LithoGRPO: Fast Inverse Lithography via GRPO Reinforced Flow Matching

Yao Lai [1 2] **Xuyuan Xiong** [1] **Zeyue Xue** [1] **Guojin Chen** [3] **Jing Wang** [4]
Xihui Liu [1] **Rui Zhang** [5] **Robert Mullins** [2] **Bei Yu** [3] **Ping Luo** [1]

Abstract

In semiconductor manufacturing, lithography projects circuit layouts onto silicon wafers through an optical mask. As circuit features shrink below the wavelength of light, optical diffraction causes the printed patterns to deviate from their intended layouts. Inverse Lithography Technology (ILT) addresses this challenge by generating optimized masks that enhance the fidelity of pattern transfer onto wafers. While ILT resembles an image synthesis task, its reliance on explicit physical metrics for mask evaluation limits the applicability of existing generative models. We introduce LithoGRPO, an ILT framework that integrates the flow-matching paradigm with GRPO-based reinforcement learning (RL) fine-tuning, enabling efficient exploration of diverse masks for a given target layout. Unlike purely generative or optimization-based approaches, RL in LithoGRPO exploits the explicitly defined, physics-based reward function of ILT, enabling optimization under complex, process-aware constraints. To the best of our knowledge, this is the first framework that unifies flow matching and RL for mask optimization. To improve RL sampling efficiency, we propose a fast shot-counting algorithm for manufacturability evaluation, achieving over 130× speedup while largely preserving the mask ranking of the traditional shot count metric. Extensive experiments demonstrate that LithoGRPO achieves state-of-the-art performance over both optimization-based and learning-based methods, while maintaining efficient mask generation. Code is available at github.com/laiyao1/LithoGRPO.

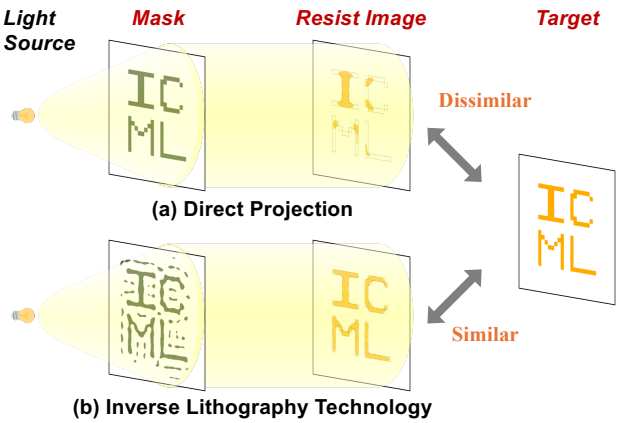

Figure 1. **Inverse Lithography.** (a) Direct projection suffers from pattern distortions in the resist image due to optical and process limitations. (b) Inverse Lithography Technology (ILT) optimizes the mask to compensate for these distortions, accurately reproducing the target layout on the wafer.

1. Introduction

Lithography is a fundamental process in semiconductor manufacturing that prints circuit layouts onto silicon wafers (Levinson, 2005; Erdmann, 2021). As illustrated in Fig. 1, a light source projects the mask pattern onto a light-sensitive photoresist layer covering the wafer. After exposure and development, the remaining resist image defines the printed circuit layout. With the continued scaling of technology nodes following Moore's law, pattern features have become smaller than the exposure wavelength. As a result, direct optical projection suffers from diffraction and optical aberrations, causing the printed patterns to deviate from the intended layouts, as shown in Fig. 1(a).

Traditionally, optical proximity correction (OPC) mitigates lithographic distortions by locally adjusting mask geometries based on empirical rules or pre-calibrated models. Rule-based OPC applies handcrafted correction heuristics

[1]The University of Hong Kong, Hong Kong [2]University of Cambridge, Cambridge, UK [3]The Chinese University of Hong Kong, Hong Kong [4]Nanjing University of Posts and Telecommunications, Nanjing, China [5]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. Correspondence to: Ping Luo <pluo@cs.hku.hk>, Bei Yu <byu@cse.cuhk.edu.hk>, Robert Mullins <Robert.Mullins@cl.cam.ac.uk>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

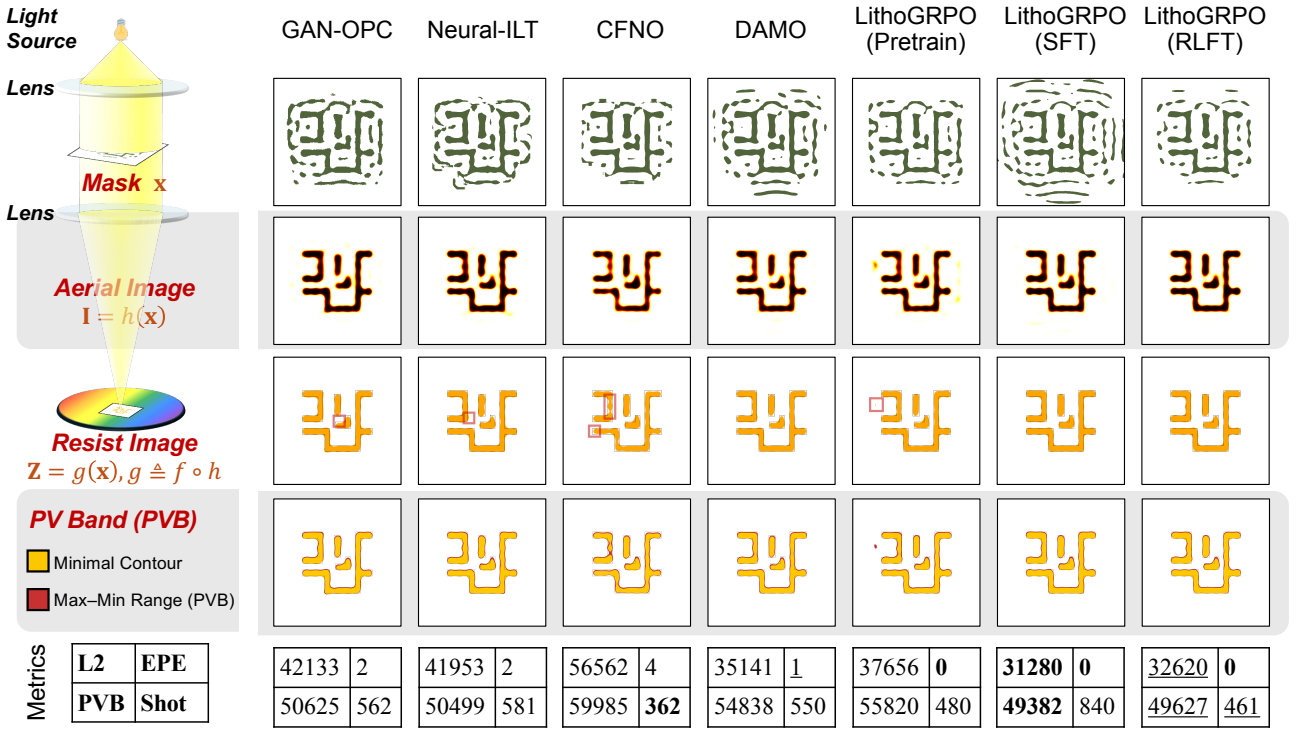

Figure 2. **ILT results visualization and comparison.** (Left) Illustration of the lithography. (Right) Results of different ILT methods for the same target layout. For the *aerial image*, color intensity indicates the light exposure level. For the *resist image*, dashed contours denote the target layout, and yellow regions indicate the simulated resist image, with critical imaging differences highlighted by red boxes. The *PV Band* highlights process-variation regions, where larger red areas indicate greater deviations. **Best** and second-best results are marked in bold and with underlines. **Better viewed at 400% zoom.**

but lacks global optimization and flexibility for complex patterns, whereas model-based OPC employs imaging simulations for iterative edge refinement, achieving higher accuracy yet remaining limited to edge-level corrections (i.e., local displacements of existing mask edges) (Poonawala & Milanfar, 2007; Pang, 2021; Yang et al., 2025). Recent efforts to enhance OPC adaptability include RL-OPC (Liang et al., 2024a), which leverages reinforcement learning to directly optimize explicit performance metrics and improve correction capability while still operating at the edge level within the OPC framework.

