EMOGen: Enhancing Mask Optimization via Pattern Generation*

Su Zheng
CUHK

Yuzhe Ma
HKUST(GZ)

Bei Yu
CUHK

Martin Wong
HKBU

Abstract

Layout pattern generation via deep generative models is a promising methodology for building practical large-scale pattern libraries. However, although improving optical proximity correction (OPC) is a major target of existing pattern generation methods, they are not explicitly trained for OPC and integrated into OPC methods. In this paper, we propose EMOGen to enable the co-evolution of layout pattern generation and learning-based OPC methods. With the novel co-evolution methodology, we achieve up to 39% enhancement in OPC and 34% improvement in pattern legalization.

1 Introduction

Large-scale layout pattern libraries are important for the evolution of various design for manufacturing (DFM) research fields, such as optical proximity correction (OPC), lithography simulation, lithography hotspot detection, etc. Since lithography simulation is a basic step in OPC while hotspot detection relies on OPC results, OPC actually becomes a major target of layout pattern generation. However, OPC is not explicitly considered in existing pattern generation methods, which are typically trained to mimic known patterns with variation and evaluated by the diversity of results.

Existing pattern generation methods can be classified into two categories, rule-based [1] and learning-based [2]. Rule-based methods employ techniques like slicing and rotation to generate new patterns based on basic units. Nevertheless, the results generated by such simple methods usually have low diversity and quantity. On the contrary, learning-based methods can create more diverse and numerous layout patterns. Pixel-based methods [3] treat the problem as creating binary images. Sequence-based learning [4] represents a layout pattern as a series of polygons and produces new polygons via sequence generation. To ensure the legality of the generated patterns, DiffPattern [5] generates the topology with a diffusion model and legalizes the geometry via nonlinear optimization.

Although existing methods can generate diverse and reliable patterns, the results are not guided to improve the performance on specific tasks. For instance, although OPC is a major target of pattern generation, existing methods do not integrate OPC-related objectives. In terms of OPC, many recent works focus on DNN-based inverse lithography technique (ILT) [6–12], which relies on advanced network architectures to do mask optimization. The model architectures for DNN-based ILT evolve from fully convolutional networks (FCNs) [7] to Fourier neural operator (FNO) [11], which brings much improvement on the performance. However, even though efforts have been made to build large-scale datasets [13], there is less research on how to collect or generate diverse and hard samples for DNN-based ILT. We treasure hard samples because they can help us find the weaknesses of the ILT models and improve the performance in extreme cases. This inspires us to enhance the performance of DNN-based ILT via layout pattern generation.

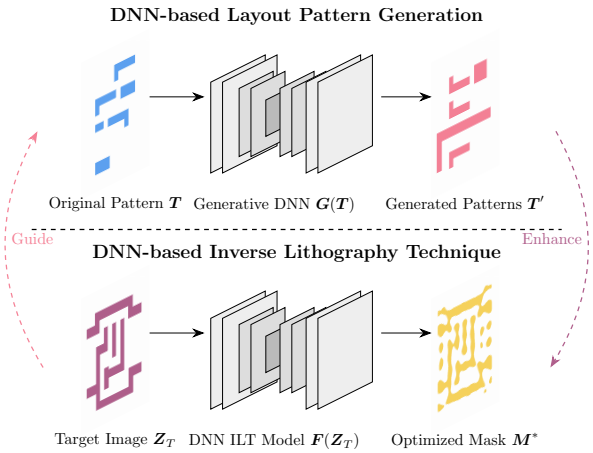


Figure 1: Motivation of the EMOGen framework. The pattern generation model outputs more diverse results with the help of the DNN-based ILT model. Meanwhile, the DNN-based ILT model is enhanced by the generated patterns.

To provide efficient and effective mask optimization, we focus on DNN-based ILT, which requires less runtime than traditional methods. We choose learning-based pattern generation because it offers better diversity. An intuitive idea for enhancing mask optimization is to collect a large quantity of data from the pattern generation model and use them to train the ILT model. However, since the pattern generation and ILT models have no information about each other, it is difficult to ensure that the generated patterns can improve the ILT performance. To overcome this challenge, we propose EMOGen to enable the co-evolution of pattern generation and ILT models, which can enhance mask optimization via layout pattern generation. As shown in Figure 1, we construct the interaction between the pattern generation and ILT models so that the pattern generation model can find the weaknesses of the ILT model. As a result, the ILT model can evolve by training on hard samples and the pattern generation model can evolve by finding new weaknesses.

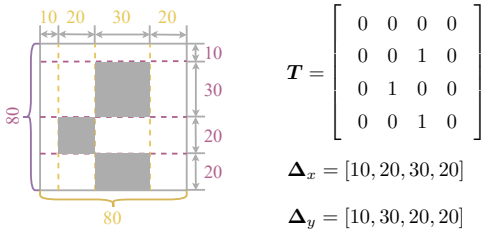
The major contribution of EMOGen can be summarized as follows.

- We propose EMOGen to enable the co-evolution of layout pattern generation and mask optimization. To our best knowledge, this is the first paper that directly utilizes layout pattern generation to improve DNN-based ILT methods, exploiting the potential of DNN-based pattern generation.
- We design the ILT-aware training and legalization schemes for DNN-based pattern generation, making it possible to discover

*This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14208021).

