GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning

Hanrui Wang[1], Kuan Wang[1], Jiacheng Yang[1], Linxiao Shen[2], Nan Sun[2], Hae-Seung Lee[1], Song Han[1]

[1]*Massachusetts Institute of Technology*
[2]*University of Texas at Austin*

Abstract—**Automatic transistor sizing is a challenging problem in circuit design due to the large design space, complex performance trade-offs, and fast technology advancements. Although there have been plenty of work on transistor sizing targeting on one circuit, limited research has been done on transferring the knowledge from one circuit to another to reduce the re-design overhead. In this paper, we present GCN-RL Circuit Designer, leveraging reinforcement learning (RL) to transfer the knowledge between different technology nodes and topologies. Moreover, inspired by the simple fact that *circuit is a graph*, we learn on the circuit topology representation with graph convolutional neural networks (GCN). The GCN-RL agent extracts features of the topology graph whose vertices are transistors, edges are wires. Our learning-based optimization consistently achieves the highest Figures of Merit (FoM) on four different circuits compared with conventional black box optimization methods (Bayesian Optimization, Evolutionary Algorithms), random search and human expert designs. Experiments on transfer learning between five technology nodes and two circuit topologies demonstrate that RL with transfer learning can achieve much higher FoMs than methods without knowledge transfer. Our transferable optimization method makes transistor sizing and design porting more effective and efficient.**

Index Terms—**Circuit Design Automation, Transistor Sizing, Reinforcement Learning, Graph Neural Network, Transfer Learning**

I. INTRODUCTION

Mixed-signal integrated circuits are ubiquitous. While digital designs can be assisted by the mature VLSI CAD tools [1], analog designs still rely on experienced human experts. It is demanding to have learning-based design automation tools.

Nonetheless, manual design is not an easy task even for seasoned designers due to the long and complicated design pipeline. Designers first need to analyze the topology and derive equations for the performance metrics. Since analog circuits have highly nonlinear properties, a large number of simplifications and approximations are necessary during the topology analysis. Based on all the equations, the initial sizes are calculated. Then, a large number of simulations for parameters fine-tuning are performed to meet the performance specifications. The whole process can be highly *labor intensive* and *time consuming* because of the large design space, slow simulation tools, and sophisticated trade-offs between different performance metrics. Therefore, automatic transistor sizing is attracting more and more research interest in the recent years [2]–[5].

With transistors rapidly scaling down, *porting* existing designs from one technology node to another become a common practice. However, although much research efforts focus on transistor sizing for a single circuit, hardly any research has explored transferring the knowledge from one topology to another, or from one technology node to another. In this work, we present GCN-RL Circuit Designer (Figure 1) to conduct the knowledge transfer. Inspired by the transfer learning ability of *Reinforcement Learning* (RL), we first train a RL agent on one circuit and then apply the same agent to size new circuits or the same circuit in new technology nodes. In this way, we can reduce the simulation cost without designing from scratch.

Moreover, prior works such as Bayesian Optimization (BO) and Evolutionary Strategy (ES) treated transistor sizing as a black box optimization problem. Inspired by the simple fact that: *circuit is a*
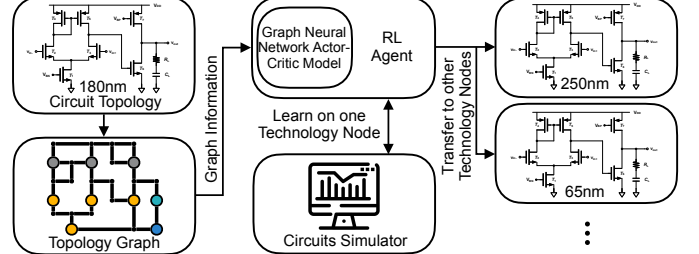

Fig. 1: Graph Convolutional Neural Networks based Reinforcement Learning for Automatic Transistor Sizing.

graph, we propose to open the black box and leverage the topology graph in the optimization loop. In order to make full use of the graph information, we propose to equip the RL agent with *Graph Convolutional Neural Network* (GCN) to process the connection relationship between components in circuits. With the proposed GCN-RL agent, we consistently achieved better performance than conventional methods such as BO and ES. Remarkably, the GCN-RL not only enables transfer knowledge between different technology nodes but also makes knowledge transfer between *different topologies* possible. Experiments demonstrate that GCN is necessary for knowledge transfer between topologies.

To our knowledge, we are the first to leverage GCN equipped RL to transfer the knowledge between different technology nodes and different topologies. The contributions of this work are as follows:

(1) **Leverage the Topology Graph Information** in the optimization loop (open-box optimization). We build a GCN based on the circuits topology graph to effectively open the optimization black box and embed the domain knowledge of circuits to improve the performance.

(2) **Reinforcement Learning as Optimization Algorithm**, which consistently achieves better performance than human expert [6], [7], random search, Evolution Strategy (ES) [8], Bayesian Optimization (BO) [9] and MACE [2].

(3) **Knowledge Transfer with GCN-RL** between different technology nodes and different circuit topologies to reduce the required number of simulations, thus shortening the design cycle.

