GAN-Place: Advancing Open Source Placers to Commercial-quality Using Generative Adversarial Networks and Transfer Learning

YI-CHEN LU, Georgia Institute of Technology, USA
HAOXING REN, Nvidia, USA
HAO-HSIANG HSIAO and SUNG KYU LIM, Georgia Institute of Technology, USA

Recently, GPU-accelerated placers such as DREAMPlace and Xplace have demonstrated their superiority over traditional CPU-reliant placers by achieving orders of magnitude speed up in placement runtime. However, due to their limited focus in placement objectives (e.g., wirelength and density), the placement quality achieved by DREAMPlace or Xplace is not comparable to that of commercial tools. In this article, to bridge the gap between open source and commercial placers, we present a novel placement optimization framework named GAN-Place that employs generative adversarial learning to transfer the placement quality of the industry-leading commercial placer, Synopsys ICC2, to existing open source GPU-accelerated placers (DREAMPlace and Xplace). Without the knowledge of the underlying proprietary algorithms or constraints used by the commercial tools, our framework facilitates transfer learning to directly enhance the open source placers by optimizing the proposed differentiable loss that denotes the "similarity" between DREAMPlace- or Xplace-generated placements and those in commercial databases. Experimental results on seven industrial designs not only show that our GAN-Place immediately improves the Power, Performance, and Area metrics at the placement stage but also demonstrates that these improvements last firmly to the post-route stage, where we observe improvements by up to 8.3% in wirelength, 7.4% in power, and 37.6% in Total Negative Slack on a commercial CPU benchmark.

CCS Concepts: • **Hardware → Placement**; **Physical design (EDA)**; **Methodologies for EDA;**

Additional Key Words and Phrases: Placement optimization, generative adversarial learning, transfer learning

ACM Reference format:
Yi-Chen Lu, Haoxing Ren, Hao-Hsiang Hsiao, and Sung Kyu Lim. 2024. GAN-Place: Advancing Open Source Placers to Commercial-quality Using Generative Adversarial Networks and Transfer Learning. *ACM Trans. Des. Autom. Electron. Syst.* 29, 2, Article 32 (February 2024), 17 pages.
https://doi.org/10.1145/3636461

1 INTRODUCTION

With the ever-increasing technology scaling driven by Moore's Law, modern **Very Large-Scale Integration (VLSI)** designs easily comprise millions, if not billions, of instances that must be placed onto constrained layouts. Unfortunately, existing commercial Electronic Design Automation tools heavily rely on CPU-intensive heuristics or analytical objectives to solve the placement

Authors' addresses: Y.-C. Lu, H.-H. Hsiao, and S. K. Lim, Georgia Institute of Technology, Georgia, USA; e-mails: yclu@gatech.edu, thsiao@gatech.edu, limsk@ece.gatech.edu; H. Ren, Nvidia, Texas, USA; e-mail: haoxingr@nvidia.com.

1084-4309/2024/02-ART32
https://doi.org/10.1145/3636461

problem, which is often inadequate in generating high-quality placements within a reasonable amount of runtime. To address this issue, open source GPU-acceleratable analytical placers, DREAMPlace [11] and Xplace [13], are proposed to exploit the parallelization capabilities of GPU kernels to significantly reduce placement runtime. Particularly, the original CPU-intensive objective functions are reformulated into GPU-acceleratable cuda kernels that leverage GPU-specific blocks and threads to expedite analytical computations without sacrificing solution quality.

Despite the significant success achieved by DREAMPlace and Xplace, the placement solution quality of these GPU-accelerated placers is still inferior to those of commercial tools. The key reason is that these placers, as there CPU counterparts, the ePlace/RePlace family [3, 14], only have limited placement objective focus on wirelength and density. Particularly, they cannot consume many other essential constraints (e.g., timing, power) as commercial **Physical Design (PD)** tools, leading to inferior placement quality in terms of full-chip **Power, Performance, and Area (PPA)** metrics. This motivates us to ask the following question: Is there any way to advance DREAMPlace toward commercial-quality without knowing the secret sauces of those black-boxed commercial engines? In this article, we prove that the answer is yes, particularly via generative adversarial learning [20].

In this work, we present the first-ever learning-driven placement optimization framework named GAN-Place that can be directly integrated with any GPU-accelerated placer (e.g., DREAMPlace or Xplace) to advance the placement quality toward commercial-quality using generative adversarial learning. The key idea is that although we do not know the underlying algorithms or constraints used by the tools, we can "quantify placement similarity" between tool-optimized placements and DREAMPlace- or Xplace-generated placements using generative learning, and by optimizing the differentiable similarity scores computed from the proposed discriminators, we can narrow the distribution gap between open source and commercial placers. Our is a simple yet highly effective, which is greatly motivated by the success of **Generative Adversarial Networks (GANs)** [5] in real-world applications, where signals in different domains (e.g., texts and images) can be converted to each other by parameterizing target distributions using differentiable frameworks.

Figure 1 presents a high-level overview of GAN-Place, where the underlying GPU-accelerated placer is considered as a "generator" whose goal is to generate the placements that follow similar distributions as the tool-optimized ones in the databases (which are optimized for different PPA objectives). To achieve this, two discriminators built upon **Convolutional Neural Networks (CNN)** and **Graph Neural Networks (GNN)** are developed to quantify the placement similarity between two types of placement (i.e., gpu placer-generated and tool-optimized). The goal of the discriminator is to make correct judgements by telling whether its input is coming from commercial databases or DREAMPlace, whereas one of the objectives of DREAMPlace (in our GAN-Place settings) is to "fool" the discriminator by generating similar (or realistic) placements as the ones in the database. Specifically, the GNN-based discriminator is dedicated to encode netlist connectivity with cell locations as node attributes, and the CNN-based discriminator is dedicated to encode bin-density maps resulted from the underlying cell locations.

The goal of this work is to demonstrate that the placement quality (or style) of a placer (e.g., commercial placer) can be transferred onto another placer without knowing the underlying algorithms or constraints through generative adversarial learning. Particularly, we show that any GPU-accelerated placer that serves as a differentiable placement engine can be elegantly considered as a generator that generates tool-alike placements of the underlying gate-level input netlists. The greatest strength of GAN-Place is that it enables any differentiable placer to optimize the underlying placement toward a commercial tool-verified (and optimized) direction without explicitly knowing the black-boxed algorithms. Although DREAMPlace or Xplace leverages fundamentally

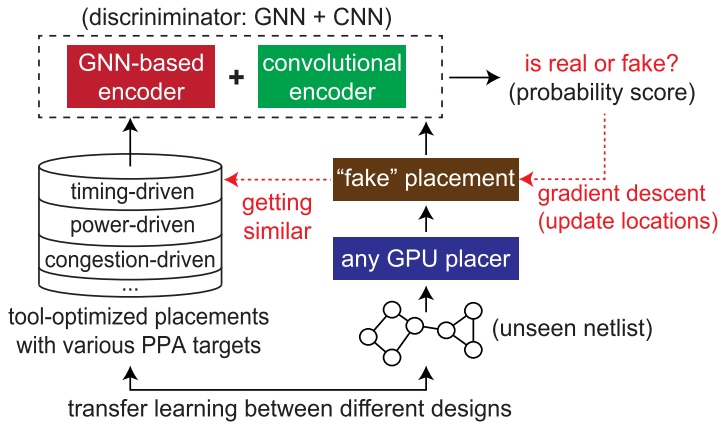

Fig. 1. High-level overview of our GAN-Place framework that performs placement optimization using generative adversarial learning. Note that GAN-Place can be integrated with any GPU-accelerated placer to improve placement quality. In this work, we particularly showcase the integration with DREAMPlace [11] and Xplace [13].

different algorithms compared with the tools, in this work, we demonstrate that the placement quality of these open source placers can be efficiently improved with the proposed framework GAN-Place.