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655680



(a)

$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{\Delta}_x = [10, 20, 30, 20]$$

$$\boldsymbol{\Delta}_y = [10, 30, 20, 20]$$

(b)

Figure 2: Illustration of squish pattern. A layout tile is converted to a topology matrix and two geometry vectors.

the weaknesses of the DNN-based ILT model while maintaining the existing diversity and legality advantages.

- Extensive experiments are conducted to verify the effectiveness of the proposed method, achieving up to 39% enhancement in DNN-based ILT and 34% improvement in pattern legalization.

2 Preliminaries

Squish Pattern. Typically, the layout patterns consist of multiple rectilinear polygons. Training DNN models on such information-sparse data incurs unnecessary computational overhead and leads to potential overfitting. To solve these problems, we adopt the squish pattern to achieve a lossless and efficient representation of the layout patterns. The squish pattern representation decomposes a layout tile into a topology matrix $\boldsymbol{T}$ and two geometry vectors $\boldsymbol{\Delta}_x, \boldsymbol{\Delta}_y$. As illustrated by Figure 2, to construct the topology matrix, we divide the layout into grids according to the scan lines defined by the edges of polygons. The topology matrix has entries of ones inside a polygon and zeros in empty space. The geometry vectors store the length of the gap between each pair of neighboring scan lines. Given an $N \times N$ layout tile, the geometry vectors should satisfy,

$$\begin{cases} \Delta_{x,i} > 0, \Delta_{y,j} > 0, \\ \sum_i \Delta_{x,i} = N, \sum_j \Delta_{y,j} = N, \end{cases} \tag{1}$$

where $\Delta_{x,i}$ and $\Delta_{y,j}$ are the elements in the two geometry vectors.

Layout Pattern Generation. Recent DNN-based methods usually formulate layout pattern generation as an image-to-image translation problem. Methods based on autoencoder [2, 3] map the topology matrices to a latent space and generate new patterns by perturbing the latent features, which can be formulated by,

$$\boldsymbol{T}' = f_{dec}\left(f_{enc}(\boldsymbol{T}) + \epsilon \mathcal{N}(0, \boldsymbol{I})\right), \tag{2}$$

where ϵ is the perturbation magnitude. The functions f_{enc} and f_{dec} represent the encoder and decoder in the autoencoder model, respectively. To avoid illegal patterns, the method based on diffusion model [5] optimizes the geometry vectors to satisfy the design rules. Specifically, it limits the minimum distance between polygons, the minimum width of a shape, and the range of a polygon's area. These rules can be formulated as constraints on the geometry vectors.

ILT for Mask Optimization. The transfer of patterns from the mask to the wafer can be modeled by lithography simulation, which consists of optical projection and photoresist models. In optical projection, the spatial information of the mask patterns $\boldsymbol{M}$ is transformed to the aerial intensity distribution $\boldsymbol{I}$ on the wafer plane. This process can be modeled by Hopkins' diffraction theory [14]:

$$\boldsymbol{I} = \boldsymbol{H}(\boldsymbol{M}) = \sum_{k=1}^{K} \mu_k |\boldsymbol{h}_k \otimes \boldsymbol{M}|^2, \tag{3}$$

where $\boldsymbol{h}_k$ is the kth optical kernel function, μ_k is the kth weight, and $\otimes$ denotes the convolution operation.

After optical projection, a photoresist model transfers the aerial image $\boldsymbol{I}$ to the printed image $\boldsymbol{Z}$. To enable gradient descent in ILT algorithms, researchers commonly design the photoresist model as $Z(x,y) = \sigma_Z(\boldsymbol{I}(x,y)) = \text{Sigmoid}(\alpha(\boldsymbol{I}(x,y) - I_{th}))$, where I_{th} is the intensity threshold and α is the steepness factor.

ILT [15–18] solves the mask patterns such that after the lithography process, the printed patterns on the silicon wafer are close to the design target. An ILT algorithm usually optimizes a parameter matrix $\boldsymbol{P}$ that can be transformed to the mask image $\boldsymbol{M}$ with $\boldsymbol{M}(x,y) = \sigma_M(\boldsymbol{P}(x,y)) = \text{Sigmoid}(\beta(\boldsymbol{P}(x,y) - \gamma))$, where β is the steepness factor and γ is a constant offset. ILT algorithms typically consider three process corners, maximum, nominal, and minimum. Three series of optical kernels $\boldsymbol{H}_{\max}$, $\boldsymbol{H}_{\text{nom}}$, and $\boldsymbol{H}_{\min}$, are employed to generate the printed images, $\boldsymbol{Z}_{\max}$, $\boldsymbol{Z}_{\text{nom}}$, and $\boldsymbol{Z}_{\min}$, respectively.

