DREAM-GAN: Advancing DREAMPlace towards Commercial-Quality using Generative Adversarial Learning

Yi-Chen Lu
yclu@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Haoxing Ren
haoxingr@nvidia.com
Nvidia
Austin, TX, USA

Hao-Hsiang Hsiao
thsiao@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Sung Kyu Lim
limsk@ece.gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

ABSTRACT

DREAMPlace is a renowned open-source placer that provides GPU-acceleratable infrastructure for placements of Very-Large-Scale-Integration (VLSI) circuits. However, due to its limited focus on wirelength and density, existing placement solutions of DREAMPlace are not applicable to industrial design flows. To improve DREAMPlace towards commercial-quality without knowing the black-boxed algorithms of the tools, in this paper, we present DREAM-GAN, a placement optimization framework that advances DREAMPlace using generative adversarial learning. At each placement iteration, aside from optimizing the wirelength and density objectives of the vanilla DREAMPlace, DREAM-GAN computes and optimizes a differentiable loss that denotes the similarity score between the underlying placement and the tool-generated placements in commercial databases. Experimental results on 5 commercial and OpenCore designs using an industrial design flow implemented by Synopsys ICC2 not only demonstrate that DREAM-GAN significantly improves the vanilla DREAMPlace at the placement stage across each benchmark, but also show that the improvements last firmly to the post-route stage, where we observe improvements by up to 8.3% in wirelength and 7.4% in total power.

CCS CONCEPTS

• **Hardware** → **Placement**; **Physical design (EDA)**.

KEYWORDS

placement optimization, generative adversarial learning

ACM Reference Format:
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 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3569052.3572993

1 INTRODUCTION

It is widely acknowledged that placement is the heart of every Physical Design (PD) flow, as the cell locations determined at this stage directly affect the on-chip interconnects and hence the capacitances and resistances induced, which are closely related to the Power, Performance, and Area (PPA) metrics achieved in the end of the design flow [9]. Driven by Moore's Law, modern Very-Large-Scale-Integration (VLSI) designs easily consist of millions of cells that are required to be placed on constrained layouts, which severely struggles Electronic Design Automation (EDA) tools to generate high-quality placements in a reasonable amount of runtime.

To overcome the runtime barrier of VLSI placement, DREAMPlace [11] and its variants [10] leverage Graph Processing Units (GPUs) to significantly improve the placement runtime by formulating CPU-intensive placement objectives into GPU-acceleratable CUDA kernels. Particularly, in the vanilla DREAMPlace, placement objectives of RePlace [3], a renowned analytical placer, are formulated as CUDA kernels and accelerated by the deep learning toolkit PyTorch [18], where a 34X speed up is achieved. However, in spite of the significant runtime improvement attained, the placement quality of DREAMPlace is not comparable to that of commercial tools' due to its limited objective focus of wirelength and density, making it hardly applicable to industrial design flows. To overcome this issue, in this paper, we present DREAM-GAN, the first-ever learning-driven placement optimization framework that directly improves DREAMPlace towards commercial-quality using generative adversarial learning. The key rationale is that even if we do not know the algorithms or constraints used by the black-boxed tools, we can quantify placement similarity between DREAMPlace-generated placements and tool-optimized placements using adversarial learning, and by optimizing the computed similarity scores, the distribution gap between them can be narrowed.

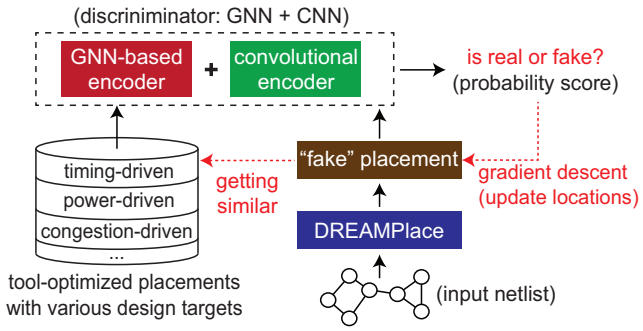

Figure 1: A high-level overview of DREAM-GAN that adopts generative adversarial learning to improve DREAMPlace. DREAMPlace improves its placement towards commercial-quality by optimizing discriminator's output.