To overcome the limited, edge-based correction capability of OPC, Inverse Lithography Technology (ILT) formulates mask synthesis as a pixel-level inverse imaging problem, enabling systematic, physically-based correction of optical and process effects beyond the reach of OPC (Yang et al., 2025), as shown in Fig. 1(b). Existing ILT methods generally fall into two categories: optimization-based and learning-based approaches. Optimization-based methods (Gao et al., 2014; Yu et al., 2022; Sun et al., 2023b; 2025; Jia & Lam, 2010) model the lithography imaging process as a differentiable system and optimize differentiable objectives through gradient descent. However, they cannot directly handle non-differentiable objectives, and the iterative optimization process results in high computational cost. Learning-based methods (Zhu et al., 2023; Yang et al., 2022b;a; Chen et al., 2020; Jiang et al., 2020; 2021; Wu et al., 2025a;b; Jin et al., 2025a) aim to improve efficiency by training image-to-image models from paired masks and target layouts, but their training data typically originates from optimization-based results, limiting the quality and generalization capability. Furthermore, these methods still rely on differentiable losses, leaving non-differentiable objectives unoptimized. Among learning-based approaches, diffusion-based variants (Wang et al., 2025; Jiang et al., 2025) achieve high-fidelity synthesis but suffer from slow multi-step sampling, constraining scalability for high-resolution ILT. ILILT (Yang & Ren, 2024) uses a hybrid optimization–learning framework to improve end-to-end prediction fidelity, but it is computationally expensive and cannot optimize non-differentiable metrics.

To tackle these challenges, we propose LithoGRPO, a flow-matching–based generative ILT framework optimized via GRPO-driven reinforcement learning (RL), inspired by recent GRPO-guided image synthesis methods (Shao et al., 2024; Guo et al., 2025; Liu et al., 2025b; Xue et al., 2025;

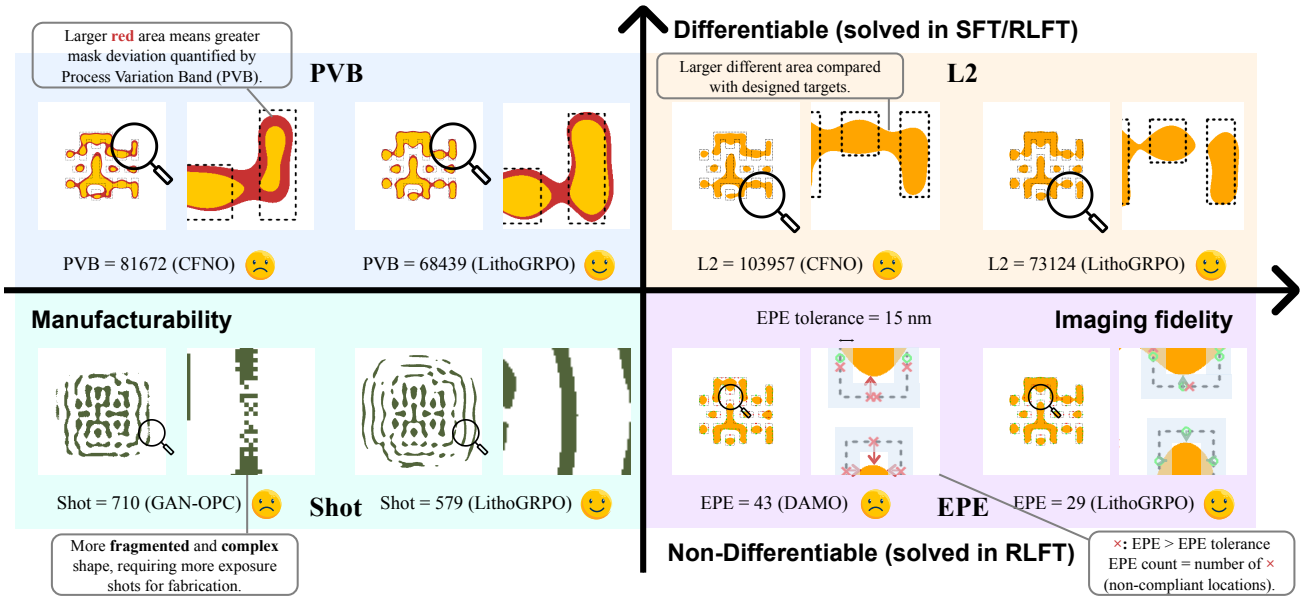

Figure 3. **ILT evaluation metrics.** L2 and EPE assess imaging fidelity: L2 measures pixel differences from the target, while EPE checks edge deviations within a threshold (Max EPE). Shot evaluates mask manufacturability, and PVB measures process-variation robustness. L2 and PVB are differentiable w.r.t. the mask (optimized in SFT/RLFT), whereas EPE and Shot are non-differentiable and handled via RLFT. These metrics are interrelated and may conflict. For instance, improving imaging fidelity (L2/EPE) typically requires finer mask features, which increases the Shot count.

Li et al., 2025). It performs reward-guided finetuning of generative models for mask synthesis, enabling unified optimization over differentiable and non-differentiable metrics. Such a setup is particularly suitable for lithography, where explicit and deterministic metrics naturally serve as well-defined rewards for reinforcement optimization. Unlike prior work, LithoGRPO models complete mask generation as a rectified flow (Liu et al., 2023), offering a deterministic one-step transport between noise and data that avoids the slow, multi-step denoising of diffusion models and improves sampling efficiency and stability. The training consists of three stages: (1) pretraining for mask-target alignment; (2) supervised finetuning optimizing differentiable metrics; and (3) RL finetuning jointly optimizing differentiable and non-differentiable metrics via SDE-based GRPO exploration. We further propose an ultra-fast shot-count evaluation based on minimal-overlap rectangular decomposition, reducing metric computation from about 1 min to 0.2 s. Overall, LithoGRPO unifies gradient-based learning and reinforcement optimization under a rectified-flow framework, jointly optimizing differentiable and non-differentiable metrics for efficient and comprehensive ILT.

Our main contributions are summarized as follows:

- To the best of our knowledge, LithoGRPO is the *first* to introduce flow matching into ILT by modeling mask generation as a rectified flow matching process conditioned on the target layout.
- A GRPO-based reinforcement learning fine-tuning (RLFT) scheme tailored for ILT tasks with explicit reward definitions, enabling joint optimization of differentiable and non-differentiable lithographic metrics under a unified framework.
- An ultra-fast shot count algorithm speeds up mask evaluation by 130×–490× while largely preserving ranking, significantly improving RL sampling efficiency.
- Extensive experiments demonstrate that LithoGRPO achieves state-of-the-art performance in imaging fidelity, manufacturability, and robustness, while maintaining high generation efficiency.

2. Preliminary

2.1. Inverse Lithography Technology

As shown on the left side of Fig. 2, the lithography process projects a mask pattern $\mathbf{x}$ through an optical system onto a photoresist-coated wafer, forming an aerial image $\mathbf{I}$ on the resist surface and a resist image $\mathbf{Z}$ on the wafer after development.

Optical Imaging and Photoresist Modeling. Instead of building a full physical simulation, people usually adopt a compact differentiable forward model that links the mask pattern $\mathbf{x}$ to the printed resist image $\mathbf{Z}$ via the optical imaging and photoresist processes. The aerial image $\mathbf{I}$ shown in Fig. 2 can be described by Hopkins' diffraction model (Hop-

kins, 1953):

$$\mathbf{I} = h(\mathbf{x}) = \sum_{k=1}^{K} \mu_k \left| h_k \otimes \mathbf{x} \right|^2, \tag{1}$$

where $\mathbf{x}$ denotes the mask pattern, h_k is the k-th coherent point-spread function, and μ_k is its corresponding illumination weight. The operator $\otimes$ denotes convolution, and $|\cdot|$ indicates the magnitude of a complex amplitude. Intuitively, Eq. (1) expresses that the mask pattern is blurred by the optical system and integrated over different illumination angles to form the aerial image $\mathbf{I}$ on the wafer.

Modeling a negative photoresist, where exposed regions are retained, the aerial image $\mathbf{I}$ is converted into a developed resist pattern through thresholding:

$$\mathbf{Z} = \mathbb{1}(\mathbf{I} > I_{\text{th}}), \tag{2}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function and I_{th} is the exposure threshold. To enable gradient-based optimization for differentiable metrics, this non-differentiable process can be replaced by an approximation:

$$\mathbf{Z} = f(\mathbf{I}) = \frac{1}{1 + \exp[-\alpha(\mathbf{I} - I_{\text{th}})]}, \tag{3}$$

where α controls the transition sharpness. Together, Eqs. (1)–(3) form a differentiable forward model that maps a mask pattern $\mathbf{x}$ to its developed resist image $\mathbf{Z}$, expressed compactly as $\mathbf{Z} = g(\mathbf{x}) = f(h(\mathbf{x}))$, where $h(\cdot)$ and $f(\cdot)$ denote the optical imaging and photoresist, respectively.

