Layout Pattern Generation and Legalization with Generative Learning Models

Xiaopeng Zhang
The Chinese University of Hong Kong
NT, Hong Kong SAR
xpzhang@cse.cuhk.edu.hk

James Shiely
Synopsys Inc.
CA, USA
Jim.Shiely@synopsys.com

Evangeline F.Y. Young
The Chinese University of Hong Kong
NT, Hong Kong SAR
fyyoung@cse.cuhk.edu.hk

ABSTRACT

VLSI layout patterns plays an important role in various research in Design for Manufacturing (DFM), such as optical proximity correction, lithography hotspot detection and so on. However, a large and diverse layout pattern library is usually not available during development stages due to the long and iterative technology life cycle, which brings potential difficulties to related research and slows down the development process. Although some previous works managed to enlarge pattern libraries with different solutions, there are still many challenges on generating complex DRC-clean two-dimensional patterns with specific styles. To address this problem, we explored the capability of generative machine learning models to learn the inherent distribution of a given set of non-trivial layouts for synthesizing diverse and realistic layout patterns with little manual guidance. For this purpose, we propose *CUP*, the *CU P*attern generation and legalization framework, which consists of two learning-based modules for pattern topology generation and design rule legalization respectively. Experiments show that CUP can generate diverse legal layout patterns which are comparable to actual design layouts in terms of resemblance in style and validity.

1 INTRODUCTION

VLSI layout patterns are significant resources for flows of various design for manufacturability (DFM) research, such as optical proximity correction (OPC) [7, 10, 13, 15, 26], layout hotspot detection [5, 27, 30, 33, 34], pattern matching [4], lithography simulation [16, 17, 31] and so on. However, VLSI layout pattern libraries are very often not available for research or testing due to the long and iterative technology life cycle, especially in the initial stage, which may slow down the technology node development [10]. Moreover, for learning-based methods which are widely adopted to lithography and hotspot detection applications, a large number of layout patterns are needed for training. Non-diversified pattern libraries will result in underlying fallacies and pitfalls in these learning-based works [21]. Therefore, research on synthesizing diversified VLSI layout pattern libraries is important for DFM solutions.

Some efforts have been made to enlarge existing layout libraries and to increase layout diversity. The works [21, 22, 32] gain new patterns by rotation, flipping and moving edges based on a few existing patterns for enhancing hotspot detection. The work [10] guides a random generator to explore new territories of the design space. These methods are coupled with complex manual guidance and can hardly increase the layout diversity due to its deterministic strategy. Some works build diversified pattern libraries based on learning-based techniques, of which one representative work is [28]. The authors proposed a pattern generation framework with transforming convolutional auto-encoder, and the work is applied only to unidirectional on-track shapes. It is still a big challenge to develop diversified and DRC-clean pattern libraries of a large number of realistic patterns for complex two-dimensional layouts.

For this target of two-dimensional pattern generation, we propose *CUP*, a novel framework based on generative learning techniques for pattern topology generation and legalization. In the generation stage, we develop a variational convolutional auto-encoder (VCAE) architecture, mapping existing pattern topologies into latent vectors on which Gaussian perturbation is applied to generate new topologies with realistic styles. After that, we will address the legality issue of the generated patterns, which is another challenging task. Previous pattern generation works filter out patterns with DRC violations [20], which will reduce the efficiency of the pattern generation process. In this paper, based on conditional generative adversarial nets (CGAN) [18], we design LegalGAN, a legalization model that transforms generated samples from blurry patterns to smooth ones, which greatly reduces DRC violation risks.

Finally, considering that we target at generating realistic layout patterns which should closely resemble actual IC layouts besides being legal, we design a pattern style detection tool based on an adversarial auto-encoder capturing the layout style in both the pattern space and the latent vector space.

The main contributions of this paper are listed as follows:

- We propose a novel two-stage generative learning-based pattern generation framework including pattern topology generation and legalization, which has the capability of learning inherent distributions of two-dimensional layouts and generate realistic patterns with little manual guidance.
- A VCAE architecture is designed for efficient generation of realistic pattern topologies by Gaussian perturbation.
- A pattern legalization model based on CGAN is developed to legalize the generated pattern topologies to reduce DRC violation risks with little manual guidance.

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 14202218).

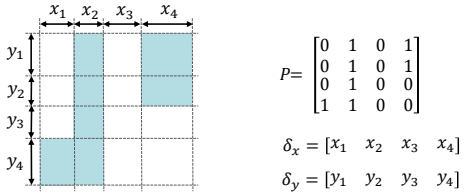
ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415607

- Based on an adversarial auto-encoder, a pattern style detection tool is designed to check pattern styles and filter out unrealistic generated patterns.
- Experimental results demonstrate that our framework has the capability of generating realistic legal patterns with high pattern diversity.

2 PRELIMINARIES

2.1 Squish Pattern Representation



(a) Splitting original pattern. **(b) Squish topology and geometry.**

Figure 1: Squish pattern representation.

Squish pattern [8] is an efficient representation in which a layout is divided into pattern topology and geometric information. Squish patterns are used in our pattern generation and legalization process, because squish pattern representation is lossless, storage-efficient and conducive to convolutional neural network models. It allows us to avoid a lot of pixel-level computations.

As shown in Figure 1a, squish pattern representation splits a layout into grids by a set of scan lines that walk along the edges of the polygons. These grids are then described by a topology matrix P and two geometry vectors δ_x and δ_y. As depicted in Figure 1b, each element of P is either a 0 or 1 that indicates space or polygon respectively, while elements of δ_x and δ_y represent the respective column widths and row heights. To make squish patterns compatible with convolutional neural networks that require matrices or tensors as input, all layout patterns are split into squish patterns with the same size by the adaptive method of [29].

2.2 Pattern Generation Problem

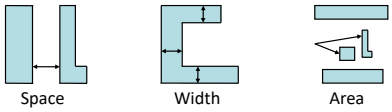


Figure 2: Layout geometry concepts for DRC rules.