L2 loss is usually adopted to measure the difference between the nominal printed image $\boldsymbol{Z}_{\text{nom}}$ and the target image $\boldsymbol{Z}_T$, defined as:

$$L2(\boldsymbol{Z}_{\text{nom}}, \boldsymbol{Z}_T) = \|\boldsymbol{Z}_{\text{nom}} - \boldsymbol{Z}_T\|^2. \tag{4}$$

To enhance the robustness against varying process conditions, ILT algorithms are required to reduce the process variation band (PVB):

$$PVB(\boldsymbol{Z}_{\max}, \boldsymbol{Z}_{\min}) = \|\boldsymbol{Z}_{\max} - \boldsymbol{Z}_{\min}\|^2. \tag{5}$$

Given the loss function, we can optimize $\boldsymbol{P}$ via gradient descent and get the optimized mask $\boldsymbol{M}^*$. In addition to L2 and PVB, we adopt edge placement error (EPE) to estimate the distortion of the printed image. To estimate EPE, we sample probe points equidistantly on the edges of target patterns, and count the points where the distance between the target and printed patterns is larger than a threshold.

Recently, researchers have proposed various DNN-based ILT methods [6–12], which can significantly boost the speed and performance of ILT with end-to-end inference. In such methods, the input and output are the target image and optimized mask, respectively.

3 Method

In this section, we first formulate the co-evolution of layout pattern generation and mask optimization. Next, we demonstrate how to achieve co-evolution by designing the loss function and training process, as well as the legalization of the generated patterns. Finally, we discuss the network architectures used in this paper.

3.1 Mathematical Formulation

Given target images $\boldsymbol{Z}_T$, the model $f_M(\cdot)$, and its parameters $\boldsymbol{\theta}_M$, the mask optimization problem is formulated as:

$$\boldsymbol{M}^* = f_M(\boldsymbol{Z}_T | \boldsymbol{\theta}_M), \tag{6}$$

where $\boldsymbol{M}^*$ represents the optimized masks given by the model.

Existing pattern generation models usually decouple the generation of topology matrices and geometry vectors. However, to find hard samples for mask optimization, we cannot ignore the interaction between them. Therefore, we integrate the generation of

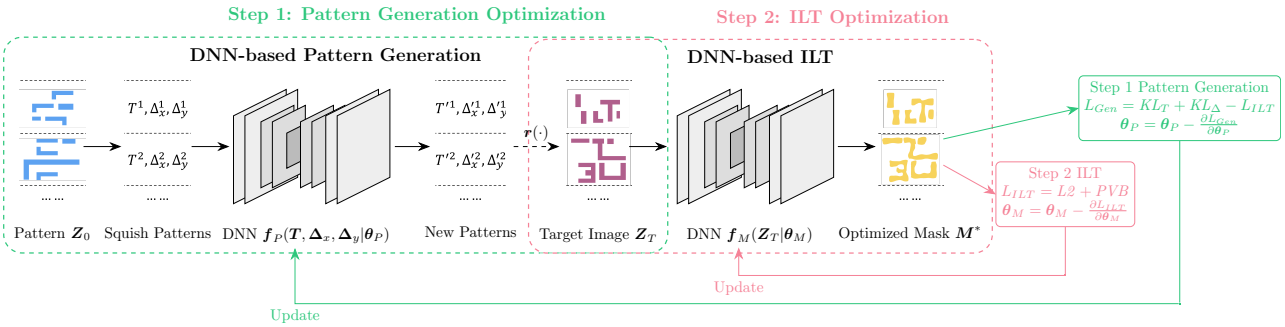


Figure 3: Overview of EMOGen training. We decouple an iteration into two steps. First, the pattern generation model is trained to deteriorate the ILT model. Second, the ILT model is optimized to achieve better performance on the generated patterns.

topology matrix and geometry vectors into one model. Finally, we can formulate the pattern generation problem as,

$$\boldsymbol{T}', \Delta_x', \Delta_y' = \boldsymbol{f}_P(\boldsymbol{T}, \Delta_x, \Delta_y|\boldsymbol{\theta}_P), \tag{7}$$

where $\boldsymbol{f}_P(\cdot)$ denotes the pattern generation model with the parameters $\boldsymbol{\theta}_P$. $\boldsymbol{T}', \Delta_x', \Delta_y'$ represent the generated topology and geometry.

In the co-evolution problem, the pattern generation and ILT models can be regarded as two players competing with each other. In terms of the pattern generation model, it makes efforts to generate hard samples that deteriorate the ILT performance. On the contrary, the target of the ILT model is to overcome the challenges from the generated patterns. In other words, the pattern generation model is expected to maximize the ILT loss while the ILT model is required to minimize it. We formulate this competition as the following equation,

$$\min_{\boldsymbol{\theta}_M} \max_{\boldsymbol{\theta}_P} L_{ILT}\left(\boldsymbol{f}_M(\boldsymbol{X}|\boldsymbol{\theta}_M), \boldsymbol{X}\right) \ s.t.\ \boldsymbol{X} = \boldsymbol{r}\left(\boldsymbol{f}_P(\boldsymbol{T}, \Delta_x, \Delta_y|\boldsymbol{\theta}_P)\right), \tag{8}$$

where L_{ILT} is the loss function of ILT. The $\boldsymbol{r}(\cdot)$ function converts the output of pattern generation to the input of the ILT model.

