TPC-GAN: Batch Topology Synthesis for Performance-Compliant Operational Amplifiers Using Generative Adversarial Networks

YuHao Leng
School of Integrated Circuits Science and Engineering
Beihang University
Beijing, China
yhleng@buaa.edu.cn

Jinglin Han
School of Integrated Circuits Science and Engineering
Beihang University
Beijing, China
jinglinhan@buaa.edu.cn

Yining Wang
Corelink Technology(Qingdao) Co.,Ltd.
Beijing, China
wyn20000407@buaa.edu.cn

Peng Wang
School of Integrated Circuits Science and Engineering
Beihang University
Beijing, China
wang.peng@buaa.edu.cn

***Abstract*—Operational amplifier is one of the most important analog basic blocks. Existing automated synthesis strategies for operational amplifiers solely focus on the optimization of single topology, making them unsuitable for scenarios requiring batch synthesis, such as dataset augmentation. In this paper, we introduce TPC-GAN, a generative model for batch topology synthesis of operational amplifiers in accordance with performance specifications. To be specific, it incorporates a reward network of circuit performance into the adversarial generative networks (GANs). This enables direct synthesis of novel and feasible circuit topology meeting performance specifications. Experimental results demonstrate that our proposed method can achieve a validity rate of 98% in circuit generation, among which 99.7% are novel relative to the training dataset. With the introduction of a reward network, a significant portion (82.8%) of the generated circuits satisfy performance specifications, which is a substantial improvement than those without. Transistor-level experimental results further demonstrate the practicality and competitiveness of our generated circuits with nearly 3x improvement over manual designs.**

***Index Terms*—Generative model, Topology generation, Analog circuit, Operational amplifier**

I. Introduction

Operational amplifier (opamp) is ubiquitous in many analog circuits but its design is often complex and time-consuming. In standard process, circuit topology design is crucial and directly affects subsequent optimization of device parameters and the final performance [1]. However, due to the ambiguous relationship between circuit topology and performance, automatic generation of valid and novel topology design is extremely challenging. This not only limits design efficiency but also restricts the size of data available for emerging Artificial Intelligence (AI) technologies, such as Large Language Models (LLMs), in analog circuits [2]–[4].

This work is supported by the National Key Research and Development Program of China (grant number 2022YFB4400402).

Over the years, two types of approaches have been developed to generate circuit topology. One focuses on transistor-level topology generation and relies on circuit libraries and design rules. Early works invoked dozens of knowledge-based rules to ensure the legitimacy of circuits [5], [6]. As circuit complexity increases, researchers started to employ random search algorithms to cope with the rising number of circuit rules [7]–[10]. More recently, Zhao et al. employ reinforcement learning to evaluate the selection of blocks [11]. However, circuit libraries are generally expensive to build and a large number of invalid redundant circuits are often generated among transistor-level methods. As a result, they would incur a substantial waste of computational resources.

The other approach focuses on behavioral-level topology generation and often employs machine learning models. For example, Lu et al. constructed the Laplacian matrix of circuit topology and utilized its eigenvectors for graph embedding [12]. A customized variational graph auto-encoder (D-VAE) [13] was employed in Lu's subsequent work, which enabled the gradient-based optimization method [14]. Following the D-VAE approach, Chen et al. proposed a reinforcement learning method [15] which was further improved with a multi-agent strategy for co-optimization of both circuit topology and device parameters [16]. In more recent works, Dong et al. proposed CktGNN, a two-level graph neural network to learn the latent representation of circuit topology [17], and Han et al. addressed the accuracy issue of topology reconstruction using an invertible graph generative model [18].

Although existing behavioral-level methods can effectively reduce faulty circuits, three issues remain unsolved. First, current studies focus on optimizing a single topology, making them unsuitable for large-scale generation of performance-compliant circuits. To be specific, they are insufficient for data generation needed for AI training. Second, methods that optimize topology in latent space are heavily dependent on the accuracy of graph encoders and decoders. However, according to the experiment in [17], even CktGNN itself achieves a reconstruction accuracy of less than 50%. In other words, their reconstruction accuracy is not yet satisfactory. Lastly, surrogate models of circuit performance are often expensive to construct in latent space and may lead to significant computational overhead.

To address the aforementioned challenges, we propose TPC-GAN, a novel topology synthesis method for three-stage operational amplifier with the following novelty:

- *GAN-based circuit topology synthesis.*
  To the best of the authors' knowledge, this is the first generative adversarial networks (GAN)-based approach for active component circuits.
- *Improved accuracy with directed acyclic graph (DAG).*
  Instead of graph encoding and decoding, TPC-GAN directly processes and generates the adjacency tensor and node feature matrix of DAGs, providing fast and accurate representation of behavioral-level circuit topology.
- *Batch topology synthesis with performance compliance.*
  The principle of GAN ensures the capability for batch generation. Performance specification is incorporated as a reward network to ensure the validity of generated circuits.
- *Reduced computational cost via offline surrogate models.*
  By using offline surrogate models for a binary classification of circuit performance, circuit performance may be determined without expensive online sizing.