II. RELATED WORK

Automatic Transistor Sizing. Automatic transistor sizing can be classified into knowledge-based and optimization-based methods. For knowledge-based such as TAGUS [10], circuits experts use their knowledge to design pre-defined plans and equations with which the transistor sizes are calculated. However, deriving a general design plan is highly time consuming, and it requires continuous maintenance to keep up with the latest technology. For optimization-based methods, they can be further categorized into model-based and simulation-based. Model-based methods such as [11], [12] model circuit performance via manual calculation or regression with sim-
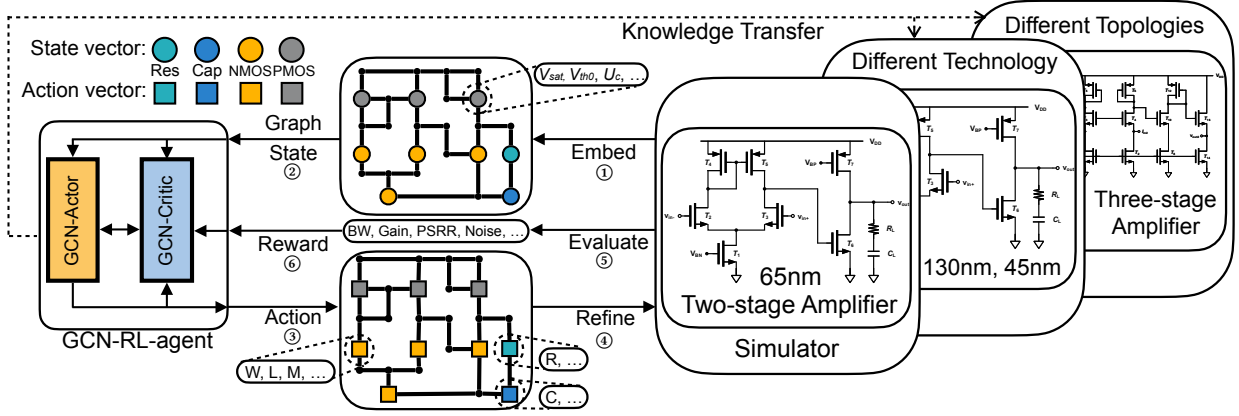
Fig. 2: GCN-RL Overview: Six steps in one optimization iteration: (1) Embed topology. (2) Generate graph and node states. (3) Generate actions (circuit parameters). (4) Refine circuit parameters. (5) Evaluate circuit parameters. (6) Update RL policy with the reward.

ulated data, and then optimize the model. The advantage is the easy-to-get global optimal. Nevertheless, building a model requires numerous simulations to improve the accuracy. For simulation-based ones, performance of circuits is evaluated with simulators (*e.g.* SPICE). The optimization algorithms such as BO [9], MACE [2], ES [8] consider the circuits as a black box and conduct optimization. Compared with ours, neither MACE nor ES leverage the topology graph information. In addition, BO and MACE have difficulties in transferring knowledge between circuits because their output space is fixed. ES cannot transfer because it keeps good samples in its population without summarizing the design knowledge.

Deep Reinforcement Learning. Recently, deep RL algorithms have been extensively applied to many problems such as game playing [13], robotics [14] and AutoML [15]. There are also environment libraries [16] using RL for system design. For different RL task domains, deep RL proves to be transferable [17]. In this work, we propose RL based transistor sizing, which makes it automated, transferable, and achieves better performance than other methods. Comparing to supervised learning, RL can continuously learn in the environment and adjust the policy.

Graph Neural Networks. Graph neural network (GNN) [18] adapts neural networks to process graph data. Several variants of GNN are proposed, including Graph Convolution Neural Networks (GCN) [19], Graph Attention Networks [20], etc. There are also accelerators [21], [22] focusing on GNN related workloads. In GCN-RL Circuit Designer, we refer to [19] to build GCN, leveraging topology graph information to benefit the optimization. [23] used GNN to replace an EM simulator for distributed circuits. By contrast, our work focuses on analog transistor sizing and exploits RL for knowledge transfer.

III. METHODOLOGY

A. Problem Definition

We handle transistor sizing problem where the topology of the analog circuits is *fixed*. The problem can be formulated as a bound-constrained optimization:

$$\underset{x \in \mathbb{D}^n}{\text{maximize}} \quad \text{FoM}(x) \qquad (1)$$

where x is the parameter vector, n is the number of parameters to search. $\mathbb{D}^n$ is the design space. Figure of Merits (FoM) is the objective we aim to optimize. We define it as the weighted sum of

the normalized performance metrics as shown in Equation 2.

$$\text{FoM} = \sum_{i=0}^{N} w_i \times \frac{\min(m_i, m_i^{\text{bound}}) - m_i^{\text{min}}}{m_i^{\text{max}} - m_i^{\text{min}}} \quad \text{if spec is satisfied} \quad (2)$$

where m_i is the measured performance metrics. m_i^{min} and m_i^{max} are pre-defined normalizing factors to normalize the performance metrics to guarantee their proper ranges. m_i^{bound} is the pre-defined upper bound for some performance aspects which do not need to be better after satisfying some requirements. w_i is the weight to adjust the importance of the ith performance metric. For some circuit baselines we use, there exists performance specification (spec) to meet, if the spec is not met, we assign a negative number as the FoM value.