In this subsection, we introduce some geometric concepts and metrics to define the pattern generation problem. Our target is to design a tool to generate diverse and realistic layout patterns that are both legal and closely resemble actual IC layouts. Figure 2 shows some common geometric concepts to describe layout design rules. *Space* represents the distance between two adjacent polygons, *Width* measures the size of a shape in one direction, and *Area* denotes the area of a polygon. A layout pattern is *legal* if and only if all these measurements are not less than some given thresholds. With appropriate values of the thresholds, a legal pattern will have much reduced DRC violation risk.

We also need to discuss the concept of diversity. In [28], the term *pattern complexity* of a given layout in squish pattern representation, denoted by a pair of values (c_x, c_y), is computed as the numbers of scan lines subtracted by one along the x-axis and the y-axis respectively. Based on this, the *pattern diversity* (denoted by H) can be defined by Shannon Entropy [24] to measure the diversity of a given pattern library as follows. Note that a larger H means that the patterns in the library are more diversified.

Definition 2.1. (Pattern Diversity) Pattern diversity, denoted by H, is the Shannon entropy of the pattern complexity sampled from the pattern library, as computed in Equation 1,

$$H = -\sum_i \sum_j P(c_{xi}, c_{yj}) \log P(c_{xi}, c_{yj}), \tag{1}$$

where $P(c_{xi}, c_{yj})$ is the probability that a pattern with complexities of (c_{xi}, c_{yj}) is sampled from the pattern library.

With the above definitions, the pattern generation problem can be described as follows.

Problem 1. *(Pattern Generation) Given a set of actual IC layout patterns and design rules, the objective of pattern generation is to generate a legal pattern library such that the pattern diversity and the number of unique realistic patterns in the library are maximized.*

3 OVERVIEW

Generating non-trivial and two-dimensional layout patterns which are diverse, legal and realistic is a challenging task for learning-based methods. We address this problem by a flow in which pattern generation, legalization and filtering are applied respectively.

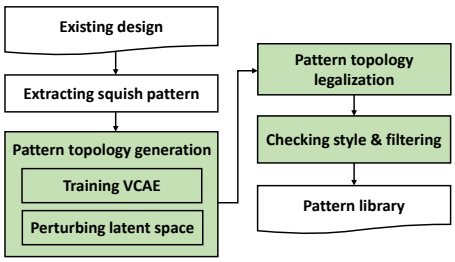


Figure 3: An overview of our CUP.

An overview of our framework is shown in Figure 3. First, we extract the squish patterns from the existing designs, where the layouts are divided into two compressed parts, topology and geometry, as described in Section 2. Our model will learn and generate new layout topologies while the geometric part will be maintained. With a dataset of pattern topologies, a learning-based model is trained to generate new topologies which are diverse and realistic by perturbation in the latent space. A legalization tool based on CGAN is then applied to make the generated topologies smooth to reduce DRC violation risk. We can then obtain new layouts in squish patterns by concatenating the generated topologies and the geometry information. Finally, a pattern style detection tool is built to check if the newly generated patterns are realistic, and those non-realistic patterns can be filtered out.

4 PATTERN TOPOLOGY GENERATION

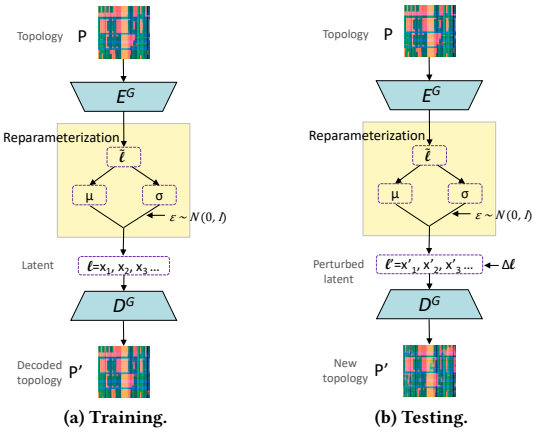


Figure 4: Topology generation in (a) training phase and (b) testing phase.

4.1 Variational Convolutional Auto-Encoder

In this section, our variational convolutional auto-encoder (VCAE) architecture is proposed that aims at generating diverse and realistic pattern topologies. VCAE is derived from the original variational auto-encoders (VAEs) [14] which is a group of densely connected encoder-decoder neural networks learning the distribution of the input for generating new ones by random sampling. In VAEs, the encoder maps the input samples into a latent Gaussian space, where new latent vectors are sampled by reparameterization, which are then reconstructed back to new samples by the decoder. Besides the schemes of VAEs, our VCAE employs additional convolutional layers and residual blocks [11] for capturing complex topology distribution of two-dimensional layouts.

Our VCAE architecture has an encoder-reparameterization-decoder pipeline for feature learning and pattern reconstruction, as depicted in Figure 4. In the training stage shown in Figure 4a, the encoder E^G of the VCAE maps a pattern topology P into a Gaussian space with mean μ and variance σ^2. In this Gaussian space, the reparameterization block samples a new latent vector l by $l = \mu + \sigma \odot \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, I)$ and $\odot$ signifies an element-wise product. After that, the decoder D^G converts the latent vector l back into the pattern topology space and reconstructs a topology P' that are expected to be close to P. The objective function of the VCAE for training is as follows:

$$min \quad D_{KL}(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, I)) + \lambda \| P - P' \|_F^2, \tag{2a}$$