Inverse Lithography and Evaluation. Inverse lithography technology (ILT) aims to find the optimal mask $\mathbf{x}$ that yields a resist image $\mathbf{Z} = g(\mathbf{x})$ matching the desired target layout $\mathbf{T}$. Its performance can mainly be assessed by four representative metrics: L2, PVB, EPE, and Shot, as illustrated in Fig. 3.

Among the differentiable objectives, the L2 loss is the most widely used metric to measure imaging fidelity. It is defined as a function of a mask $\mathbf{x}$ and a target layout $\mathbf{T}$, reflecting how well the simulated resist image $\mathbf{Z} = g(\mathbf{x})$ reproduces the target layout $\mathbf{T}$:

$$\text{L2}(\mathbf{x}, \mathbf{T}) = \|g(\mathbf{x}) - \mathbf{T}\|_2^2. \tag{4}$$

To account for process variations such as focus and dose fluctuation, the process variation band (PVB) evaluates the stability of the printed results for a given mask $\mathbf{x}$ across different process corners. It measures the intensity range between the best and worst process conditions:

$$\text{PVB}(\mathbf{x}) = \|g_{\max}(\mathbf{x}) - g_{\min}(\mathbf{x})\|_2^2, \tag{5}$$

where $g_{\max}(\mathbf{x})$ and $g_{\min}(\mathbf{x})$ are the simulated images from the same $\mathbf{x}$ under the best and worst process conditions, respectively. A larger PVB indicates higher sensitivity to process variations and thus lower robustness.

Also, two non-differentiable metrics are critical for evaluating the local geometric fidelity and mask manufacturability. Edge Placement Error (EPE) measures the local deviation between the printed contour and the target edge. It is typically evaluated at discrete sampling points along the target boundary by counting the number of sites where the deviation exceeds a specified tolerance (Max EPE), with smaller EPE values indicating higher pattern fidelity. The shot count (Shot) assesses mask manufacturability. It is the minimum number of rectangular shots required to cover the ILT pattern during e-beam mask writing, directly impacting fabrication time and cost. These metrics often exhibit inherent trade-offs: pursuing higher imaging fidelity (e.g., lower L2 / EPE) typically requires more complex mask geometries, which in turn increases mask writing complexity and degrades manufacturability, leading to a larger Shot count.

2.2. Flow Matching

Flow-based generative modeling (Lipman et al., 2024) has recently shown remarkable success in image and video synthesis, with models like Stable Diffusion 3 (Esser et al., 2024) and FLUX (Labs, 2024). These models operate by learning a continuous transformation that maps a simple prior distribution p_0 (e.g., a standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$) to a complex target data distribution p_1. This transformation is defined by an ordinary differential equation (ODE):

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_\theta(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0, \tag{6}$$

where $\mathbf{v}_\theta(\mathbf{x}, t)$ is a neural network parameterized by θ that represents a time-dependent velocity field. Generation of new samples is accomplished by integrating this ODE from $t = 0$ to $t = 1$ using numerical solvers. For instance, the Euler method provides a first-order approximation:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t * \mathbf{v}_\theta(\mathbf{x}_t, t), \tag{7}$$

which discretely simulates the probability path, or flow, that transports the prior distribution p_0 to the target distribution p_1. In contrast to diffusion models, this formulation establishes a more direct theoretical connection between the latent space and the data manifold, often enabling significantly faster sampling with fewer function evaluations.

Flow matching (Lipman et al., 2023) introduces a general and simulation-free training paradigm for learning such ODEs. It trains the neural velocity field $\mathbf{v}_\theta$ by regressing it onto a predefined target vector field. A prominent variant, Rectified flow (Liu et al., 2023), simplifies the underlying dynamics by constructing linear interpolation paths between pairs of samples $(\mathbf{x}_0, \mathbf{x}_1)$ drawn from the prior and target

distributions, respectively. The trajectory and its corresponding target velocity field are defined as:

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \quad \mathbf{v}(\mathbf{x}_t, t) = \mathbf{x}_1 - \mathbf{x}_0. \tag{8}$$

The training objective is thus formulated as a simple regression loss:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{x}_0\sim p_0,\mathbf{x}_1\sim p_1}\left[\|\mathbf{v}_\theta(\mathbf{x}_t,t) - \mathbf{v}(\mathbf{x}_t,t)\|_2^2\right]. \tag{9}$$

where the target velocity $\mathbf{x}_1 - \mathbf{x}_0$ remains constant along each individual linear trajectory. This rectification process yields substantially straighter and smoother flows, which are inherently easier to integrate and exhibit improved numerical stability during sampling.

3. Method

Overview. LithoGRPO is trained in three stages: pretraining, supervised fine-tuning (SFT), and reinforcement-learning fine-tuning (RLFT), as shown in Fig. 4. Pretraining learns an initial ILT generator from a limited mask dataset. SFT adds differentiable lithographic metrics to the loss to improve pattern fidelity, at the cost of increased shot count due to metric trade-offs. RLFT then optimizes all metrics via a single reward, reducing shot count while largely preserving the other metrics near saturation.

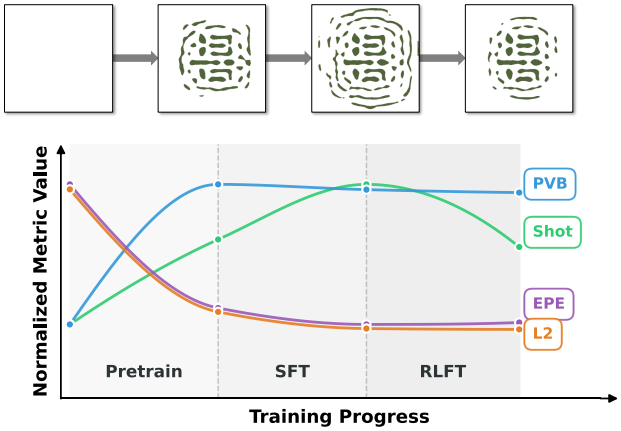

Figure 4. **Training dynamics of key metrics.** L2 and EPE decrease during pre-training and SFT, while Shot rises; the RLFT stage subsequently reduces Shot without degrading other metrics, highlighting the benefit of the three-stage strategy. Curves are fitted from stage-end metric values.

Pretraining and SFT. Following the two-stage training framework in previous works (Zheng et al., 2023), we replace the original GAN-based or pixel-space generation modeling approach with rectified flow.

During pretraining, we use dataset-provided pairs $(\mathbf{T}, \mathbf{x}_1)$, where $\mathbf{T}$ denotes the target layout and $\mathbf{x}_1$ is the corresponding mask pattern. For each pair, Gaussian noise $\mathbf{x}_0$ is sampled, and the model is trained with the standard rectified-flow mean-squared-error objective to predict the velocity field $\mathbf{v}_\theta$ from $(\mathbf{x}_0, \mathbf{x}_1)$ conditioned on $\mathbf{T}$, as shown in Eq. 9. In practice, the target layout $\mathbf{T}$ is concatenated with the noisy input $\mathbf{x}_t$ along the channel dimension.

In the supervised fine-tuning (SFT) stage, we incorporate differentiable, task-specific metrics into the optimization. At each training step, a random time $t \sim \mathcal{U}(0,1)$ is sampled, and the model predicts the velocity $\mathbf{v}_\theta(\mathbf{x}_t, t; \mathbf{T})$ from the interpolated state $\mathbf{x}_t$ conditioned on the target layout $\mathbf{T}$. The predicted velocity is then projected to the end of the trajectory to estimate the final mask:

$$\mathbf{x}_1 = \mathbf{x}_t + (1-t)\mathbf{v}_\theta(\mathbf{x}_t, t; \mathbf{T}), \tag{10}$$

where $\mathbf{x}_1$ denotes the predicted mask. While the flow-matching loss $\mathcal{L}_{\text{flow}}$ in Eq. 9 provides supervision at the intermediate time t, the differentiable metrics (L2/PVB) are computed on $\mathbf{x}_1$. The L2 metric is computed between the target layout $\mathbf{T}$ and the resist image $\mathbf{Z}$ (Eq. 4), where $\mathbf{Z} = g(\mathbf{x}_1)$ is obtained via the differentiable lithography simulator. The PVB metric is similarly evaluated based on $\mathbf{x}_1$ (Eq. 5). The total SFT loss is a weighted combination:

$$\mathcal{L}_{\text{sft}} = \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{L2}}\text{L2}(\mathbf{x}_1, \mathbf{T}) + \lambda_{\text{PVB}}\text{PVB}(\mathbf{x}_1). \tag{11}$$