B. Framework Overview

An overview of the proposed framework is shown in Figure 2. In each iteration, (1) Circuit environment embeds the topology into a graph whose vertices are components and edges are wires; (2) The environment generates a state vector for each transistor and passes the graph with the state vectors (refer to the graph on the top with circle nodes) to the RL agent; (3) The RL agent processes each vertex in the graph and generates an action vector for each node. Then the agent passes the graph with the node action vectors (refer to the graph with square vertices) to the environment; (4) The environment then denormalizes actions ([-1, 1] range) to parameters and refines them. We refine the transistor parameters to guarantee the transistor *matching*. We also round and truncate parameters according to minimum precision, lower and upper bounds of the technology node; (5) Simulate the circuit; (6) Compute an FoM value and feed to RL agent to update policy. We *do not need* the initial parameters as in the human design flow. The detailed RL agent will be discussed in Section III-D.

C. Reinforcement Learning Formulation

We apply the actor-critic RL agent in GCN-RL. The critic can be considered as a differentiable model for the circuit simulator. The actor looks for points with best performance according to the model.

State Space. The RL agent processes the circuit graph component by component. For a circuit with n components in topology graph G, the state $\mathbf{s_k}$ for the kth component is defined as $\mathbf{s_k} = (\mathbf{k}, \mathbf{t}, \mathbf{h})$, where $\mathbf{k}$ is the one-hot representation of the transistor index, $\mathbf{t}$ is the one-hot representation of component type and $\mathbf{h}$ is the selected model feature vector for the component which further distinguishes different component types. For the NMOS and PMOS, the model parameters
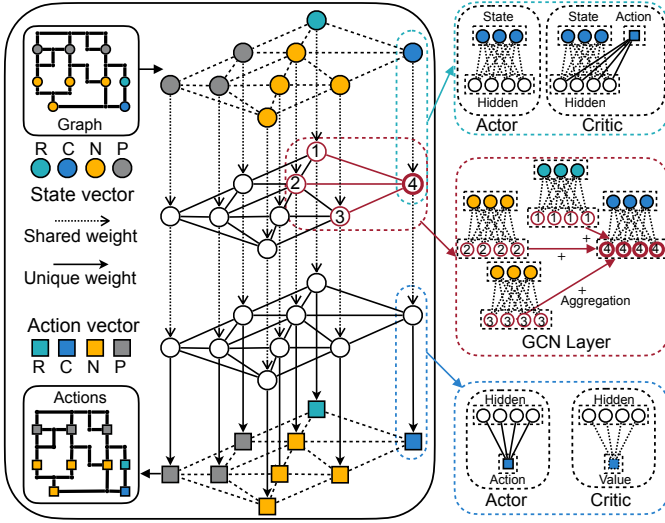
Fig. 3: Reinforcement learning agent with multi-layer GCN.

we use are V_{sat}, V_{th0}, V_{fb}, μ_0 and U_c. For the capacitor and resistor, we set the model parameters to zeros. For instance, for a circuit with ten components of four different kinds (NMOS, PMOS, R, C) and a five-dimensional model feature vector, the state vector for the third component (an NMOS transistor) is,

$$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, V_{\text{sat}}, V_{\text{th0}}, V_{\text{fb}}, \mu_0, U_c] \quad (3)$$

For each dimension in the observation vector $\mathbf{s_k}$, we normalize them by the mean and standard deviation across different components.

Action Space. The action vector varies for different types of components because the parameters needed to search are not the same. For the kth component, if it is NMOS or PMOS transistors, the action vector is formulated as $\mathbf{a_k^{MOS}} = (W, L, M)$, where W and L are the width and length of the transistor gate, M is the multiplexer; for resistors, the action vector is formulated as: $\mathbf{a_k^R} = (r)$, where r is the resistance value; for capacitors, the action vector is formulated as: $\mathbf{a_k^C} = (c)$, where c is the capacitance value.

We use a *continuous* action space to determine the transistor sizes even though we will round them to discrete values. The reason why we do not use a *discrete* action space is because that will lose the *relative order* information, also because the discrete action space is too large. For instance, for a typical operational amplifier with 20 transistors, each with three parameters, and each size with 1000 value options, the size of the discrete space is about 1000^{60}.

Reward. The reward is the FoM defined in Equation 2. It is a weighted sum of the normalized performance metrics. In our default setup, all the metrics are equally weighted. We also studied the effect of assigning different weights to different metrics in the experiments. Our method is *flexible* to accommodate different reward setups.

D. Enhancing RL Agent with Graph Convolutional Neural Network

To embed the graph adjacency information into the optimization loop, we leverage GCN [19] to process the topology graph in the RL agent. As shown in Figure 3, one GCN layer calculates each transistor's hidden representation by aggregating feature vectors from its neighbors. By stacking multiple layers, one node can receive information from farther and farther nodes. In our framework, we apply seven GCN layers to make sure the last layer have a *global receptive field* over the entire topology graph. The parameters for one component are primarily influenced by nearby components, but

---

Algorithm 1: Proposed GCN-RL Method.

