{th} deployment step. In the following steps, the RL agent focuses on optimizing phase margin and bandwidth. The similar analysis also applies to the design of the RF PA. We also analyze a few failed cases where our trained policy cannot converge to the optimal device parameters. We observe that in these cases, some specifications can converge to a neighborhood of the desired ones, but after which they deviate a bit from the goal. Fortunately, we find that by slightly tuning the device parameters with manual effort at that particular step, the design goal is also easily achieved. In this way, the design accuracy can be improved to 100%. These results show that human designers can still greatly benefit from the trained policy, if used as an efficient warm-start for the manual tuning, even if an automated deployment fails.

Generalization to Unseen Specifications: We also evaluate the generalization ability of our GCN-FC-based policy by deploying it with unseen specifications out of the sampling space in Table 1. Figure 6 shows such an example, where the horizontal dashed lines denote these unseen specifications: gain ($G = 225$), bandwidth ($B = 2.6 \cdot 10^7$ Hz), phase margin ($PM = 65°$), power consumption ($P = 6 \cdot 10^{-3}$ W) for the two-stage Op-Amp; output power ($P = 2.9$ W), and power efficiency ($E = 69\%$) for the RF PA. Compared to policy deployment with the specifications coming from the sampling space, the deployment with unseen specifications usually requires more search steps. For example, the generalization for the RF PA needs 49 steps to achieve the design goals while 11 steps are enough for the normal deployment in Figure 5. This is because that unseen specifications are beyond the scope of training datasets, thereby demanding more steps to reach optimal parameters. We also analyze the generalization ability of baseline methods (not shown here) and find that they often do not generalize well as ours even with a higher number of search steps. The better generalization ability of ours is attributed to the fact that it is capable of capturing key domain knowledge from state space, hence can better apply the learned experiences to unseen specifications at the inference time.

FoM Optimization: We also compare all methods in optimizing FoM by using the RF PA as an example. To apply the methods to this

Table 2: Comparison summarization of different design automation methods.

Methods	Sufficient key domain knowledge (?)		P2S optimization			FoM optimization
			Design accuracy	Mean # of design steps		FoM value
				Two-stage Op-Amp	RF PA	RF PA
Genetic Algorithm [6]	NO		76.7%	370	389	2.53
Bayesian Optimization [8]	NO		83.7%	86	105	2.61
Supervised learning [10]	NO		79%	1	1	N/A
RL method (Baseline A) [13]	NO		92%	27	23^{a}	2.92^{a}
RL method (Baseline B) [16]	NO^{b}		84% (87%)	32 (31)	25 $(23)^{a}$	2.81 $(2.86)^{a}$
Our RL method	**YES: Full circuit topology + Specification couplings**	**GCN + FCNN**	**98%**	**24**	**19**	**3.18**
		GAT + FCNN	**99%**	**21**	**16**	**3.25**

[a] They originally cannot design RF circuits. We leverage our transferring learning technique to enable them to design RF circuits.

[b] Implemented with our GCN (GAT) part: full circuit topology + device parameters as key node features.

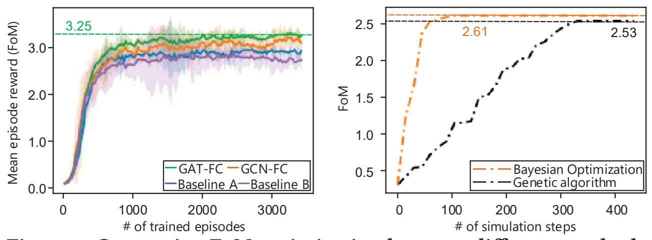

Figure 7: Comparing FoM optimization between different methods. All results of RL methods are reported based on 6 random seeds.

problem, we use the FoM definition [8] of RF PAs, i.e., $r_i = P_i + 3 \cdot E_i$ to revise the reward function in Eq. (1). Here, P_i, E_i are the intermediate specifications at time step i. In the training, we use references P_r, E_r for normalization, i.e., $r_i = (P_i - P_r)/(P_i + P_r) + 3 \cdot (E_i - E_r)/(E_i + E_r)$. For each RL method, we train the corresponding RL agent with 3.5×10^3 episodes. Figure 7 shows the optimization curves of all methods. Our methods (GAT-FC/GCN-FC) obtains a higher FoM and the GAT-FC-based policy attains the highest one, showing the superiority of our methods.