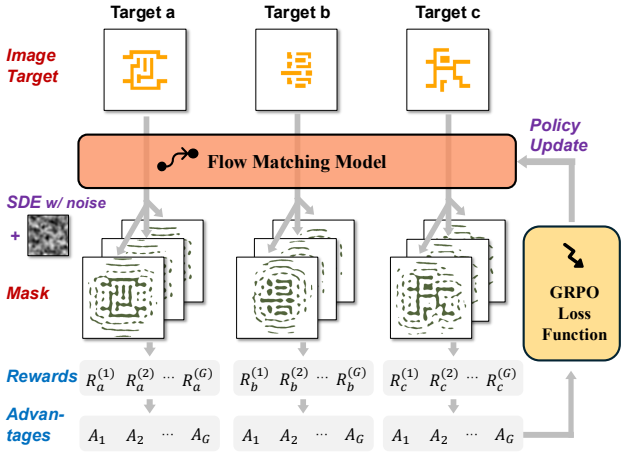

Figure 5. **Overview of the RLFT stage in LithoGRPO.** For each target layout, the flow matching model generates multiple mask samples by simulating SDE dynamics conditioned on the target and noise. Each group of generated masks yields corresponding rewards $\{R^{(1)}, \ldots, R^{(G)}\}$, which are normalized to compute group-level advantages $\{A^{(i)}\}$. These advantages are then used to update the policy via the GRPO loss.

GRPO-based RLFT. In the third stage, we employ GRPO-based RL finetuning (RLFT) to jointly optimize all four metrics, including the non-differentiable ones ignored in earlier stages. To achieve this, we draw inspiration from recent GRPO-based generative methods (Xue et al., 2025;

Liu et al., 2025b). We reformulate the model's deterministic Ordinary Differential Equation (ODE) into an equivalent Stochastic Differential Equation (SDE). This SDE is constructed to preserve the marginal distributions of the original ODE at all time steps, while its inherent stochasticity enables the generation of multiple mask candidates for each target layout during RL optimization, as illustrated in Fig. 5. Specifically, for a given target layout $\mathbf{T}$, the stochastic flow model samples a group of G candidate mask patterns $\{\mathbf{x}_1^{(i)}\}_{i=1}^G$, for which we compute corresponding rewards $R(\mathbf{x}_1^{(i)}, \mathbf{T})$ using the all four performance metrics. To reduce the variance of the policy gradient, we calculate the advantage for each mask candidate $\mathbf{x}_1^{(i)}$ by normalizing its reward against the statistics of the entire group:

$$A_i = \frac{R(\mathbf{x}_1^{(i)}, \mathbf{T}) - \text{mean}(\{R(\mathbf{x}_1^{(i)}, \mathbf{T})\}_{i=1}^G)}{\text{std}(\{R(\mathbf{x}_1^{(i)}, \mathbf{T})\}_{i=1}^G) + \varepsilon}. \tag{12}$$

This advantage A_i then weights the score function of the corresponding trajectory, guiding the model to favor generations that yield higher rewards.

The reward signal for a generated mask $\mathbf{x}_1^{(i)}$ given a target $\mathbf{T}$ is defined as the negative sum of normalized metric scores. The metric set is $\mathcal{K} = \{\text{L2}, \text{PVB}, \text{EPE}, \text{Shot}\}$. Each metric k is normalized by its baseline value k_0, obtained from the model at the end of the SFT stage. As we adopt uniform weights, the reward is:

$$R(\mathbf{x}_1^{(i)}, \mathbf{T}) = -\sum_{k \in \mathcal{K}} k(\mathbf{x}_1^{(i)}, \mathbf{T})/k_0. \tag{13}$$

The Shot and PVB metrics depend only on the generated mask $\mathbf{x}_1^{(i)}$, whereas L2 and EPE additionally depend on the target layout $\mathbf{T}$, as introduced in the preliminaries. For computational efficiency, we use a fast shot count approximation (Shot(fast)) during RL training; the traditional shot count is used only for final evaluation.

We treat the generative model as a policy $\pi_\theta(\mathbf{x} \mid \mathbf{T})$, representing the conditional probability of generating layout $\mathbf{x}$ given a target $\mathbf{T}$, and optimize its parameters θ by minimizing the GRPO loss, defined as an expectation over the dataset of target layouts $\mathcal{D}$:

$$\mathcal{L}_{\text{grpo}}(\theta) = \mathbb{E}_{\mathbf{T} \sim \mathcal{D}}\Big[f(\theta; \mathbf{T})\Big], \quad \text{where}$$

$$f(\theta; \mathbf{T}) = -\sum_{i=1}^{G} \min\Big(r_i(\theta)A_i,\ \text{clip}\big(r_i(\theta), 1-\varepsilon, 1+\varepsilon\big)A_i\Big),$$

$$r_i(\theta) = \frac{\pi_\theta(\mathbf{x}_1^{(i)}|\mathbf{T})}{\pi_{\theta_{\text{old}}}(\mathbf{x}_1^{(i)}|\mathbf{T})}. \tag{14}$$

Here A_i is the advantage defined in Eq. 12, ε is the clipping threshold, and $\pi_{\theta_{\text{old}}}$ denotes the policy from the previous iteration used to compute the importance ratio.

SDE-based Sampling in GRPO. To overcome the limited exploration inherent in the deterministic ODE-based sampling of our flow model, we introduce a stochastic generation process. Inspired by FlowGRPO (Liu et al., 2025b), we reformulate the generation ODE into a Stochastic Differential Equation (SDE). This SDE is carefully designed to ensure its marginal distribution matches that of the original flow model at each timestep. By doing so, our sampler can explore a diverse range of candidates for GRPO optimization without deviating from the learned data manifold, thus preventing policy stagnation in local optima.

Specifically, the reverse-time SDE is discretized using the Euler–Maruyama scheme (Kloeden & Pearson, 1977), yielding the following update rule for generating a sample $\mathbf{x}_{t+\Delta t}$ from the previous state $\mathbf{x}_t$:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Big[\mathbf{v}_\theta(\mathbf{x}_t, t) + \frac{\sigma_t^2}{2t}\big(\mathbf{x}_t + (1-t)\mathbf{v}_\theta(\mathbf{x}_t, t)\big)\Big]\Delta t + \sigma_t\sqrt{\Delta t}\,\boldsymbol{\varepsilon}. \tag{15}$$

Here, the stochastic term $\boldsymbol{\varepsilon}$ is realized as low-frequency colored noise, with scale $\sigma_t = a\sqrt{(1-t)/t}$ where a is a scalar hyperparameter. Unlike the conventional setting of injecting Gaussian white noise ($\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$), uncorrelated noise introduces high-frequency perturbations in the mask domain, yielding fragmented regions that degrade manufacturability and increase shot count. To alleviate this, colored noise is obtained by low-pass filtering white noise in the Fourier domain, producing spatially correlated perturbations that preserve mask topology and suppress spurious fragments. This design is crucial for ILT mask generation, where smooth and continuous features are essential. Fig. 6 qualitatively compares (a) unstructured white noise and (b) colored noise with correlated patterns. For GRPO, we approximate each transition with an independent Gaussian for log-probability computation, $\mathbf{x}_{t+\Delta t} \sim \mathcal{N}(\boldsymbol{\mu}_t, \sigma_t^2 \Delta t\, \mathbf{I})$, where $\boldsymbol{\mu}_t$ denotes the deterministic component in Eq. 15. Details are in Appendix C.

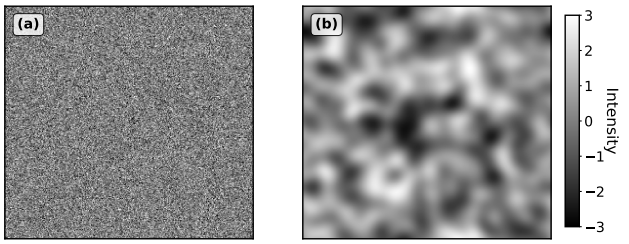

Figure 6. **Comparison of noise types.** (a) Gaussian white noise: high-frequency and spatially uncorrelated, leading to fragmented masks with high shot counts. (b) Colored Gaussian noise: low-frequency and spatially correlated, preserving the spatial structure of lithography masks.