Random Initialize critic network $Q(S, A \mid \theta^Q)$ and actor network $\mu(S \mid \theta^\mu)$ with critic weights θ^Q and actor weights θ^μ ;
Initialize replay buffer P ;
for *episode = 1, M* **do**
    Initialize random process $\mathcal{N}$; Receive observation state S;
    if *episode $< W$* **then**
        Warm-up: randomly sample an action $\widetilde{A}$;
    end
    else
        Select action $\widetilde{A} = \mu(S \mid \theta^\mu) + \mathcal{N}$ according to the current policy and exploration noise;
    end
    Denormalize and refine $\widetilde{A}$ with design constrains to get A;
    Simulate the A to get reward R;
    Store transition (S, A, R) in P;
    if *episode $> W$* **then**
        Sample a batch of $(\widehat{S}, \widehat{A}, \widehat{R})$ from P (batch size = N_s);
        Update the critic by minimizing the loss: $L = \frac{1}{N_s} \sum_{k=1}^{N_s} (\widehat{R}_k - B - Q(\widehat{S}_k, \widehat{A}_k \mid \theta^Q))^2$;
        Update the actor using the sampled policy gradient: $\nabla_{\theta^\mu} J \approx \frac{1}{N_s} \sum_{k=1}^{N_s} \nabla_a Q(S, A \mid \theta^Q)|_{\widehat{S}_k, \mu(\widehat{S}_k)} \nabla_{\theta^\mu} \mu(S \mid \theta^\mu)|_{\widehat{S}_k}$;
    end
end

---

are also influenced by farther components. The GCN layer can be formulated as:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}) H^{(l)} W^{(l)}, \quad (4)$$

Here, $\widetilde{A} = A + I_N$ is the adjacency matrix (A) of the topology graph G plus identity matrix (I_N). Adding the identity matrix is common in GCN networks [19]. $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ and W^l is a layer-specific trainable weight matrix, echoing with the shared weights in Figure 3. $\sigma(\cdot)$ is an activation function such as ReLU($\cdot$) [19]. $H^{(l)} \in \mathbb{R}^{n \times d}$ is the hidden features in the lth layer (n: number of nodes, d: feature dimension). $H^0 = S$, which are the input state vectors for actor.

The actor and critic models have slightly different architectures (Figure 3). The actor's first layer is a FC layer shared among all components. The critic's first layer is a shared FC layer with a *component-specific* encoder to encode different actions. The actor's last layer has a *component-specific* decoder to decode hidden activations to different actions, while the critic has a shared FC layer to compute the predicted reward value. We design those specific encoder/decoder layers because different components have *different* kinds of actions (parameters). The output of the last layer of the actor is a pre-refined parameter vector for each component ranging [-1, 1]. We denormalize and refine them to get the final parameters.

For RL agent training, we leverage DDPG [24], which is an off-policy actor-critic algorithm for continuous control problem. The details are illustrated in Algorithm 1. N_s denotes the number of sampled data batch in one episode, $S = \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_n}\}$ denotes states, $A = \{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}\}$ denotes actions. The baseline B is defined as an exponential moving average of all previous rewards in order to *reduce the variance* of the gradient estimation. M is the max search episodes and W is the warm-up episodes. $\mathcal{N}$ is a truncated norm noise with exponential decay.
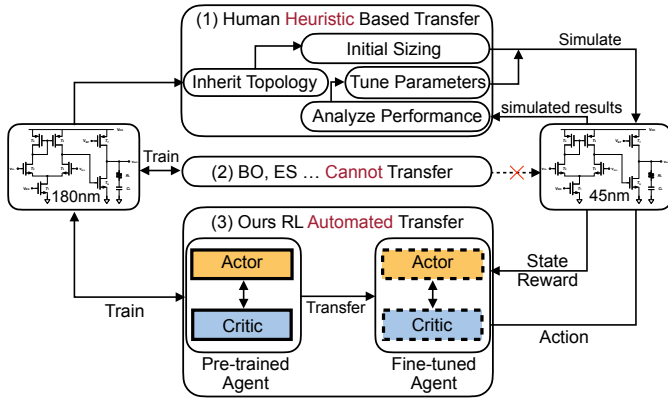
Fig. 4: Our RL agent can transfer design knowledge by inheriting the pretrained weights of the actor-critic model. Human need use heuristic based methods to re-design the parameters. BO and ES cannot transfer.

We implemented two types of RL agent to show the effectiveness of GCN: one is the proposed GCN-RL and the other is non-GCN RL (NG-RL), which does not operate the *aggregation* step in Figure 3, thus topology information is not used.

E. Knowledge Transfer

Transfer between Technology Nodes. The rapid transistor scaling down makes *porting* existing designs from one technology node to another a common practice. As in Figure 4 top, human designers first inherit the topology from one node and compute initial parameters, then iteratively tune the parameters, simulate and analyze performance. In contrast, our method can automate this process by training an RL agent on one technology node and then *directly applying* the trained agent to search the same circuit under different technology nodes by virtue of similar design principles among different technology. For instance, the agent can learn to change the gain by tuning the input pair transistors of an amplifier. This feature does not vary among technology nodes.