2 RELATED WORKS AND MOTIVATIONS

Analytical placers [2, 3, 7, 8, 14] have brought tremendous success to the semiconductor industry since the early 2010s [10]. Nevertheless, recent advancements in ML Theory and its applications have further pushed modern placers to an unprecedented frontier [9]. Based on the characteristics of learning strategies, existing learning-driven frameworks for VLSI placement can be categorized into the following streams:

— *Supervised Placement Quality Prediction*: These categories include the works who strive to predict crucial placement-related metrics in early stages of the design flow, which typically rely on pre-built databases with comprehensive data. A recent work [4] developed an ensemble framework to predict post-place congestion and timing metrics based on a massive database that contains millions of instances from 72 industrial designs. To extend the prediction scope from single-stage to full-flow modeling [15, 18], developed learning-driven flow-based graph modeling techniques using GNNs to predict end-of-flow PPA metrics for each intermediate placement stage, allowing designers to perform efficient design space exploration.

— *Unsupervised Placement Optimization*: This category refers to the works who aim to develop ML frameworks that directly perform placement optimization without a pre-built database. In other words, they often directly consider ML models as optimization rather than prediction frameworks. Reference [19] presented the PL-GNN framework that generates cell clustering constraints as placement guidance using GNNs to advance commercial placers. The key idea is to drive commercial placers to spend additional effort in placing the cells belonging to the same ML-predicted cluster closer to each other to improve the overall PPA metrics. However, the framework in Reference [19] is not "goal-directed" as the graph learning and the clustering steps are not end-to-end-differentiable, which prohibits GNN models

from utilizing feedback from the achieved optimization results. To overcome this problem, the authors of Reference [21] developed differentiable PPA-inspired ML loss functions to improve the GNN learning process with direct optimization feedback, which includes timing, power, and congestion evaluations of the clustering results.

— ***Reinforcement Learning (RL)*** *Placement Optimization*: RL is a promising paradigm that has shown superior optimization results in high-dimensional control problems. The authors of Reference [22] presented a seminal work that leveraged RL for macro placement to replace human labor, which significantly improves the chip design turn-around time. Another work [1] utilized RL to efficiently tune the placement parameters of commercial placers through thousands of iterations. Still another work [24] further combined simulated annealing and RL in a cyclic fashion to conduct iterative placement optimization. However, given that RL algorithms typically demand significant training time, this article focuses on placement optimization techniques that do not incur excessive overhead to enhance placement quality.

In this work, we take a brand new perspective to solve the VLSI placement optimization problem using generative adversarial learning [5]. In the realm of PD, previous work [16] has shown that the generative learning conducted by GAN can elegantly optimize the **Clock Tree Synthesis (CTS)** process of commercial tools without knowing the algorithms inside the black-boxed CTS engine. Motivated by Reference [16], in this article, we aim to leverage GAN-based models to demystify commercial black-boxed placers so as to improve open source placers: DREAMPlace [12] and Xplace [13] toward commercial-quality. Particularly, we consider any vanilla open source placer as a generator in a conventional GAN-based framework whose goal is to generate commercial-quality placements from an unplaced input netlist. To achieve this, we first build a relevant database containing commercial-quality placements with different objectives (e.g., timing-driven, power-driven, congestion-driven, etc.) using *Synopsys ICC2*, an industry-leading commercial tool. We then develop a discriminator that serves as a teacher to improve DREAMPlace by optimizing the similarity scores between DREAMPlace-generated placements and the ones in the commercial database as shown in Figure 1. By minimizing the similarity loss, the underlying cell locations of DREAMPlace- or Xplace-generated placements will be updated through gradient descent, which effectively become more similar to the tool-generated placements in the database.

3 GAN-PLACE OVERVIEW

It is widely acknowledged that generative adversarial learning is a promising paradigm that effectively captures complicated distributions using generative and discriminative models that have "opposite" objectives. Generally speaking, the goal of the generator is to generate target distribution alike data from random (or non-meaningful) distributions, while the goal of the discriminator is to distinguish the source of its inputs (i.e., from the generator or from the target distribution). Note that both generator and discriminator can be realized by any differentiable system (i.e., not necessarily neural networks). As aforementioned, in this article we consider DREAMPlace, a differentiable placement system, as a generator whose goal is to generate placements that follow similar distributions as the ones in a tool-optimized database.

Our discriminator leverages CNNs and GNNs to determine the origin of its input source that alternates in each iteration of GAN training, where CNNs are responsible to encode cell bin-density maps and GNNs are responsible to encode netlist connectivity. The rationale behind using GNNs for netlist encoding is that netlists are essentially hypergraphs whose node connectivity is critical to placers. The motivation behind using CNNs for bin-density map encoding is shown in Figure 2.

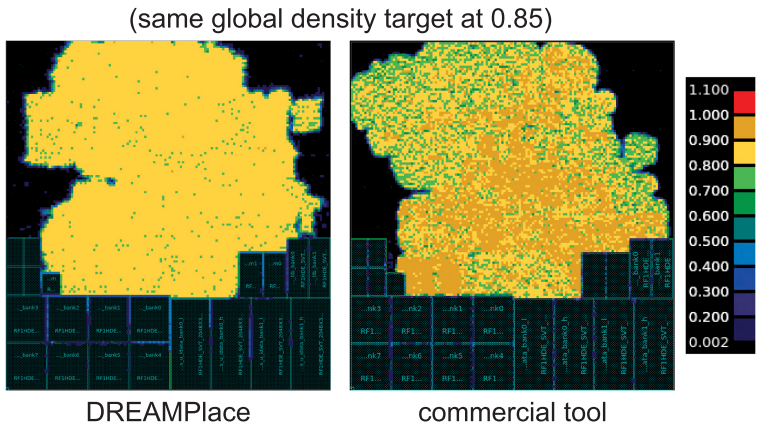

Fig. 2. Comparison of cell density maps between DREAMPlace and Synopsys ICC2 under the same global placement density target. It is observed that the commercial tool has extra intelligence on where to locally aggregate or spread out cells to optimize crucial PPA metrics while satisfying the global density constraints.

