Building up End-to-end Mask Optimization Framework with Self-training

Bentian Jiang
CSE Department, CUHK
btjiang@cse.cuhk.edu.hk

Xiaopeng Zhang
CSE Department, CUHK
xpzhang@cse.cuhk.edu.hk

Lixin Liu
CSE Department, CUHK
lxliu@cse.cuhk.edu.hk

Evangeline F.Y. Young
CSE Department, CUHK
fyyoung@cse.cuhk.edu.hk

ABSTRACT

With the continuous shrinkage of device technology node, the tremendously increasing demands for resolution enhancement technologies (RETs) have created severe concerns over the balance between computational affordability and model accuracy. Having realized the analogies between computational lithography tasks and deep learning-based computer vision applications (e.g., medical image analysis), both industry and academia start gradually migrating various RETs to deep learning-enabled platforms. In this paper, we propose a unified self-training paradigm for building up an end-to-end mask optimization framework from undisclosable layout patterns. Our proposed flow comprises (1) a learning-based pattern generation stage to massively synthesize diverse and realistic layout patterns following the distribution of the undisclosable target layouts, while keeping these confidential layouts blind for any successive training stage, and (2) a complete self-training stage for building up an end-to-end on-neural-network mask optimization framework from scratch, which only requires the aforementioned generated patterns and a compact lithography simulation model as the inputs. Quantitative results demonstrate that our proposed flow achieves comparable state-of-the-art (SOTA) performance in terms of both mask printability and mask correction time while reducing 66% of the turn around time for flow construction.

CCS CONCEPTS

• **Hardware → Design for manufacturability**.

KEYWORDS

Mask Optimization; Optical Proximity Correction; Deep Learning

ACM Reference Format:
Bentian Jiang, Xiaopeng Zhang, Lixin Liu, and Evangeline F.Y. Young. 2021. Building up End-to-end Mask Optimization Framework with Self-training . In *Proceedings of the 2021 International Symposium on Physical Design (ISPD '21), March 22–24, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3439706.3447050

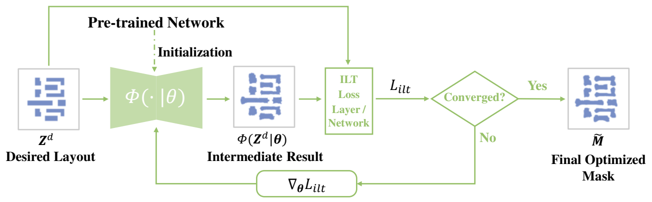


Figure 1: DNN-ILT, a typical end-to-end on-neural-network ILT correction (mask optimization) framework [1].

1 INTRODUCTION

The mismatches between lithography system wavelengths and ever-shrinking device feature sizes lead to inevitable distortion effects on the printed patterns. In the past two decades, industry has relied more on the computational lithography simulation models and resolution enhancement techniques (RETs) to guarantee the functionality and high-volume yields for complex circuit designs in advanced technology nodes.

Under the guidance of rigorous and compact lithography simulation models, RETs, including but not limited to, optical proximity correction (OPC), sub-resolution assist feature (SRAF) insertion, phase-shifting masks (PSM), etc., aim to help all on-mask patterns achieving proper exposures. As one of the most prevailing RETs, current mainstream OPC compensates the printed image errors (caused by non-linear effects, optical diffraction and interference effects, resist, etc.) by modifying the on-mask features with either empirical or numerical approaches.

Empirical OPC approach usually refers to rule-based OPC, where for a given lithography process, the corrections are made empirically by modifying the features according to a set of pre-determined rules [2]. Rule-based OPC is extremely fast, but only suitable for less complex designs due to its faltering accuracy and robustness for advanced technology nodes. Starting from sub-90nm generation, industry began to switch to forward model-based OPC, where the predicted resist image from a compact lithography simulator drives the movements of the on-mask feature edges iteratively until a preset tolerance of printing error is met [2]. Model-based OPC has been adopted ubiquitously for 193i layers since it achieves a reasonable balance between model accuracy and computational overhead. On the other hand, the numerical OPC approach, which is also referred as inverse lithography technique (ILT), treats OPC as an inverse imaging problem and performs pixel-wise mask optimization by gradient descent to search for the optimum mask shape yielding the desired resist shape. ILT correction is more aggressive and is able to achieve the best possible process window obtainable for 193i layers [3], and recent study shows even EUV patterning for sub-7nm also has similar demands for ILT-style corrections [4].

Apparently, the effectiveness of either model-based OPC or ILT-based correction heavily relies on the accuracy of the lithography simulation model. Among various computational simulation recipes, rigorous lithography simulation is accomplished by solving Maxwell's equations for the electromagnetic field (EMF) with rigorous EMF solvers such as rigorous coupled wave analysis (RCWA), finite-difference time-domain (FDTD) and pseudo-spectral time-domain (PSTD) [5]. Rigorous lithography simulation is capable of modeling the mask topography effects (3D mask effects), aerial imaging and resist development. Despite their superior accuracy, performing full-chip rigorous simulations in every iteration of the OPC process is generally not practical. In contrast to rigorous models, compact lithography simulation models combine a reasonably accurate and physically correct aerial imaging model (e.g., solving partially coherent imaging calculations with Hopkins approach) with a simple resist model [2] (e.g., a constant or variable threshold resist model). In practice, integrating compact lithography simulation models into model-based OPC and ILT generally can be regarded as a compromising solution to trade-off between result quality and computational affordability.

Inspired by the analogies between lithography imaging and typical deep learning-based computer vision applications such as medical imaging, deep neural networks (DNNs) has prevalently become a proper alternative to carry out various lithographic-related tasks, which includes (1) approximating rigorous lithography simulations on 3D mask-aware aerial imaging [6, 7] and resist effects modeling [8]; (2) performing end-to-end mask optimization on deep learning platforms [1, 9, 10]; (3) applying DNNs on fast early stage hotspots detection [11], etc. Among them, Ye *et al.* [6] proposes a generative learning-based framework which is capable of predicting 3D aerial images with accurate modeling of the mask topography effects while achieving orders of magnitude speedup. Jiang *et al.* [1] casts a compact lithography simulation model as a neural network layer and hereby converts the ILT correction process as a DNN training procedure to achieve end-to-end mask optimization. Liu *et al.* [10] pre-trains multiple DNN-based rigorous lithography simulators (DRLS) for modeling 3D mask effects, aerial imaging and resist effects, these DRLSs are then integrated into the mask synthesis flow for guiding the entire mask optimization process.

As proof-of-concept (POC) studies, the above seminal works have successfully verified the feasibility and significant runtime superiority of leveraging deep learning paradigm to carry out computational lithographic tasks. However, there are still gaps between POC evaluations and realistic massive deployments. On one hand, academia researchers can hardly obtain the opportunity to evaluate ideas using industrial resources (e.g., design layouts and lithography simulation recipes) without restrictive permissions. As a result, some recent works from academia [1, 6, 8, 9, 12–14] are still widely using legacy problem settings or evaluating on contact-layers only, which may not be able to reflect the newest demands from industry. On the other hand, implementing numerical optimization flow such as ILT from scratch is generally time-consuming. Extensive efforts, including fine-tuning the parameters, enabling proper parallelism, polishing the efficiency optimizer, etc. (which are well-developed features in deep learning toolkits), are required to achieve a reasonable performance.