$$s.t. \quad \tilde{l} = E^G(P, W_E), \tag{2b}$$

$$\mu = f_R^{\mu}(\tilde{l}, W_R^{\mu}), \tag{2c}$$

$$\sigma = f_R^{\sigma}(\tilde{l}, W_R^{\sigma}), \tag{2d}$$

$$l = \mu + \sigma \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{2e}$$

$$P' = D^G(l, W_D). \tag{2f}$$

The first part of Equation 2a aims at making the distribution with mean μ and variance σ^2 close to the standard Gaussian distribution $\mathcal{N}(0, I)$, where D_{KL} represents the Kullback-Leibler (KL) divergence which measures the difference between two distributions. The second part of Equation 2a represents the reconstruction error of this auto-encoder, where λ is a trade-off parameter and $\| \cdot \|_F$ represents the Frobenius norm. Equation 2b and Equation 2f are the computation of the encoder and the decoder respectively, configurations of which are described in Section 4.3. Equation 2c and Equation 2d are reparameterization operations for μ and σ respectively, both of which are simple linear transformations. The terms μ and σ are trained by the encoder and the reparameterization block, and the ε is sampled from a standard Gaussian distribution. The variable $\tilde{l}$ is an internal representation of the output from the encoder, while the variable l is the sampled latent vector with Gaussian distribution by the reparameterization block. In addition, W_R^{μ}, W_R^{σ}, W_E and W_D are the trainable weights associated with the respective units. By minimizing this objective function, the latent representation l computed by Equation 2e is nearly in Gaussian distribution because $\mathcal{N}(\mu, \sigma^2)$ is close to $\mathcal{N}(0, I)$ and ε is also in Gaussian distribution. After training, our VCAE decoder is expected to have the ability of converting a latent vector in the Gaussian space back to a realistic pattern topology.

4.2 New Pattern Topology Generation

Different from the random sampling in the original VAEs, we generate new pattern topologies by applying limited Gaussian perturbations to the latent vector from existing pattern topologies as shown in Figure 4b.

At this stage, the encoder E^G and reparameterization block map an existing pattern topology into a latent vector l in a Gaussian space, which allows perturbation by adding a random Gaussian vector $\triangle l$. Apparently the new vector $l + \triangle l$ is still in Gaussian distribution, which implies that it can be converted back to pattern topology space by the trained decoder D^G. This reconstruction process is described by Equation 3:

$$P' = D^G(l + \triangle l, W_D), \quad \triangle l_i \sim \mathcal{N}(0, c^2), \quad i = 1 \cdots N, \tag{3}$$

where each element of $\triangle l$ is sampled from a simple Gaussian distribution with mean 0. The variance c^2 is a user-defined parameter that controls the range of perturbation. A larger variance c^2 implies a larger Gaussian perturbation, which leads to a larger difference between the original pattern topology and the generated topologies. In our experiments, the variance c^2 is set to 0.5.

4.3 Network Architecture Configuration

Instead of having densely connected layers as in the original VAEs, more powerful techniques, such as convolutional layers and residual blocks, are used in our VCAE to capture complex layout topology information. Table 1 shows the detailed configurations of the encoder, decoder and reparameterization block of our VCAE. The encoder uses convolution layers (Conv), while the decoder employs deconvolution layers (Deconv). Residual blocks (ResBlock) are employed in both the encoder and decoder. Instance normalization (IN) is selectively applied on certain convolutional layers, and ReLU is used as the activation function. For computing μ and σ respectively, the reparameterization block includes two simple linear transformation layers, both of which have dimension sizes of 1024.

Table 1: The network configurations of our VCAE.

Encoder		**Decoder**	
Layer	Output Size	Layer	Output Size
Input	$1 \times 128 \times 128$	Latent	$4 \times 16 \times 16$
Conv-IN-ReLU	$64 \times 128 \times 128$	Deconv-IN-ReLU	$32 \times 16 \times 16$
Conv-IN-ReLU	$128 \times 64 \times 64$	Deconv-IN-ReLU	$64 \times 16 \times 16$
Conv-IN-ReLU	$256 \times 32 \times 32$	Deconv-IN-ReLU	$128 \times 16 \times 16$
ResBlock	$256 \times 32 \times 32$	ResBlock	$128 \times 16 \times 16$
Conv-IN-ReLU	$256 \times 16 \times 16$	ResBlock	$128 \times 16 \times 16$
Conv-IN-ReLU	$128 \times 16 \times 16$	Deconv-IN-ReLU	$256 \times 16 \times 16$
ResBlock	$128 \times 16 \times 16$	Deconv-IN-ReLU	$256 \times 32 \times 32$
ResBlock	$128 \times 16 \times 16$	ResBlock	$256 \times 32 \times 32$
Conv-IN-ReLU	$64 \times 16 \times 16$	Deconv-IN-ReLU	$128 \times 64 \times 64$
Conv-IN-ReLU	$32 \times 16 \times 16$	Deconv-IN-ReLU	$64 \times 128 \times 128$
Conv-IN-ReLU	$4 \times 16 \times 16$	Deconv-Tanh	$1 \times 128 \times 128$
Reparameterization for μ		**Reparameterization** for σ	
Layer	Output Size	Layer	Output Size
Input	1024	Input	1024
Linear	1024	Linear	1024

5 PATTERN TOPOLOGY LEGALIZATION

Due to the complexity of two-dimensional layout patterns, the patterns generated by VCAE may not be legal. To repair the generated patterns to legal ones, we propose a CGAN-based topology legalization model, named LegalGAN, to eliminate DRC-violations for given layout patterns with little manual guidance.

5.1 Motivation and Data Preparation

VAEs will suffer from blurry generation for non-trivial datasets because of the injected noise and imperfect element-wise measures [2]. Our proposed generation model VCAE, based on VAEs, also faces the challenge of blurry flaws in generated pattern topologies, which will increase DRC violation risk. This happens primarily due to the reconstruction loss of the decoder, which is hard for VAE-based models to avoid. To repair the generated topologies, we design a legalization tool LegalGAN to transform generated samples from blurry patterns to smooth ones. The basic idea of LegalGAN is to eliminate blurry flaws and DRC violation risks by reducing the reconstruction loss with CGAN techniques.

For training LegalGAN, a dataset consisting of pairs of samples corresponding to real pattern topologies and decoded pattern topologies is needed. The real pattern topologies are those topologies P extracted from existing layout patterns, which are also used as input for training VCAE as shown in Figure 4a, while the decoded pattern topologies P' are the output of VCAE in Figure 4a. Due to the reconstruction loss, the topologies P' may be illegal, while those real topologies P are not. Hence, LegalGAN learning to transform P' to P will have the ability of repairing blurry pattern topologies and reducing DRC violation risks by reducing the reconstruction loss.