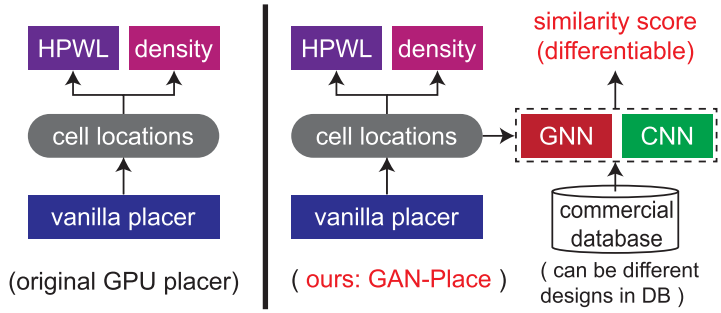

Fig. 3. Difference in placement objectives between a vanilla GPU-accelerated (i.e., differentiable) placer and GAN-Place at each placement iteration. The cell locations generated in GAN-Place are encouraged to follow the ones in commercial database by optimizing the similarity scores.

We observe that under the same global density target, the commercial tool has extra intelligence on locally aggregating or loosening cells to improve design PPA metrics globally, where DREAMPlace naively strives to make every local bin to have the same local density target as the global density target. This density variation is proven to be critical to the success of placement optimization in Reference [21]. The observation shown in Figure 2 strongly motivates us to leverage bin-density map as one the indicators of placement similarity, and CNNs thus become the second-to-none choice to perform encoding on it as they are well known for grid signals (e.g., images) classification.

Figure 3 shows the key objective difference between the proposed framework GAN-Place and the vanilla GPU-acceleratable (i.e., differentiable) placer. Aside from optimizing the **wirelength (WL)** and density objectives as in the vanilla placer on the left, in each placement iteration, GAN-Place further computes a differentiable similarity loss using GNN-based and CNN-based discriminators. Note that the in our generative settings, the input design of GAN-Place and the designs in the commercial database are not necessarily the same. That is, GAN-Place facilitates transfer learning to perform placement optimization on unseen netlists, which is the greatest strength of the proposed method.

Table 1. Parameters We Leverage for Database Generation

ICC2 parameters	type (values)	description
set_qor_strategy	enum (3)	set optimization priority
low_power_effort	enum (4)	effort in low power optimization
congestion_effort	enum (3)	effort in congestion optimization
is_timing_driven	bool (2)	is timing-driven placement
is_power_driven	bool (2)	is power-driven placement
buffer_aware	bool (2)	buffering of high-fanout nets
coarse_density	float ([0.7,0.9])	density of global placement
target_route_density	float ([0.7,0.9])	density of early global routing

3.1 Commercial Database Construction

A high-quality placer is helpful to demonstrate the proposed placement optimization technique using generative adversarial learning. In this article, we take *Synopsys ICC2*, an industry-leading commercial tool, as our reference tool for database construction. Particularly, we sweep around essential placement parameters offered by *ICC2* as shown in Table 1. The combinations of these parameters form a high-dimensional space, leading to a variety of placements that have distinct PPA focus, including performance-driven placements, low-power placements, routability-driven placements, and so on. Nonetheless, we want to emphasize that our framework GAN-Place is not limited to any objective focus. It can be equipped with any database to transfer the placement quality (or style) within onto the targeted placer (e.g., DREAMPlace or Xpalce). More importantly, the designs in the database and the designs that the targeted placer is optimizing do not have to be the same as GAN-Place facilitates transfer learning for placement optimization. The goal of this work is to demonstrate that GAN-Place can perform efficient placement optimization on unseen netlists using a tool-optimized database where the placements are not necessarily coming from the same design.

4 GAN-PLACE ALGORITHMS

The key idea of GAN-Place is to transfer the placement quality (or more precisely, style) of one placer onto another through generative adversarial learning. To achieve this, a discriminator built upon GNNs and CNNs is developed to quantify similarity of placements between different placers. Since the similarity scores have to be differentiable to update the underlying cell locations (so as to improve DREAMPlace), a differentiable graph pooling methodology is adopted, and a differentiable bin-density map transformation technique named Soft-Bin is proposed.

4.1 VLSI Netlist Encoding Using GNNs

Graph representation learning conducted by GNNs is an effective technique to encode netlist connectivity and the underlying attributes into a meaningful graph- or node-level representations [17] In the realm of PD, such encoded representations have been leveraged to successfully solve many tasks that are once considered extremely hard to solve [23]. In this article, we specifically leverage GNNs to encode netlist graph $G = (V, E)$ of the underlying placement while considering cell locations as initial attributes. The results graph-level vectors are further taken as the input of the proceeding GNN-based discriminator network to decide the similarity between different placements.

To apply GNNs for solving VLSI problems, a netlist transformation technique is required to transform the netlist hypergraphs into normal graphs. A naive transformation that assigns a GNN message passing edge between each cell on the same net will introduce $O(n^2)$ edges, which may

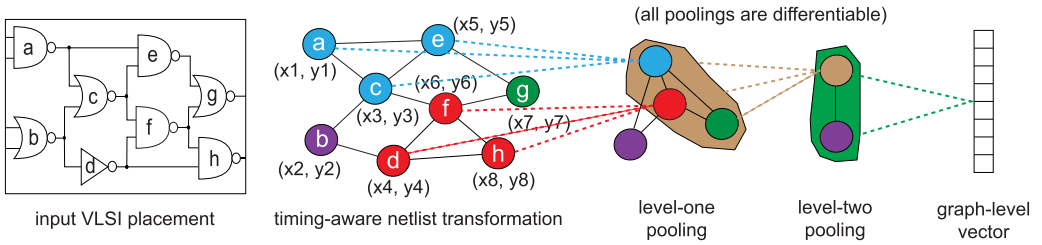

Fig. 4. Illustration of GNN netlist transformation and graph pooling. First, following Reference [21], we perform timing-aware netlist transformation by only taking timing arcs as GNN message passing edges. Then, we perform attention graph pooling to construct the final graph-level vector that characterizes the input placement. Note that the poolings are differentiable (i.e, learnable), meaning that the clustering results (e.g., {a, c, e} is now a cluster) will change across training iterations. The obtained graph-level vector is fed to downstream networks to quantify placement similarity.

damage the quality of graph learning as redundant edges limit the expressiveness of GNNs [25]. To overcome this issue, in this work, we adopt the netlist transformation technique proposed in Reference [21] that for every net in the original netlist graph, only driver-to-load connections are added in the transformed graph, and beyond that, artificial edges are introduced between every start points and end points of timing paths.

Figure 4 illustrates the graph learning process of the proposed GAN-Place framework. Starting from an input placement, we first perform timing-aware netlist transformation as in Reference [21]. The transformation only preserves timing arcs as GNN message passing edges. Then, following from GraphSAGE [6], an inductive based message passing process is leveraged to transform the initial features of each node into better representations through neighborhood aggregation as

$$\begin{aligned} h_{Neigh(v)}^{k-1} &= reduce\_mean\left(\{\mathbf{W}_k^{agg} h_u^{k-1}, \forall u \in Neigh(v)\}\right), \\ h_v^k &= \sigma\left(\mathbf{W}_k^{proj} \cdot \text{concat}\left[h_v^{k-1}, h_{Neigh(v)}^{k-1}\right]\right), \end{aligned} \tag{1}$$

where k denotes the transformation level, σ denotes the sigmoid function, $Neigh(v)$ denotes the neighbors of node v, and W_k^{agg} and W_k^{proj} denote the aggregation and projection matrices at the kth layer of the GNN model. Note that Equation (1) is repeated for every cell in the design. In the implementation, our GNN has three layers ($K = 3$), and each of them has a 32 hidden dimensions. Hence, the final node embeddings $\{h_v^3, \forall v \in V\}$ has 32 dimensions.