The remainder of this paper is organized as follows. Section II provides the general background with problem formulation and a short review of the generative adversarial networks (GANs). Section III introduces our GAN-based operational amplifier generation method and its experimental results are presented and discussed in Section IV. Finally we conclude this paper in section V.

II. BACKGROUND

A. Problem Formulation

The synthesis of opamps can be divided into two steps. The first is to generate a circuit topology that meets performance specifications, whilst the second is performance verification through device sizes tuning, also known as sizing.

Let t denote the topology of an opamp, belonging to a topology space $t \in \mathcal{T}$. Each topology t can be represented as a directed acyclic graph (DAG) with a set of nodes and edges. Each node of the opamp topology corresponds to a node n_i, i = 1, 2, ...M in the DAG, where M is the total number of nodes. There are E different edges between two nodes, representing different circuit devices and directions. For an opamp with N types of nodes and M number of nodes, its topology can be summarized with a node feature matrix, $\boldsymbol{X} \in \mathbb{R}^{M\times N}$, and an adjacency tensor, $\boldsymbol{A} \in \mathbb{R}^{M\times M\times E}$. Here, $\boldsymbol{A}_{ij} \in \mathbb{R}^{E}$ is a one-hot vector indicating the type of the edge between the node n_i and n_j.

For a given topology, we define the device sizing space that depends on the topology space as $S(\mathcal{T})$. Let $f(\cdot)$ represents the difference between actual circuit response and given specifications across various metrics, including but not limited to gain, phase margin (PM), and gain bandwidth product (GBW). Now sizing can be described as an optimization problem:

$$\min_{\mathbf{s}\in S(\mathcal{T})} \quad f(\mathbf{s}:t), \qquad \text{s.t.} \quad c_s(\mathbf{s}:t) \leq 0, \tag{1}$$

where $\mathbf{s}:t$ is the device sizes in the given topology and $c_s(\mathbf{s}:t)$ represents constraints on devices, e.g. the range of device sizes.

B. Generative Adversarial Networks

Generative adversarial network (GAN) is a type of implicit generative model with proven potential in graph generation [19]. It typically consists of two components: a generative model called *generator* (G) and a discriminative model called *discriminator* (D). The former is to learn the mapping from a prior distribution p_{z} and aims to approximate the true data distribution p_{d} as closely as possible. The latter learns to distinguish whether samples are from the true data or the generator. Both models are deployed as customized neural networks and trained in an adversarial manner. To be specific, the training of a GAN can be described as a minimax problem:

$$\min_G \max_D \mathbb{E}_x\left[\log D(x)\right] + \mathbb{E}_z\left[\log(1-D(G(z)))\right], \tag{2}$$

where $\mathbb{E}[\cdot]$ represents ensemble average. Samples from the prior distribution and true data are denoted as $\boldsymbol{z} \sim p_{\mathrm{z}}$ and $\boldsymbol{x} \sim p_{\mathrm{d}}$, respectively.

In practice, GAN training may be unstable and prone to unexpected behaviors such as mode collapse. To improve its training stability, WGAN is often employed [20] and utilizes Kantorovich-Rubinstein duality to express the distance between two distributions p_1 and p_2, also known as the *Earth-Mover* (EM) distance:

$$EM = \frac{1}{K}\sup_{\|f\|_L\leq K}\left(\mathbb{E}_{x\sim p_1(x)}[g(x)] - \mathbb{E}_{x\sim p_2(x)}[g(x)]\right), \tag{3}$$

Here, the supremum is over all the K-Lipschitz functions $g(\cdot)$.

In addition, one may further improve stability by incorporating gradient penalty as an alternative soft constraint on the 1-Lipschitz continuity [21]. It leads to the popular WGAN-GP, in which the final loss function of generator remains same as in WGAN but the loss function of discriminator becomes:

$$Loss = -D(x) + D(G(z)) + \alpha\left(\|\nabla_{\hat{x}}D(\hat{x})\| - 1\right)^2. \tag{4}$$

The first two terms represent the original discriminator loss. The third term is the gradient penalty with the hyperparameter α in our experiments, and variable $\hat{\boldsymbol{x}}$, which is sampled linear combination between real data $\boldsymbol{x}$ and generated data $G(\boldsymbol{z})$.