Table 1. **Performance comparison on ILT for Metal and Via layers. Best** and second-best are in bold and underlined. LithoGRPO (RLFT) results are mean ± std over 4 independent trainings with different random seeds. Avg. Rank is computed over 4 datasets × 4 metrics + Time (17 columns), with ties assigned the average of the occupied ranks.

Method	Metal Layer								Via Layer								Avg. Time (s)↓	Avg. Rank↓
	MetalSet				StdMetal (in-distribution)				ViaSet				StdContact (OOD)					
	L2↓	PVB↓	EPE↓	Shot↓	L2↓	PVB↓	EPE↓	Shot↓	L2↓	PVB↓	EPE↓	Shot↓	L2↓	PVB↓	EPE↓	Shot↓		
Optimization Approaches																		
MOSAIC	35860	48080	6.6	361	21733	27222	3.2	229	-	-	-	-	-	-	-	-	0.940	9.8
LevelSet	34712	50113	5.4	**263**	21526	27769	3.5	**145**	9632	10197	4.7	**64**	50770	32134	34.7	275	2.290	6.9
MultiLevel	**27893**	41372	**2.5**	1250	**13203**	23755	**0.1**	1240	**4268**	8370	**0.0**	1434	20497	43405	0.4	1473	1.030	5.6
Learning Approaches																		
GAN-OPC	43414	41290	8.7	574	25929	23715	4.6	457	14767	**6686**	8.3	166	81378	**4931**	73.2	276	**0.010**	7.4
Neural-ILT	36670	42666	7.3	476	20045	23548	2.4	373	12723	8537	6.2	263	25422	41537	3.2	**265**	0.025	6.5
DAMO	32579	41173	5.4	523	16120	23796	0.2	418	5081	9962	**0.0**	176	50445	35673	26.7	458	0.028	5.7
DOINN	36409	41929	7.4	519	25913	25749	4.5	313	4382	7836	**0.0**	968	72058	17968	55.8	493	0.107	7.3
CFNO	47814	46131	12.5	302	26809	26814	4.2	232	8949	9890	0.1	184	70740	17950	55.1	396	0.040	7.4
Mixed Approach																		
ILILT	30353	45353	3.2	433	14596	24969	**0.1**	368	4666	10065	**0.0**	238	38957	43869	7.1	510	0.441	5.9
Our Approach																		
LithoGRPO (Pretrain)	32824	42320	5.4	487	17362	23837	0.4	386	11595	8776	3.4	170	77289	15445	63.9	377	0.104	6.6
LithoGRPO (SFT)	29123	40722	2.8	803	14756	**22468**	0.2	675	4270	7751	**0.0**	989	**18720**	39291	**0.0**	1546	0.104	4.7
LithoGRPO (RLFT)	28933	**39838**	3.1	444	14840	22497	0.2	358	4276	8376	**0.0**	398	19102	41359	0.2	889	0.104	**4.3**
± std dev	±541	±281	±0.2	±24	±643	±160	±0.1	±21	±68	±122	±0.0	±31	±373	±26	±0.1	±16	-	-

Fast Shot Count. The shot count, a critical manufacturability metric, is traditionally obtained by partitioning a mask pattern into the minimum set of *non-overlapping* rectangles. This minimum-rectangle partitioning problem is NP-hard (Kim et al., 2023), and existing implementations (Raghuvanshi et al., 2009) often take over one minute per mask. To enable fast evaluation, we compute an *overlapping* shot-count proxy by formulating rectangle selection as a minimum set cover problem and solving it with an ILP (Fig. 7). Our pipeline has three steps. (1) Extract a comprehensive set of locally maximal rectangles using a histogram-based search in $O(N^2)$ for an N^2-pixel mask, ensuring that every valid shot is contained within at least one candidate. (2) Redundant rectangles that are fully contained in others are removed via a pruning step in $O(K^2)$, where K is the number of candidate rectangles. (3) Remaining rectangles form a compact Set Cover ILP, where a row-wise scan-line optimization generates all covering constraints in $O(NK^2)$. This method achieves up to 490× speedup over traditional non-overlapping partitioning while preserving the relative ordering of the shot count, serving as an efficient proxy. This approximation is well-suited to GRPO: the group-wise advantage normalization in Eq. 12 cancels any constant offset between Shot(fast) and the traditional shot, so only the within-group ranking of candidates matters for the policy gradient. As Fig. 9 shows, this ranking is strongly preserved. Moreover, allowing overlapping shots aligns with modern multi-beam mask writing practice (Chua et al., 2011; Zable et al., 2010; Fujimura et al., 2010; Chan et al., 2016; Zhang et al., 2024). Details are in Appendix B.

Inference. During inference, masks are generated by integrating the velocity field using the Euler update in Eq. 7, starting from Gaussian noise $\mathbf{x}_0$. For efficiency, we use a single-step update ($\Delta t = 1$) by default, while multi-step inference is also supported and evaluated in the ablation study.

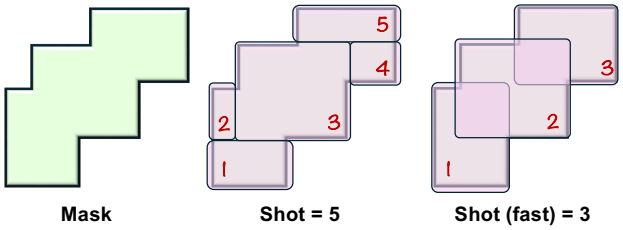

Figure 7. **Shot-counting methods.** Traditional counting tiles the mask with non-overlapping rectangles; ours uses a minimal overlapping cover, yielding faster computation while largely preserving relative shot-count ordering.

4. Experimental Results

Experimental Settings. We evaluate LithoGRPO and baselines on the LithoBench (Zheng et al., 2023) dataset, which contains metal-layer (MetalSet, StdMetal) and via-layer (ViaSet, StdContact) patterns, each covering a 2048×2048 nm^2 region. All patterns are processed at 512×512 resolution. For metal layers, LithoGRPO is pretrained for 50 epochs on MetalSet, followed by 25 epochs of SFT and 1000 RLFT steps, and evaluated on both MetalSet and in-distribution StdMetal. For via layers, 10 epochs of pretraining, 1 epoch of SFT, and 1000 RLFT steps are performed on ViaSet; StdContact, being out-of-distribution, is fine-tuned from the ViaSet-pretrained model for 50 epochs of SFT with 1000 RLFT steps following the LithoBench protocol. We evaluate EPE with a 15 nm tolerance, and EPE can saturate

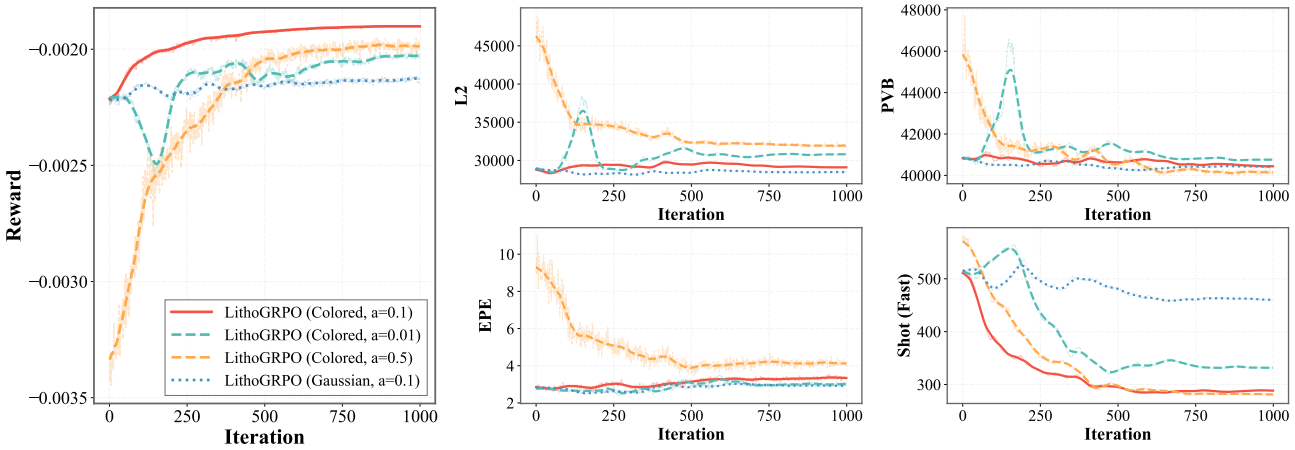

Figure 8. **Noise configurations in SDE.** We compare noise types (colored vs. Gaussian) and noise levels ($a \in \{0.01, 0.1, 0.5\}$) for LithoGRPO. At $a = 0.5$, mask quality degrades, leading to poor initialization; at $a = 0.01$, exploration slows. Gaussian noise yields inferior shot-count results, while colored noise with $a = 0.1$ achieves the best overall balance.