At each level k of the node representation learning, a differentiable graph attention pooling mechanism [26] is applied to coarsen the graph via a soft clustering assignment C_k, where nodes belonging to the same cluster will be merged into a super-node through mean pooling, which can be derived as

$$H_{k+1} = C_k H_k, \quad A_{k+1} = {A_k}^T C_k A_k, \tag{2}$$

where $H_k = \{h_v^k, \forall v \in V\}$, A_k denotes the adjacency matrix at level k, and $C_k \in R^{|V|_{k+1} \text{x} |V|_k}$ denotes the mapping of nodes between level k and level $k+1$, where a node v may be merged into a super-node or stay as itself. At the end of the entire graph representation learning, a mean pooling is applied across the remaining nodes to obtain the final graph-level vector g, which is taken as one of the indicators of placement similarity. A graphical illustration of the entire graph learning process is shown in Figure 4. With the graph pooling strategy, netlist information will be iteratively condensed into a graph-level vector g that characterizes the underlying input placement G.

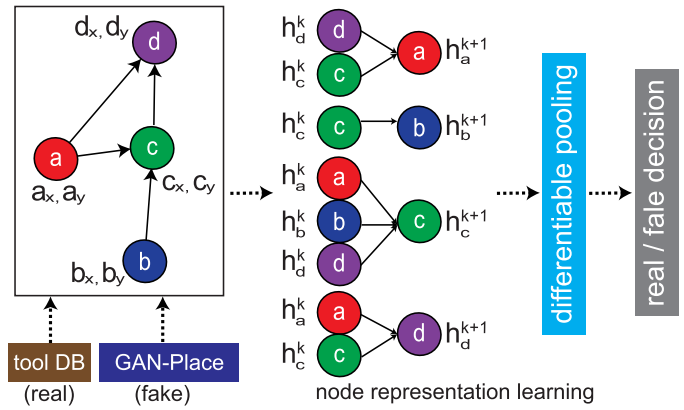

Fig. 5. GNN-based discriminator that outputs "fake/real" decision using node representation learning based on different input placement sources.

Finally, we want to emphasize that both GAN-Place generated placements and tool-optimized placements go through the same graph learning process as shown in Figure 4. Let D_{gnn} denote the discriminator network parameters that are related to graph representation learning, the graph representation learning objective L_{gnn} can be derived using cross-entropy as

$$\begin{aligned}\mathcal{L}_{gnn} =& \mathbb{E}_{(G,x,y)\sim database}\left[log(D_{gnn}(G,x,y))\right] \\ &+ \mathbb{E}_{(G,x,y)\sim DREAM}\left[log(1-D_{gnn}(G,x,y))\right].\end{aligned} \tag{3}$$

Equation (3) reflects the adversarial nature in our GAN-Place framework that depending on the sources of input, we train the discriminator to make corresponding correct judgements. With enough training iterations, our GNN module is able to extract the information that tells the key difference between DREAMPlace-generated placement and the tool-optimized ones.

4.2 GNN-based Discriminator

Figure 5 demonstrates the adversarial learning conducted by GNNs in the proposed framework GAN-Place. By applying node representation learning on the input placement, which is either from the commercial database or from the differentiable placement engine, we train the GNN model to differentiate various placement sources based on netlist connectivity and assigned cell locations. Note that the goal of GAN-Place is to generate "realistic" placements that are similar to the ones in the database so as to "fool" the discriminator to predict its generated placements as coming from the database. Nonetheless, GNN alone is not enough to characterize and differentiate different placements. For example, to truly encode the difference of the bin-density distribution as shown in Figure 2, a more direct approach that quantifies the layout information is needed. Hence, in this work, we leverage CNN-based networks to encode such information.

4.3 Soft-Bin: Differentiable Two-dimensional Bin-Density Map Transformation

So far, we have introduced how GAN-Place encodes netlist connectivity using GNNs by leveraging a differentiable attention pooling mechanism with cell locations as initial node features. Now, we present the details of how GAN-Place leverages bin-density maps to characterize and differentiate different placements with CNNs, where the key motivation behind is illustrated in Figure 2.

It should first be mentioned that most common bin-density calculation method using the simple formula of dividing the total cell area in a bin by the total bin area is not differentiable, as

ALGORITHM 1: Soft-Bin Transformation.

Input: $G = (V, E)$: input netlist, $\{X_v, Y_v \; \forall v \in V\}$: cell locations of the input placement, W: with of floorplan, H: height of floorplan, b_width: bin width, b_height: bin height.

Output: $M \in R^{|V|x|V|}$: differentiable two-dimensional bin-density map.

1: $M[*][*] \leftarrow 0$ ▷ initialize M to **0**
2: $bin\_area = b\_width * b\_height$
3: $num\_w \leftarrow floor(\frac{W}{b\_width})$
4: $num\_h \leftarrow floor(\frac{H}{b\_height})$
5: **for** $i = 0; i < num\_w; ++i$ **do**
6: **for** $j = 0; j < num\_h; ++j$ **do**
7: $V' \leftarrow filter\{i * num\_w \leq X_v < (i+1) * num\_w \forall v \in V\}$
8: $V' \leftarrow filter\{j * num\_h \leq Y_v < (j+1) * num\_h \forall v \in V'\}$
9: **for** $v \in V'$ **do**
10: $b \leftarrow M[i][j]$
11: $neigh\_bins \leftarrow$ get adjacent or diagonal bins of b
12: $dist\_vec \leftarrow []$ ▷ distance vector of cell v to each bin
13: $dist\_vec.push\_back(\|b_x - v_x, b_y - v_y\|^2)$
14: **for** $nb \in neigh\_bins$ **do**
15: $dist\_vec.push\_back(\|nb_x - v_x, nb_y - v_y\|^2)$
16: **end for**
17: $prob\_vec \leftarrow softmax\left(dist\_vec^{-1}\right)$
18: $area\_vec \leftarrow area_v * prob\_vec$ ▷ expected values
19: update M by $area\_vec$ ▷ add area of each bin to M
20: **end for**
21: **end for**
22: **end for**
23: $M \leftarrow \frac{M}{bin\_area}$ ▷ convert expected area to expected density

the act of assigning a cell to a particular bin is deterministic. Although such computation is "exact" and accurate, a probabilistic calculation method is needed to compute gradients based on cell (x, y) locations so as to improve the underlying placement through gradient descent. To achieve this, we develop a differentiable bin-density transformation technique named Soft-Bin as shown in Algorithm 1.