5.2 LegalGAN Architecture Design

Generative adversarial nets (GAN) [9] are networks that use a set of training samples to learn their distribution and generate new samples from the learned distribution. At the highest level, GANs are composed of a generator and a discriminator that compete with each other. The generator G generates fake samples based on a random noise to fool the discriminator, while the discriminator D tries to classify the real samples as 1 (real) and fake samples as 0 (fake). With this competition, the discriminator guides the generator on what are close to real samples, while also improves itself by learning how to distinguish real samples from fake ones.

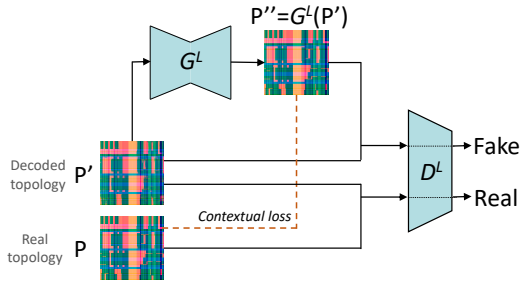


Figure 5: Architecture of LegalGAN.

CGANs are conditional model extended from GANs. Unlike the unconditional GANs, CGANs generate fake samples with a specific condition rather than generic samples based on random noises [12]. Specifically, for our legalization task, with the decoded topologies P' as input, CGAN requires the generated new topologies P'' to not only fool the discriminator but also to be close to the respective real topologies P.

Figure 5 shows the training process of our proposed LegalGAN model based on CGAN. This model is trained to transform a decoded topology P' to a legal one like the real topology P. Fed with P', the generator outputs a repaired pattern topology P'' where $P'' = G^L(P')$. The discriminator is responsible for classifying this topology pair $(P', G^L(P'))$ as fake, and meanwhile, it needs to regard the topology pair (P', P) as real. Mathematically, the objective function of the discriminator D^L is given by

$$\max_{D^L} \mathbb{E}_{P,P'}\left[\log(D^L(P', P))\right] + \mathbb{E}_{P'}\left[\log(1 - D^L(P', G^L(P')))\right]. \tag{4}$$

On the other hand, the objective of the generator is to generate pattern topology to fool the discriminator by maximizing the $D^L(x', G^L(x'))$. Besides, the generator wishes that the generated pattern topology $G^L(x')$ is close to the real pattern topology x by minimizing the contextual loss with $L1$ norm. The objective of G^L is defined as

$$\min_{G^L} \mathbb{E}_{P'}\left[\log(1 - D^L(P', G^L(P')))\right] + \gamma \cdot \mathbb{E}_{P'}\left[||P - G^L(P')||_1\right], \tag{5}$$

where γ is a weighting parameter to trade-off the impact of different parts of the objective function. Combining Equation 4 and Equation 5, we have the following objective function for our legalization model,

$$\min_{G^L}\max_{D^L} \mathbb{E}_{P,P'}\left[\log(D^L(P', P))\right] + \mathbb{E}_{P'}\left[\log(1 - D^L(P', G^L(P')))\right] + \gamma \cdot \mathbb{E}_{P'}\left[||P - G^L(P')||_1\right], \tag{6}$$

Besides, LegalGAN shares network configurations with the image-to-image CGAN in [12].

In the testing phase, the trained generator repairs the generated pattern topology P' to a new topology P'' with few blurry flaws and lower DRC violation risk.

5.3 Overall Flow in Testing Stage

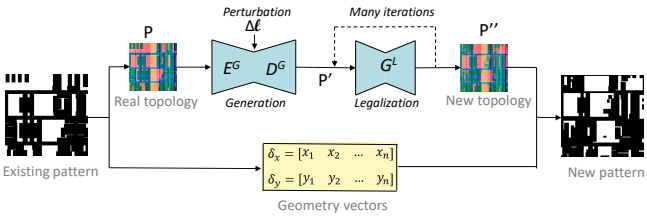
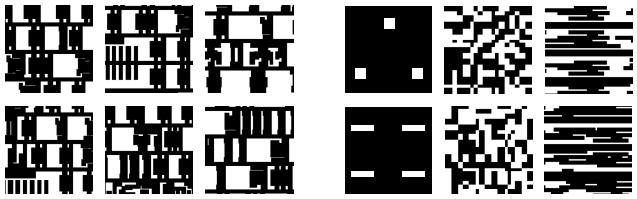


Figure 6: The flow of generation and legalization of CUP in the testing stage.

We summarize the pattern topology generation and legalization flow as shown in Figure 6, where an existing pattern is input, and finally a legalized new pattern is output. First an input existing pattern is transformed to a real topology P and two geometry vectors. Through the encoder E^G and decoder D^G of the trained VCAE, a topology P' is generated by applying a random Gaussian perturbation $\triangle l$ in the latent representation. Then the topology P' is legalized to a new topology P'' by the trained generator G^L of LegalGAN. Considering the much reconstruction loss due to the perturbation in VCAE, the trained generator G^L often needs to be recursively executed for multiple iterations, where the input is the partially legalized topology from the previous iteration. Finally, we concatenate the new topology P'' and the original geometry vectors to form a new smooth and realistic pattern with lower DRC violation risk.

6 PATTERN STYLE DETECTION

6.1 Pattern Validity



(a) Realistic. (b) Unrealistic.

Figure 7: Legal patterns with (a) realistic styles and (b) unrealistic styles.

Previous pattern generation efforts usually evaluate a generated layout pattern by only design rule checking. However, the features and styles of existing design layouts are resulted from not only design rule constraints but also inherent characteristic of their prior procedures of the development flow, such as logic synthesis physical design. Many DFM researches/flows need not only legal patterns but also realistic patterns. For a specific actual IC layout, a *realistic pattern* means a pattern that closely resembles the actual layout and shares the same inherent style with the layout.