ISPD '23, March 26–29, 2023, Virtual Event, USA

ACM ISBN 978-1-4503-9978-4/23/03.
https://doi.org/10.1145/3569052.3572993

Figure 1 presents a high-level overview of DREAM-GAN, where the vanilla DREAMPlace is considered as a "generator" whose goal is to generate placements that follow similar distributions as the tool-optimized ones in the database (which are optimized for different PPA objectives with various placement parameters). To achieve this, a discriminator built upon Convolutional Neural Networks (CNN) and Graph Neural Networks (GNN) is developed to quantify the placement similarity between the two "types (or styles)" of placement (i.e., DREAMPlace-generated and tool-optimized). The goal of the discriminator is to make correct judgements in deciding whether its input originates from commercial databases or DREAMPlace, whereas aside from wirelength and density optimization, one of the objectives of DREAMPlace in our settings is to "fool" the discriminator by generating placements that are similar to the ones in the database, which effectively forms an adversarial setting.

The greatest strength of DREAM-GAN is that it enables DREAMPlace to optimize the underlying placement towards a tool-verified (and optimized) direction without explicitly knowing commercial tools' black-boxed algorithms. Although DREAMPlace leverages fundamentally different algorithms compared with the tools, in this work, we demonstrate that the placement quality of commercial placers can be efficiently transferred to DREAMPlace with the proposed framework DREAM-GAN.

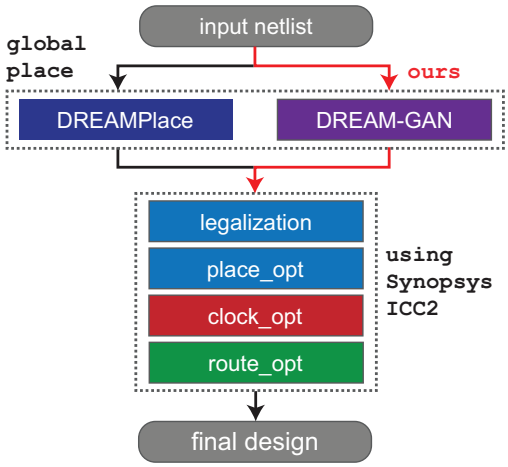

Figure 2: Illustration of integrating DREAM-GAN with an industrial PD flow implemented by Synopsys ICC2.

2 RELATED WORKS AND MOTIVATIONS

Analytical placers [2, 3, 7] have brought tremendous success to the semiconductor industry in the last decade [9]. Nevertheless, recent advancements in ML Theory and its applications have further pushed modern placers to an unprecedented frontier [8]. 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, [12, 14] 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 (DSE).
- 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. [15] 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 in order to improve the overall PPA metrics. However, the framework in [15] 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 [16] 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 [17] 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 [20] further combined simulated annealing and RL in a cyclic fashion to conduct iterative placement optimization. Nonetheless, the runtime of modern VLSI placements easily takes from hours to days, which greatly limits the usage of RL-driven optimization algorithms.

Unlike the previous works presented above, in this work, we take a brand new approach to perform VLSI placement optimization using generative adversarial learning [5]. In the realm of PD, previous work [13] 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 [13], in this paper, we aim to leverage GAN-based models to demystify commercial black-boxed placers so as to improve DREAMPlace [11], a renowned open-source placer, to commercial-quality. Particularly, we consider DREAMPlace 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 profound database containing commercial-quality placements with different objectives (e.g., timing-driven, power-driven, etc.) using *Synopsys ICC2*, an industry-leading commercial tool. We then develop a discriminator that serves as a teacher to improve DREAMPlace by

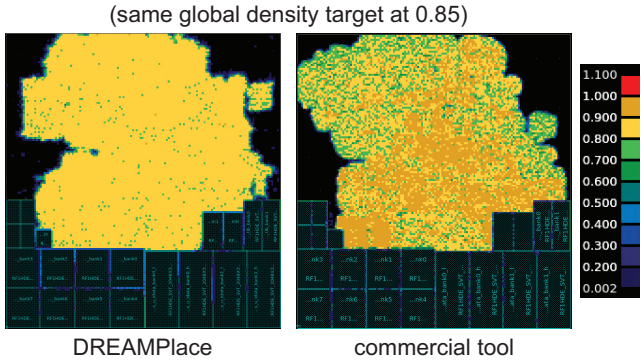

Figure 3: Comparison of cell density maps between DREAMPlace and Synopsys ICC2 under the same global placement density target. It is observed that the commercial tool has extra intelligence on where to locally aggregate or spread out cells in order to optimize crucial PPA metrics while satisfying the global density constraints.