Although Equation (8) is enough for enhancing ILT, we need to model design rules in pattern generation to ensure the legality of the generated results. The first rule is about the minimum space between shapes, formulated by,

$$\sum_{k\in\boldsymbol{k}_x} \Delta_{x,k}' \geq \text{Space}_{\text{min}}, \sum_{k\in\boldsymbol{k}_y} \Delta_{y,k}' \geq \text{Space}_{\text{min}}, \forall \boldsymbol{k}_x, \boldsymbol{k}_y \in \boldsymbol{S}_{\text{min}}, \tag{9}$$

where $\mathbf{S}_{\text{min}}$ is a set containing the continuous horizontal or vertical segments in the empty space. $\text{Space}_{\text{min}}$ is the minimum space between shapes. Similarly, we have the constraint on the minimum width of the shapes, represented by,

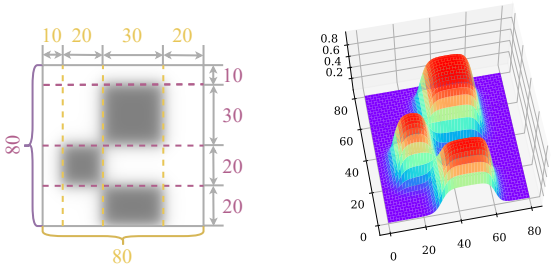
$$\sum_{l\in\boldsymbol{l}_x} \Delta_{x,l}' \geq \text{Width}_{\text{min}}, \sum_{l\in\boldsymbol{l}_y} \Delta_{y,l}' \geq \text{Width}_{\text{min}}, \forall \boldsymbol{l}_x, \boldsymbol{l}_y \in \boldsymbol{W}_{\text{min}}, \tag{10}$$

where $\mathbf{W}_{\text{min}}$ includes the continuous segments on the polygon edges and $\text{Width}_{\text{min}}$ is the minimum width. Finally, the area of a polygon is limited to a proper range, formulated by,

$$\sum_{(i,j)\in\boldsymbol{p}} \Delta_{x,i}'\Delta_{y,j}' \in [\text{Area}_{\text{min}}, \text{Area}_{\text{max}}], \forall \text{ polygon } \boldsymbol{p}, \tag{11}$$

where $[\text{Area}_{\text{min}}, \text{Area}_{\text{max}}]$ specifies the legal range of the areas. In terms of pattern generation, we require the results to not only deteriorate the performance of ILT but also satisfy these constraints.

The legalization process can be generalized to universal design rules because most rules are the combination of width, spacing, area, and enclosure checking [19]. Given the squish patterns, we can easily represent distances with the addition or subtraction among geometry vector elements, as in Equation (9) and Equation (10). It helps us to model the legalization of width, spacing, and enclosure rules. With the multiplication between geometry vector elements, we can further model area rules, like Equation (11). We solve the legalization by converting the rules to penalty terms, which enables us to implement the logical operations between different rules. For example, the logical *AND* of two rules can be accomplished by the addition of their penalty terms. The logical *OR* of two rules can be implemented by the multiplication of them. Therefore, the combination of width, spacing, area, and enclosure rules can also be modeled and solved in our legalization process.



(a) (b)

Figure 4: Illustration of the bell-shaped function for relaxing the conversion from squish patterns to images. (a) is the image converted from the squish pattern in Figure 2, (b) shows the 3D views of the image, formed by the bell-shape functions.

3.2 Co-evolution of Pattern Generation and Mask Optimization

To solve Equation (8), we need to overcome several challenges.

(1) The way to feed the generated patterns to the ILT model. The output of pattern generation includes the topology matrix and geometry vectors, while the input of ILT is an image containing the target patterns. The conversion between these two representations is not differentiable, which prohibits us from back-propagating the gradients from ILT to pattern generation.

(2) A training scheme that can solve the min-max optimization problem. Since the training of a neural network is typically a minimization problem, we need to convert Equation (8) to the minimization of loss functions in order to train the two models.

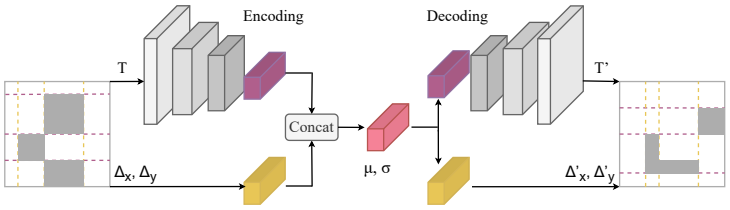


Figure 5: Illustration of the pattern generation model.

Table 1: Pattern Generation Model Architecture

Encoding		Decoding	
Layer	Output Size	Layer	Output Size
Input	$1 \times 32 \times 32$	Linear×2	128
Conv×2	$64 \times 16 \times 16$	Conv×2	$256 \times 4 \times 4$
Conv×2	$128 \times 8 \times 8$	Conv×2	$128 \times 8 \times 8$
Conv×2	$256 \times 4 \times 4$	Conv×2	$64 \times 16 \times 16$
Linear×2	128	Conv×2	$1 \times 32 \times 32$

(3) The way to satisfy the constraints in Equation (9)-(11). To ensure the validity of the generated patterns, we need a method to legalize them. Although [5] has proposed a legalization method, it is not applicable in our setting because we require the generated patterns to deteriorate the performance of the ILT model.