Table 2. **Effect of inference steps on Metal-layer benchmarks.** On both MetalSet and its OOD counterpart StdMetal, one-step inference already matches or beats more-step variants on the main metrics while being 2–10× faster. Avg. Rank is computed over 2 datasets × 4 metrics + Runtime (9 columns), with ties assigned the average of the occupied ranks. LithoGRPO (RLFT) results are the mean over 4 independent seeds, matching the main table.

Method	Step	MetalSet				StdMetal				Runtime (s)↓	Avg. Rank↓
		L2	PVB	EPE	Shot	L2	PVB	EPE	Shot		
ILILT	-	30353	45353	3.2	433	14596	24969	0.1	368	0.441	7.3
LithoGRPO (Pretrain)	1	32824	42320	5.4	**487**	17362	**23837**	0.4	**386**	**0.104**	9.1
	2	31540	**42167**	4.2	506	16404	23903	0.2	402	0.203	9.1
	5	**31420**	42352	4.2	512	**16242**	23916	0.2	414	0.508	9.8
	10	31790	42400	**2.8**	507	16330	24040	**0.2**	415	1.014	9.9
LithoGRPO (SFT)	1	29123	40722	2.8	803	14756	**22468**	0.2	675	**0.104**	7.4
	2	**28185**	40566	2.7	774	**13906**	22582	0.1	644	0.203	5.4
	5	28520	**40456**	**2.6**	776	14388	22486	0.1	626	0.508	5.9
	10	28396	40556	2.6	**758**	14121	22544	**0.1**	**618**	1.014	5.9
LithoGRPO (RLFT)	1	28933	**39838**	3.1	**444**	14840	**22497**	0.2	**358**	**0.104**	4.7
	2	**28386**	40152	**3.0**	460	**13881**	22743	**0.1**	368	0.203	3.8
	5	28630	40035	3.1	483	14076	22653	0.2	390	0.508	5.9
	10	28658	39927	3.1	491	14128	22617	0.2	400	1.014	6.7

at 0 when all edge samples fall within this tolerance. We use a group size $G = 6$ in GRPO. An 87M-parameter U-Net backbone uses a single denoising step at inference. We use PuLP to solve the ILP formulation for fast shot count. Fast shot count is used for GRPO training, while traditional shot count is reported at inference. All experiments are conducted on four NVIDIA RTX 3090 GPUs with Intel Xeon Silver 4216 CPUs, with each training stage taking under 8 hours. Details are in Appendix E.

Main Results. Table 1 compares LithoGRPO with baseline methods. Most baseline results are taken from the original papers (Zheng et al., 2023; Yang & Ren, 2024), while metrics not reported in the literature (e.g., Shot for ILILT and optimization-based methods) are reproduced using the same evaluation protocol. MOSAIC results for the via layer are omitted because its hyperparameters failed to generalize from metal to via patterns, resulting in blank resist images. The results show that the three training stages of LithoGRPO contribute sequentially: Pre-training establishes a basic mask-generation capability, SFT, which directly optimizes differentiable metrics (L2 and PVB), rapidly reduces them but increases Shot complexity, and RLFT balances all metrics, notably lowering Shot while keeping L2, PVB, and EPE stable. With one-step inference, LithoGRPO maintains high efficiency (0.1 s per sample) and achieves the best overall trade-off among conflicting metrics, attaining the lowest Avg. Rank (**4.3**), ahead of all baselines (best at 5.6), setting a new state of the art. Additional results are shown in Appendix F.

Noise Type and Level. To enable stochastic exploration in RL, we extend the deterministic ODE sampling (Eq. 7) to an SDE formulation (Eq. 15). The injected noise is controlled by two factors: the noise level a and the noise type (Gaussian white vs. low-frequency colored). We analyze their effects on MetalSet in Fig. 8. The default setting is $a = 0.1$ with colored noise. When decreasing a to 0.01, the exploration becomes noticeably slower. Conversely, increasing a to 0.5 yields noisier, less stable masks, leading to poorer initial rewards. Even under higher noise levels, our method converges to close results, demonstrating robustness. In contrast, Gaussian noise produces more fragmented masks than low-frequency colored noise, hindering exploration of lower-shot mask patterns.

Inference Steps. Table 2 compares different rectified flow sampling steps on the Metal benchmarks. RLFT with two-step sampling attains the best overall rank (3.8), narrowly ahead of one-step (4.7); the marginal quality gain does not offset the $2\times$ runtime, so one-step sampling is adopted as the default for efficient generation.

Fast Shot Count. We compare the traditional shot-count implementation in LithoBench with our proposed fast version in correlation and efficiency. The comparison uses masks generated by LithoGRPO (RLFT). As shown in Fig. 9, the two methods exhibit a strong correlation ($R^2 = 0.994$), confirming the reliability of our fast estimation scheme. The ILP formulation is also fully deterministic by construction, yielding perfectly reproducible shot counts. Table 3 further reports the runtime, showing a $134\times$–$491\times$ speed-up over the traditional implementation, greatly improving RL sampling efficiency. A detailed error analysis between traditional and fast shot count is in Appendix F.2.

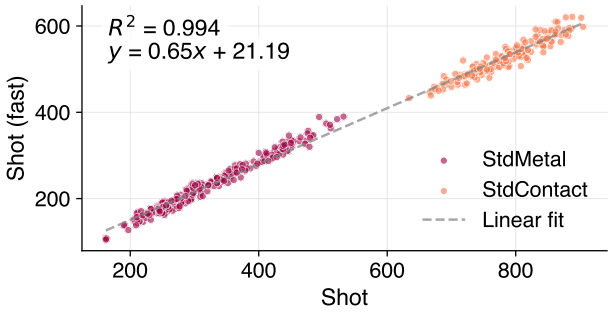

Figure 9. **Correlation between shot and shot (fast).** Strong linear correlation demonstrates that the fast shot method accurately approximates the traditional shot count.

Table 3. **Runtime comparison for shot computation.**

Method	MetalSet	StdMetal	ViaSet	StdContact
Shot (s)	34.01 ± 8.10	64.82 ± 17.69	44.81 ± 15.88	148.35 ± 14.32
Shot (fast) (s)	0.25 ± 0.19	0.16 ± 0.06	0.18 ± 0.12	0.30 ± 0.06
Speedup	×134.6	×398.2	×251.1	×491.3

5. Conclusion

We present LithoGRPO, the first ILT framework that integrates flow matching with GRPO-based RL fine-tuning. It formulates mask generation as a rectified flow, enabling the joint optimization of differentiable and non-differentiable metrics via reward feedback. A fast shot-count algorithm further accelerates mask evaluation by over 130×, ensuring efficient exploration. Experiments demonstrate high accuracy, robustness, manufacturability, and efficiency.

Limitations. Our multi-stage training adds computation but yields the best overall trade-off, and jointly optimizing conflicting metrics remains inherently challenging (Table 1). Evaluation is limited to public academic benchmarks (LithoBench); industrial-scale validation and extreme layout densities are left for future work.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. Our experiments rely on open-source academic benchmarks at mature technology nodes, far from cutting-edge industrial processes. We do not foresee additional societal consequences requiring specific discussion.

Acknowledgment

This paper is partially supported by the National Key R&D Program of China No.2022ZD0161000 and the General Research Fund of Hong Kong No.17200622 and 17209324.

References

Alawieh, M. B., Lin, Y., Zhang, Z., Li, M., Huang, Q., and Pan, D. Z. Gan-sraf: Sub-resolution assist feature generation using conditional generative adversarial networks. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.

Black, K., Janner, M., Du, Y., Kostrikov, I., and Levine, S. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Chan, T.-B., Gupta, P., Han, K., Kagalwalla, A. A., and Kahng, A. B. Benchmarking of mask fracturing heuristics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 36(1):170–183, 2016.

Chen, G., Chen, W., Ma, Y., Yang, H., and Yu, B. Damo: Deep agile mask optimization for full chip scale. In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2020.

Chua, G. S., Wang, W. L., Choi, B. I., Zou, Y., Tabery, C., Bork, I., Nguyen, T., and Fujimura, A. Optimiza-

tion of mask shot count using mb-mdp and lithography simulation. In *Photomask Technology*, volume 8166, pp. 830–840. SPIE, 2011.

Dong, X. and Zhang, L. Analog layout retargeting with process-variation-aware rule-based opc. In *International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4. IEEE, 2017.