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-generated placements will be updated through gradient descent, which effectively become more similar to the tool-generated placements in the database.

Finally, Figure 2 shows a graphical illustration of integrating the proposed framework DREAM-GAN into an industrial design flow implemented by *Synopsys ICC2*. To perform fair comparisons between the vanilla DREAMPlace and the proposed DREAM-GAN, we use the same ICC2 parameters and seeds to implement the rest of the design flow after global placement. In the experiments, we show that DREAM-GAN outperforms DREAMPlace in critical PPA metrics at each major stage of the flow.

3 DREAM-GAN OVERVIEW

It is widely acknowledged that generative adversarial learning is a promising paradigm that effectively captures complicated distributions using generative and discriminative models which 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 paper we consider DREAMPlace, a differentiable placement system, as a generator whose goal is to generate placements that follow similar distributions as the ones in the 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

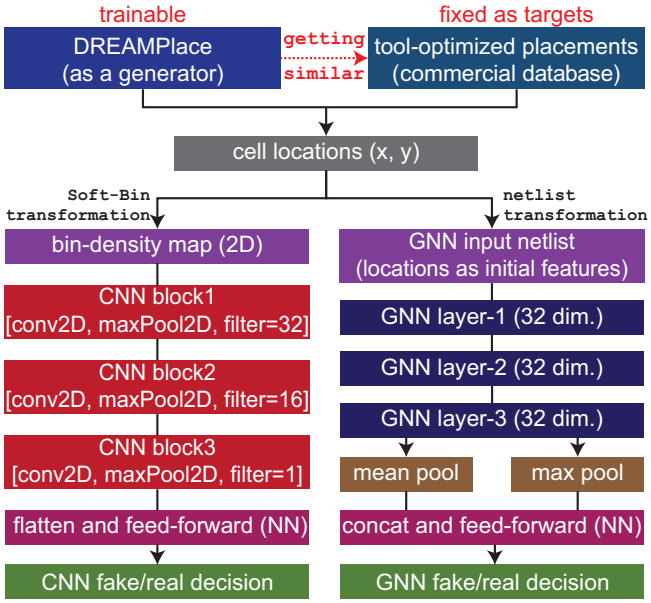

Figure 4: Detailed architecture of DREAM-GAN. Given placements in terms of (x, y) locations, the CNN-based and GNN-based discriminators leverage bin-density maps and netlist graphs, respectively, to determine the placement origins (i.e., either from DREAMPlace or from the commercial database).

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

encoding is shown in Figure 3. We observe that under the same global density target, the commercial tool has extra intelligence on locally aggregating or loosening cells in order 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 [16]. The observation shown in Figure 3 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 such encoding as they are well-known for grid signals (e.g., images) classification.

Figure 4 illustrates the detailed architecture of DREAM-GAN. As aforementioned, the key idea of DREAM-GAN is to improve the placement quality of DREAMPlace to commercial-quality through generative adversarial learning. To achieve this goal, we first build a commercial database containing various tool-optimized placements with different objectives, including performance-driven placements,

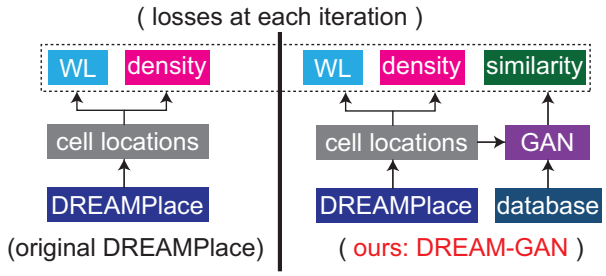

Figure 5: Difference in objectives between DREAMPlace [11] and the proposed DREAM-GAN at each placement iteration. The similarity loss encourages the placements from DREAMPlace follow the distributions as the ones in the database.