In accordance with these observations, we are aiming at finding a unified solution for above concerns by fully leveraging the efficiency of off-the-shelf deep learning software and hardware, and achieves reasonably good performance with breakthough reduction on the development overheads. In this paper, we presented a unified self-contained training paradigm for building up an end-to-end learning-based mask optimization framework from scratch (a typical flow is shown in Figure 1). Our proposed flow makes use of (1) a learning-based pattern generation stage to massively synthesize diverse and realistic layout patterns following the distribution of the undisclosable target layouts, while keeping these confidential layouts blind for any successive training stage, and (2) a complete self-training stage for building up an end-to-end learning-based mask optimization framework from scratch, which only requires the aforementioned generated patterns and a compact lithography simulation model as the inputs.

To the best of knowledge, this is the first work that attempts to systematically build up an end-to-end mask optimization flow entirely on deep learning platform (i.e., `PyTorch`) **without any pre-labelled data**. Note that, the only open-sourced industrial package (with 32*nm* or below tech node) available for us is from ICCAD 2013 mask optimization contest [15]. Considering such limitation, we perform all evaluations based on the compact lithography simulation model and benchmarks provided by them. Nevertheless, the proposed concept is a unified paradigm for fab and vendor as it can be naturally migrated to rigorous simulation environment by replacing the compact lithography simulation model with other DRLSs [10]. Our key contributions are highlighted as follows.

- We build up an end-to-end learning-based mask optimization framework from scratch (completely on deep learning platform) with the self-training scheme.
- A generative learning-based pattern generator is applied to massively synthesize diverse layout patterns while preserving the major characteristics of the given confidential layouts. A simple third-party reverse attack is also performed to empirically study the effectiveness of the generator for protecting the confidential layouts from disclosure.
- Our self-training scheme does not required any pre-labelled mask data. Only a set of generated patterns and a compact litho-simulation model are needed as the inputs.
- Experimental evaluation demonstrate that our proposed flow achieves comparable SOTA performance in terms of both mask printability and mask correction time while reducing 66% of the turn around time for flow construction.

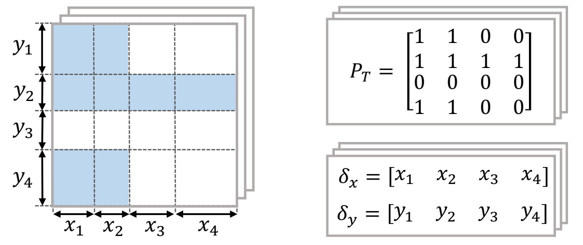
The rest of the paper is organized as follows. Section 2 lists some preliminaries. Section 3 discusses the details of the framework and algorithms. Section 4 presents experimental results, followed by a conclusion in Section 5.

2 PRELIMINARIES

In this section, we will provide preliminaries for the background and formulate the problem. For convenience, notations used in this paper are listed in table 1.

Table 1: Notations used in this paper.

Z^{d}	Desired target layout
$\boldsymbol{M}$	Optimized mask
$\boldsymbol{I}$	Aerial images
$\boldsymbol{R}_{\mathrm{nom}}$	Nominal process condition: nominal dose and focus $\boldsymbol{H}$
$\boldsymbol{R}_{\mathrm{max}}$	Max process condition: max dose and nominal focus $\boldsymbol{H}$
$\boldsymbol{R}_{\mathrm{min}}$	Min process condition: min dose and defocus $\boldsymbol{H}_{\mathrm{def}}$
$Z,\ Z_{\mathrm{in}},\ Z_{\mathrm{out}}$	Binary wafer image under the $\boldsymbol{R}_{\mathrm{nom}}, \boldsymbol{R}_{\mathrm{min}}, \boldsymbol{R}_{\mathrm{max}}$
$\boldsymbol{H}$	A set of optical kernels $\{\boldsymbol{h}_1, \dots, \boldsymbol{h}_K\}$
$\otimes$	Convolution operator
$f(\cdot \mid \boldsymbol{R})$	Lithography simulation under a process condition $\boldsymbol{R}$
$\Phi(\cdot \mid \boldsymbol{\theta})$	Neural network forward function with weights $\boldsymbol{\theta}$



$$P_T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\delta_x = [x_1 \quad x_2 \quad x_3 \quad x_4]$$
$$\delta_y = [y_1 \quad y_2 \quad y_3 \quad y_4]$$

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

Figure 2: Squish pattern representation.

2.1 Compact Lithography Simulation Model

A compact lithography model combines a reasonably accurate and physically correct aerial imaging model with a very simple resist mode [2]. In practice, Hopkins' approach is widely used for modeling the partially coherent lithography systems (193i system with annular illumination in this work), where an approximate solution to the Hopkins imaging equations called Sum of Coherent Sources (SOCS) can be applied to calculate the aerial image. Theoretically, the aerial image can be obtained by convolving the mask $\boldsymbol{M}$ with a set of pre-computed coherent convolution kernels $\boldsymbol{H}$ and taking the weighted summation with corresponding coefficients $\boldsymbol{\omega}$:

$$\boldsymbol{I}(x,y) = \sum_{k=1}^{\infty} \omega_k |\boldsymbol{M}(x,y) \otimes \boldsymbol{h}_k(x,y)|^2 \approx \sum_{k=1}^{N} \omega_k |\boldsymbol{M}(x,y) \otimes \boldsymbol{h}_k(x,y)|^2, \tag{1}$$

where $\boldsymbol{h}_k$ and ω_k are the k^{th} kernel and weight. As the coefficients quickly decaying to zero, the infinite summation will be cut off at some number N, where $N = 24$ in our implementation. The constant threshold resist (CTR) model is then used to model the resist action for developing the final binary wafer image $\boldsymbol{Z}$. It is described with a step function $\Gamma(\cdot)$:

$$Z(x,y) = \Gamma(\boldsymbol{I}(x,y)) = \begin{cases} 1, & \text{if } \boldsymbol{I}(x,y) \geq I_{th}, \\ 0, & \text{if } \boldsymbol{I}(x,y) < I_{th}, \end{cases} \tag{2}$$

where $I_{th} = 0.225$ is a constant in our implementation.

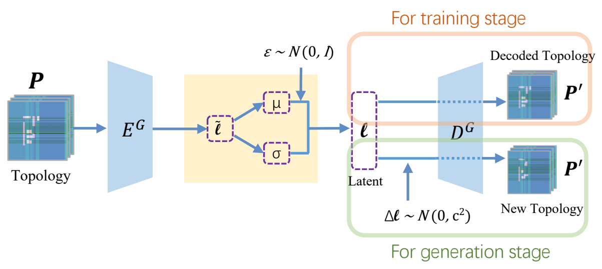


Figure 3: A typical pattern topology generation flow [16].