III. TPC-GAN FRAMEWORK

Fig. 1 illustrates the general workflow of our TPC-GAN framework. It consists of three major parts: a generator G, a discriminator D and a reward network R. From left of Fig. 1, the generator samples from the prior distribution and generates outputs, $\boldsymbol{z} \sim p_{\mathrm{z}}$. Subsequently, Gumbel-Softmax [22] is employed to perform categorical sampling to generate the adjacency tensor $\boldsymbol{A}$ and node feature matrix $\boldsymbol{X}$. Their combination would yield the preliminary circuit topology, which is then evaluated by a discriminator and a reward network, separately. The former distinguishes between samples from the dataset and the generator, while the latter determines whether the topology meets performance specifications via reinforcement learning. Both discriminator and the reward network incorporate Relational-GCN (R-GCN) [23] to facilitate learning of different types of circuit connections. The reward network is pre-trained by the true topology dataset while the discriminator is trained using both dataset and the generated topology. A linear combination of values from original WGAN loss and the reward network is employed to train the generator. Ultimately, the fully trained generator has the capability to produce performance-compliant circuits topology in batch. Those preliminary circuit topologies are later sent for sizing.

A. Graph Representation of Circuit Topology

Following the topology mapping method in [14], each stage of opamp is modeled by an ideal voltage controlled current source (VCCS) with a pair of parasitic capacitor C and resistor R connected to the ground. Without loss of generality, there are five fixed nodes in a three-stage opamp: the input node V_{in}, the node between the first and second stages (node1), the node between the second and third stages (node2), the output node V_{out} and the ground node. Those five nodes have a total of $C_5^2 = 10$ tunable edges between them. For opamp compensation schemes, the most common available devices are resistors R, capacitors C, forward transconductance , backward transconductance, and their serial and parallel combinations. Based on these devices, there are 24 possible connection in total. Moreover, the absence of a connection between two nodes should also be considered. Overall, the number of edge types is $E = 25$, the number of nodes is $M = 5$ and the number of node types is $N = 1$.

B. Generator

In TPC-GAN framework, generator G takes samples ($\boldsymbol{z}$) from a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(0,1)$. Since the size of circuit topology is often fixed and relatively small, we implement the generator with multi-layer perception (MLP) [24]. In order to eliminate errors introduced by encoding and decoding, we represent each node and edge type with probabilities of categorical over types. Outputs of the generator is continuous and processed via categorical sampling. This enables direct generation of the adjacency tensor and node feature matrix without encoding and decoding. To be specific, TPC-GAN employs a straight-through gradient based on categorical reparameterization with the Gumbel-Softmax. By doing so, it ensures proper gradient flow and enables automatic selection of the most appropriate edge types. Unless specified otherwise, we denote $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ as the continuous adjacency tensor and node feature matrix, outputs from the MLP, while their one-hot counterparts, $\boldsymbol{A}$ and $\boldsymbol{X}$, are from Gumbel-Softmax processing.

The generator aims to generate circuit topology closely resembling the true dataset in terms of its graph properties, and meets the evaluation of the reward network at the same time. Henceforth, its loss function (L_G) is designed as follows:

$$L_G = \beta\, D(G(\boldsymbol{z})) + (1-\beta)\, R(\boldsymbol{z}). \tag{5}$$

where β is a model hyperparameter, $R(\boldsymbol{z})$ is the output computed by the reward network.. The generator is trained to learn design expertise from the dataset with the help of discriminator and reward network. Upon completing thorough adversarial training, the generator can rapidly and efficiently synthesize performance-compliant topology in batch without evaluation.

C. Discriminator

The discriminator (D) takes the adjacency tensor ($\boldsymbol{A}$) and node feature matrix ($\boldsymbol{X}$) as inputs and aims to distinguish true data from the generated topology. It must process information from 25 different edge types of the three-stage opamp and correctly guide the generator to produce analogous data. To tackle this issue, we deploy a series of graph convolution layers based on Relational-GCN (R-GCN), a convolution network that has demonstrated potential in handling multiple edge types. R-GCN assigns weight to each edge type and conducts convolution of the node signals ($\boldsymbol{X}$) at each layer:

$$h_i^{'(l+1)} = f_s^{(l)}\left(h_i^{(l)}, x_i\right) + \sum_{j=1}^{M}\sum_{m=1}^{E} \frac{\mathbf{A}_{ijm}}{|N_i|} f_m^{(l)}\left(h_j^{(l)}, x_j\right) \tag{6a}$$

$$h_i^{(l+1)} = \tanh\left[h_i^{'(l+1)}\right]. \tag{6b}$$

Here $h_i^{(l)}$ denotes the state of i-th node at the l-th layer, $f_s^{(l)}$ and $f_m^{(l)}$ are linear transformation functions that affine the self-connections at each layer and the connections with different edges from other nodes. $|N_i|$ denotes the number of neighbors of the i-th node.

Outputs of graph convolutions ($h_i^{(L)}$) are aggregated into a graph representation vector:

$$h_G^{'} = \sum_{i=1}^{N} \text{sigmoid}\left[f_j\left(h_i^{(L)}, x_i\right)\right] \odot \tanh\left[f_k\left(h_i^{(L)}, x_i\right)\right] \tag{7a}$$

$$h_G = \tanh\left(h_G^{'}\right) \tag{7b}$$

where f_j and f_k are MLPs with a linear output layer, $\odot$ denotes element-wise multiplication. The final vector representation of the circuit graph (h_G) would be further processed by another MLP to produce a scalar form. We employ the WGAN loss function as the discriminator loss L_D (4).