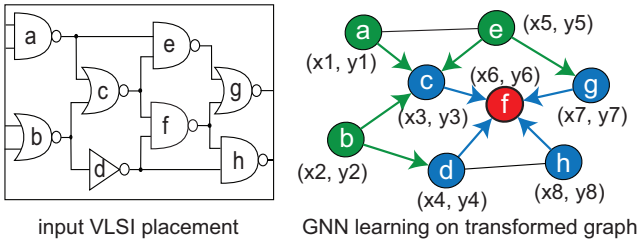

Figure 6: Illustration of netlist transformation and GNN learning on node f, where cell locations (x,y) are taken as initial node features. The blue nodes denote the 1-hop neighbors, the green nodes denote the 2-hop neighbors, and the arrows denote the message passing directions to compute the final representations of the target node colored in red.

low-power placements, routability-driven placements, etc., which can be obtained by permuting placement parameters offered by *Synopsys ICC2*. Table 1 shows the parameters we utilize in this work to generate commercial-quality placements. As shown in the table, the combination of different parameters forms a high-dimensional space, leading to a variety of tool-optimized placements that have distinct PPA objectives, and thus resulting in a diversified database. During placement iterations, DREAM-GAN guides DREAMPlace to directly improve cell locations by optimizing the computed similarity scores so that the achieved placements can follow a similar distribution to the ones in the database.

To directly update cell locations by optimizing similarity scores, the inputs of DREAM-GAN have to be differentiable with respect to locations. GNNs naturally satisfy this requirement as they are end-to-end models and cell locations are taken as node features. However, normally, the computation of a bin-density map is not differentiable as it is not continuous. To overcome this issue, in this paper, we propose a differentiable bin-density map transformation technique named "Soft-Bin", which transforms cell locations into "probabilistic" 2D bin-density maps using a softmax function so that the CNN-computed gradients can be directly used to update cell locations in each placement iteration.

Finally, the key difference in objectives between DREAM-GAN and the vanilla DREAMPlace is illustrated in Figure 5. In each placement iteration of DREAM-GAN, aside from optimizing the existing objectives of DREAMPlace: wirelength and density, it computes and optimizes a similarity loss using the architecture shown in Figure 4 to guide the generated placements towards commercial-quality.

4 ALGORITHMS

In this section, we will delve into the details of each transformation technique and main components of the proposed framework DREAM-GAN as shown in Figure 4, which include GNN learning for cell locations encoding, the proposed Soft-Bin algorithm that generates differentiable 2D bin-density map, and the objectives of the CNN and GNN-based discriminator networks.

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. In the realm of PD, such encoded representations have been leveraged to successfully solve many tasks that are once considered extremely hard to solve [19]. In this paper, 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 damage the quality of graph learning as redundant edges limit the expressiveness of GNNs [21]. To overcome this issue, in this work, we adopt the netlist transformation technique proposed in [16] 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 6 shows the illustration of the graph learning process in the proposed DREAM-GAN framework. Given an input placement, GNNs are leveraged to perform node representation learning by using cell locations as initial node features. Considering the benefits of runtime and computational memory, in this work, we adopt the message passing scheme of GraphSAGE [6] to perform node representation learning on a target node v as (which is repeated for each cell in the design):

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

where k denotes the transformation level, σ denotes the sigmoid function, $Neigh(v)$ denotes the neighbors of node v, W_k^{agg} and W_k^{proj} denote the aggregation and projection matrices at the k-th layer of the GNN model. In this work, the GNN-encoded node embeddings have 32 dimensions and the message passing transformation has a total number of three levels, which is subject to the number of GNN layers in the framework as shown in Figure 4. After the node representation learning, we obtain the final graph-level encoding g by performing mean and max pooling of the transformed