2.2 Learning-based Pattern Generation

To produce diverse new patterns, we apply the approach in [16], a generative learning-based pattern generation framework for pattern topology generation and legalization. This approach has the capability of learning inherent distributions of two-dimensional layouts and generate realistic patterns.

In [16], the layout patterns are firstly converted to squish pattern representation composed of a topology matrix and two geometry vectors (as shown in Figure 2). The topology matrix contains most of the information of a layout pattern and serves as the input of the neural networks in [16].

As depicted in Figure 3, the pattern topology generation model has an encoder-reparameterization-decoder pipeline for feature learning and pattern reconstruction. In the training stage, the encoder E^G maps a pattern topology $\boldsymbol{P}$ into a Gaussian space with mean μ and variance σ^2. Then the reparameterization block samples a latent vector l by $l = \mu + \sigma \odot \varepsilon$, where ε follows a standard normal distribution, i.e., $\varepsilon \sim \mathcal{N}(0, I)$ 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 $\boldsymbol{P'}$ that are expected to be close to $\boldsymbol{P}$. The objective function of training is as follows:

$$\begin{aligned} min \quad & D_{KL}(\mathcal{N}(\mu, \sigma^2) || \mathcal{N}(0, I)) + \lambda ||\boldsymbol{P} - \boldsymbol{P'}||_F^2, & \text{(3a)} \\ s.t. \quad & \mu = f_R^{\mu}(\tilde{l}, W_R^{\mu}), \quad \sigma = f_R^{\sigma}(\tilde{l}, W_R^{\sigma}), & \text{(3b)} \\ & \tilde{l} = E^G(\boldsymbol{P}, W_E), \quad l = \mu + \sigma \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), & \text{(3c)} \\ & \boldsymbol{P'} = D^G(l, W_D). & \text{(3d)} \end{aligned}$$

The D_{KL} in Equation 3a represents the Kullback-Leibler (KL) divergence which measures the difference between the distribution $\mathcal{N}(\mu, \sigma^2)$ and a standard Gaussian distribution $\mathcal{N}(0, I)$. $\tilde{l}$ is an internal representation of the output from the encoder, while l is a sampled latent vector with Gaussian distribution from the reparameterization block. Besides, W_R^{μ}, W_R^{σ}, W_E and W_D represent the trainable weights associated with the respective units.

In the topology generation stage, the latent vector l in Gaussian space allows perturbation by adding a random Gaussian vector $\triangle l$, as shown in Figure 3. Apparently the new vector $l + \triangle l$ is still in Gaussian distribution, which implies it can be converted back to the pattern topology space by the trained decoder D^G and generate a new pattern topology with Equation 4:

$$\boldsymbol{P'} = D^G(l + \triangle l, W_D), \quad \triangle l_i \sim \mathcal{N}(0, c^2), \quad i = 1, ..., K, \tag{4}$$

where $\triangle l$ is sampled from a Gaussian distribution with mean 0 and variance c^2. The c^2 is a user-defined parameter that controls the range of perturbation. where c^2 is set to 0.5 in our implementation.

2.3 End-to-end Learning-based Mask Optimization by ILT

As depicted in Figure 1, DNN-ILT is a typical end-to-end on-neural-network ILT correction framework [1] which consists of (1) a pre-trained backbone neural network to perform layout-to-mask translation, (2) an ILT correction layer to calculate ILT loss and corresponding gradient, and (3) other customized layers for optimizing user-specific objectives (e.g., process variation band).

In DNN-ILT, a compact lithography simulation model (section 2.3) is re-implemented for GPU with `CUDA`. The CUDA-based litho-simulator is then used to implement an ILT correction layer, where its forward function calculates the forward propagation ILT loss:

$$L_{\text{ilt}}(\boldsymbol{Z}^{\text{d}} \mid \boldsymbol{\theta}) = ||\boldsymbol{Z} - \boldsymbol{Z}^{\text{d}}||_2^2 = ||f(\Phi(\boldsymbol{Z}^{\text{d}}|\boldsymbol{\theta}) \mid \boldsymbol{R}_{\text{nom}}) - \boldsymbol{Z}^{\text{d}}||_2^2, \quad (5)$$

and the backward function computes the backward propagation ILT gradient $\nabla_{\boldsymbol{M}} L_{\text{ilt}} = \frac{\partial L_{\text{ilt}}}{\partial \boldsymbol{M}}$ with respect to mask $\boldsymbol{M}$. Since the predecessor backbone network's forward prediction $\Phi(\boldsymbol{Z}^{\text{d}}|\boldsymbol{\theta})$ yields a mask $\boldsymbol{M}$ (we omit the sigmoid approximation here for simplicity), its weights $\boldsymbol{\theta}$ can be updated accordingly through the chain rule $\nabla_{\boldsymbol{\theta}} L_{\text{ilt}} = \frac{\partial L_{\text{ilt}}}{\partial \boldsymbol{M}} \frac{\partial \boldsymbol{M}}{\partial \Phi(\boldsymbol{Z}^{\text{d}}|\boldsymbol{\theta})} \frac{\partial \Phi(\boldsymbol{Z}^{\text{d}}|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

As a consequence, given a desired target layout $\boldsymbol{Z}^{\text{d}}$ as input, the on-neural-network ILT correction process is essentially an unsupervised training procedure for the backbone network to minimize the ILT loss in Equation (5). This is an end-to-end procedure, since the mask optimization process will be performed iteratively until the preset tolerance is met.

2.4 Term Definitions

We further define some concepts used in this paper to highlight the features of our proposed flow.

2.4.1 Semi-supervised Learning. Semi-supervised learning (SSL) is a learning paradigm concerned with the design of models using a small amount of labeled data in conjunction with a large amount unlabeled data for training [17].

2.4.2 Self-training. Self-training can be regarded as a wrapper method for semi-supervised learning [17]. Considering the labelled dataset with m samples $\boldsymbol{L} = \{(\boldsymbol{x}^1, \boldsymbol{y}^1), (\boldsymbol{x}^2, \boldsymbol{y}^2), ..., (\boldsymbol{x}^m, \boldsymbol{y}^m)\}$ where $\boldsymbol{x}^i \in \mathbb{X}, \boldsymbol{y}^i \in \mathbb{Y}, \forall i \in \{1, ..., m\}$, and an unlabelled dataset with l samples $\boldsymbol{U} = \{\boldsymbol{x}^{m+1}, \boldsymbol{x}^{m+2}, ..., \boldsymbol{x}^{m+l}\}$, where $l >> m$. Self-training aims to seek a mapping $\Phi : \mathbb{X} \rightarrow \mathbb{Y}$ by repeating the followings:

(1) Train a model Φ from set $\boldsymbol{L}$.
(2) Apply Φ to predict the pseudo-labels of k samples in set $\boldsymbol{U}$.
(3) Remove the predicted k samples from set $\boldsymbol{U}$ and add them to set $\boldsymbol{L}$, until there is no sample in set $\boldsymbol{U}$.