D. Reward Network

Reward network (R) determines whether a generated topology meets performance specifications. Instead of conducting complete sizing for each topology, the reward network is pre-trained and incorporates a binary classification model with identical structure as the discriminator. By doing so, we also avoid the complex and often elusive task of constructing an accurate surrogate model between circuit graph and its performance, and in turn, improve training efficiency and reduce computational costs. Although the reward network has identical structure as that the discriminator, its aggregated output is further processed through an activation layer to a normalized range of $(0, 1)$. Its

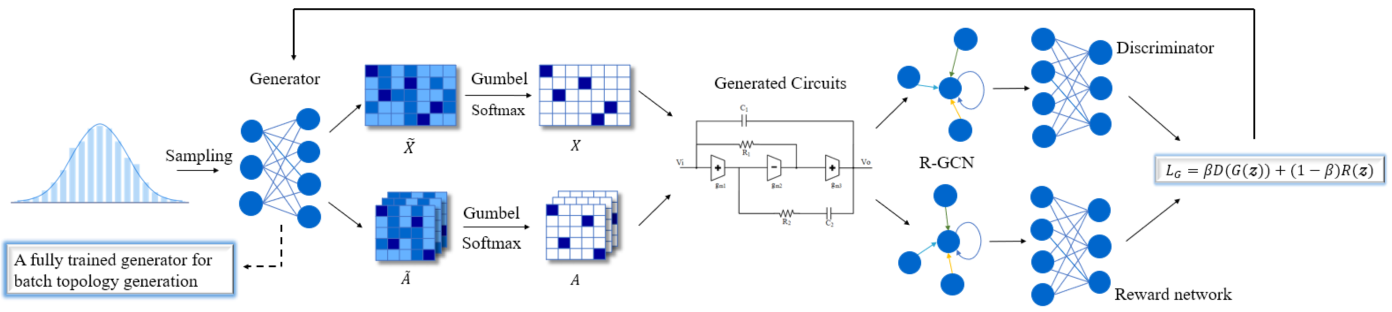


Fig. 1. General workflow of TPC-GAN framework.

loss function L_R is defined as the following cross-entropy loss, which is a popular choice in binary classification problems:

$$L_R = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y})), \tag{8}$$

where y denotes the label of true dataset, and $\hat{y}$ is the predicted value of the reward network.

E. Fine Tuning of Device Parameters

Upon completing the training of TPC-GAN, it is essential to evaluate the performance of the fully trained generator. To accomplish this task, we conduct device sizing on the circuits generated by the generator to assess the actual performance of each circuit. Considering n types of performance metrics, the objective of the sizing problem (1) can be defined as follows:

$$f(\mathbf{s}:t) = -\sum_{i=1}^{n}\lambda_i e_i(\mathbf{s}:t). \tag{9}$$

Here $e_i(\cdot)$ represents the difference between actual and target performance. Each is assigned with a corresponding weight λ_i. When actual performance exceeds performance specifications, its e_i value is set to zero. TPC-GAN solves the sizing problem (1) with a combined Levenberg-Marquardt (LM) and simulated annealing [25]. As outlined in Algorithm 1, we omit t from $s:t$ since the topology is fixed during the sizing step. Without loss of generality, one may employ other optimization algorithms for specialized circuits in the TPC-GAN framework.

IV. EXPERIMENTS AND DISCUSSION

We evaluate the proposed framework via three-stage opamp and compare with some baseline methods. Using the same training dataset as [18], power supply voltage V_{source}, load capacitor C_L, and load resistance R_L are set to 1.8V, 10nF, and 0.15MΩ, respectively.

Our study is conducted on a Linux server with an AMD EPYC 7F52 CPU, RTX 4080 GPU, and 512 GB of memory. The generator samples a 32-dimensional vector from a standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(0,1)$ and processes it further through a three-layer MLP of [128,256,512] hidden units with hyperbolic tangent function (tanh) as activation functions. The discriminator uses a two-layer R-GCN with [128,64] hidden units to process input graph and produces a 128-dimensional graph representation. This result is then processed through a three-layer MLP of [128,64,1] hidden units with tanh$(\cdot)$ as