The key idea behind Algorithm 1 is that instead of deterministically assigning each cell to an exact bin purely based on its location, we can probabilistically distribute the area of a cell onto its neighboring bins, which can be achieved by any activation function that maps real values into probabilities. In this article, we use softmax for such computation. The beauty of our probabilistic bin-density calculation approach is that it enables the underlying cell locations to be updated along with any operation (i.e., maximization or minimization) of the computed density. That is, with the proposed Soft-Bin technique, cell locations can be directly updated using gradient descent by optimizing the similarity loss computed from the CNN-based discriminator.

The proposed Soft-Bin algorithm works as follows. In Lines 1–4, we initialize the bin density map M based on the specifications from inputs. The cells corresponding to each bin can be obtained using Lines 7 and 8. Now, as shown in Lines 10–15, for each cell that deterministically belongs to a target bin b, we first identify its neighboring bins (adjacent or diagonal) $neigh\_bins$ and then compute a distance vector $dist\_vec$ that denotes the Euclidean distance between the target cell to

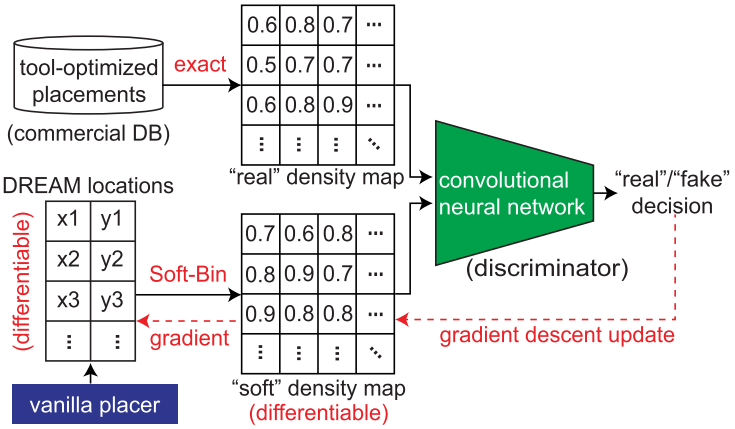

Fig. 6. Illustration of our CNN-based discriminator using the Soft-Bin transformation technique that transforms cell locations to differentiable bin-density maps. The goal of DREAMPlace is to generate placements that have similar bin-density maps as the ones in the database.

each bin (including b and *neigh_bins*). Finally, we transform the distance vector *dist_vec* into a probability vector *prob_vec* using the softmax function as shown in Line 16, and the "expected" area contribution can be calculated using Line 17. After updating the bin-density map M by the expected area contribution of each cell in the design, we obtain the final density of each bin using Line 19.

4.4 CNN-based Discriminator

With the proposed Soft-Bin technique for differentiable bin-density map transformation, we now describe the adversarial learning conducted by the CNN-based discriminator as shown in Figure 6. Note that the key motivation for using CNNs to encode and differentiate different placements is originated from Figure 2, where we clearly observe that the commercial tool is having additional intelligence on locally spreading or coarsening cells for PPA optimization that DREAMPlace is lacking of. Therefore, to advance DREAMPlace toward commercial quality, we aim to improve the DREAMPlace generated locations by following the bin-density distributions as the ones generated by the commercial tool.

As shown in Figure 6, we use the proposed Soft-Bin technique to transform the DREAMPlace generated placement into a differentiable "soft" bin-density map, while using exact (i.e., deterministic) computation to obtain the bin-density map from the commercial database, as it does not need to be differentiable. By considering the input density maps as grid signals, CNNs are used to perform convolution to extract key information that can be used to decide whether the input is coming from DREAMPlace or the database.

4.5 End-to-End GAN-Place Training

Figure 7 shows the detailed architecture of the CNN-based and the GNN-based discriminators. Note that placements originating from GAN-Place do not have to be the same as the ones from the commercial database, since our framework does not explicitly take design information such as number of cells or number of nets as features (i.e., not memorizing). Instead, GAN-Place learns to parameterize placement distributions in terms of netlist connectivity and the achieved bin-density maps. All the computations involved are "size-independent," meaning that netlists in different sizes

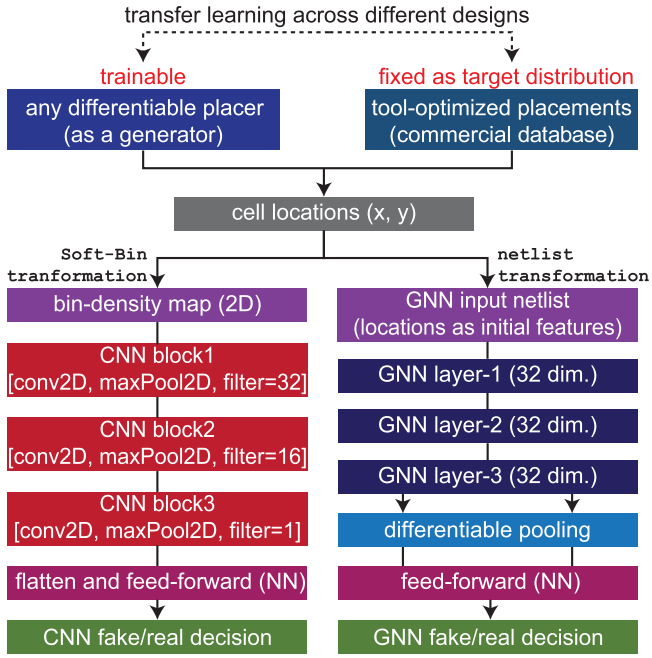

Fig. 7. Detailed architecture of GAN-Place that leverages CNN-based and GNN-based discriminators to quantify placement similarity between different placement sources.

will be encoded to vectors in the same dimensions, which is true for both CNN-based and GNN-based discriminator networks.

The key difference between the proposed GAN-Place and a vanilla GPU-accelerated (differentiable) placer is shown in Figure 3. Beyond the conventional placement objectives that are wirelength and density, we consider the underlying placement engine in our GAN-Place framework as a generator to maximize the probability of the generated placement being classified as "tool-optimized" by the GNN-based and CNN-based discriminators. The similarity loss L_{sim} computed by the entire discriminator (GNN based and CNN based) can be obtained as

$$\begin{aligned}\mathcal{L}_{sim} =& \mathbb{E}_{(G,x,y)\sim database}\left[log(D_{gnn}(G,x,y))\right] \\ &+ \mathbb{E}_{(G,x,y)\sim DREAM}\left[log(1-D_{gnn}(G,x,y))\right] \\ &+ \mathbb{E}_{(G,x,y)\sim database}\left[log(D_{cnn}\left(G, exact\_map(x,y)\right))\right] \\ &+ \mathbb{E}_{(G,x,y)\sim DREAM}\left[log(1-D_{cnn}\left(G, Soft\text{-}Bin(x,y)\right))\right],\end{aligned} \tag{4}$$

where D_{gnn} and D_{cnn} denote the GNN-based and CNN-based discriminators, respectively; (G, x, y) denotes the sampled (either from the database or from DREAMPlace) netlist graph with annotated cell locations; $exact\_map$ denotes the deterministic bin-density map; and $Soft - Bin$ denotes the proposed transformation technique. Finally, the similarity loss L_{sim} will be jointly optimized with the original wirelength and density objectives (L_{WL} and $L_{density}$) in any vanilla GPU-accelerated placer (e.g., DREAMPlace or Xplace) as

$$L = L_{WL} + \lambda_1 L_{density} + \lambda_2 L_{sim}, \tag{5}$$

where L_{WL} and $L_{density}$ denote the wirelength and density objectives computed from the vanilla DREAMPlace and $\{\lambda\}$ denote the hyper-parameters for loss weighting. At each placement