2.4.3 Complete Self-training. In this paper, we further extend the concept of self-training to complete self-training. In contrast to classic self-training, we add two more features to complete self-training:

(1) The input training dataset for complete self-training paradigm **only contains unlabelled** data.
(2) In contrast to directly generating the pseudo-labels for unlabelled samples by the network inference, the complete self-training paradigm produces labels by performing unsupervised training on the network itself.

2.5 Problem Formulation

The ultimate goal for OPC (mask optimization) is to enhance the mask printability, which can be measured by squared L$_2$ error and process variation band (PVBand).

Definition 1 (Squared L$_2$ Error)**.** Given the desired target layout $\boldsymbol{Z}^{\text{d}}$ and the wafer image $\boldsymbol{Z}$ of a mask $\boldsymbol{M}$ where $\boldsymbol{Z} = f(\boldsymbol{M} \mid \boldsymbol{R}_{\text{nom}})$, the squared L$_2$ error is given by $||\boldsymbol{Z} - \boldsymbol{Z}^{\text{d}}||_2^2$.

Definition 2 (Process Variation Band)**.** Process variation band (PVBand) denotes the contour area variations under ±2% dose error, which is measured by the summation of bitwise-XOR between $\boldsymbol{Z}_{\text{in}} = f(\boldsymbol{M} \mid \boldsymbol{R}_{\text{min}})$ and $\boldsymbol{Z}_{\text{out}} = f(\boldsymbol{M} \mid \boldsymbol{R}_{\text{max}})$.

Problem 1 (End-to-end Learning-based Mask Optimization)**.** Given an undisclosable layout pattern set $\mathfrak{Z}^{\text{u}}$. Build up an end-to-end learning-based ILT correction framework for $\mathfrak{Z}^{\text{u}}$ without directly involving $\mathfrak{Z}^{\text{u}}$ in the training. The constructed framework should be able to generate optimized mask solutions for testing layouts to simultaneously minimizes (1) squared L2 error, (2) PVBand and (3) turn around mask optimization time.

3 PROPOSED FLOW

3.1 Overview

Figure 4 presents the overall flow of how to build up an end-to-end learning-based mask optimization framework from scratch. Our proposed flow consists of two stages, (1) a pattern generation stage that applies a generative learning-based model to sample from the distribution of a given realistic but confidential layouts and to massively synthesize diverse fake layouts with similar characteristics (Figure 4(b)); and (2) a complete self-training stage that only leverages the aforementioned generated fake layout patterns and a GPU-based compact lithography simulation model to construct an entire on-neural-network ILT correction (DNN-ILT) framework from scratch (Figure 4(a), (c), (d), (e)).

As depicted in Figure 4, given an undisclosable layout pattern set $\mathbb{Z}^{\text{u}}$, the pattern generation stage first synthesizes a new layout pattern set $\mathbb{Z}^{\text{d}}$ following the distribution of $\mathbb{Z}^{\text{u}}$ (i.e., similar design style and design space coverage). We then split $\mathbb{Z}^{\text{d}}$ into T batches randomly, i.e., $\mathbb{Z}^{\text{d}} = \bigcup_{i=1}^{T} \mathbb{Z}_i^{\text{d}}$. At the self-training stage, the t^{th} batch of generated layout patterns $\mathbb{Z}_t^{\text{d}}$ is fed to the DNN-ILT self-training procedure at t^{th} iteration, through which the mask labels $\tilde{\mathbb{M}}_t^{\text{d}}$ for $\mathbb{Z}_t^{\text{d}}$ is generated by performing on-neural-network ILT correction (Section 2.3) using our pre-trained backbone model $\boldsymbol{\theta}_{t-1}$ obtained from the last iteration, and $(\mathbb{Z}_t^{\text{d}}, \tilde{\mathbb{M}}_t^{\text{d}})$ are then leveraged to pre-train a new backbone model $\boldsymbol{\theta}_t$ in preparation for the DNN-ILT self-training at the $(t+1)^{\text{th}}$ iteration. This DNN-ILT self-training procedure (Figure 4(c)) is conducted in an iterative manner until all the batches in $\mathbb{Z}^{\text{d}}$ are consumed and the final output model $\boldsymbol{\theta}_{\text{pt}}^*$ is deployed for testing. As a result, the self-training stage is able to construct an end-to-end learning-based mask optimization framework (based on $\boldsymbol{\theta}_{\text{pt}}^*$) from scratch with an unlabelled fake pattern set $\mathbb{Z}^{\text{d}}$ without seeing any confidential pattern in set $\mathbb{Z}^{\text{u}}$.