Transfer between Topologies. We can also transfer knowledge between different topologies if they share similar design principles, such as between a two-stage transimpedance amplifier and a three-stage transimpedance amplifier. This is enabled by GCN which can extract features from the circuit topologies. Concretely, we slightly modify the state vector mentioned in Section III-C. $\mathbf{k}$ is modified to a one-dimension index value instead of a one-hot index vector. In this way, the dimension of the state vector of each component remains the *same* among different topologies. We will show that without GCN, knowledge transfer between different topologies cannot be achieved.

IV. Experiments

A. Comparison between GCN-RL and others

To demonstrate the effectiveness of the proposed GCN-RL method, we applied GCN-RL to 4 real-world circuits (Figure 6): a two-stage transimpedance amplifier (Two-TIA), a two-stage voltage amplifier (Two-Volt), a three-stage transimpedance amplifier (Three-TIA) and a low-dropout regulator (LDO). The Two-Volt and LDO were designed in commercial 180nm TSMC technology, and simulated with Cadence Spectre. Two-TIA and Three-TIA were also designed in 180nm technology, simulated with Synopsys Hspice.

We compared FoMs of GCN-RL with human expert design, random search, non-GCN RL (NG-RL), Evolutionary Strategy (ES) [8], Bayesian Optimization (BO) [9], and MACE [2]. The human expert designs for Two-TIA and Three-TIA are strong baselines from [6], [7], which achieved the "Most Innovative Design" award [25]. The

	Two-TIA	Two-Volt	Three-TIA	LDO
Human [6], [7]	2.32	2.02	1.15	0.61
Random	2.46 ± 0.02	1.74 ± 0.06	0.74 ± 0.03	0.27 ± 0.03
ES [8]	2.66 ± 0.03	1.91 ± 0.02	1.30 ± 0.03	0.40 ± 0.07
BO [9]	2.48 ± 0.03	1.85 ± 0.19	1.24 ± 0.14	0.45 ± 0.05
MACE [2]	2.54 ± 0.01	1.70 ± 0.08	1.27 ± 0.04	0.58 ± 0.04
NG-RL	2.59 ± 0.06	1.98 ± 0.12	1.39 ± 0.01	0.71 ± 0.05
GCN-RL	**2.69 ± 0.03**	**2.23 ± 0.11**	**1.40 ± 0.01**	**0.79 ± 0.02**

TABLE I: FoM comparison between different algorithms. Our method consistently achieves the highest FoM values in four circuits baselines.

human expert designs for Two-Volt and LDO come from a seasoned designer with 5-year experience designing for 6 hours. MACE is a parallel BO method with multi-objective acquisition ensemble. For ES, BO and MACE, we used open-sourced frameworks to guarantee the *unbiased* implementations. For GCN-RL, NG-RL, ES and random search, we ran 10000 steps. In these methods, the circuit simulation time accounts for over 95% of the total runtime. For BO and MACE, it is *impossible* to run 10000 steps because the computation complexity is $\mathcal{O}(N^3)$, thus we ran them for the *same runtime* (around 5 hours) with GCN-RL for fair comparisons.

We ran each experiment three times to show significance. We computed the FoM values based on Equation 2. The $m_i^{\max}$ and $m_i^{\min}$ were obtained by random sampling 5000 designs, and choosing the max and min of each metric. We assigned $w_i = 1$ to one performance metric if larger is better, such as gain; and assigned $w_i = -1$ if smaller is better, such as power. The experimental results are shown in Table I. We plot the max FoM values among three runs for each method in Figure 5. GCN-RL consistently achieves the highest FoM. The *convergence speed* of GCN-RL is also faster than NG-RL, which benefits from the fact that GCN is better than pure FC on extracting the topology features, similar to CNN being better than pure FC on extracting image features.

Two-Stage Transimpedance Amplifier. Diode-connected input transistors are used to convert its drain current to a voltage at the drain of its output stage. Bandwidth, gain, power, noise and peaking are selected as metrics. The FoM results are shown in the 'Two-TIA' column of Table I. Performance metrics are show in Table II top part. All the models satisfy the spec. Since the FoM encourages *balanced* performance among metrics, the GCN-RL design forms a good balance among five metrics, resulting in the highest FoM. Notably, GBW of GCN-RL is also the highest.

We often need to trade-off different performance metrics. In order to demonstrate the *flexibility* of the GCN-RL with different design focuses, we assigned 10× larger weight for one metric then other metrics: for GCN-RL-1 to GCN-RL-5, 10× larger weight on each of BW, gain, power, noise and peaking. Since one spec is only suitable for one design requirement, in those five experiments, we *did not* limit the results with the spec to show its general effectiveness. Except for GCN-RL-4, all designs achieve highest performance on the single aspect we care about. For GCN-RL-4, it achieves the second best.

Two-Stage Voltage Amplifier. The amplifier is connected in a closed-loop configuration for PVT-stable voltage gain, which is set by the capacitor ratio. Miller compensation is used to stabilize the closed-loop amplifier. Bandwidth, common mode phase margin (CPM), differential mode phase margin (DPM), power, noise and open loop gain are selected as metrics. The FoM results are shown in the 'Two-Volt' column of Table I and details in Table III. The GCN-RL achieves highest CPM, DPM; second highest Gain and GBW.