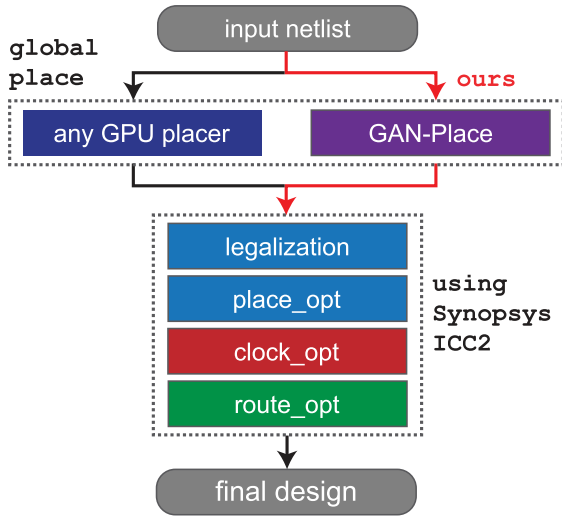

Fig. 8. Illustration of integrating open source placers into an industrial design flow using Synopsys ICC2, where the global placement stage is replaced with proceeding stages remain the same.

optimization, DREAMPlace will optimize Equation (5) to improve the underlying placement toward commercial-quality.

4.6 Full-Flow Integration

Figure 8 shows a graphical illustration of integrating the proposed framework GAN-Place into an industrial design flow implemented by *Synopsys ICC2*. To perform fair comparisons between vanilla placers (e.g., DREAMPlace and Xplace) and the proposed GAN-Place, we use the same ICC2 parameters and seeds to implement the rest of the design flow after global placement to remove run-to-run variation. In the experiments, we demonstrate that GAN-Place efficiently improves both DREAMPlace [11] and Xplace [13] in critical PPA metrics at each major stage of the PD flow. Note that in this work, we not only demonstrate that GAN-Place immediately improves the PPA metrics measured at the placement stage but also show that the improvements last firmly to the post-route stage.

5 EXPERIMENTAL RESULTS

In this article, we validate GAN-Place with three commercial CPU benchmarks and five OpenCore benchmarks using the TSMC 28-nm technology node. The commercial database is generated by randomly sampling the parameters listed in Table 1 using *Synopsys ICC2*, where per benchmark, 100 placements are generated with different PPA objectives. In the experiments, we not only demonstrate that GAN-Place significantly improves vanilla DREAMPlace in single-design optimization but also show that it achieves superior optimization results on unseen designs that are not in the database.

5.1 Single-design Optimization Results on DREAMPlace [11]

In this experiment, we perform “single-design” optimization to demonstrate the considerable placement quality enhancements offered by the proposed GAN-Place framework by taking DREAMPlace [11] as a generator. The term "single-design" implies that the design that GAN-Place optimizes is same as the design in the target database. Our objective in this experiment is

Table 2. Eight Benchmarks Used in This Work and Their Attributes in TSMC 28 nm

Design Name	# Nets	# FFs	# Cells	# macros
AES	90,905	10,688	113,168	0
CPU-1	206,224	22,366	202,791	21
CPU-2	542,391	47,522	537,085	29
CPU-3	194657	33,085	121,682	16
DMA	10,898	2,062	10,215	0
LDPC	42,018	2,048	39,377	0
ECG	85,058	14,018	84,127	0
VGA	56,279	17,054	56,194	0

Table 3. Single-Design Optimization Results on DREAMPlace [11]

design (# cells)	PD stage	vanilla DREAMPlace [11] wns (ns)	TNS (ns)	# vios	total WL (um)	total power (mW)	GAN-Place enhanced (ours) wns (ns)	TNS (ns)	# vios	total WL (um)	total power (mW)
CPU-1 (202K)	global place	−2.05	−13498	19558	374130	200.1	−1.42	−10175	18038	3532961	190.8
	place opt	−1.74	−6197	13018	4034908	194.7	−1.45	−5975	12605	3765982	175.3
	clock opt	−0.30	−45.89	681	4163129	144.4	−0.26	−33.42	489	3920346	136.5
	route opt	−0.26	−22.4	464	4166459	144.3	−0.17	−19.03	412	4035806 (−3.0%)	139.3 (−3.5%)
CPU-2 (537K)	global place	−432.97	−5634543	48869	12382802	25142.4	−428.41	−5216911	44382	11067230	24647.2
	place opt	−608.91	−7218793	40780	12654907	13244.1	−606.4	−6867935	39059	11318429	11820.07
	clock opt	−0.20	−61.48	1726	17769476	488.1	−0.18	−45.12	1477	16317518	452.3
	route opt	−0.17	−45.83	1405	17765081	490.5	−0.13	−29.07	906	16150284 (−10.2%)	441.5 (−10.0%)
CPU-3 (121K)	global place	−2.13	−8437.48	11730	1711937	149.2	−1.88	−7763.08	10963	1637906	140.6
	place opt	−0.54	−164.78	2466	1439469	155.8	−0.48	−135.48	1948	1387534	150.9
	clock opt	−0.51	−37.68	414	1588135	141.9	−0.54	−33.45	350	1462754	138.0
	route opt	−0.49	−41.21	1207	1582822	143.0	−0.33	−36.56	972	1540096 (−2.7%)	139.1 (−2.7%)
LDPC (46K)	global place	−1.14	−1411.74	2184	1289738	225.8	−1.06	−1322.54	2076	1182512	216.3
	place opt	−0.25	−292.49	2192	1454863	255.5	−0.20	−221.88	1911	1416226	251.3
	clock opt	−0.20	−156.62	1897	1857624	255.4	−0.14	−92.70	1693	1713233	249.5
	route opt	−0.24	−198.72	1976	1878969	261.8	−0.15	−112.06	1783	1819348 (−3.2%)	256.9 (−1.9%)

GAN-Place optimizes the same design as in the database.

to establish the efficacy of employing generative adversarial learning for placement optimization. Subsequent sections will provide evidence of transfer learning experiments that further support our claims. Table 3 demonstrates the detailed optimization results, where we clearly observe that GAN-Place consistently outperforms vanilla DREAMPlace at each major PD stage across all four industrial and OpenCore benchmarks. Although the same design is used in the database during training, we still believe the achieved results are highly remarkable as *GAN-Place is NOT using any "memorization" technique* such as explicit net-matching, cell-alignment, and so on. The superior results are purely achieved by optimizing placement similarity scores via generative adversarial learning. Figure 9 further shows the bin-density map comparison between GAN-Place and *ICC2* on the design CPU-2 that shows the largest PPA improvements. We observe that although the underlying placements are visually different, the achieved bin-density maps are arguably similar, which demonstrates the effectiveness of the proposed CNN-based discriminator.