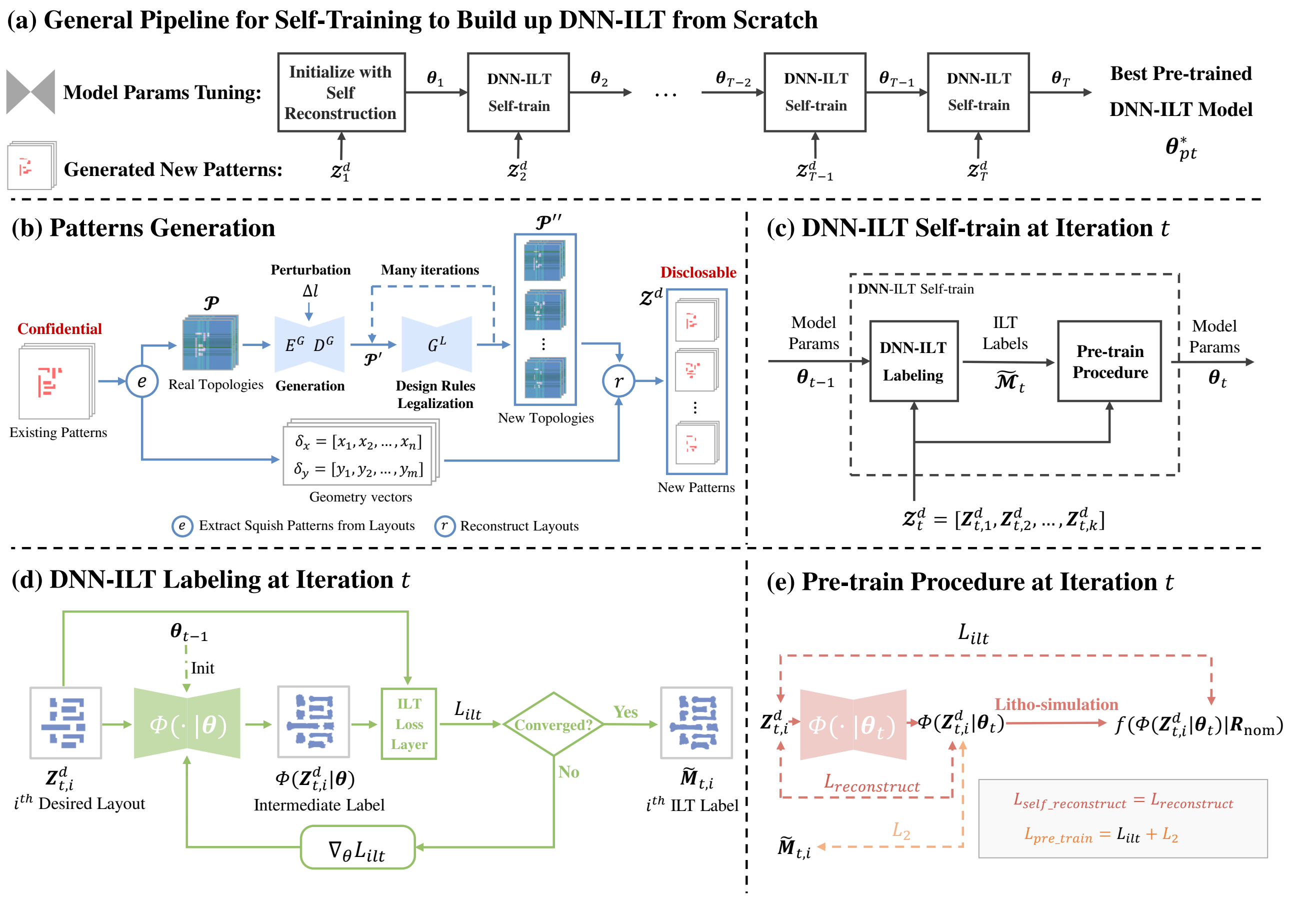


Figure 4: The self-training paradigm for constructing end-to-end learning-based mask optimization framework from scratch.

However, we are going to show that, it is capable of handling end-to-end mask optimizations for layouts from $\mathcal{Z}^u$, since $\mathcal{Z}^d$ is synthesized following the distribution of $\mathcal{Z}^u$.

In the rest of this section, we will present the algorithmic details of the proposed flow.

3.2 Learning-based Pattern Generation and Third-party Reverse Attack

3.2.1 Learning-based Pattern Generation. The primary objective of integrating a generative learning-based pattern generation model is to massively synthesize diverse fake layout patterns while preserving the characteristics of the given confidential realistic layouts (i.e., similar design style and design space coverage), and those useful characteristics can hopefully be captured by the later training procedures using directly the generated layouts, thereby maintaining the confidentiality of the undisclosable layouts to other third-party applications (mask optimization in our case).

As depicted in Figure 4(b), this forward pattern generation process can be achieved with the topology generation-legalization pipeline. Given an undisclosable layout pattern set $\mathcal{Z}^u$, the corresponding squish-representation set, which consists of a pattern topology set $\mathcal{P}$ and a geometry vector set $\boldsymbol{\delta}$, can be extracted accordingly. The encoder-reparameterization-decoder pipeline (Section 2.2) can be trained by the pattern topology set $\mathcal{P}$ with the objective of minimizing the loss in Equation (3). By introducing the random perturbations $\Delta \boldsymbol{l} \sim \mathcal{N}(0, c^2)$ on each latent vector l (an internal representation before decoding), a set of new pattern topologies can be generated by the decoder D^G as $\mathcal{P}' = D^G(\boldsymbol{l} + \triangle \boldsymbol{l}, W_D)$. The new pattern topology set $\boldsymbol{P}'$ will further go through the design rules legalization stage (please refer to [16]). Finally, the newly generated pattern set $\mathcal{Z}^d$ can be uniquely reconstructed by combining the legalized topology set $\mathcal{P}''$ and geometry vector set $\boldsymbol{\delta}$.

The capabilities of this generative learning-based model for preserving similar design style and design space coverage have been quantitatively verified in [16]. From the perspective of layout owner, the reversibility of these fake patterns to retrieve the original true patterns emerges as a critical concern. To what extent can the generated patterns $\mathcal{Z}^d$ prevent the mother templates $\mathcal{Z}^u$ from being re-constructed?

3.2.2 Backward Third-party Attack. We empirically study the reversibility of the pattern generation process through a hacking heuristic under the assumption that all the associated generative models belong to the patterns owner, and **only** the generated patterns $\mathcal{Z}^d$ are exposed to other end-users. Observing that the geometry vectors remain unchanged during the pattern generation process, recovering an undisclosable layout $\boldsymbol{Z}^u$ is essentially to recover the corresponding $\boldsymbol{P}^u$ from its generated topologies $\mathcal{P}'_u$. Apparently, when we have sufficiently large number of randomly

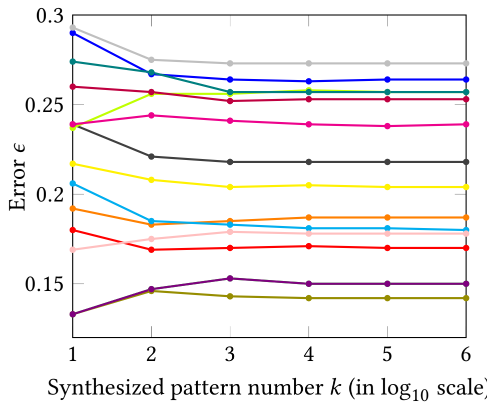


Figure 5: Errors of reverse attacks on 15 samples layout patterns are getting converged with respect to the growth of synthesized patterns.

perturbed latent vectors, the difference between the mean of $\boldsymbol{l}'_{\mathrm{u}}$ and l_{u} will be vanished, and hereby recover the $\boldsymbol{P}^{\mathrm{u}}$, that is, as $k \longrightarrow \infty$,

$$D^G\left(\frac{\sum_{i=1}^{k} l'_{\mathrm{u},i}}{k}, W_D\right) = D^G\left(\frac{\sum_{i=1}^{k}(l_{\mathrm{u}} + \triangle l_i)}{k}, W_D\right) = D^G(l_{\mathrm{u}}, W_D) = \boldsymbol{P}^{\mathrm{u}}, \tag{6}$$

where $\Delta l_i \sim \mathcal{N}(0, c^2)$, $\forall i \in \{1, ..., k\}$. Inspired by this observation, a hacking heuristic in the view of a third-party user can be described as follows to reversely attack the disclosed generated patterns $\mathcal{Z}^{\mathrm{d}}$: (1) extract the corresponding topology matrix set $\mathcal{P}'$ and geometry vectors set $\boldsymbol{\delta}$; (2) assign all topology matrices with the same δ into a batch; (3) for each batch, calculate the mean topology matrix and combine it with δ to reconstruct the pattern.

Note that we omit the design rules legalization steps in Figure 4(b) due to the hardness of modeling the complicated legalization behaviors in [16]. In general, legalization steps introduce more nonlinearities which may further hinder the pattern reversibility.