Three-Stage Transimpedance Amplifier. A common-mode input pairs is used to convert its differential source current to a voltage at
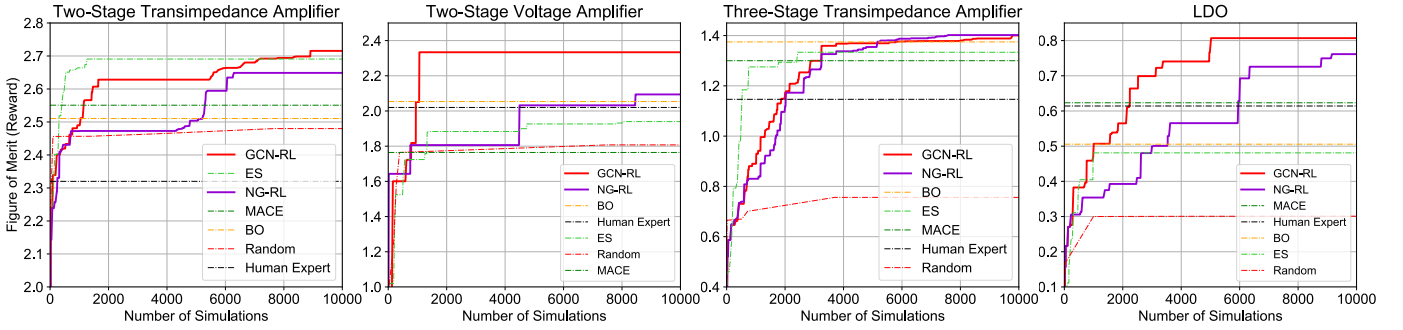
Fig. 5: Learning curves of different algorithms. The proposed GCN-RL achieves the highest FoM in four baselines.
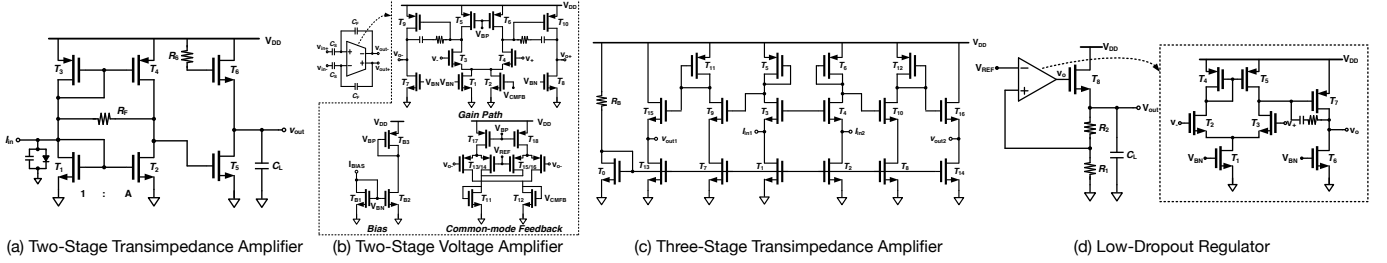

Fig. 6: Topologies for four circuit baselines we optimize.

	BW (GHz)	Gain ($\times 10^2\Omega$)	Power (mW)	Noise (pA/$\sqrt{Hz}$)	Peaking (dB)	GBW (THz$\times\Omega$)	FoM
Spec [25]	max	>7.58	<18.0	<19.3	<1.00	max	-
Human [6]	5.95	7.68	8.11	18.6	0.93	4.57	2.32
Random	1.41	35.9	2.30	9.28	0.16	5.05	2.48
ES [8]	1.18	104	4.25	3.77	0	12.3	2.69
BO [9]	0.16	123	2.76	**1.68**	0	1.99	2.51
MACE [2]	0.97	83.1	2.74	7.36	0	8.07	2.55
NG-RL	0.75	156	2.31	3.85	0.068	11.7	2.65
GCN-RL	1.03	167	3.44	3.72	0.0003	**17.2**	**2.72**
GCN-RL-1	**13.6**	0.09	41.2	295	0.036	0.13	-
GCN-RL-2	0.20	**266**	2.58	5.73	0	5.18	-
GCN-RL-3	0.42	249	**0.58**	4.78	0	10.3	-
GCN-RL-4	0.86	124	3.67	**3.64**	1.0	10.7	-
GCN-RL-5	0.57	89.0	0.94	11.7	**0**	5.10	-

TABLE II: Performance metrics comparison of two-stage transimpedance amplifier. First 8 rows: GCN-RL forms a good balance between different metrics and achieves highest GBW and FoM. Last 5 rows: GCN-RL is *flexible* for different FoMs. It can achieve best single-metric performance if the FoM value is biased to one performance metric.