Erdmann, A. Optical and euv lithography: A modeling perspective. 2021.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Fujimura, A., Pierrat, C., Kiuchi, T., Komagata, T., and Nakagawa, Y. Efficiently writing circular contacts on production reticles. *Photomask Japan*, 2010, 2010.

Gao, J.-R., Xu, X., Yu, B., and Pan, D. Z. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2014.

Geng, H., Yang, H., Ma, Y., Mitra, J., and Yu, B. SRAF insertion via supervised dictionary learning. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 406–411, 2019.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

He, H., Ye, Y., Liu, J., Liang, J., Wang, Z., Yuan, Z., Wang, X., Mao, H., Wan, P., and Pan, L. GARDO: Reinforcing diffusion models without reward hacking. *arXiv preprint arXiv:2512.24138*, 2025a.

He, X., Fu, S., Zhao, Y., Li, W., Yang, J., Yin, D., Rao, F., and Zhang, B. TempFlow-GRPO: When timing matters for GRPO in flow models. *arXiv preprint arXiv:2508.04324*, 2025b.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems (NeurIPS)*, 33:6840–6851, 2020.

Hopkins, H. H. On the diffraction theory of optical images. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 217:408 – 432, 1953.

Jia, N. and Lam, E. Y. Machine learning for inverse lithography: using stochastic gradient descent for robustphotomask synthesis. *Journal of Optics*, 12(4):045601, 2010.

Jiang, B., Liu, L., Ma, Y., Zhang, H., Yu, B., and Young, E. F. Neural-ilt: Migrating ilt to neural networks for mask printability and complexity co-optimization. In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2020.

Jiang, B., Liu, L., Ma, Y., Yu, B., and Young, E. F. Neural-ilt 2.0: Migrating ilt to domain-specific and multitask-enabled neural network. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 41(8):2671–2684, 2021.

Jiang, H., Yang, J., Xu, R., Fang, M., and Liu, D. BBDM-ILT: Brownian bridge diffusion model for inverse lithography technology. *Journal of Micro/Nanopatterning, Materials, and Metrology*, 24(2):023202, 2025. doi: 10.1117/1.JMM.24.2.023202.

Jin, Q., Liu, Y., Jiang, Y., Sun, Q., and Zhuo, C. Unitho: A unified multi-task framework for computational lithography. *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, 2025a.

Jin, Q., Peng, Q., Liu, Y., Qiu, X., and Sun, Q. Recent advances in computational lithography technology. *Moore and More*, 2(1):1–18, 2025b.

Kim, H., Lee, J., and Ahn, H.-K. Rectangular partitions of a rectilinear polygon. *Computational Geometry*, 110: 101965, 2023.

Kloeden, P. E. and Pearson, R. The numerical solution of stochastic differential equations. *The ANZIAM Journal*, 20(1):8–12, 1977.

Labs, B. F. Flux. https://github.com/black-forest-labs/flux, 2024.

Lam, S. K., Pitrou, A., and Seibert, S. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015.

Levinson, H. J. *Principles of lithography*, volume 146. SPIE press, 2005.

Li, J., Cui, Y., Huang, T., Ma, Y., Fan, C., Yang, M., and Zhong, Z. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE. *arXiv preprint arXiv:2507.21802*, 2025.

Li, Y., Wang, Y., Zhu, Y., Zhao, Z., Lu, M., She, Q., and Zhang, S. BranchGRPO: Stable and efficient GRPO with structured branching in diffusion models. *International Conference on Learning Representations (ICLR)*, 2026.

Liang, X., Ouyang, Y., Yang, H., Yu, B., and Ma, Y. RL-OPC: Mask optimization with deep reinforcement learning. *IEEE Transactions on Computer-Aided Design of*

Integrated Circuits and Systems (TCAD), 43(1):340–351, 2024a.

Liang, X., Yang, H., Liu, K., Yu, B., and Ma, Y. Camo: Correlation-aware mask optimization with modulated reinforcement learning. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2024b.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Liu, G.-T., Tai, W.-C., Lin, Y.-T., Jiang, I. H.-R., Shiely, J. P., and Cheng, P.-J. Sub-resolution assist feature generation with reinforcement learning and transfer learning. In *International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2022.

Liu, H., Huang, H., Wang, J., Liu, C., Li, X., and Ji, X. DiverseGRPO: Mitigating mode collapse in image generation via diversity-aware GRPO. *arXiv preprint arXiv:2512.21514*, 2025a.

Liu, J., Liu, G., Liang, J., Li, Y., Liu, J., Wang, X., Wan, P., ZHANG, D., and Ouyang, W. Flow-GRPO: Training flow matching models via online RL. *Advances in Neural Information Processing Systems (NeurIPS)*, 38:40783–40818, 2025b.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *International Conference on Learning Representations (ICLR)*, 2023.

Luo, Y., Liang, X., and Ma, Y. Enabling robust inverse lithography with rigorous multi-objective optimization. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2024.

Lyu, Q., Chen, Z., Wang, C., Shi, H., Gao, S., Piao, R., Zeng, Y., Si, J., Ding, F., Li, J., et al. Multi-GRPO: Multi-group advantage estimation for text-to-image generation with tree-based trajectories and multiple rewards. *arXiv preprint arXiv:2512.00743*, 2025.

Mack, C. *Fundamental principles of optical lithography: the science of microfabrication*. John Wiley & Sons, 2008.

Moreau, W. M. *Semiconductor lithography: principles, practices, and materials*. Springer Science & Business Media, 2012.

Otto, O. W., Garofalo, J. G., Low, K., Yuan, C.-M., Henderson, R. C., Pierrat, C., Kostelak, R. L., Vaidya, S., and Vasudev, P. Automated optical proximity correction: a rules-based approach. In *Optical/Laser Microlithography VII*, volume 2197, pp. 278–293. SPIE, 1994.

Pang, L. Inverse lithography technology: 30 years from concept to practical, full-chip reality. *Journal of Micro/Nanopatterning, Materials, and Metrology*, 20(3): 030901–030901, 2021.

Poonawala, A. and Milanfar, P. Mask design for optical microlithography—an inverse imaging problem. *IEEE Transactions on Image Processing (TIP)*, 16(3):774–788, 2007.

Raghuvanshi, N., Narain, R., and Lin, M. C. Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(5):789–801, 2009.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10684–10695, 2022.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

Sun, S., Chen, X., Yang, F., Yu, B., Li, S., and Zeng, X. Efficient model-based opc via graph neural network. In *2023 International Symposium of Electronics Design Automation (ISEDA)*, pp. 449–455. IEEE, 2023a.

Sun, S., Yang, F., Yu, B., Shang, L., and Zeng, X. Efficient ilt via multi-level lithography simulation. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2023b.

Sun, S., Yang, F., Yu, B., Shang, L., and Zeng, X. Adaptive ilt via multi-level lithography simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.

Wallace, B., Dang, M., Rafailov, R., Zhou, L., Lou, A., Purushwalkam, S., Ermon, S., Xiong, C., Joty, S., and Naik, N. Diffusion model alignment using direct preference optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8228–8238, 2024.

Wang, F. and Yu, Z. Coefficients-Preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Wang, H., Xu, Y., Feng, Y., Geng, H., and Xiong, S. HiFiMO: a high-resolution flexible mask optimization framework with diffusion model. *Journal of Micro/Nanopatterning, Materials, and Metrology*, 24(3): 033202, 2025.

Wang, J., Liang, J., Liu, J., Liu, H., Liu, G., Zheng, J., Pang, W., Ma, A., Xie, Z., Wang, X., et al. GRPO-Guard: Mitigating implicit over-optimization in flow matching via regulated clipping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5988–5998, 2026.

Weng, C.-L., Wu, C.-Y., and Lee, Y.-C. Enhanced corner sharpness in dmd-based scanning maskless lithography using optical proximity correction and genetic algorithm. *Optics Express*, 32(25):45357–45372, 2024.

Wu, R., Dong, L., and Wei, Y. Method for optical proximity correction based on a vector imaging model. *Applied Optics*, 63(10):2719–2727, 2024.

Wu, Y., Chen, Y., Xia, Y., Zhao, Y., Wang, J., He, X., GENG, H., and Yu, J. TokMan:tokenize manhattan mask optimization for inverse lithography. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Wu, Y., Chen, Y., Yin, S., Wang, N., Wu, T., He, X., Geng, H., and Yu, J. Lvm-mo: A large vision model pioneer on full-chip mask optimization. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7, 2025b.