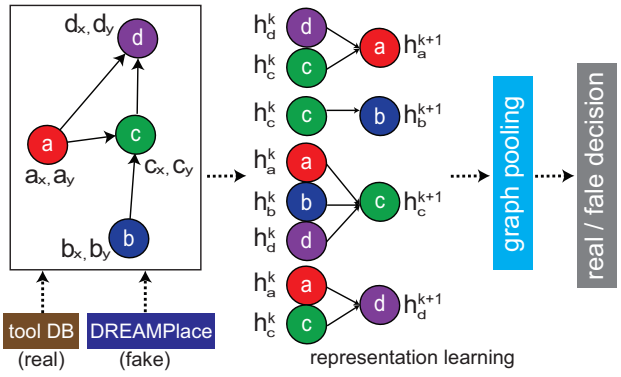

Figure 7: GNN-based discriminator that outputs "fake/real" decision using node representation learning based on different input placement sources.

node embeddings $\{h_v^3\ \forall v \in V\}$ as:

$$g = \text{concat}\left[mean\_pooling\left(\{h_v^3\}\right), max\_pooling\left(\{h_v^3\}\right)\right], \quad (2)$$

which is taken as the input of the proceeding feedforward neural networks to make the final decision of whether the underlying placement is similar to the ones in the database in terms of the GNN-encoded representations.

4.2 GNN-Based Discriminator

Figure 7 demonstrates the adversarial learning conducted by GNNs in the proposed framework DREAM-GAN. By applying node representation learning on the input placement, which is either from the commercial database or from the DREAMPlace engine, we train the GNN model to differentiate various placement sources based on netlist connectivity and assigned cell locations. Note that the goal of DREAMPlace 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 3, 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: Transforming Cell Locations to Differentiable 2D Bin-Density Map

As aforementioned, the key motivation behind using bin-density distribution as one of the placement similarity metrics is shown in Figure 3, where we observe that commercial placer has extra intelligence on where to locally aggregate or loosen cells to improve the overall PPA metrics, while the vanilla DREAMPlace naively makes every local bin-density satisfies the global density constraints.

It should first be mentioned that a normal 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 the assignment of locating a cell to a bin is purely based on the cell's locations (i.e., deterministic). Although this deterministic way to generate 2D bin-density maps from cell locations is "exact" and accurate, a probabilistic bin-density calculation method is needed in our learning-driven framework in order to update the cell locations by using the gradients calculated from the similarity loss. To achieve this, we develop a differentiable bin-density transformation technique that is termed Soft-Bin as shown in Algorithm 1.

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: $M \in R^{|V|\text{x}|V|}$: differentiable 2D 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: $V' \leftarrow filter\{i*num\_w \le X_v < (i+1)*num\_w \forall v \in V\}$
8: $V' \leftarrow filter\{j*num\_h \le Y_v < (j+1)*num\_h \forall v \in V'\}$
9: **for** $v \in V'$ **do**
10: $b \leftarrow M[i][j]$
11: $neigh\_bins \leftarrow$ get adjacent and 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: $prob\_vec \leftarrow softmax\left(dist\_vec^{-1}\right)$
17: $area\_vec \leftarrow area_v * prob\_vec$ ▷ expected values
18: update M by $area\_vec$ ▷ add area of each bin to M
19: $M \leftarrow \frac{M}{bin\_area}$ ▷ convert expected area to expected density

The key idea behind Algorithm 1 is that instead of deterministically assigning each cell to an exact bin based on its (x, y) location, we can probabilistically distribute the cell area to its neighboring bins using the softmax function, which enables cell locations to be updated along with any maximization or minimization operation of the computed bin-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 as shown in Figure 4.

The proposed Soft-Bin algorithm works as follows. In Lines 1–4, we initialize the bin density map M based on the input specifications. The cells corresponding to each bin can be obtained using Lines 7–8. 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 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 M by the expected area contribution of each cell in the design, we obtain the final density of each bin using Line 19.

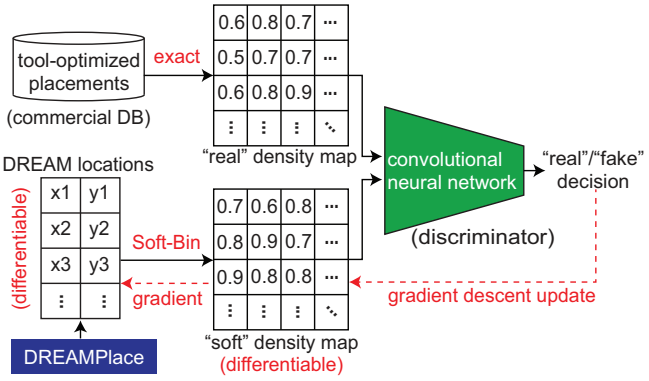

Figure 8: Illustration of our CNN-based discriminator using the Soft-Bin transformation technique that transforms cell locations to differentiable bin-density maps. The goal of DREAMPlace is to generate placements that have similar bin-density maps as the ones in the database.