Figure 7a shows existing realistic patterns of an actual layout from ICCAD contest 2014. Besides legal by satisfying the design rules described in Section 2.2, these patterns share inherent features and styles that cannot be completely described by intuitive design rules. Meanwhile, some legal but unrealistic patterns are shown in Figure 7b, where the leftmost two patterns are sparse ones that satisfy nearly all common design rules, the middle two patterns are randomly generated under the above design rules, and the rightmost two patterns are existing patterns from another layout with different style. All patterns in Figure 7b satisfy the design rules in Section 2.2, but apparently they share different inherent features and styles from the realistic patterns in Figure 7a. Note that, for the layout of ICCAD contest 2014, the right 2 existing patterns in Figure 7b are also regarded as unrealistic ones due to the different layout style. It is not reasonable that a DFM model designed based on the unrealistic patterns in Figure 7b will also work well with the realistic patterns in Figure 7a.

For pattern generation, the generated legal layout patterns cannot be just regarded as realistic patterns, and the legal but unrealistic patterns should be filtered out. To evaluate how realistic the generated patterns are, we introduce a concept of *pattern validity* by measuring the proportion of realistic patterns in a give pattern library:

Definition 6.1. (Pattern Validity). The pattern validity, denoted by V, is the proportion of realistic patterns to the total patterns, as shown in 7,

$$V = \frac{The\ number\ of\ realistic\ patterns}{The\ number\ of\ total\ patterns}. \tag{7}$$

A good pattern generation method can develop a pattern library with not only large pattern diversity but also high pattern validity.

6.2 Pattern Style Detection Model

Considering the fact that the inherent style of a layout is subjective and lack of measurement, a natural question then arises: how to evaluate the layout style of new patterns. In this section, we propose a pattern style detection model to check if the give patterns are realistic or not.

The pattern style detection task can be considered as an Anomaly Detection problem [3], where a realistic pattern that conforms to a particular layout style is regarded as normal, otherwise it's anomalous. Generative learning based models make great progress in anomaly detection problems [6]. In such problems, A CGAN is trained on normal samples only and makes the generator learn the distribution of normal samples. Given a trained generator, when an anomalous pattern is encoded, the generator will fail to reconstruct abnormalities, because it has not learnt anomalous features. The anomalous sample can then be detected by the difference between itself and its reconstructed sample. With this basic idea, we adopt the CGAN-based anomaly detection scheme [1] for the pattern style detection problem.

The training process of our pattern style detection model based on CGANs [18] is shown in Figure 8. Unlike vanilla CGANs, this model employs an adversarial auto-encoder within an encoder-decoder-encoder pipeline in its generator network capturing the

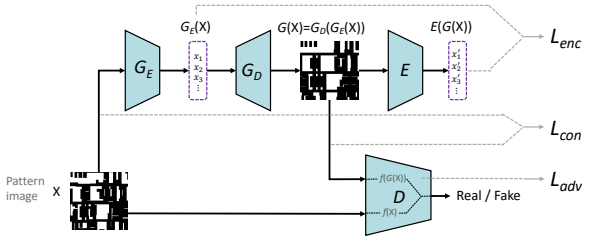


Figure 8: Architecture of pattern style detection model.

layout distribution within both the pattern and latent vector space. The additional encoder E maps the generated pattern to its latent representation, which is used to compute the style detection scores for new patterns.

In Figure 8, G_E, G_D and E are respectively the encoder, decoder and the second encoder in the generator, while D is the discriminator. First, the encoder G_E encodes the input pattern image X to a latent vector $G_E(X)$. $G_E(X)$ is then decoded to a fake pattern $G(X)$ by decoder G_D, where $G(X) = G_D(G_E(X))$. After that, the second encoder E generates another latent vector $E(G(X))$ when fed with fake pattern $G(X)$. With different parametrization, E has the same architectural details as $G_E(X)$. Meanwhile, the discriminator D is responsible for classifying the input pattern X and the generated pattern $G(X)$ as real or fake respectively.

Our objective function consists of three loss functions which are L_{adv}, L_{con} and L_{enc}. On the generator side, besides fooling the discriminator, the generator wants that the reconstructed pattern $G(X)$ is close to the input pattern X, which leads to the contextual loss L_{con} as follows:

$$L_{con} = \mathbb{E}_X \left[||X - G(X)||_1 \right]. \tag{8}$$

Then, an encoder loss L_{enc} is applied to minimize the difference between the latent features of the input pattern and the reconstructed one. It is formally defined as

$$L_{enc} = \mathbb{E}_X \left[||G_E(X) - E(G(X))||_2 \right]. \tag{9}$$

On the discriminator side, since [23] shows that feature matching has the ability of reducing the instability of training for GANs, we use feature matching loss [1] for adversarial learning, and update G based on the internal representation of D. We define f to represent a function that outputs the internal representation of D. Mathematically, the adversarial loss L_{adv} is given by

$$L_{adv} = \mathbb{E}_X \left[||f(X) - f(G(X))||_2 \right]. \tag{10}$$

Combining the three loss functions, we get the following objective function:

$$\min_{G_E, G_D, E} \max_{D} L_{adv} + \alpha \cdot L_{con} + \beta \cdot L_{enc}, \tag{11}$$

where α and β are weighting parameters to trade-off the impact of individual losses.

After training, we assume that when an unrealistic pattern image $\hat{X}$ is fed into the trained generator, G_D cannot generate the unrealistic patterns even though G_E manages to map $\hat{X}$ into the latent vector space. This is because the generator has learned styles and features of the realistic layout only, and its parametrization is not suitable for reconstructing unrealistic features. Then, the $E(G(\hat{X}))$ from encoder E will also miss unrealistic feature representation, leading to a dissimilarity between $G_E(\hat{X})$ and $E(G(\hat{X}))$. Based on the dissimilarity, we can classify $\hat{X}$ as an unrealistic pattern. More specifically, we define a pattern anomaly score as

$$score(\hat{X}) = ||G_E(\hat{X}) - E(G(\hat{X}))||_1. \tag{12}$$

The pattern $\hat{X}$ is considered to be realistic, when the $score(\hat{X})$ satisfies Equation 13 as follows:

$$score(\hat{X}) < T, \tag{13}$$

where T is a user-defined threshold value. A smaller T implies a stricter metric.