We simulate this attack on 15 sample layouts following the design specifications for 32nm M1 layout topologies provided in ICCAD 2013 mask optimization contest [15], and measure the reconstruction error ϵ by $\epsilon = \frac{||\frac{\sum_{i=1}^{k} P'_i}{k} - \boldsymbol{P}||_F}{||\boldsymbol{P}||_F}$, where for each undisclosable pattern topology $\boldsymbol{P}$, $k = \{10, 10^2, ..., 10^6\}$ is the number of synthesised pattern topologies $\boldsymbol{P}'$ exposed to the attacker. Quantitative results are shown in Figure 5. We can see that, the reconstruction error ϵ for each sample layout topology $\boldsymbol{P}$ converges when the number of synthesized patterns increases, but **none** of these attacks has achieved a sufficiently small ϵ at convergence. Such phenomenon indicates that some biases (unknown for the end-users) may be introduced by the nonlinear decoding procedure D^G in the layout pattern generator, and as a result, the property in Equation (6) does not follow accurately.

3.3 Complete Self-training Paradigm

As mentioned in Section 2.3, the end-to-end on-neural-network ILT correction process (DNN-ILT) is essentially an unsupervised training procedure of a pre-trained backbone network $\boldsymbol{\theta}$ for minimizing the ILT loss

$$L_{\mathrm{ilt}}(Z^{\mathrm{d}} \mid \boldsymbol{\theta}) = ||Z - Z^{\mathrm{d}}||_2^2 = ||f(\Phi(Z^{\mathrm{d}}|\boldsymbol{\theta}) \mid \boldsymbol{R}_{\mathrm{nom}}) - Z^{\mathrm{d}}||_2^2,$$

where Z^{d} denotes the desired target layout, $f(\cdot \mid \boldsymbol{R})$ is lithography function under a nominal process condition, and $\Phi(\cdot \mid \boldsymbol{\theta})$ is forward prediction of model $\boldsymbol{\theta}$. Apparently, the backbone network $\boldsymbol{\theta}$ is capable of conducting the layout-to-mask translation for DNN-ILT, while different pre-trained backbone networks $\boldsymbol{\theta}_{\mathrm{pt}}$ will yield different **initial** mask predictions for DNN-ILT.

According to the studies in [1], two major motivations drive us to perform ILT correction on a pre-trained neural network.

(1) Comparing to conventional ILT, DNN-ILT possesses higher searching efficiency (with DL platforms built-in optimizers) and significantly larger search space (optimizing network neurons vs. optimizing binary on-mask pixels), which enable effective numerical optimization.
(2) ILT is generally sensitive to initial solution quality. A well pre-trained model $\boldsymbol{\theta}_{\mathrm{pt}}$ is capable of predicting a better initial mask solution which helps to achieve faster and better convergence.

However, the DNN-ILT described in [1] still suffers from the extremely time-consuming data-preparation stage during the flow construction, since its backbone model is pre-trained purely in a supervised manner, where all the mask labels are synthesized by a conventional ILT tool [18]. In contrast to this approach, which constructs DNN-ILT via supervised-learning, our complete self-training paradigm aims to build up DNN-ILT from scratch **without any pre-labelled mask data** and hereby "eliminate" the need of a conventional data-preparation procedure.

As depicted in Figure 4(a), the pipeline of our complete self-training stage consumes T iterations of DNN-ILT self-training processes (Figure 4(c)) to produce the final pre-trained backbone model $\boldsymbol{\theta}^*_{\mathrm{pt}}$ for DNN-ILT. At the t^{th} DNN-ILT self-training iteration, the t^{th} batch of target layout set $\mathcal{Z}^{\mathrm{d}}_t$ is fed to DNN-ILT to generate the corresponding mask labels $\tilde{\mathcal{M}}^{\mathrm{d}}_t$ that can minimize the following mask printability loss,

$$\begin{aligned} L_{\mathrm{msk\_print}} =& \alpha \cdot L_{\mathrm{ilt}} + \beta \cdot L_{\mathrm{pvband}} \\ =& \alpha \cdot ||f(\Phi(\mathcal{Z}^{\mathrm{d}}_t|\boldsymbol{\theta}_{t-1}) \mid \boldsymbol{R}_{\mathrm{nom}}) - \mathcal{Z}^{\mathrm{d}}_t||_2^2 + \\ & \beta \cdot ||f(\Phi(\mathcal{Z}^{\mathrm{d}}_t|\boldsymbol{\theta}_{t-1}) \mid \boldsymbol{R}_{\mathrm{min}}) - f(\Phi(\mathcal{Z}^{\mathrm{d}}_t|\boldsymbol{\theta}_{t-1}) \mid \boldsymbol{R}_{\mathrm{max}})||_2^2, \end{aligned} \tag{7}$$

where α, β are configurable hyper-parameters, $\boldsymbol{\theta}_{t-1}$ is the pre-trained backbone model obtained from the previous DNN-ILT self-training iteration, $\boldsymbol{R}_{\mathrm{nom}}$, $\boldsymbol{R}_{\mathrm{min}}$, $\boldsymbol{R}_{\mathrm{max}}$ are the nominal, minmum and maximum process conditions defined by the lithography simulation model, respectively (we omit the gradient for Equation (7) here due to the limited space, please refer [1] for details).

We use the DNN-ILT (based on $\boldsymbol{\theta}_{t-1}$) to generate the labels for the t^{th} iteration ourselves. This self-labelled dataset $\boldsymbol{L}_t = (\mathcal{Z}^{\mathrm{d}}_t, \tilde{\mathcal{M}}^{\mathrm{d}}_t)$ is applied to pre-train a new backbone model $\boldsymbol{\theta}_t$ to minimize the pre-training loss (Figure 4(e))

$$L_{\mathrm{pre\_train}} = ||\Phi(\mathcal{Z}^{\mathrm{d}}_t|\boldsymbol{\theta}_t) - \tilde{\mathcal{M}}^{\mathrm{d}}_t||_2^2 + \eta||f(\Phi(\mathcal{Z}^{\mathrm{d}}_t|\boldsymbol{\theta}_t) \mid \boldsymbol{R}_{\mathrm{nom}}) - \mathcal{Z}^{\mathrm{d}}_t||_2^2, \tag{8}$$

where η is a configurable hyper-parameter. As ILT is ill-posed, the second term in Equation (8) essentially introduces the domain-knowledge of the partially coherent imaging system and hereby guides $\boldsymbol{\theta}_t$ gradually converged towards a domain-specific direction.