Combining Pattern Generation and ILT Models. As shown in Figure 3, we combine the networks for pattern generation and ILT by using the generated patterns as the inputs of the ILT model. Typically, to convert squish patterns to images, we look at each entry in the topology matrix $T_{i,j}$ and the elements $(\Delta_{x,i}, \Delta_{y,j})$ in the geometry vectors. If $T_{i,j} == 1$, we put a $\Delta_{x,i} \times \Delta_{y,j}$ rectangle on the image, whose bottom-left and top-right corners are $(l_{i,j}, b_{i,j})$ and $(r_{i,j}, t_{i,j})$, respectively. The points $l_{i,j}$, $b_{i,j}$, $r_{i,j}$, and $t_{i,j}$ can be computed by,

$$l_{i,j} = \sum_{k=0}^{i-1} \Delta_{x,k}, \ b_{i,j} = \sum_{k=0}^{j-1} \Delta_{y,k}, \ r_{i,j} = l_{i,j} + \Delta_{x,i}, \ t_{i,j} = b_{i,j} + \Delta_{y,j}. \tag{12}$$

However, this conversion is not differentiable for $\boldsymbol{\Delta}_x$ and $\boldsymbol{\Delta}_y$ so we need to relax the problem. Thus, we design the $\boldsymbol{r}(\cdot)$ function to convert the output of pattern generation to the input of the ILT model. In the $\boldsymbol{r}(\cdot)$ function, if $T_{i,j} == 1$, we put a bell-shaped function rather than a rectangle. The bell-shaped function can be defined as,

$$\Omega_{i,j}(x,y) = \frac{1}{\left(1+e^{\kappa(l_{i,j}-x)}\right)\left(1+e^{\kappa(x-r_{i,j})}\right)\left(1+e^{\kappa(b_{i,j}-y)}\right)\left(1+e^{\kappa(y-t_{i,j})}\right)}, \tag{13}$$

where κ is the steepness factor. Figure 4 shows an example of relaxing the conversion via differentiable bell-shaped functions.

Training Method for the Min-max Problem. To make Equation (8) implementable, we decouple the problem into the training of two loss functions like GAN [20]. The pattern generation loss is,

$$L_{Gen} = \alpha KL(\boldsymbol{p}_{\boldsymbol{T}'} \| \boldsymbol{p}_{\boldsymbol{T}}) + \beta KL(\boldsymbol{p}_{\boldsymbol{\Delta}'} \| \boldsymbol{p}_{\boldsymbol{\Delta}}) - \gamma L_{ILT}, \tag{14}$$

where $KL(\boldsymbol{p}_{\boldsymbol{T}'} \| \boldsymbol{p}_{\boldsymbol{T}})$ computes the KL divergence between the distribution of the original and generated topology matrices. It is employed to preserve the similarity between the new patterns and the given data. Similarly, $KL(\boldsymbol{p}_{\boldsymbol{\Delta}'} \| \boldsymbol{p}_{\boldsymbol{\Delta}})$ retains the similarity between the original and generated geometry vectors. The term $-L_{ILT}$ aims to guide the model to deteriorate the ILT performance using the generated patterns. α, β, and γ are the weights of the three terms.

For the ILT model, the loss function is defined as,

$$L_{ILT} = L2(\boldsymbol{Z}_{\text{nom}}, \boldsymbol{Z}_T) + PVB(\boldsymbol{Z}_{\max}, \boldsymbol{Z}_{\min}). \tag{15}$$

At each iteration, we first train the pattern generation model with L_{Gen} and then train the ILT model with L_{ILT}. Via Equation (14), we progress towards $\max_{\boldsymbol{\theta}_P} L_{ILT}(\boldsymbol{f}_M(\boldsymbol{X}|\boldsymbol{\theta}_M), \boldsymbol{X})$ in Equation (8). Next, Equation (15) guides ILT to achieve the minimization in Equation (8).

Algorithm 1 Training Process of the Models

Input: Pattern generation model $\boldsymbol{f}_P(\cdot|\boldsymbol{\theta}_P)$, ILT model $\boldsymbol{f}_M(\cdot|\boldsymbol{\theta}_M)$.

1: //Pretraining phase;
2: Train the pattern generation model to recover original patterns;
3: Train the ILT model on the original target images;
4: //Co-evolution phase;
5: **for** *epoch* $\in [1, 2, ..., \#epochs]$ **do**
6: **for** *iter* $\in [1, 2, ..., \#iterations]$ **do**
7: //Training pattern generation model;
8: Freeze the weights $\boldsymbol{\theta}_M$, make the weights $\boldsymbol{\theta}_P$ trainable;
9: Compute L_{Gen} by Equation (14);
10: Update the parameters of $\boldsymbol{f}_P$ by $\boldsymbol{\theta}_P = \boldsymbol{\theta}_P - \frac{\partial L_{Gen}}{\partial \boldsymbol{\theta}_P}$;
11: //Training ILT model;
12: Freeze the weights $\boldsymbol{\theta}_P$, make the weights $\boldsymbol{\theta}_M$ trainable;
13: Compute L_{ILT} by Equation (15);
14: Update the parameters of $\boldsymbol{f}_M$ by $\boldsymbol{\theta}_M = \boldsymbol{\theta}_M - \frac{\partial L_{ILT}}{\partial \boldsymbol{\theta}_M}$;
15: **end for**
16: **end for**