5.2 Transfer Learning on Unseen Designs Using DREAMPlace [11] and Xplace [13]

Due to the intrinsic characteristics of the PD implementation process, wherein almost each realization necessitates a long runtime for completion in industrial design flows, the employment of transfer learning is the most favored approach in leveraging ML models to optimize critical PPA metrics. In this article, we refer transfer learning as the paradigm that aims to harness the knowledge acquired from training designs into unseen ones to perform PPA optimization. As

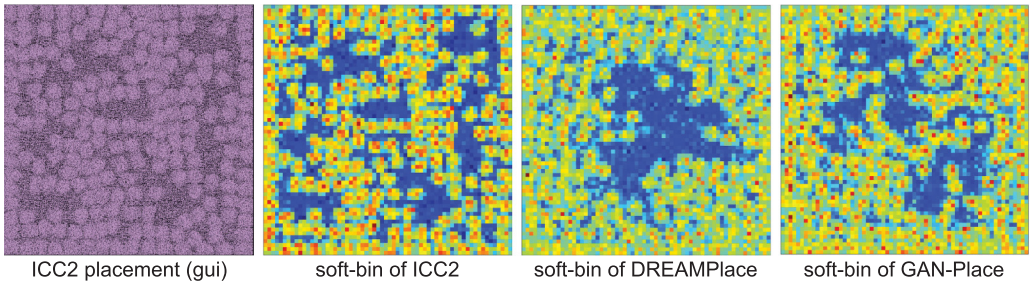

Fig. 9. Visualization of the proposed Soft-Bin technique in placements of ICC2 default placer, vanilla DREAMPlace, and the proposed GAN-Place on the AES benchmark. We observe GAN-Place produces a visually more similar bin-density map distribution to ICC2 than the vanilla DREAMPlace.

Table 4. Transfer Learning Optimization Results on DREAMPlace [11] and Xplace [13]

design (# cells)	global place metrics	vanilla DREAMPlace	DREAMPlace+GAN (ours)	vanilla Xplace [13]	Xplace+GAN (ours)	ICC2 default placer
AES (111K)	WL (um)	1946366	1755417 (−9.8%)	1410691	1315794 (−6.7%)	1393520
	WNS (ns)	−0.32	−0.31	−0.17	−0.14	−0.05
	TNS (ns)	−139.36	−125.01	−39.06	−33.95	−10.38
	# vios	3012	2830	1736	1649	928
	power (mW)	605.4	597.8	571.8	565.9	558.8
	runtime (min)	< 1	8	< 1	8	26
DMA (11K)	WL (um)	196988	189688 (−3.7%)	188629	176664 (−12.5%)	165069
	WNS (ns)	−0.19	−0.17	−0.18	−0.16	−0.12
	TNS (ns)	−35.06	−29.61	−15.91	−13.34	−6.48
	# vios	488	442	268	242	192
	power (mW)	31.1	30.9	31.0	30.8	30.9
	runtime (min)	< 1	5	< 1	5	19
ECG (85K)	WL (um)	1527118	1423859 (−6.8%)	1031059	965658 (−6.3%)	883419
	WNS (ns)	−2.24	−1.86	−1.18	−0.95	−1.03
	TNS (ns)	−6783.55	−6309.96	−1402.53	−1264.36	−728.90
	# vios	10862	9854	5678	5131	3801
	power (mW)	195.7	193.9	173.5	171.6	169.9
	runtime (min)	< 1	8	< 1	8	32
VGA (56K)	WL (um)	2202288	2043376 (−7.2%)	2023368	1895024 (−6.4%)	1756611
	WNS (ns)	−1.61	−1.33	−1.32	−1.29	−1.28
	TNS (ns)	−10001.56	−9803.32	−8092.88	−7295.62	−5165.15
	# vios	16251	15973	15981	15942	15970
	power (mW)	257	255	256	253	249
	runtime (min)	< 1	6	< 1	6	25

The "DREAMPlace+GAN" and "Xplace+GAN" columns are achieved by using different designs in the target database (i.e., each underlying design is unseen during GAN optimization). All metrics are reported by *Synopsys ICC2*.

aforementioned, the proposed GAN-Place framework does not require the target database contains the exact same design as the one being optimized by the GPU-accelerated placer. In the experiment, we demonstrate the enablement of transfer learning ability of GAN-Place using non-macro designs.

Table 4 demonstrates our transfer learning optimization results. Note that each design is unseen during the optimization. The target database is built by other non-macro designs in the benchmarks as shown in Table 2. We clearly observe that the proposed GAN-Place framework

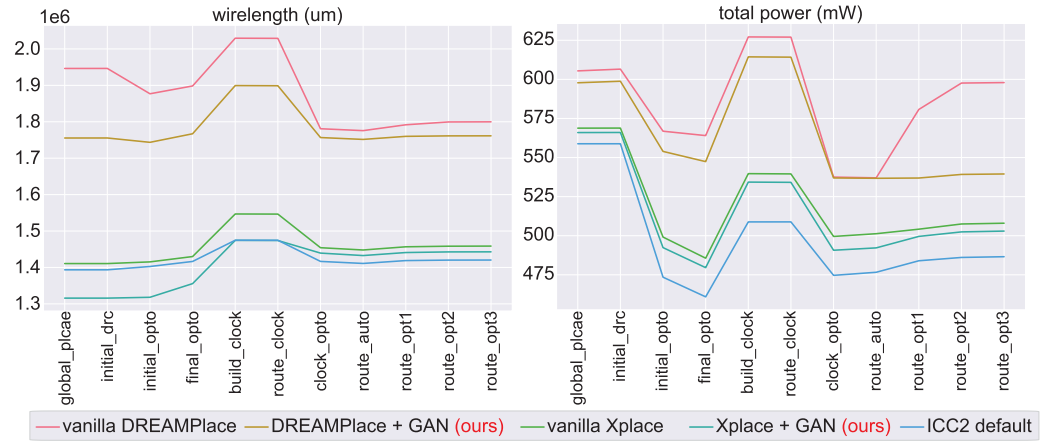

Fig. 10. Stage-by-stage wirelength and power metrics on AES. Note that each implementation starts from a different global placement, but all rely the exact same script to complete the full-chip design flow.

immediately improves each critical PPA metric at the global placement stage across every benchmark. Figure 10 further demonstrates the full-flow impact on wirelength and power, where we clearly observe that GAN-Place significantly improves not only the global placement metrics but also the post-route ones.

5.3 Discussion of Optimization Results

We believe the tremendous success of the proposed framework, GAN-Place, has three major implications: (1) "Placement style" can be transferred from one placer to another. This argument is supported by Table 3, which show that by using single-design optimization, GAN-Place can significantly improve the vanilla GPU placer on industrial benchmarks in consistent. (2) "Placement style" is more related to a placer itself than the designs being placed. This argument is validated by Table 4, where we observe that placement quality can not only be transferred in the same designs but also in completely different ones. (3) Without knowing the underlying algorithms or constraints, the "placement style" of a black-boxed placer can still be parameterized through generative adversarial learning.