Algorithm 1 Combined LM and simulated annealing algorithm

```
Require: Initial device sizes $s_0$, total iteration number of LM $N_{\text{LM}}$, total iteration number of simulated annealing $N_{\text{sa}}$, total iterations $N$, initial radius $r$, expansion factor $\alpha_{\text{exp}}$, contraction factor $\alpha_{\text{con}}$, initial temperature $T_0$, cooling factor $\gamma$.
Ensure: Solution of the device sizes $s^*$
 1: Initialize $s_0$, $T_0$, $r$
 2: for $i = 1, 2, \ldots, N$ do
 3:     $s_i \leftarrow$ Results of LM algorithm after $N_{\text{LM}}$ iterations
 4:     for $j = 1, 2, \ldots, N_{\text{sa}}$ do
 5:         Sample random points $s_x$, $s_y$, $s_z$ on $(s_i, r)$, $(s_i, r\cdot\alpha_{\text{exp}})$ and $(s_i, r\cdot\alpha_{\text{con}})$, respectively.
 6:         if $\max\{f(s_x), f(s_y), f(s_z)\} > q(s_i)$ then
 7:             $s_i \leftarrow \arg\max_{s_x,s_y,s_z}\{f(s_x), f(s_y), f(s_z)\},$
 8:         else
 9:             Sample $d$ from the uniform distribution on (0,1)
10:             if $d < \exp\left(-\frac{f(s_i)-\max\{f(s_x),f(s_y),f(s_z)\}}{T_0}\right)$ then
11:                 $s_i \leftarrow \arg\max_{s_x,s_y,s_z}\{f(s_x), f(s_y), f(s_z)\},$
12:     $T_0 \leftarrow T_0\cdot\gamma$ , $r \leftarrow r\cdot 0.5$.
13: return $s^*$   % the last result of the iteration
```

activation functions. The hyperparameter α in (4) is set to 10 in our expriment. The reward network shares the same architecture as the discriminator, with an added sigmoid activation layer for output processing. Using Adam optimizer with a learning rate of 10^{-5}, the reward network and entire GAN model are both trained for 150 epochs with a batch size of 256.

A. Pre-training of the Reward Network

To optimize the training of the reward network, we establish performance thresholds to balance number of topologies in the dataset between those meet the performance criteria and those not. They are set as:

$$\text{GAIN} > 40\text{dB}, \quad \text{PM} > 30^\circ, \quad \text{GBW} > 0.7\text{MHz}. \tag{10}$$

In the dataset, we label topology design that meets the performance thresholds as 1 and the others as 0. As a result, there are $4,584$ designs of label 1 and $5,416$ of 0. A total of $1,000$ samples are set for validation, among which 500 are labeled as 1 and 500 as 0. The remaining $9,000$ samples from the dataset are used for training.

Fig. 2 shows the convergence of the reward network via its loss and accuracy rate against the number of epoch. To reduce the impacts of randomness in training, those are averaged results over ten runs. Prediction variance are included as shading around the plots. It shrinks to tiny values as the number of epoch increases and stabilizes around 60 epochs. Meanwhile, prediction accuracy also elevates in tandem with rising epochs and settles after 60 epochs. Both demonstrates the stability of the training and convergence processes. We note here that dataset quality directly affects our selection of performance thresholds and thus training of the reward network. One may adjust the performance thresholds and retraining the reward network with ease, since the entire training process are efficiently designed and may take less than **2 minutes** on the work station aforementioned.

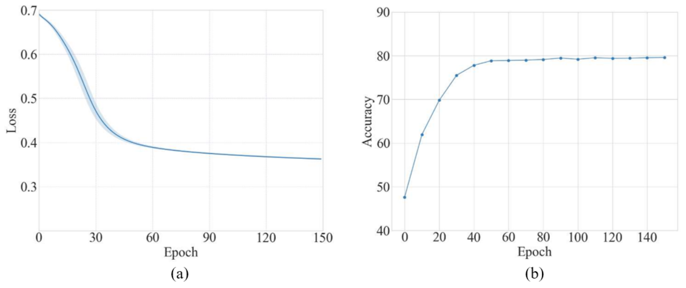


Fig. 2. (a) Training loss of the reward network with variance indicated as shading. (b) Prediction accuracy of the reward network. All results are averaged over ten runs.

B. Selection of Hyperparameter β

Hyperparameter β in (5) controls the trade-off between desired performance and alignment of the generator with the prior data distribution. A larger value of β will enhance the similarity between the generated topology and those in the dataset. Alternatively, a smaller β facilitates topology generation closer to performance specifications, but may potentially reduce the diversity of the generated designs.

To determine the appropriate value of β, a spectrum of its value, $\beta \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$, are examined. Three evaluation metrics, namely, valid, novel and unique, are employed. Here *Valid* is the percentage of generated topology whose gain, phase margin (PM), and gain-bandwidth product (GBW) are greater than zero. *Novel* represents the ratio between the number of valid topology not in the dataset and the total number of generated topology. Those two metrics are popular measures in baseline studies [17]& [18]. We also include textitunique, the ratio between the number of unique topology and that of generated topology, to measure the degree of variety.

Table I presents the averaged results of valid, novel and unique from different values of the hyperparameters β. One can see that as β increases from 0 to 1, its *Valid* ratio drops from 100% to 82.8%, while the *Unique* ratio rises from 61.3% to 100%. This phenomenon arises because the reward network tends to guide the generator towards producing identical topology structures to meet the performance criteria. In the extreme scenario, the generator may produce only a handful of topological designs in order to achieve an extremely low loss value, which is known as mode collapse in GAN training. This may occur in our experiments when the number of epochs is relatively large. To avoid such problem, one must carefully choose appropriate values for β, epochs and learning rate. We also note that a rising value of β would come with a decreasing *Novel* ratio. In other words, the generated topology increasingly conforms to the prior distribution from the dataset. However, this change in novelty is fairly small from 100% to 98.7%. Finally, we select $\beta = 0.5$ for subsequent experiments to ensure the optimal balance among the three ratios.