Algorithm 1 summarizes the training process, which includes the pretraining and co-evolution phases. In the pretraining phase (lines 2–3), the pattern generation model and the ILT model are trained separately to achieve a moderate performance on their own tasks. In the pretraining of the pattern generation model, we adopt the following loss function inspired by variational autoencoder [3, 21],

$$\begin{aligned} L_{Rec} =& \text{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \| \mathcal{N}(0, \boldsymbol{I})) + \\ & \eta \left(\|\boldsymbol{T}' - \boldsymbol{T}\|^2 + \|\boldsymbol{\Delta}'_x - \boldsymbol{\Delta}_x\|^2 + \|\boldsymbol{\Delta}'_y - \boldsymbol{\Delta}_y\|^2 \right), \end{aligned} \tag{16}$$

where $\text{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \| \mathcal{N}(0, \boldsymbol{I}))$ is the KL divergence that minimizes the distance between a standard Gaussian distribution and the distribution given by the latent features. $\|\boldsymbol{T}' - \boldsymbol{T}\|^2 + \|\boldsymbol{\Delta}'_x - \boldsymbol{\Delta}_x\|^2 + \|\boldsymbol{\Delta}'_y - \boldsymbol{\Delta}_y\|^2$ guides the model to recover the input patterns. It is weight is η. For the pretraining of ILT, we use L_{ILT} as the loss function.

Lines 5–16 in Algorithm 1 present the co-evolution phase, where we alternatively optimize the pattern generation model with L_{Gen} and the ILT model with L_{ILT}. As illustrated by Figure 3, we have two steps at each training iteration. In the first step, we evaluate the ILT performance on the generated patterns and optimize the pattern generation model to deteriorate the ILT results while keeping the similarity between the new and original patterns. In the second step, we generate patterns to optimize the ILT performance on these new data. Note that before training a model, we freeze the weights of the other one. This two-step training scheme creates a competition between the two models, achieving the co-evolution of pattern generation and mask optimization. In this way, we can enhance mask optimization via pattern generation in a differentiable manner.

Table 2: Comparison Between ILT Models With and Without Co-evolution

Method	GAN-OPC		NeuralILT		DAMO		CFNO		Average		Ratio	
Setting	Finetune	**Co-evolve**	Finetune	**Co-evolve**	Finetune	**Co-evolve**	Finetune	**Co-evolve**	Finetune	**Co-evolve**	Finetune	**Co-evolve**
L2	50,388	**38,288**	50,804	**45,313**	53,448	**33,757**	46,280	**45,863**	50,230	**40,805**	1.00	**0.81**
PVB	69,549	**53,987**	61,464	**55,082**	65,447	**54,703**	62,514	**61,811**	64,743	**56,395**	1.00	**0.87**
EPE	7.1	**4.0**	7.5	**6.4**	10.2	**3.7**	5.9	**4.9**	7.7	**4.7**	1.00	**0.61**

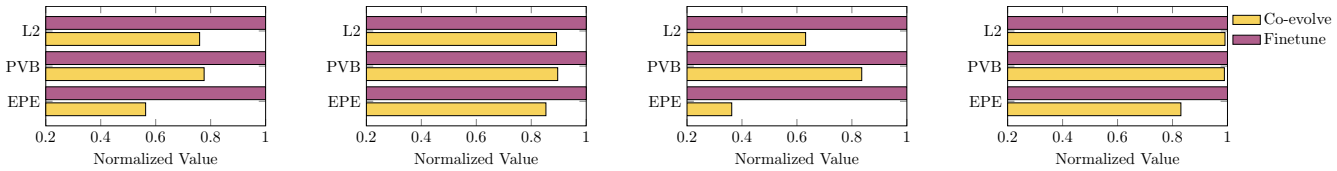


Figure 6: Comparison between ILT models with and without co-evolution on the normalized L2, PVB, and EPE. The ILT models are (a) GAN-OPC; (b) NeuralILT; (c) DAMO; (d) CFNO.

Legalization of the Generated Patterns. The pattern generation model can not only serve as a critic of the ILT model, but also be used as a standalone layout pattern generator. To ensure the validity of the generated patterns, we legalize the geometry vectors using the penalty function method. The constraints in Equation (9)-(11) can be classified into three categories, (1) $f(\boldsymbol{x}) \geq a$; (2) $g(\boldsymbol{x}) \leq b$; (3) $h(\boldsymbol{x}) = c$. Their penalty terms are designed as, (1) $\lambda_f \| \max(a - f(\boldsymbol{x}), 0)\|^2$; (2) $\lambda_g \| \max(g(\boldsymbol{x}) - b, 0)\|^2$; (3) $\lambda_h \|h(\boldsymbol{x}) - c\|^2$. Finally, the optimization objective is defined as,

$$\begin{aligned} L_{legal} = &\alpha\, KL(\boldsymbol{p}_{\boldsymbol{T}'} \| \boldsymbol{p}_{\boldsymbol{T}}) + \beta\, KL(\boldsymbol{p}_{\Delta'} \| \boldsymbol{p}_{\Delta}) - \gamma\, L_{ILT} + \\ &\sum_f \lambda_f \| \max(a - f(\boldsymbol{x}), 0)\|^2 + \sum_g \lambda_g \| \max(g(\boldsymbol{x}) - b, 0)\|^2 + \\ &\sum_h \lambda_h \|h(\boldsymbol{x}) - c\|^2. \end{aligned} \tag{17}$$

We solve Equation (17) via gradient descent. Other methods like augmented Lagrangian method is also applicable, but we find that the penalty function method generalizes better.