7 EXPERIMENTAL RESULT

We implement our flows including pattern generation, legalization and style detection with Python and PyTorch library [19]. The framework is tested on a platform with a TITAN V Graphic Card. For the benchmark, we get small layout patterns by splitting a $160 \times 400 \mu m^2$ layout from ICCAD contest 2014 [25]. After filtering out parts of patterns that maintain little information, we build a dataset including 13869 layout pattern images with size of $2.048 \times 2.048 \mu m^2$. 3000 of these patterns are regarded as the testing data, while others are in the training set. In our experiments, each layout pattern is represented as a lossless image with 2048×2048 pixels, which is then extracted to a squish pattern including one topology matrix and two geometry vectors. Each squish pattern topology is zero-padded to a matrix of size 128×128.

VCAE and the style detection model are trained with the above training set. LegalGAN is trained with pairs of topologies of training patterns and decoded topologies by the trained VCAE model. After training, CUP (VCAE + LegalGAN) will be applied to generate new patterns based on the existing patterns in the training set. The testing set is used for validating the trained pattern style detection model.

7.1 Results on Pattern Generation

In this subsection, we show experimental results of our pattern generation framework by the VCAE + LegalGAN model. First we randomly choose 100 existing patterns as the basic patterns. With each method, 100000 patterns are generated by perturbing latent representations of the 100 basic patterns with random Gaussian noise as shown in Equation 3, where the variance c^2 is set to 0.5. After that, we use the *KLayout* to check the legality of the generated patterns based on the design rules described in Section 2.2, and obtain the generated legal patterns.

To demonstrate the contribution of our VCAE model, we compare the pattern diversity between VCAE (+ LegalGAN) and a baseline which is a vanilla convolutional auto-encoder (CAE) (+ LegalGAN) with similar scale of network weights. Besides, the number of iterations in LegalGAN is set to 100.

Table 2 shows the comparison of pattern diversity (H) and the number of legal patterns among different pattern generation methods. *Existing patterns* consisting of both training and testing data are all legal patterns. All the 4 methods generate new patterns based on the 100 *basic patterns*. The *existing patterns* are highly diversified, while the 100 *basic patterns* have low pattern diversity because they only share parts of knowledge of the whole pattern set.

Table 2: Comparison of evaluation metrics among different methods.

Set/Method	All patterns		Legal patterns	
	Pattern #	Diversity (H)	Pattern #	Diversity (H)
Existing patterns	13869	10.7767	-	-
Basic patterns	100	6.6039	-	-
CAE	100000	4.5875	19	3.7871
VCAE	100000	10.9311	2126	9.9775
CAE+LegalGAN	100000	5.8465	3740	5.8142
CUP(VCAE+LegalGAN)	100000	9.8692	84510	9.8669

[1] Pattern# means the number of patterns.

We observe that our CUP (VCAE + LegalGAN) model can expand the low-diversified *basic patterns* to a legal and high-diversified pattern library with $H = 9.8669$, while CAE + LegalGAN generates a lower-diversified library with $H = 5.8142$. This demonstrates that our pattern topology generation model VCAE has the ability of generating diverse layout patterns.

On the other hand, although the patterns generated by VCAE without legalization obtain the highest pattern diversity, most of them suffer from DRC violations and need to be filtered out, which leads to a low generation efficiency. Meanwhile, the LegalGAN model can legalize most of such illegal patterns to legal ones, and finally CUP (VCAE + LegalGAN) gets 84510 patterns with high pattern diversity $H = 9.8669$.

7.2 LegalGAN Validation

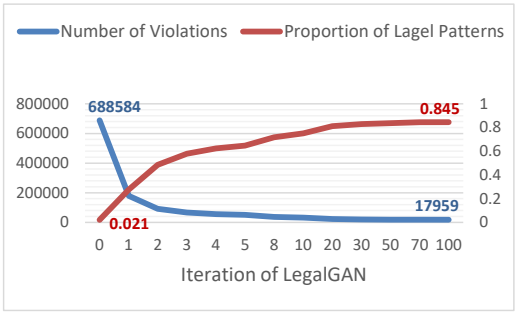


Figure 9: The number of DRC violations and the proportion of legal patterns in the process of legalization by LegalGAN.

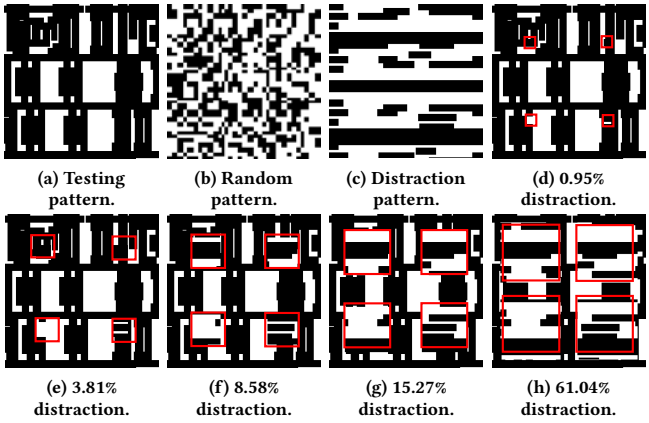
In this subsection, we further demonstrate the ability of our LegalGAN for removing DRC violations by making the generated patterns smooth. By the VCAE, we generate 100000 raw pattern topologies, which are then legalized by the LegalGAN in 100 iterations. Figure 9 shows the number of DRC violations and the proportion of legal patterns in different iterations. We observe that the legalization process converges after 30 iterations and produces a large proportion of legal patterns. Specifically, 97.39% DRC violations are removed (from 688584 to 17959). Note that there may be many DRC violations in a single pattern. Besides, the proportion of legal patterns in the total changes from 2.1% to 84.5%, which manifests the effectiveness of our LegalGAN model.