TABLE I
EVALUATION RESULTS OF THE HYPERPARAMETER β.

β	Valid (%)	Unique (%)	Novel (%)
0.0	100.0	61.3	100.0
0.25	100.0	88.1	100.0
0.5	98.0	99.4	99.7
0.75	90.3	98.7	99.6
1.0	82.8	100.0	98.7

C. Batch Topology Generation

We now examine the effectiveness of trained TPC-GAN model via its generated topology in batch. For comparison, two state-of-the-arts methods, CKtGNN [17] and TSO-flow [18], are also included. Table II presents the results of the three methods. Compared to CktGNN and TSO-flow, our model provides the best scores of *valid*, *unique* and *novel*, representing an improvements of 4.9%, 7.1% and 0.2%, respectively. In contrast to graph embedding-based methods, TPC-GAN is an implicit model that avoids errors introduced by graph reconstruction during encoding and decoding. Instead of constructing surrogate models in latent space, our reward network learns from the intrinsic graph structure and provides more accurate estimation on circuit performance without sizing.

TABLE II
EVALUATION OF BATCH TOPOLOGY GENERATED FROM THREE METHODS.

Algorithm	Valid (%)	Unique (%)	Novel (%)
CktGNN [17]	91.3	92.2	96.3
TSO-flow [18]	93.1	91.6	99.5
TPC-GAN	**98.0**	**99.3**	**99.7**

To better elucidate the crucial role of the reward network in our model, an ablation experiment is conducted. Here we investigate the performance of our model with and without the reward network. Three level performance thresholds are set for circuit performance evaluation:

1) Level 1: GAIN$>$ 0dB, PM$>$ $0°$, and GBW$>$ 0MHz.
2) Level 2: GAIN$>$ 40dB, PM$>$ $30°$, and GBW$>$ 0.7MHz.
3) Level 3: GAIN$>$ 85dB, PM$>$ $55°$, and GBW$>$ 0.7MHz.

The first threshold is the definition of valid circuits. The second is identical to those for the reward network (10) and is considered a medium performance requirement. Lastly, the third threshold represents high-quality circuit performance.

Table III presents the achievement rates of three different thresholds from the topology generated with and without the reward network. It is clear that by including the reward network, achievement rates increase significantly for all three performance thresholds. Among them, improvements of the first threshold is the slightest due to its low requirements of valid. As threshold requirement becomes more stringent, benefits from the reward network to topology generation becomes more significant. For example, the reward network more than doubles the achievement rate of level 2 from 37.3% to 82.8%, and brings a nearly twenty-fold increase at level 3 to 45.7%. Overall, design expertise embedded in the reward network can be correctly conveyed to the generator through the loss function, which in turn assists the generator produce performance-compliant circuit topology.

TABLE III
ACHIEVEMENT RATES OF THREE DIFFERENT THRESHOLDS FROM THE TOPOLOGY GENERATED WITH AND WITHOUT THE REWARD NETWORK.

	Level 1 (%)	Level 2 (%)	Level 3 (%)
without reward network	82.8	37.3	2.7
with reward network	**98.0**	**82.8**	**45.7**