3.3 Neural Network Architecture

In this paper, we design the pattern generation network based on variational autoencoder [21], following [3]. Unlike the previous work that preserves the original geometry vectors, our model optimizes the topology matrix and geometry vectors together to better achieve the goal of enhancing ILT. Figure 5 presents the architecture of the pattern generation model. The topology matrix of the original layout is input to the upper encoding branch while the geometry vectors are input to the lower encoding branch. After obtaining the features from the two branches, we concatenate them and get a uniform latent representation via fully connected layers. Next, we generate the new topology and geometry information via the two decoding branches and convert the results to the new layout image.

Table 1 shows the encoding and decoding branches for the topology matrix. The branches for the geometry vectors have the same configuration as the linear layers in Table 1. Since our major target is to enhance ILT with pattern generation, we use an input/output size of 32×32, which can generate $1024 \times 1024nm^2$ layout tiles to meet the demand of ILT. Due to the small input/output size, we design such a relatively simple model architecture to avoid overfitting.

Since most DNN-based ILT methods are image-to-image translation models, we can adopt various state-of-the-art (SOTA) models [6–8, 11] to verify the effectiveness of EMOGen.

4 Experiments

We implement all DNN models with PyTorch 1.10 [22] and test them on RTX Titan GPU. The ILT models are implemented based on the OpenILT framework [23]. We pretrain the pattern generation and ILT models for 32 epochs using a batch size of 8 and the AdamW optimizer [24] with a learning rate of 10^{-3}. For finetuning, we train the models for 8 epochs with a learning rate of 10^{-5}.

Following SOTA ILT works AdaOPC [25] and LithoBench [13], we obtain data by cropping the metal 1 layer of the *gcd* layout generated by OpenROAD [26]. Each tile has a size of $1024 \times 1024nm^2$ and the cropping stride is 256. For the training of pattern generation, the tiles are converted to 32×32 squish patterns. For the ILT model, each tile is padded to 2048×2048 and then scaled to 512×512. The numbers of training and testing data are 7,592 and 844, respectively.

To show that EMOGen can enhance the performance of mask optimization, we compare the ILT models under two training schemes. In the finetuning scheme, we train the ILT model without the help of pattern generation. The co-evolution scheme refers to the proposed method. To make a fair comparison, the hyper-parameters under the two training schemes are kept the same. We test the following ILT models in this experiment.

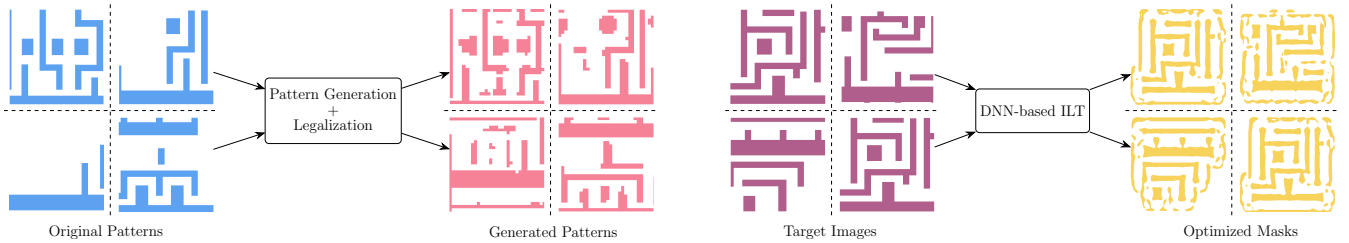
GAN-OPC [6] It follows the design of generative adversarial network (GAN) [20]. In addition to the GAN loss functions, it uses the $L2(\boldsymbol{Z}_{\text{nom}}, \boldsymbol{Z}_T)$ as an additional objective.

Neural-ILT [7] A UNet [27] is utilized in Neural-ILT to predict the optimized mask, minimizing $L2(\boldsymbol{Z}_{\text{nom}}, \boldsymbol{Z}_T) + PVB(\boldsymbol{Z}_{\max}, \boldsymbol{Z}_{\min})$.

DAMO [8] It improves the GAN for ILT with the backbone based on UNet++ [28] and a multiscale discriminator.

CFNO [11] Combining the basic principles of Vision Transformer (ViT) [29] and Fourier Neural Operator (FNO) [30], the CFNO model is designed for efficient and effective ILT.

Table 2 presents the ILT results under the finetuning and co-evolution schemes. EMOGen achieves huge improvement in L2,



(a) (b)

Figure 7: Examples from the trained pattern generation and ILT models. (a) The pattern generation model generates new patterns after legalization. (b) The ILT model takes target images as input and outputs the optimized masks.