Table 3 gives a visualization of how raw patterns progressively become more and more smooth during the the topology legalization process of our LegalGAN. In this experiment, we choose an existing pattern, and generate 3 raw patterns by VCAE. Then, the raw patterns are legalized recursively, and Table 3 shows the legalization results of the first several iterations of LegalGAN. We can observe that LegalGAN has the ability of removing blurry flaws, which can reduce the DRC violation risks intuitively.

7.3 Results on Pattern Style Detection

In this subsection, we verify the validity of our trained pattern style detection model that is expected to provide a detection score for each testing pattern. A smaller detection score implies a more realistic pattern. Then, based on this tool, we demonstrate the ability of our pattern generation framework to generate realistic layout patterns.

In this experiment, we have designed 8 different pattern sets, each of which contains 3000 different layout patterns. The first set is the above testing set, in which the patterns are regarded as realistic patterns. The second set contains random patterns which are randomly generated but still satisfying the design rules described in Section 2.2. The third set, named distraction set, is extracted from another actual layout of ICCAD contest 2014, in which the patterns are also legal, but they maintain different pattern styles from the testing set. The other sets are mixed sets obtained by mixing the testing patterns and distraction patterns with different proportions, as shown in Figure 10. For example, Figure 10d shows a pattern that is mixed with 99.05% (area) testing pattern and 0.95% (area) distraction pattern, where the regions in red boxes are the parts from a distraction pattern. Intuitively, the pattern validity of the mixed sets by Equation 7 would be larger when the ratio of distraction patterns is smaller.



(a) Testing pattern. (b) Random pattern. (c) Distraction pattern. (d) 0.95% distraction.

(e) 3.81% distraction. (f) 8.58% distraction. (g) 15.27% distraction. (h) 61.04% distraction.

Figure 10: Testing pattern, random pattern, distraction pattern, and mixed patterns.

As shown in Table 4, our trained pattern style detection model predicts the pattern validity of the above 8 datasets. In this table, T is the validity threshold defined in Equation 13, and a smaller T implies stricter a metric. We observe that the pattern validity of testing patterns is large, while the pattern validity of random

Table 3: Visualization of the LegalGAN process.

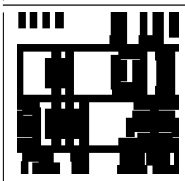
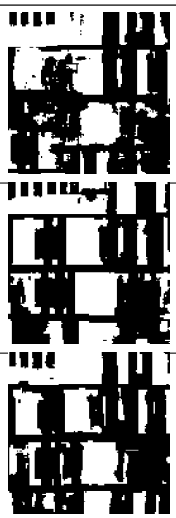
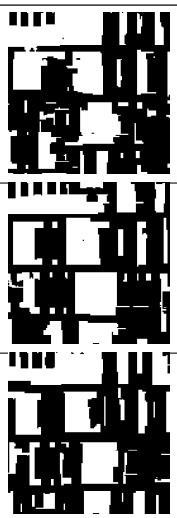
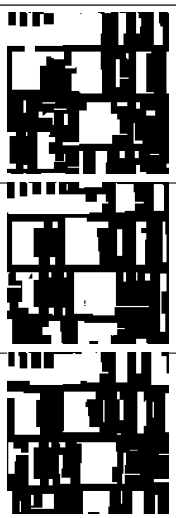
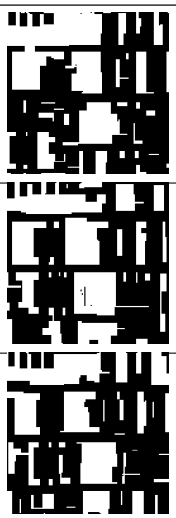
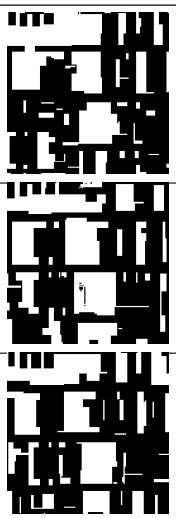
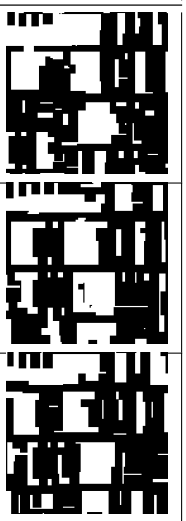
Existing pattern	Generated patterns	Legalized patterns				
		Iteration 1	Iteration 3	Iteration 5	Iteration 7	Iteration 9
						

Table 4: Pattern validity comparison between different mixed dataset for verifying pattern style detection.

Pattern Set	Pattern #	Pattern Validity (V)		
		$T = 0.6$	$T = 0.7$	$T = 0.8$
Testing patterns	3000	0.6493	0.8485	0.9306
Random patterns	3000	0	0	0.007
Distraction patterns	3000	0.0119	0.0143	0.0246
0.95% distraction	3000	0.5981	0.8236	0.9198
3.81% distraction	3000	0.4227	0.6943	0.8536
8.58% distraction	3000	0.0710	0.2204	0.4371
15.27% distraction	3000	0.0058	0.0365	0.1041
61.04% distraction	3000	0.0003	0.0041	0.0191

Table 5: Pattern validity comparison between different methods.

Set/Method	Pattern #	Pattern Validity (V)		
		$T = 0.6$	$T = 0.7$	$T = 0.8$
Testing patterns	3000	0.6493	0.8485	0.9306
CAE+LegalGAN	2126	0.0003	0.0027	0.0167
CUP(VCAE+LegalGAN)	84510	0.5430	0.7840	0.9057

patterns and distraction patterns is close to 0. For the mixed sets, if the ratio of distraction patterns is smaller, the pattern validity of the sets will be larger, which agrees with our intuition. Such results show that the trained pattern style detection model has the ability of checking the layout style of new patterns.