Algorithm 1 Complete Self-training Paradigm for DNN-ILT

```
Input: Undisclosable layout pattern set $\mathcal{Z}^{u}$, lithography simulation model $f(\cdot \mid \boldsymbol{R})$.
Output: The best pre-trained DNN-ILT model $\boldsymbol{\theta}^{*}_{pt}$.
1: $\mathcal{Z}^{d} \leftarrow$ Pattern_Generation($\mathcal{Z}^{u}, c^2$);    ▷ $c^2 = 0.5$
2: Split $\mathcal{Z}^{d}$ into $T$ batches, such that $\mathcal{Z}^{d} = \bigcup_{i=1}^{T} \mathcal{Z}^{d}_{i}$;
3: Pre-train $\boldsymbol{\theta}_1$ with $(\mathcal{Z}^{d}_{1}, \mathcal{Z}^{d}_{1})$ to minimize Equation (9);
4: for Iteration $t = 2,...,T$ do
5:     Initialize DNN-ILT with $\boldsymbol{\theta}_{t-1}$;
6:     $\tilde{\mathcal{M}}^{d}_{t} \leftarrow$ DNN_ILT_Labeling($\mathcal{Z}^{d}_{t}, \boldsymbol{\theta}_{t-1}$);    ▷ Equation (7)
7:     Pre-train $\boldsymbol{\theta}_t$ with $(\mathcal{Z}^{d}_{t}, \tilde{\mathcal{M}}^{d}_{t})$ to minimize Equation (8);
8:     if $\boldsymbol{\theta}_t$ outperforms $\boldsymbol{\theta}^{*}_{pt}$ in evaluation then
9:         $\boldsymbol{\theta}^{*}_{pt} \leftarrow \boldsymbol{\theta}_t$;
10: return best pre-trained DNN-ILT model $\boldsymbol{\theta}^{*}_{pt}$;
```

Note that, in the first DNN-ILT self-training iteration, there is **no** $\boldsymbol{\theta}_0$ to conduct the on-neural-network ILT corrections for $\mathcal{Z}^{d}_{1}$, since we are completely building up the DNN-ILT from scratch with only unlabelled layout dataset $\mathcal{Z}^{d}$. Moreover, we observe that on-neural-network ILT correction process (Equation (7)) **cannot** be directly applied on the network with common default initialization, i.e., initializing the network weights with random values under a certain distribution, because in practice, the ILT gradient will eliminate all on-mask shapes and fall into the trap of a local minimum. In order to avoid the local minimum trap, we enforce the forward prediction of $\boldsymbol{\theta}_1$ to be the first batch data itself, i.e., $\Phi(\mathcal{Z}^{d}_{1}|\boldsymbol{\theta}_1) = \mathcal{Z}^{d}_{1}$. As a result, the $\boldsymbol{\theta}_1$'s pre-training step in the first round self-training is to minimize the self-reconstruction loss

$$L_{\text{self\_reconstruct}} = ||\Phi(\mathcal{Z}^{d}_{1}|\boldsymbol{\theta}_1) - \mathcal{Z}^{d}_{1}||^{2}_{2}. \tag{9}$$

Algorithm 1 depicts the complete flow of our proposed complete self-training paradigm.

4 EXPERIMENTAL RESULTS

In this work, we explore how to build up an end-to-end on-neural-network ILT correction framework from an undisclosable layout pattern set. The proposed flow is developed with `PyTorch` and `CUDA`, and tested on a Linux machine with a 2.2GHz Intel Xeon CPU and an Nvidia Titan V GPU.

4.1 Dataset

4.1.1 Undisclosable layout pattern set. We obtain a layout pattern library from the authors of [12], which consists of 4300 instances of size $2\mu m \times 2\mu m$ based on the design specifications for 32nm M1 layout topologies provided in the ICCAD 2013 mask optimization contest benchmark suite [15]. We regard it as the undisclosable layout pattern set $\mathcal{Z}^{u}$ in the following experiments.

4.1.2 Generated layout pattern set. Taking the undisclosable layout pattern set $\mathcal{Z}^{u}$ as inputs, we generated a new layout pattern set $\mathcal{Z}^{d}$ using the flow described in Section 2.2. For each layout pattern in $\mathcal{Z}^{u}$, we synthesized 4 new pattern topologies with a perturbation variance of $c^2 = 0.5$.

4.1.3 Test dataset. We use the ICCAD 2013 mask optimization contest benchmark suite (blind for training) and the corresponding lithography simulation engine `lithosim_v4` package [15] to evaluate the effectiveness of our proposed flow. The test benchmarks consists of ten $2\mu m \times 2\mu m$ clips for 32nm M1 layout designs.

4.2 Effectiveness of Proposed Flow

The baseline flow [1] pre-trains its backbone model $\boldsymbol{\theta}_{\text{base}}$ directly with the training dataset $(\mathcal{Z}^{u}, \tilde{\mathcal{M}}^{u})$, where a conventional ILT tool [18] is used to synthesize the mask labels $\tilde{\mathcal{M}}^{u}$ for all the 4300 instances in $\mathcal{Z}^{u}$, and it takes approximately 10 days (> 240 hours) to finish the entire construction flow which includes data-preparation and model training.

For our proposed flow, we build up DNN-ILT directly with the unlabelled dataset $\mathcal{Z}^{d}$ which contained 17200 newly generated layout patterns. We randomly split $\mathcal{Z}^{d}$ into 4 batches, i.e., $\mathcal{Z}^{d} = \mathcal{Z}^{d}_{1} \bigcup \mathcal{Z}^{d}_{2} \bigcup \mathcal{Z}^{d}_{3} \bigcup \mathcal{Z}^{d}_{4}$, and pre-train the backbone models $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4$ following the pipeline described in Section 3.3. The entire pipeline synthesizes mask labels for around 13000 $2\mu m \times 2\mu m$ layout patterns (recall that $\mathcal{Z}^{d}_{1}$ does not require labelling) by DNN-ILT, which takes us approximately 3 days (< 72 hours) to finish the entire construction flow (includes data-preparation and model training).

To verify the effectiveness of our proposed flow, we optimize ten 32nm M1 layout masks in the ICCAD 2013 benchmark suite [15] based on the final pre-trained model $\boldsymbol{\theta}^{*}_{pt} = \boldsymbol{\theta}_3$ (i.e., performing DNN-ILT on $\boldsymbol{\theta}_3$) and compare the results with the baseline flow [1]. Quantitative results are listed in Table 2, where column "Runtime" lists end-to-end mask optimization (ILT) time for a given test layout. Columns "L_2", "PVB", "TAT" denote the squared L_2 error, the process variation band, and the total turn around time for the entire flow construction (i.e., time to obtain the final DNN-ILT from scratch), respectively. We uses a consistent configuration for every test input, where the initial learning rate = 0.002, DNN-ILT iterations = 60, $\alpha = 1.0$ and $\beta = 1.0$ (Equation (7)). It can be observed that, in comparing to the state-of-the-art (SOTA) baseline flow [1], our flow achieves comparable performance in terms of "Runtime", "L_2" and "PVB" metrics, while reduce 66% of the total turn around flow construction time. As mentioned in Section 1, an important feature of our flow is that, in the self-training paradigm, the end-to-end learning-based mask optimization framework is built up from the undiscloable layout pattern set $\mathcal{Z}^{u}$ without actually "seeing" it. Unlike the baseline flow which directly uses $(\mathcal{Z}^{u}, \tilde{\mathcal{M}}^{u})$, none of these undiscloable layout patterns in $\mathcal{Z}^{u}$ is directly involved in the training of $\boldsymbol{\theta}^{*}_{pt}$, but the resulted DNN-ILT is still capable of conducting mask optimization for $\mathcal{Z}^{u}$.