Furthermore, in the experiments, we observe that GAN-Place not only improves the wirelength significantly but also introduces notable improvements in power. We think this is because by following the placement distribution of *ICC2* in terms of cell locations, GAN-Place will introduce less buffers and logic fixing than the vanilla DREAMPlace during many optimization steps throughout the flow, which effectively results in less power consumption. Finally, we would like to emphasize that the runtime difference between the vanilla DREAMPlace and the proposed GAN-Place is no more than 10 minutes across all benchmarks, which is *practically negligible* compared with the global placement runtime of *ICC2*.

5.4 Why Use Generative Adversarial Learning to Advance VLSI Placement?

The placement problem in PD has been a central focus of extensive scientific investigation for several decades. A multitude of research groups have devoted considerable resources to this area, necessitating continuous innovation in methodological approaches. This persistent development is essential to address the emerging complexities inherent in design paradigms that arise as a consequence of advancing technology scaling. Over the years, heuristic-based and analytical

placement methodologies have substantially advanced the VLSI placement domain; however, limited research has been conducted on the analysis of placement distributions across various placement algorithms and the subsequent utilization of such information for optimizing placement quality. In this study, we highlight the significance and applicability of such information for enhancing placement quality in an efficient manner, employing generative adversarial learning techniques.

Notably, the primary advantage of GAN-based learning lies in its ability to perform optimization without necessitating explicit objectives. This allows us to concentrate solely on the acquisition of high-quality designs for constructing the database. In our specific context, despite lacking knowledge of the proprietary algorithms employed by commercial tools, we successfully enhance the placement performance of open source placers to approach commercial-quality standards. We achieve this by guiding them to emulate the placement distribution in terms of netlist connectivity and bin-density maps, using the tool-verified, high-quality placements present in our database.

6 CONCLUSION

In this article, we have presented GAN-Place, which, as far as we know, is the first-ever learning-driven framework that improves an open source placer toward commercial-quality using generative adversarial learning. We showcase that GAN-Place can be easily integrated with state-of-the-art open source GPU-accelerated placers DREAMPlace and Xplace to significantly improves the achieved placement quality. In the experiments, we demonstrate that GAN-Place not only improves the optimization results in single-design optimization but also facilitates transfer learning to perform optimization on unseen designs. The main assumption of this work is that "placement style (quality)" is born with a placer itself, which can be parameterized and transferred to another placer through generative adversarial learning. We have provided comprehensive experiments and analyses to prove this assumption empirically. We believe this work shall open the gate for future endeavor on advancing PD using generative adversarial learning.

REFERENCES

[1] Anthony Agnesina, Kyungwook Chang, and Sung Kyu Lim. 2020. VLSI placement parameter optimization using deep reinforcement learning. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.

[2] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. 2008. NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 27, 7 (2008), 1228–1240.

[3] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. 2018. Replace: Advancing solution quality and routability validation in global placement. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 38, 9 (2018), 1717–1730.

[4] Xiang Gao, Yi-Min Jiang, Lixin Shao, Pedja Raspopovic, Menno E. Verbeek, Manish Sharma, Vineet Rashingkar, and Amit Jalota. 2022. Congestion and timing aware macro placement using machine learning predictions from different data sources: Cross-design model applicability and the discerning ensemble. In *Proceedings of the International Symposium on Physical Design*. 195–202.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* 30 (2017).

[7] Xu He, Yao Wang, Yang Guo, and Evangeline F.Y. Young. 2016. Ripple 2.0: Improved movement of cells in routability-driven placement. *ACM Trans. Des. Autom. Electr. Syst.* 22, 1 (2016), 1–26.

[8] Meng-Kai Hsu, Yi-Fang Chen, Chau-Chin Huang, Sheng Chou, Tzu-Hen Lin, Tung-Chieh Chen, and Yao-Wen Chang. 2014. NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 33, 12 (2014), 1914–1927.

[9] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, et al. 2021. Machine learning for electronic design automation: A survey. *ACM Trans. Des. Autom. Electr. Syst.* 26, 5 (2021), 1–46.

[10] Andrew B. Kahng. 2021. Advancing placement. In *Proceedings of the International Symposium on Physical Design*. 15–22.

[11] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2019. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.

[12] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2020. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 40, 4 (2020), 748–761.

[13] Lixin Liu, Bangqi Fu, Martin D.F. Wong, and Evangeline F.Y. Young. 2022. Xplace: An extremely fast and extensible global placement framework. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 1309–1314.

[14] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method. *ACM Trans. Des. Autom. Electr. Syst.* 20, 2 (2015), 1–34.

[15] Yi-Chen Lu, Wei-Ting Chan, Vishal Khandelwal, and Sung Kyu Lim. 2022. Driving early physical synthesis exploration through end-of-flow total power prediction. In *Proceedings of the ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD '22)*. IEEE, 97–102.

[16] Yi-Chen Lu, Jeehyun Lee, Anthony Agnesina, Kambiz Samadi, and Sung Kyu Lim. 2019. GAN-CTS: A generative adversarial framework for clock tree prediction and optimization. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '19)*. IEEE, 1–8.

[17] Yi-Chen Lu and Sung Kyu Lim. 2022. On advancing physical design using graph neural networks. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–7.

[18] Yi-Chen Lu, Siddhartha Nath, Vishal Khandelwal, and Sung Kyu Lim. 2021. Doomed run prediction in physical design by exploiting sequential flow and graph learning. In *Proceedings of the IEEE/ACM International Conference On Computer Aided Design (ICCAD '21)*. IEEE, 1–9.

[19] Yi-Chen Lu, Sai Pentapati, and Sung Kyu Lim. 2021. The law of attraction: Affinity-aware placement optimization using graph neural networks. In *Proceedings of the International Symposium on Physical Design*. 7–14.

[20] Yi-Chen Lu, Haoxing Ren, Hao-Hsiang Hsiao, and Sung Kyu Lim. 2023. DREAM-GAN: Advancing dreamplace towards commercial-quality using generative adversarial learning. In *Proceedings of the International Symposium on Physical Design*. 141–148.

[21] Yi-Chen Lu, Tian Yang, Sung Kyu Lim, and Haoxing Ren. 2022. Placement optimization via ppa-directed graph clustering. In *Proceedings of the ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD '22)*. IEEE, 1–6.

[22] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. 2021. A graph placement methodology for fast chip design. *Nature* 594, 7862 (2021), 207–212.

[23] Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z. Pan, Marilyn Wolf, and Jörg Henkel. 2021. Mlcad: A survey of research in machine learning for cad keynote paper. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* (2021).

[24] Dhruv Vashisht, Harshit Rampal, Haiguang Liao, Yang Lu, Devika Shanbhag, Elias Fallon, and Levent Burak Kara. 2020. Placement in integrated circuits using cyclic reinforcement learning and simulated annealing. *arXiv preprint arXiv:2011.07577* (2020).

[25] Ziyi Wang, Chen Bai, Zhuolun He, Guangliang Zhang, Qiang Xu, Tsung-Yi Ho, Bei Yu, and Yu Huang. 2022. Functionality matters in netlist representation learning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 61–66.

[26] Zhitao Ying et al. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems* (2018).

Received 21 April 2023; revised 18 August 2023; accepted 30 November 2023