D. Transistor-level Simulation Results

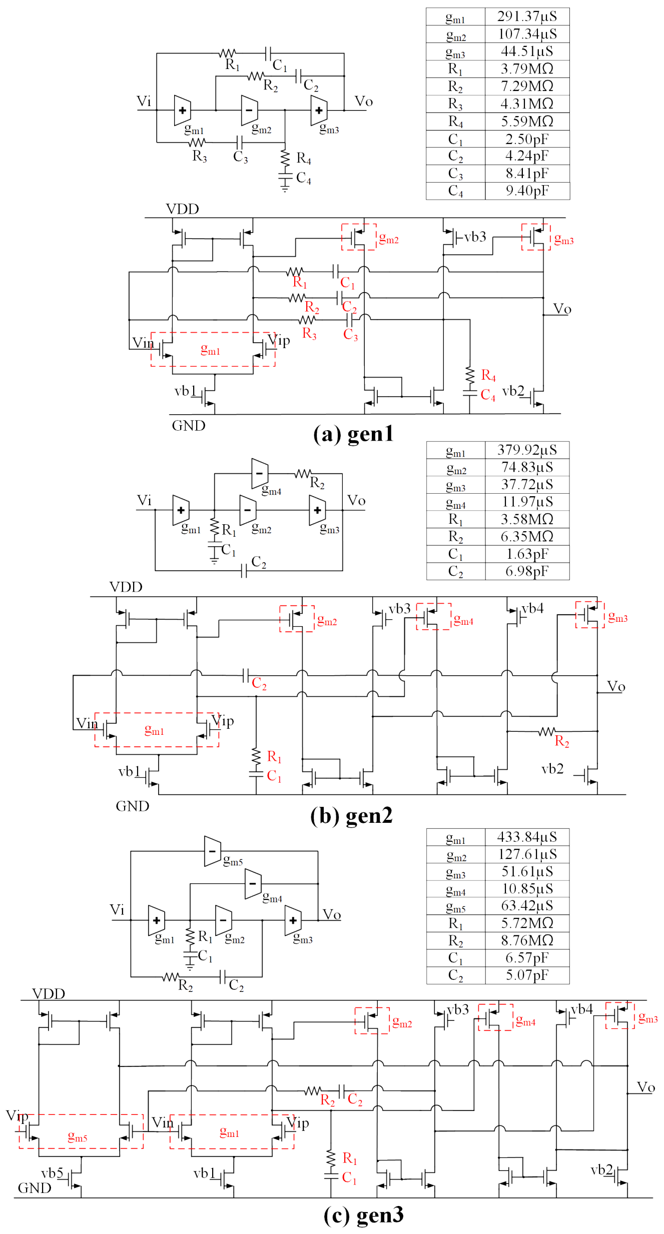
Previous experiments demonstrate the batch generation capability of TPC-GAN. To further demonstrate the practicality of our generated circuits, we produce three opamps that meets the Level 3 specifications and map them into transistor level. Here, the mapping strategy and Figure of Merit (FoM) formula are the same as [14]& [15].

$$\text{FoM} = \frac{\text{GBW [MHz]} \times C_L \text{ [pF]}}{\text{Power [mW]}}. \tag{11}$$

Fig. 3 presents the schematics of our three generated circuits. Their transistor-level simulation results are presented in Table IV along with those from Manual [26]–[28], VAEBO [14] and TOTAL [15]. Although Manual designs achieved better Gain and PM, our generated designs demonstrate strong competitiveness with nearly three times improvement of FoM, along with better GBW and Power. Compared to the other two automatic methods, TPC-GAN leads to the best GBW (2.1) and lowest Power (69), with equally good PM (57.5). In total, its average FoM shows improvements over Manual (181%), VAEBO (112%) and TOTAL (18%).

TABLE IV
TRANSISTOR-LEVEL SIMULATION RESULTS

Methods		FoM	Gain (dB)	PM (degree)	GBW (MHz)	Power (μW)	Tech (nm)
Manual	[26]	73611	>100	57.2	1.1	144	350
	[27]	180441	>100	45	3.4	13	130
	[28]	74569	>100	75.5	1.5	70	180
	avg	**109540**	**>100**	**59.2**	**2**	**76**	/
VAEBO [14]	gen1	282051	80.8	47	1.1	39	40
	gen2	52083	80.2	50	1	192	40
	gen3	101064	85	55	1.9	188	40
	avg	**145066**	**82**	**51**	**1.3**	**140**	/
TOTAL [15]	gen1	264753	89.6	57.3	1.7	62	180
	gen2	220949	89.2	56.7	1.5	70	180
	gen3	295041	100.5	57.2	2.4	80	180
	avg	**260248**	**93.1**	**57.0**	**1.9**	**71**	/
Ours	gen1	346468	91.2	56.4	2	56	180
	gen2	282984	87.3	56.1	2.2	77	180
	gen3	294364	87.5	60.0	2.2	74	180
	avg	**307938**	**88.7**	**57.5**	**2.1**	**69**	/



g_{m1}	291.37μS
g_{m2}	107.34μS
g_{m3}	44.51μS
R_1	3.79MΩ
R_2	7.29MΩ
R_3	4.31MΩ
R_4	5.59MΩ
C_1	2.50pF
C_2	4.24pF
C_3	8.41pF
C_4	9.40pF

g_{m1}	379.92μS
g_{m2}	74.83μS
g_{m3}	37.72μS
g_{m4}	11.97μS
R_1	3.58MΩ
R_2	6.35MΩ
C_1	1.63pF
C_2	6.98pF

g_{m1}	433.84μS
g_{m2}	127.61μS
g_{m3}	51.61μS
g_{m4}	10.85μS
g_{m5}	63.42μS
R_1	5.72MΩ
R_2	8.76MΩ
C_1	6.57pF
C_2	5.07pF

Fig. 3. The behavioral-level circuits and their mapped transistor-level schematics. Behavioral-level device parameters are given in the adjacent tables.

V. CONCLUSION

This paper proposes a novel framework, TPC-GAN, for batch topology synthesis of performance-compliant operational amplifiers. It employs generative adversarial networks (GANs) to directly process the adjacency matrices and node feature matrices of behavioral-level circuits. By doing so, TPC-GAN eliminates the need for graph encoding and decoding and thus reduces training time and avoids reconstruction errors. A reward network is incorporated into GAN, enabling the framework to determine whether a generated topology meets performance specifications without sizing. The experiment results demonstrate that TPC-GAN is capable of generating performance-compliant behavior-level circuits at scale, with significant improvement than state-of-the-art methods. Such may facilitate the training of emerging AI technologies.