Finally, to evaluate the ability of our CUP (VCAE + LegalGAN) framework for generating realistic patterns, based on the trained pattern style detection model, we compute the pattern validity of generated legal patterns by different methods. The legal patterns are obtained from the 100000 generated patterns as described in Section 7.1. Table 5 shows the comparison results on pattern validity. We observe that the CAE + LegalGAN hardly generates realistic patterns, while CUP (VCAE + LegalGAN) produces legal patterns with high pattern validity. Besides, the pattern validity of the legal patterns generated by CUP (VCAE + LegalGAN) is close to that of the existing patterns in the testing set. Therefore, our pattern generation framework has ability of generating a DRC-clean pattern library with both high pattern diversity and high pattern validity.

8 CONCLUSION

In this work, we address the pattern library generation problem for DFM flows/researches based on different generative learning technologies. We propose a VCAE model that captures both pattern styles and features, which inspires a pattern topology generation flow with perturbation in the latent Gaussian representation space. We also design a pattern topology tool, named LegalGAN, to make the generated patterns smooth and DRC-clean. Besides, a pattern style detection tool is developed to check if the generated and legalized patterns are realistic, which also helps filtering the generated patterns to ensure the quality of the new pattern library. Our experiments show that our flow can generate legal patterns with high pattern validity and high pattern diversity.

REFERENCES

[1] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. 2018. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*. Springer, 622–637.

[2] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. 2017. CVAE-GAN: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE International Conference on Computer Vision*. 2745–2754.

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[4] Jingsong Chen, James Shiely, and Evangeline FY Young. 2019. Fast detection of largest repeating layout pattern. In *Design-Process-Technology Co-optimization for Manufacturability XIII*, Vol. 10962. International Society for Optics and Photonics, 109620A.

[5] Ran Chen, Wei Zhong, Haoyu Yang, Hao Geng, Xuan Zeng, and Bei Yu. 2019. Faster region-based hotspot detection. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[6] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. 2019. A survey on gans for anomaly detection. *arXiv preprint arXiv:1906.11632* (2019).

[7] Jhih-Rong Gao, Xiaoqing Xu, Bei Yu, and David Z Pan. 2014. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[8] Frank E Gennari and Ya-Chieh Lai. 2014. Topology design using squish patterns. US Patent 8,832,621.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[10] Ayman Hamouda, Mohamed Bahnas, Dan Schumacher, Ioana Graur, Ao Chen, Kareem Madkour, Hussein Ali, Jason Meiring, Neal Lafferty, and Chris McGinty. 2017. Enhanced OPC recipe coverage and early hotspot detection through automated layout generation and analysis. In *Optical Microlithography XXX*, Vol. 10147. International Society for Optics and Photonics, 101470R.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[13] Bentian Jiang, Hang Zhang, Jinglei Yang, and Evangeline FY Young. 2019. A fast machine learning-based mask printability predictor for opc acceleration. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 412–419.

[14] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations, ICLR 2014*.

[15] Jian Kuang, Wing-Kai Chow, and Evangeline FY Young. 2015. A robust approach for process variation aware mask optimization. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1591–1594.

[16] Jian Kuang, Junjie Ye, and Evangeline FY Young. 2016. Simultaneous template optimization and mask assignment for DSA with multiple patterning. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 75–82.

[17] Jian Kuang and Evangeline FY Young. 2013. An efficient layout decomposition approach for triple patterning lithography. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[18] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *CoRR* abs/1912.01703 (2019). arXiv:1912.01703 http://arxiv.org/abs/1912.01703

[20] Gaurav Rajavendra Reddy, Mohammad-Mahdi Bidmeshki, and Yiorgos Makris. 2019. VIPER: A Versatile and Intuitive Pattern GenERator for Early Design Space Exploration. In *IEEE International Test Conference, ITC 2019, Washington, DC, USA, November 9-15, 2019*. 1–7.

[21] Gaurav Rajavendra Reddy, Kareem Madkour, and Yiorgos Makris. 2019. Machine Learning-Based Hotspot Detection: Fallacies, Pitfalls and Marching Orders. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[22] Gaurav Rajavendra Reddy, Constantinos Xanthopoulos, and Yiorgos Makris. 2018. Enhanced hotspot detection through synthetic pattern generation and design of experiments. In *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 1–6.

[23] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 2226–2234.

[24] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423.

[25] Rasit O Topaloglu. 2014. ICCAD-2014 CAD contest in design for manufacturability flow for advanced semiconductor nodes and benchmark suite. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 367–368.

[26] Haoyu Yang, Shuhe Li, Yuzhe Ma, Bei Yu, and Evangeline FY Young. 2018. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.

[27] Haoyu Yang, Yajun Lin, Bei Yu, and Evangeline FY Young. 2017. Lithography hotspot detection: From shallow to deep learning. In *2017 30th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 233–238.

[28] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2019. Deepattern: Layout pattern generation with transforming convolutional auto-encoder. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.

[29] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2019. Detecting multi-layer layout hotspots with adaptive squish patterns. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 299–304.

[30] Haoyu Yang, Jing Su, Yi Zou, Yuzhe Ma, Bei Yu, and Evangeline FY Young. 2018. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 6 (2018), 1175–1187.

[31] Wei Ye, Mohamed Baker Alawieh, Yibo Lin, and David Z Pan. 2019. Lithogan: End-to-end lithography modeling with generative adversarial networks. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[32] Wei Ye, Yibo Lin, Meng Li, Qiang Liu, and David Z Pan. 2019. LithoROC: lithography hotspot detection with explicit ROC optimization. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 292–298.

[33] Hang Zhang, Bei Yu, and Evangeline FY Young. 2016. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. In *Proceedings of the 35th International Conference on Computer-Aided Design*. 1–8.

[34] Hang Zhang, Fengyuan Zhu, Haocheng Li, Evangeline FY Young, and Bei Yu. 2017. Bilinear lithography hotspot detection. In *Proceedings of the 2017 ACM on International Symposium on Physical Design*. 7–14.