Comparison Summarization: We summarize the comparisons in Table 2. In tackling the P2S optimization, our method achieves the highest design accuracy. Optimization methods [6, 8] cannot ensure a high design accuracy because of their limitations, e.g., being stuck at a local optimum (caused by non convexity) or divergence of the algorithms. Due to the inherent approximation errors, SL methods [10] suffer from a low design accuracy. RL methods [13, 16] excluding the key domain knowledge cannot reach the human-level design accuracy as ours. Due to such limitations, these methods show a weaker generalization ability than ours, either. Despite not excelling the design efficiency of SL methods with one-step prediction, once trained our method uses fewer steps to find the optimal device parameters for the same desired specifications, improving the design efficiency by average 1.5× compared to the prior RL methods [13, 16] and average 10× compared to optimization methods [6, 8]. In the application of FoM optimization, our method also achieves higher FoM value than prior RL methods and optimization methods. In summary, our RL method inspired by key domain knowledge of analog circuit design and human-like multiple tuning steps achieves the best balance between the design accuracy and efficiency as well as the best optimality.

5 CONCLUSION

We have shown a deep learning method for the automated design of analog circuits. The key property of our framework is to incorporate domain knowledge of practical analog circuit design (i.e., the underlying physical topology of a given circuit and the trade-offs between specifications) into the newly proposed combined GNN (GCN/GAT)-FC-based multimodal policy network. We show that such a method is superior to other methods without such considerations in designing various analog circuits with higher accuracy, efficiency, and optimality. We expect that our method will assist IC industry to accelerate the analog/RF chip design, with artificial agents that master massive circuitry optimization experiences via continuous learning.

Acknowledgment: Weidong Cao was an intern at MERL. This work is supported by MERL with additional support for Weidong Cao, Xuan Zhang in part by National Science Foundation grant no. CCF-1942900.

REFERENCES

[1] Minjie Wang et al. 2020. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. arXiv:cs.LG/1909.01315

[2] Q. Diduck et al. 2016. A 300MHz to 1200MHz Saturated Broadband Amplifier in GaN for 2W Applications. In *2016 Texas Symposium on Wireless and Microwave Circuits and Systems (WMCS).* 1–4.

[3] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[4] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations.*

[5] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin. 2020. An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2640–2653.

[6] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V. Fernández. 2009. Analog Circuit Optimization System Based on Hybrid Evolutionary Algorithms. *Integration* 42, 2 (2009), 137 – 148.

[7] Mingjie et al. Liu. 2021. Parasitic-Aware Analog Circuit Sizing with Graph Neural Networks and Bayesian Optimization. In *2021 Design, Automation & Test in Europe Conference Exhibition (DATE).* 1372–1377.

[8] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. 2018. Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design. In *Proceedings of the 35th International Conference on Machine Learning.* 3306–3314.

[9] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning.* 1928–1937.

[10] H. M.V. and B. P. Harish. 2020. Artificial Neural Network Model for Design Optimization of 2-stage Op-amp. In *2020 24th International Symposium on VLSI Design and Test (VDAT).* 1–5.

[11] Haoxing et al. Ren. 2020. ParaGraph: Layout Parasitics and Device Parameter Prediction Using Graph Neural Networks. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference.* 1–6.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:cs.LG/1707.06347

[13] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic. 2020. AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs. In *2020 Design, Automation & Test in Europe Conference Exhibition (DATE).* 490–495.

[14] Stuart Thomas. 2020. Gallium nitride gets wrapped up. *Nature Electronics* 3, 12 (01 Dec 2020), 729–729.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations.*

[16] H. Wang, K. Wang, J. Yang, and L. Shen et al. 2020. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC).* 1–6.

[17] Guo Zhang, Hao He, and Dina Katabi. 2019. Circuit-GNN: Graph Neural Networks for Distributed Circuit Design. In *Proceedings of the 36th International Conference on Machine Learning.* 7364–7373.