	BW (MHz)	CPM (°)	DPM (°)	Power ($\times 10^{-4}$W)	Noise (nA/$\sqrt{Hz}$)	Gain ($\times 10^3$)	GBW (THz)	FoM
Human	**242**	180	83.9	2.94	47.1	3.94	0.95	2.02
Random	187	180	2.51	7.85	23.8	8.77	1.64	1.80
ES [8]	27.3	180	4.43	1.46	74.2	**50.0**	1.37	1.93
BO [9]	151	166	2.77	**1.45**	46.3	25.0	**3.77**	2.05
MACE [2]	99.4	180	4.44	8.45	**16.1**	8.93	0.89	1.76
NG-RL	96.2	180	23.7	4.02	19.2	17.2	1.66	2.09
GCN-RL	84.7	**180**	**96.3**	2.56	58.7	29.4	2.57	**2.33**

TABLE III: Performance metrics comparison of two-stage voltage amplifier. GCN-RL forms a good balance between metrics and achieves best CPM and DPM and second highest Gain and GBW.

the drain of its output stage. Three-stage configuration is used to boost the I-V conversion gain. Bandwidth, gain and power are selected as metrics. The FoM results are shown in the 'Three-TIA' column of Table I. GCN-RL achieves lowest power and second highest GBW.

Low-Dropout Regulator. LDO regulates the output voltage and minimizes the effect of either supply change or load change. Settling time for load increase (T_{L+}), settling time for load decrease (T_{L-}),

	250nm	130nm	65nm	45nm
Two-TIA No Transfer	2.36 ± 0.05	2.43 ± 0.03	2.36 ± 0.09	2.36 ± 0.06
Two-TIA Transfer from 180nm	**2.55 ± 0.01**	**2.56 ± 0.02**	**2.52 ± 0.04**	**2.51 ± 0.04**
Three-TIA No Transfer	0.69 ± 0.25	0.65 ± 0.14	0.55 ± 0.03	0.53 ± 0.05
Three-TIA Transfer from 180nm	**1.27 ± 0.02**	**1.29 ± 0.05**	**1.20 ± 0.09**	**1.06 ± 0.07**

TABLE IV: Knowledge transfer from 180nm to other technology nodes on Two-TIA and Three-TIA. After the *same* steps, performance with transfer is *consistently better* than without transfer.

Load Regulation (LR), settling time for supply increase (T_{V+}), settling time for supply decrease (T_{V+}), Power Supply Rejection Ratio (PSRR) and power are the metrics. The FoM results are shown in the 'LDO' column of Table I. The proposed GCN-RL method achieves the shortest T_{L+} and T_{L-}, second largest LR and PSRR.

B. Knowledge Transfer Between Technology Nodes

We also conducted experiments on knowledge transfer between technology nodes, which is a great feature enabled by reinforcement learning. The agents are trained on 180nm and transferred to both larger node – 250nm and smaller nodes – 130, 65 and 45nm to verify its extensive effectiveness.

Transfer on Two-Stage Transimpedance Amplifier. We directly applied the RL agent trained in Section IV-A on searching parameters for Two-TIA on other technology nodes. We compared the transfer learning results with no transfer after limited number of training steps (300 in total: 100 warm-up, 200 exploration). As in Table IV top part, transfer learning results are much better than without transfer.

Transfer on Three-Stage Transimpedance Amplifier. We also applied the RL agent in Section IV-A to search the parameters on other nodes. We show the results in Table IV bottom part. We also plot the max FoM value learning curves in Figure 7. We use the same random seeds for two methods so they have the same FoMs in the warm-up stage. After exploration, transfer learning results are consistently better than no transfer after the *same* steps.
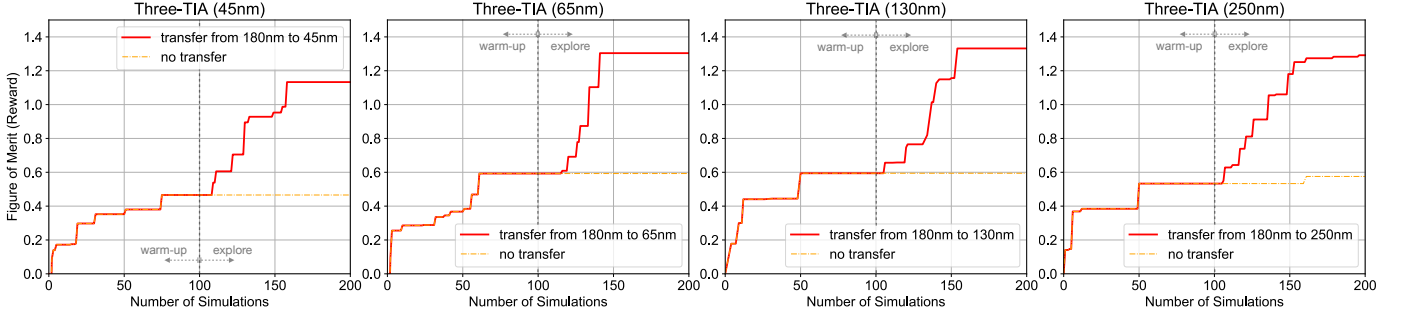
Fig. 7: Knowledge transfer between technology nodes on Three-TIA. We transfer the design learned on 180nm to 45nm, 65nm, 130nm, and 250nm. After 100 warm-up steps, the FoM with knowledge transfer increases rapidly and finally converges at a higher level than without knowledge transfer.
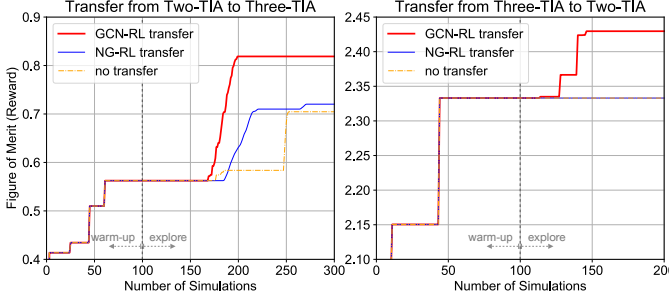

Fig. 8: Knowledge transfer between Two-TIA and Three-TIA. After 100 warm-up steps, the FoM of GCN-RL increases rapidly and finally converges at a higher value than NG-RL transfer and no transfer.