References

[1] Anshu Gupta, UBS Chandrawat, DK Mishra, R Khatri, and Preet Jain. A two stage and three stage cmos opamp with fast settling, high dc gain and low power designed in 180nm technology. In *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, pages 448–453. IEEE, 2010.

[2] Yao Lai, Sungyoung Lee, Guojin Chen, Souradip Poddar, Mengkang Hu, David Z Pan, and Ping Luo. Analogcoder: Analog circuit design via training-free code generation. *arXiv preprint arXiv:2405.14918*, 2024.

[3] Bingyang Liu, Haoyi Zhang, Xiaohan Gao, Zichen Kong, Xiyuan Tang, Yibo Lin, Runsheng Wang, and Ru Huang. Layoutcopilot: An llm-powered multi-agent collaborative framework for interactive analog layout design. *arXiv preprint arXiv:2406.18873*, 2024.

[4] Yuxuan Yin, Yu Wang, Boxun Xu, and Peng Li. Ado-llm: Analog design bayesian optimization with in-context learning of large language models. *arXiv preprint arXiv:2406.18770*, 2024.

[5] Prabir C Maulik, L Richard Carley, and Rob A Rutenbar. Integer programming based topology selection of cell-level analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401–412, 1995.

[6] Federico Bruccoleri, Eric AM Klumperink, and Bram Nauta. Generating all two-mos-transistor amplifiers leads to new wide-band lnas. *IEEE Journal of Solid-State Circuits*, 36(7):1032–1040, 2001.

[7] Cindy Goh and Yun Li. Ga automated design and synthesis of analog circuits with practical constraints. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 170–177. IEEE, 2001.

[8] Trent McConaghy, Pieter Palmers, Michiel Steyaert, and Georges GE Gielen. Variation-aware structural synthesis of analog circuits via hierarchical building blocks and structural homotopy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(9):1281–1294, 2009.

[9] Trent McConaghy, Pieter Palmers, Michiel Steyaert, and Georges GE Gielen. Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks. *IEEE Transactions on Evolutionary Computation*, 15(4):557–570, 2011.

[10] Zhenxin Zhao and Lihong Zhang. An automated topology synthesis framework for analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4325–4337, 2020.

[11] Zhenxin Zhao and Lihong Zhang. Analog integrated circuit topology synthesis with deep reinforcement learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(12):5138–5151, 2022.

[12] Jialin Lu, Liangbo Lei, Fan Yang, Changhao Yan, and Xuan Zeng. Automated compensation scheme design for operational amplifier via bayesian optimization. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 517–522. IEEE, 2021.

[13] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in neural information processing systems*, 32, 2019.

[14] Jialin Lu, Liangbo Lei, Fan Yang, Li Shang, and Xuan Zeng. Topology optimization of operational amplifier in continuous space via graph embedding. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 142–147. IEEE, 2022.

[15] Zihao Chen, Songlei Meng, Fan Yang, Li Shang, and Xuan Zeng. Total: Topology optimization of operational amplifier via reinforcement learning. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pages 1–8. IEEE, 2023.

[16] Zihao Chen, Songlei Meng, Fan Yang, Li Shang, and Xuan Zeng. Macro: Multi-agent reinforcement learning-based cross-layer optimization of operational amplifier. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 423–428. IEEE, 2024.

[17] Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. Cktgnn: Circuit graph neural network for electronic design automation. *arXiv preprint arXiv:2308.16406*, 2023.

[18] Jinglin Han, Yuhao Leng, Xiuli Zhang, and Peng Wang. Tso-flow: A topology synthesis and optimization workflow for operational amplifiers with invertible graph generative model. In *2024 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. ACM, 2024.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[20] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[21] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[22] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[23] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer, 2018.

[24] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[25] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*, pages 105–116. Springer, 2006.

[26] Zushu Yan, Pui-In Mak, Man-Kay Law, and Rui P Martins. A 0.016-mm^2 144-μw three-stage amplifier capable of driving 1-to-15 nf capacitive load with $>$0.95-mhz gbw. *IEEE journal of solid-state circuits*, 48(2):527–540, 2013.

[27] Min Tan and Wing-Hung Ki. A cascode miller-compensated three-stage amplifier with local impedance attenuation for optimized complex-pole control. *IEEE Journal of Solid-State Circuits*, 50(2):440–449, 2014.

[28] Wanyuan Qu, Shashank Singh, Yongjin Lee, Young-Suk Son, and Gyu-Hyeong Cho. Design-oriented analysis for miller compensation and its application to multistage amplifier design. *IEEE Journal of Solid-State Circuits*, 52(2):517–527, 2016.