Here we further randomly pick up 20 layout patterns from $\mathcal{Z}^{u}$ to empirically verify the above statement. We conduct DNN-ILT corrections for the selected 20 sample layouts based on our pre-trained models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4\}$ and the baseline model $\boldsymbol{\theta}_{\text{base}}$ from [1]. Their printability scores (L_2 + PVBand) are depicted in Figure 6. As expected, the printability score of $\boldsymbol{\theta}_1$ (red line) is not as good as that of $\boldsymbol{\theta}_{\text{base}}$ (green line), but the score gets significant improved on $\boldsymbol{\theta}_2$ (blue line) and gradually converged to baseline that was constructed with mask labels (green line) at $\boldsymbol{\theta}_3$ (purple line) and $\boldsymbol{\theta}_4$ (yellow line), while our approach does not need the provision of any input labels. The above empirical results indicate that our proposed

Table 2: Mask Printability and Runtime Performance Comparison with the SOTA Method [1] on ICCAD13 Benchmark Suite

Benchmarks		DNN-ILT with labelled real data [1]				DNN-ILT by complete self-training (ours)			
ID	Area (nm^2)	Runtime (s)	L_2 (nm^2)	PVB (nm^2)	TAT (hours)	Runtime (s)	L_2 (nm^2)	PVB (nm^2)	TAT (hours)
case1	215344	13.57	50795	63695	-	14.92	52583	62753	-
case2	169280	14.37	36969	60232	-	13.98	41471	56559	-
case3	213504	9.72	94447	85358	-	15.04	82360	111507	-
case4	82560	10.40	17420	32287	-	21.42	17597	32607	-
case5	281958	10.04	42337	65536	-	21.48	39405	64828	-
case6	286234	11.11	39601	59247	-	11.50	41535	56210	-
case7	229149	9.67	25424	50109	-	17.26	25884	50956	-
case8	128544	11.81	15588	25826	-	21.60	16562	26016	-
case9	317581	9.68	52304	68650	-	12.25	53319	67376	-
case10	102400	11.46	10153	22443	-	21.59	12199	21790	-
Average	-	11.18	38504	53338	> 240	17.10	38292	55060	< 72
Ratio	-	**1.00**	1.00	**1.00**	1.00	1.53	**0.99**	1.03	**< 0.33**

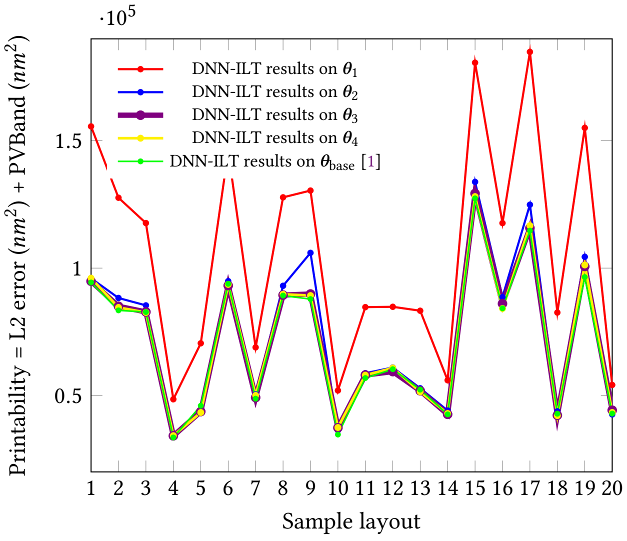


Figure 6: The DNN-ILT correction results of 20 sample layouts based on the pre-trained models $\{\theta_1, \theta_2, \theta_3, \theta_4\}$ and the baseline model θ_{base} [1], where the printability score = L_2 error (nm^2) + PVBand (nm^2).

flow can achieve fast convergence (e.g., from θ_1 to θ_4) to good performance by this iterative self-training approach even without any input mask labels. Meanwhile, our complete self-training can efficiently capture the useful characteristics of undisclosable layouts in a secure manner.

5 CONCLUSION

In this paper[1], we present a unified self-training paradigm to build up an end-to-end mask optimization framework from undisclosable layout patterns. Quantitative results verified its effectiveness. The presented flow can be easily extended to rigorous simulation environment in fab and vendor for their internal use. Future works would include detailed analysis on the reversibility of generated patterns and the empirical risk of the self-training flow.

[1]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. CUHK14209320).

REFERENCES

[1] B. Jiang, L. Liu, Y. Ma, H. Zhang, B. Yu, and E. F. Y. Young, "Neural-ilt: Migrating ilt to neural networks for mask printability and complexity co-optimization," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020.
[2] C. Mack, *Fundamental Principles of Optical Lithography: The Science of Microfabrication.* John Wiley & Sons, 2008.
[3] R. Pearman, J. Ungar, N. Shirali, A. Shendre, M. Niewczas, L. Pang, and A. Fujimura, "How curvilinear mask patterning will enhance the euv process window: a study using rigorous wafer+ mask dual simulation," in *Proceedings of SPIE*, vol. 11178, 2019.
[4] K. Hooker, B. Kuechler, A. Kazarian, G. Xiao, and K. Lucas, "ILT optimization of EUV masks for sub-7nm lithography," in *Proceedings of SPIE*, vol. 10446, 2017.
[5] M. Yeung and E. Barouch, "Development of fast rigorous simulator for large-area EUV lithography simulation," in *Extreme Ultraviolet (EUV) Lithography X*, vol. 10957, International Society for Optics and Photonics. SPIE, 2019, pp. 286 – 295.
[6] W. Ye, M. B. Alawieh, Y. Watanabe, S. Nojima, Y. Lin, and D. Z. Pan, "Tempo: Fast mask topography effect modeling with deep learning," in *ACM International Symposium on Physical Design (ISPD)*, 2020, p. 127–134.
[7] H. Tanabe, S. Sato, and A. Takahashi, "Fast 3D lithography simulation by convolutional neural network: POC study," in *Photomask Technology 2020*, vol. 11518. SPIE, 2020, pp. 1 – 7.
[8] Y. Lin, M. Li, Y. Watanabe, T. Kimura, T. Matsunawa, S. Nojima, and D. Z. Pan, "Data efficient lithography modeling with transfer learning and active data selection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2019.
[9] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "Damo: Deep agile mask optimization for full chip scale," *arXiv preprint arXiv:2008.00806*, 2020.
[10] P. Liu, "Mask synthesis using machine learning software and hardware platforms," in *Optical Microlithography XXXIII*, vol. 11327. International Society for Optics and Photonics, 2020, p. 1132707.
[11] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.
[12] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 131:1–131:6.
[13] W. Zhong, S. Hu, Y. Ma, H. Yang, X. Ma, and B. Yu, "Deep learning-driven simultaneous layout decomposition and mask optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.
[14] Y. Ma, W. Zhong, S. Hu, J. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020.
[15] S. Banerjee, Z. Li, and S. R. Nassif, "ICCAD-2013 CAD contest in mask optimization and benchmark suite," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 271–274.
[16] X. Zhang, J. Shiely, and E. F. Y. Young, "Layout pattern generation and legalization with generative learning models," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020.
[17] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information systems*, vol. 42, no. 2, pp. 245–284, 2015.
[18] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 52:1–52:6.