Table 3: Comparison on the Legalization of Generated Patterns

Metric	μ_V	L2	PVB	EPE
No Legalization [2]	0.094	51777	57149	22.0
Design Rules Only [5]	0.070	50782	57335	21.4
Design Rules + ILT (ours)	**0.062**	**64800**	**65394**	**44.1**

PVB, and EPE on each model. On average, L2 and PVB are reduced by 19% and 13%, respectively. In terms of EPE, EMOGen makes 39% improvement. Figure 6 visualizes the improvement by showing the normalized L2, PVB, and EPE. The enhancement brought by pattern generation is significant, especially for the DAMO model. According to the results in Table 2, although CFNO is the best model under finetuning, EMOGen helps DAMO to surpass it after co-evolution.

In addition to ILT, we also compare different legalization schemes for pattern generation. We test three schemes, (1) no legalization [2], (2) design rules only [5], (3) design rules + ILT (ours). The metrics include the average violations and ILT performance deterioration. The average violations μ_V refers to the average number of constraint violations among the generated patterns. In terms of ILT performance deterioration, we expect the generated patterns to make the L2, PVB, and EPE worse. In this experiment, we use the pretrained NeuralILT model for ILT.

As presented in Table 3, our method has the lowest average violations and highest ILT performance deterioration. It shows that the ILT-aware legalization in EMOGen can not only reduce the number of design constraint violations by 34%, but also deteriorate the ILT performance by up to 100%. Figure 7 presents some examples from our pattern generation and ILT flows. The pattern generation model generates new patterns after legalization. The ILT model takes target images as input and outputs the optimized masks.

5 Conclusion

We propose EMOGen to enable the co-evolution of pattern generation and mask optimization. The novel ILT-aware pattern generation method achieves up to 39% enhancement in mask optimization. Our legalization step brings up to 34% improvement in design constraint violations. We hope EMOGen can provide a new methodology for enhancing DNN-based mask optimization.

References

[1] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *Proc. VTS*, 2018.

[2] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "DeePattern: Layout pattern generation with transforming convolutional auto-encoder," in *Proc. DAC*, 2019.

[3] X. Zhang, J. Shiely, and E. F. Young, "Layout pattern generation and legalization with generative learning models," in *Proc. ICCAD*, 2020.

[4] L. Wen *et al.*, "LayouTransformer: generating layout patterns with transformer via sequential pattern modeling," in *Proc. ICCAD*, 2022.

[5] Z. Wang *et al.*, "Diffpattern: Layout pattern generation via discrete diffusion," *Proc. DAC*, 2023.

[6] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. DAC*, 2018.

[7] B. Jiang, L. Liu, Y. Ma, H. Zhang, B. Yu, and E. F. Young, "Neural-ILT: Migrating ILT to neural networks for mask printability and complexity co-optimization," in *Proc. ICCAD*, 2020.

[8] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *Proc. ICCAD*, 2020.

[9] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "DevelSet: Deep neural level set for instant mask optimization," in *Proc. ICCAD*, 2021.

[10] Q. Wang, B. Jiang, M. D. F. Wong, and E. F. Y. Young, "A2-ILT: GPU accelerated ilt with spatial attention mechanism," in *Proc. DAC*, 2022.

[11] H. Yang and H. Ren, "Enabling scalable ai computational lithography with physics-inspired models," in *Proc. ASPDAC*, 2023.

[12] H.-C. Shao, C.-W. Lin, and S.-Y. Fang, "Data-driven approaches for process simulation and optical proximity correction," in *Proc. ASPDAC*, 2023.

[13] S. Zheng, H. Yang, B. Zhu, B. Yu, and M. Wong, "LithoBench: Benchmarking ai computational lithography for semiconductor manufacturing," in *Proc. NeurIPS*, 2023.

[14] H. H. Hopkins, "The concept of partial coherence in optics," in *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 208, no. 1093, 1951.

[15] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *Proc. ICCAD*, 2013.

[16] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *Proc. DAC*, 2014.

[17] S. Sun, F. Yang, B. Yu, L. Shang, and X. Zeng, "Efficient ilt via multi-level lithography simulation," in *Proc. DAC*, 2023.

[18] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A gpu-enabled level set method for mask optimization," in *Proc. DATE*, 2021.

[19] Z. He, Y. Zuo, J. Jiang, H. Zheng, Y. Ma, and B. Yu, "OpenDRC: An efficient open-source design rule checking engine with hierarchical GPU acceleration," in *Proc. DAC*, 2023.

[20] I. Goodfellow *et al.*, "Generative adversarial nets," *Proc. NIPS*, vol. 27, 2014.

[21] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. ICLR*, 2014.

[22] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Proc. NeurIPS*, vol. 32, 2019.

[23] S. Zheng, B. Yu, and M. Wong, "OpenILT: An open source inverse lithography technique framework," in *Proc. ASICON*, 2023.

[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2018.

[25] W. Zhao *et al.*, "AdaOPC: A self-adaptive mask optimization framework for real design patterns," in *Proc. ICCAD*, 2022.

[26] T. Ajayi *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *Proc. DAC*, 2019.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015.

[28] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," in *Proc. DLMIA*, 2018.

[29] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Proc. ICLR*, 2021.

[30] Z. Li *et al.*, "Fourier neural operator for parametric partial differential equations," in *Proc. ICLR*, 2020.