	Two-TIA → Three-TIA	Three-TIA → Two-TIA
No Transfer	0.63 ± 0.07	2.37 ± 0.01
NG-RL Transfer	0.62 ± 0.09	2.40 ± 0.07
GCN-RL Transfer	**0.78 ± 0.12**	**2.45 ± 0.02**

TABLE V: Knowledge transfer between Two-TIA and Three-TIA. GCN-RL consistently achieves higher FoM than NG-RL. NG-RL transfer has similar FoM as no transfer.

C. Knowledge Transfer Between Topologies

We can also transfer the knowledge learned from one topology to another. We chose Two-TIA and Three-TIA as they are both transimpedance amplifier, thus sharing some common knowledge. We first trained both GCN-RL and NG-RL agents on Two-TIA for 10000 steps. Then, we directly applied the agents to the Three-TIA and trained for only 300 steps. We also conducted the reverse experiments, learning from Three-TIA and transferring to Two-TIA. We compared (1) transfer learning with GCN-RL, (2) transfer learning without GCN (NG-RL), (3) no transfer with GCN-RL, as in Table V, and learning curves in Figure 8. GCN-RL consistently achieves higher FoMs than NG-RL. Without GCN, the FoM of NG-RL is barely at the *same* level as without transfer, which shows that it is *critical* to use GCN to extract the knowledge from the graph, and the graph information extracted by GCN can help improve knowledge transfer performance.

V. CONCLUSION

We present GCN-RL Circuit Designer, a transferable automatic transistor sizing method with GCN and RL. Benefiting from transferability of RL, we can transfer the knowledge between different technology nodes and even different topologies, which is difficult with other methods. We also use GCN to involve topology information into the RL agent. Extensive experiments demonstrate that our method can achieve better FoM than others, with knowledge transfer ability. Therefore, GCN-RL Circuit Designer enables more effective and efficient transistor sizing and design porting.

REFERENCES

[1] I. A. M. Elfadel *et al.*, *Machine Learning in VLSI Computer-Aided Design*. Springer, 2018.

[2] W. Lyu *et al.*, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *ICML*, 2018.

[3] T. Liao *et al.*, "Parasitic-aware gp-based many-objective sizing methodology for analog and rf integrated circuits," in *ASP-DAC*, 2017.

[4] B. Liu *et al.*, "An enhanced moea/d-de and its application to multiobjective analog cell sizing," in *CEC*, 2010.

[5] H. Wang *et al.*, "Learning to design circuits," in *NeurIPS Machine Learning for Systems Workshop*, 2018.

[6] D. Bankman. (2013) Stanford EE214B Advanced Analog IC Design Contest "Most Innovative Design Award". (Available upon request).

[7] D. Bankman *et al.* (2013) Stanford EE214A Fundamentals of Analog IC Design, Design Project Report. (Available upon request).

[8] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[9] J. Snoek *et al.*, "Practical bayesian optimization of machine learning algorithms," in *NIPS*, 2012.

[10] N. Horta, "Analogue and mixed-signal systems topologies exploration using symbolic methods," *AICSP*, 2002.

[11] E. Deniz *et al.*, "Hierarchical performance estimation of analog blocks using pareto fronts," in *6th Conference on Ph.D. Research in Microelectronics Electronics*, 2010.

[12] R. Castro-López *et al.*, "Multimode pareto fronts for design of reconfigurable analogue circuits," *Electronics Letters*, 2009.

[13] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, 2017.

[14] S. Levine *et al.*, "End-to-end training of deep visuomotor policies," *JMLR*, 2016.

[15] Y. He *et al.*, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices," in *ECCV*, 2018.

[16] H. Mao *et al.*, "Park: An open platform for learning-augmented computer systems," in *NeurIPS*, 2019.

[17] M. E. Taylor *et al.*, "Transfer learning for reinforcement learning domains: A survey," *JMLR*, 2009.

[18] F. Scarselli *et al.*, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[19] T. N. Kipf *et al.*, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[20] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[21] M. Yan *et al.*, "Hygcn: A gcn accelerator with hybrid architecture," in *HPCA*. IEEE, 2020.

[22] Z. Zhang *et al.*, "Sparch: Efficient architecture for sparse matrix multiplication," in *HPCA*. IEEE, 2020.

[23] G. Zhang *et al.*, "Circuit-GNN: Graph neural networks for distributed circuit design," in *ICML*, 2019.

[24] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning." in *ICLR*, 2016.

[25] B. Murmann. (2013) Stanford ee214b advanced analog ic design contest. [Online]. Available: https://web.stanford.edu/class/ee214b/contest