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 8. Note that the key motivation for using CNNs to encode and differentiate different placements is originated from Figure 3, where we clearly observe that the commercial tool is having additional intelligence on locally spreading or coarsening cells for PPA optimization which DREAMPlace is lacking of. Therefore, to advance DREAMPlace towards 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 8, 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 DREAM-GAN Training

The key difference between the "training" of the vanilla DREAMPlace and our proposed DREAM-GAN is shown in Figure 5. Beyond the conventional placement objectives which are wirelength and density, we consider DREAMPlace 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:

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

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 existing DREAMPlace objectives as

$$L = L_{WL} + \lambda_1 L_{density} + \lambda_2 L_{sim}, \tag{4}$$

where L_{WL} and $L_{density}$ denote the wirelength and density objectives computed from the vanilla DREAMPlace, $\{\lambda\}$ denote the hyper-parameters for loss weighting. At each placement optimization, DREAMPlace will optimize Equation 4 to improve the underlying placement towards commercial-quality.

5 EXPERIMENTAL RESULTS

In this paper, we validate the proposed framework DREAM-GAN on 5 commercial CPU and OpenCore benchmarks under the TSMC 28*nm* technology node. In the experiments, we perform head-to-head comparisons between DREAMPlace [11] and DREAM-GAN at each major stage of an industrial PD flow implemented by *Synopsys ICC2*. The commercial database is generated by randomly sampling the parameters in Table 1, where for each benchmark, we complete 50 commercial runs to obtain the tool-optimized placements with different PPA objectives. The entire DREAM-GAN framework is implemented upon the infrastructure of DREAMPlace using PyTorch. For each benchmark, the runtime difference between DREAM-GAN and DREAMPlace is less than 2 minutes on a CPU-only machine.

5.1 DREAM-GAN Optimization Results

To perform head-to-head comparisons between DREAMPlace [11] and DREAM-GAN, for both approaches, we use the exact same seeds and parameters offered by *ICC2* to implement the full-chip design after global placement. An illustration of the comparison flow is shown in Figure 2. Table 2 demonstrates the detailed optimization results, where we clearly observe that across all the benchmarks, DREAM-GAN consistently outperforms DREAMPlace at each major PD stage in critical PPA metrics. Specifically, at our largest benchmark (i.e, CPU-2 with 580K cells), we observe that DREAM-GAN significantly improves the *post-route* wirelength by 8.3% and the total power by 7.4%. Among all the benchmarks, DREAM-GAN delivers impressive average improvements of 4.34% in post-route wirelength, and 3.16% in post-route total power. We believe the experimental results have empirically proved that DREAM-GAN successfully advances DREAMPlace towards commercial-quality. Finally, we want to emphasize that although DREAM-GAN requires a pre-built database to perform similarity-based learning, it does NOT use any "memorization" technique such as net-matching, cell-alignment etc. to perform the optimization, as even for the same benchmark, the number of cells or nets may vary from run to run in the database due to the realization of various PPA objectives. Our superior optimization results are purely achieved by optimizing placement similarity scores via generative adversarial learning, which does not require the number of cells or nets to be matched between the designs that DREAM-GAN optimizes and the ones in the database.

Table 2: Detailed PPA comparison results between DREAMPlace [11] and DREAM-GAN at each major PD stage. The comparison flow is illustrated in Figure 2. In this work, we use Synopsys ICC2 to perform the entire PD except for the global placement that is either performed by DREAMPlace (left column) or DREAM-GAN (right column). The runtime difference in global placement between the two approaches is no more than 2 minutes across all commercial and OpenCore benchmarks.

design (# cells)	PD stage	DREAMPlace [11] wns (ns)	TNS (ns)	# vios	total WL (um)	total power (mW)	DREAM-GAN (ours) wns (ns)	TNS (ns)	# vios	total WL (um)	total power (mW)
CPU-1 (220K)	global place	-2.05	-13498	19558	374130	200.1	-1.46	-10601	18425	3546577	193.5
	place opt	-1.74	-6197	13018	4034908	194.7	-1.52	-6024	12697	3870333	179.6
	clock opt	-0.30	-45.89	681	4163129	144.4	-0.24	-34.28	473	4041709	140.1
	route opt	-0.26	-22.4	464	4166459	144.3	-0.18	-21.11	446	4050908 (-2.7%)	141.9 (-1.6%)
CPU-2 (580K)	global place	-432.97	-5634543	48869	12382802	25142.4	-432.98	-5324323	45644	11110278	25098.2
	place opt	-608.91	-7218793	40780	12654907	13244.1	-608.74	-7202230	40544	11493278	12431.0
	clock opt	-0.20	-61.48	1726	17769476	488.1	-0.23	-48.28	1505	16305060	455.0
	route opt	-0.17	-45.83	1405	17765081	490.5	-0.14	-28.61	942	16287654 (-8.3%)	454.2 (-7.4%)
CPU-3 (121K)	global place	-2.13	-8437.48	11730	1711937	149.2	-1.96	-8057.19	11435	1691131	147.8
	place opt	-0.54	-164.78	2466	1439469	155.8	-0.48	-138.74	1981	1413154	153.1
	clock opt	-0.51	-37.68	414	1588135	141.9	-0.57	-32.98	359	1518498	137.7
	route opt	-0.49	-41.21	1207	1582822	143.0	-0.35	-36.24	1023	1520481 (-3.9%)	138.9 (-2.9%)
VGA (57K)	global place	-2.2	-13999.49	16630	2418386	345.5	-1.65	-8057.19	11435	1691131	342.2
	place opt	-0.07	-2.06	188	1426981	279.8	-0.10	-2.55	171	1456516	276.5
	clock opt	-0.16	-7.46	441	1579559	327.8	-0.14	-5.37	398	1536218	322.4
	route opt	-0.17	-13.73	1712	1586940	333.5	-0.14	-7.06	1050	1542569 (-2.8%)	329.1 (-1.3%)
LDPC (46K)	global place	-1.14	-1411.74	2184	1289738	225.8	-1.10	-1331.30	2048	1233014	219.5
	place opt	-0.25	-292.49	2192	1454863	255.5	-0.21	-217.76	-217.76	1390693	248.6
	clock opt	-0.20	-156.62	1897	1857624	255.4	-0.16	-98.47	1757	1785355	248.4
	route opt	-0.24	-198.72	1976	1878969	261.8	-0.18	-123.94	1846	1803729 (-4.0%)	255.0 (-2.6%)

6 DISCUSSION: WHY DREAM-GAN WORKS?

We believe the success of our framework DREAM-GAN immediately implies an important message. That is, just as "image stlyes" can be transferred between different domains in recent GAN applications, "placement stlyes" can be transferred among different placers using generate adversarial learning without knowing the underlying algorithms or constraints. Particularly, in this work, we leverage bin-density maps along with netlist graphs to quantify "placement styles" of different placers, which are encoded by CNN-based and GNN-based discriminators, respectively. Figure 9 shows the bin-density map comparison between DREAM-GAN and Synopsys ICC2, where we observe that the achieved bin-density maps are indeed similar to each other, demonstrating the effectiveness of the proposed framework.

Finally, in Table 2, we observe that DREAM-GAN not only improves the wirelength significantly, but also introduces notable improvements in power and timing. We think this is because by following the placement distribution of *ICC2* in terms of cell locations, DREAM-GAN introduces less buffers and delay fixing procedures than the vanilla DREAMPlace during many optimization steps throughout the flow, which effectively leads to less power consumption and better timing results.

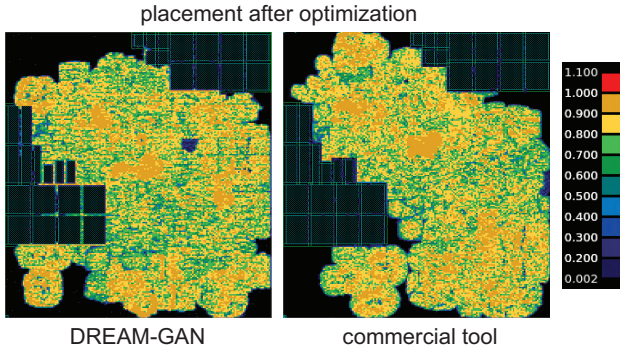

Figure 9: Density map comparison between the proposed DREAM-GAN and Synopsys ICC2 on the CPU-2 benchmark.

7 CONCLUSION AND FUTURE WORK

In this paper, we have presented DREAM-GAN which is the first-ever learning-driven framework that improves DREAMPlace, an open-source placer, towards commercial-quality using generative adversarial learning. In the experiments, we show that DREAM-GAN not only immediately improves major PPA metrics at the placement stage, but also demonstrate that the improvements last firmly to the post-route stage. The main assumption of this work is that "placement style (quality)" is an inherent attribute of a placer, which can be parameterized and transferred to another placer using generative adversarial learning. Although this assumption is proved empirically in this paper, in the future, we aim to provide more rigorous analyses and further enable effective transfer learning across different designs. Finally, we believe this work shall present new directions in advancing VLSI placement.

REFERENCES

[1] A. Agnesina, K. Chang, and S. K. Lim. Vlsi placement parameter optimization using deep reinforcement learning. In Proceedings of the 39th International Conference on Computer-Aided Design, pages 1–9, 2020.

[2] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(7):1228–1240, 2008.

[3] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang. Replace: Advancing solution quality and routability validation in global placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38(9):1717–1730, 2018.

[4] X. Gao, Y.-M. Jiang, L. Shao, P. Raspopovic, M. E. Verbeek, M. Sharma, V. Rashingkar, and A. Jalota. 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 2022 International Symposium on Physical Design, pages 195–202, 2022.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. Communications of the ACM, 63(11):139–144, 2020.

[6] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

[7] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang. Ntuplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(12):1914–1927, 2014.

[8] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, et al. Machine learning for electronic design automation: A survey. ACM Transactions on Design Automation of Electronic Systems (TODAES), 26(5):1–46, 2021.

[9] A. B. Kahng. Advancing placement. In Proceedings of the 2021 International Symposium on Physical Design, pages 15–22, 2021.

[10] P. Liao, S. Liu, Z. Chen, W. Lv, Y. Lin, and B. Yu. Dreamplace 4.0: timing-driven global placement with momentum-based net weighting. In 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE.

[11] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 40(4):748–761, 2020.

[12] Y.-C. Lu, W.-T. Chan, V. Khandelwal, and S. K. Lim. Driving early physical synthesis exploration through end-of-flow total power prediction. In 2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD), pages 97–102. IEEE.

[13] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim. Gan-cts: A generative adversarial framework for clock tree prediction and optimization. In 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 1–8. IEEE.

[14] Y.-C. Lu, S. Nath, V. Khandelwal, and S. K. Lim. Doomed run prediction in physical design by exploiting sequential flow and graph learning. In 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), pages 1–9. IEEE.

[15] Y.-C. Lu, S. Pentapati, and S. K. Lim. The law of attraction: Affinity-aware placement optimization using graph neural networks. In Proceedings of the 2021 International Symposium on Physical Design, pages 7–14, 2021.

[16] Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren. Placement optimization via ppa-directed graph clustering. In 2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD), pages 1–6. IEEE, 2022.

[17] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. Nature, 594(7862):207–212, 2021.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 2019.

[19] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf, and J. Henkel. Mlcad: A survey of research in machine learning for cad keynote paper. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021.

[20] D. Vashisht, H. Rampal, H. Liao, Y. Lu, D. Shanbhag, E. Fallon, and L. B. Kara. Placement in integrated circuits using cyclic reinforcement learning and simulated annealing. arXiv preprint arXiv:2011.07577, 2020.

[21] Z. Wang, C. Bai, Z. He, G. Zhang, Q. Xu, T.-Y. Ho, B. Yu, and Y. Huang. Functionality matters in netlist representation learning. In Proceedings of the 59th ACM/IEEE Design Automation Conference, pages 61–66, 2022.