Xu, X., Matsunawa, T., Nojima, S., Kodama, C., Kotani, T., and Pan, D. Z. A machine learning based framework for sub-resolution assist feature generation. In *International Symposium on Physical Design (ISPD)*, pp. 161–168, 2016.

Xue, Z., Wu, J., Gao, Y., Kong, F., Zhu, L., Chen, M., Liu, Z., Liu, W., Guo, Q., Huang, W., et al. DanceGRPO: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Yang, H. and Ren, H. ILILT: Implicit learning of inverse lithography technologies. *International Conference on Machine Learning (ICML)*, 235:56319–56331, 2024.

Yang, H. and Ren, H. Pushing the limits of inverse lithography with generative reinforcement learning. *arXiv preprint arXiv:2602.19027*, 2026.

Yang, H., Li, S., Ma, Y., Yu, B., and Young, E. F. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

Yang, H., Li, Z., Sastry, K., Mukhopadhyay, S., Anandkumar, A., Khailany, B., Singh, V., and Ren, H. Large scale mask optimization via convolutional fourier neural operator and litho-guided self training. *arXiv preprint arXiv:2207.04056*, 2022a.

Yang, H., Li, Z., Sastry, K., Mukhopadhyay, S., Kilgard, M., Anandkumar, A., Khailany, B., Singh, V., and Ren, H. Generic lithography modeling with dual-band optics-inspired neural networks. In *ACM/IEEE Design Automation Conference (DAC)*, pp. 973–978, 2022b.

Yang, Y., Liu, K., Gao, Y., Wang, C., and Cao, L. Advancements and challenges in inverse lithography technology: a review of artificial intelligence-based approaches. *Light: Science & Applications*, 14(1):250, 2025.

Yang, Y.-F. and Hsiao, H.-H. Photolithography overlay map generation with implicit knowledge distillation diffusion transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15288–15297, 2025.

Yu, Z., Chen, G., Ma, Y., and Yu, B. A gpu-enabled level-set method for mask optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 42(2):594–605, 2022.

Yu, Z., Liao, P., Ma, Y., Yu, B., and Wong, M. D. Ctm-sraf: Continuous transmission mask-based constraint-aware subresolution assist feature generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 42(10):3402–3411, 2023.

Zable, H. R., Fujimura, A., Komagata, T., Nakagawa, Y., and Petersen, J. S. Writing wavy metal 1 shapes on 22-nm logic wafers with less shot count. In *Photomask and Next-Generation Lithography Mask Technology XVII*, volume 7748, pp. 289–298. SPIE, 2010.

Zhang, S., Zhang, Z., Dai, C., and Duan, Y. E-GRPO: High entropy steps drive effective reinforcement learning for flow models. *arXiv preprint arXiv:2601.00423*, 2026.

Zhang, X., Zheng, S., Chen, G., Zhu, B., Xu, H., and Yu, B. Fracturing-aware curvilinear ilt via circular e-beam mask writer. In *ACM/IEEE Design Automation Conference (DAC)*, 2024.

Zheng, S., Yang, H., Zhu, B., Yu, B., and Wong, M. Lithobench: Benchmarking ai computational lithography for semiconductor manufacturing. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:30243–30254, 2023.

Zhou, Y., Ling, P., Bu, J., Wang, Y., Zang, Y., Wang, J., Niu, L., and Zhai, G. Fine-Grained GRPO for precise preference alignment in flow models. *IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2026.

Zhu, B., Zheng, S., Yu, Z., Chen, G., Ma, Y., Yang, F., Yu, B., and Wong, M. D. L2O-ILT: Learning to optimize inverse lithography techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 43(3):944–955, 2023.

Zhu, J., Yue, L., Li, Y., Liu, X., Wu, Q., Wang, Q., and Li, Y. Machine learning-enhanced model-based optical proximity correction framework with convolutional neural network-based variable threshold method near the diffraction limit. *IEEE Access*, 2025.

A. Related Work

A.1. Optical Proximity Correction

Optical proximity correction (OPC) (Levinson, 2005; Moreau, 2012; Mack, 2008; Jin et al., 2025b) is the mainstream mask correction technique in optical lithography to compensate for imaging distortions and process variations. Rule-based OPC (Otto et al., 1994; Dong & Zhang, 2017; Weng et al., 2024; Liang et al., 2024b) applies pre-defined geometric recipes (e.g., bias tables and pattern-dependent corrections) to modify mask features, offering high throughput but limited fidelity under complex imaging conditions. Model-based OPC (Wu et al., 2024; Zhu et al., 2025; Sun et al., 2023a; Geng et al., 2019) uses calibrated lithography and resist models to predict wafer contours and iteratively updates mask edges to minimize objective functions such as EPE and CD errors, achieving higher accuracy at the cost of heavier computation. Sub-resolution assist features (SRAFs) (Alawieh et al., 2019; Xu et al., 2016; Liu et al., 2022) are commonly inserted as a resolution enhancement technique to improve image contrast and process window; in practice, SRAF insertion can be performed as a standalone step or co-optimized within the OPC flow. In contrast to edge-centric OPC, inverse lithography technology (ILT) formulates mask synthesis as a global optimization problem and can naturally generate complex mask geometries, including SRAF-like patterns (Yu et al., 2023), to better satisfy lithographic objectives. Beyond mask optimization, learning-based methods have also been explored to model and generate photolithography overlay maps for process monitoring and correction (Yang & Hsiao, 2025). We give an overview of the relationship between traditional OPC, ILT, and SRAF in Fig. 10.

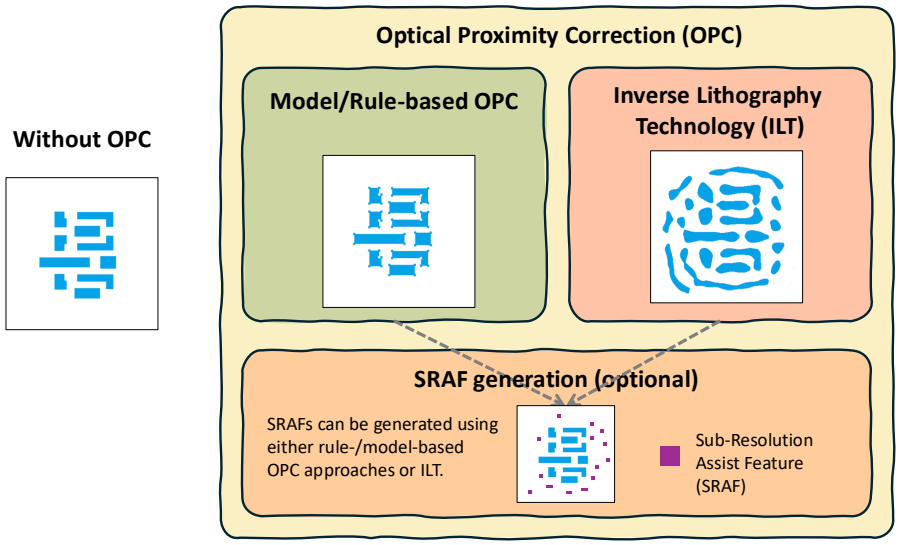

Figure 10. **Relationship between OPC, ILT, and optional SRAF insertion/generation.**

A.2. Inverse Lithography Technology

ILT approaches can be broadly categorized into two paradigms. Optimization-based methods (Gao et al., 2014; Yu et al., 2022; Sun et al., 2023b; 2025; Jia & Lam, 2010; Luo et al., 2024) iteratively refine mask patterns through gradient-based optimization of differentiable metrics such as L2 distance and process variation band (PVB). Representative works include MOSAIC (Gao et al., 2014), which incorporates process window awareness, and GPU-accelerated level-set methods (Yu et al., 2022) that improve computational efficiency. Multi-level simulation approaches (Sun et al., 2023b; 2025) further enhance optimization speed while maintaining accuracy. However, these methods are limited to differentiable objectives and require multiple costly lithography simulations.

Learning-based approaches (Yang et al., 2018; Chen et al., 2020; Jiang et al., 2020; 2021; Yang et al., 2022b;a; Zhu et al., 2023; Wu et al., 2025a;b; Liang et al., 2024a) leverage neural networks to learn direct mappings from target patterns to masks, significantly accelerating mask generation. GAN-OPC (Yang et al., 2018) pioneered the use of generative adversarial networks for mask optimization, while Neural-ILT (Jiang et al., 2020; 2021) and DAMO (Chen et al., 2020) demonstrated full-chip scalability. More recent works incorporate domain-specific architectures (Yang et al., 2022b), learning-to-optimize frameworks (Zhu et al., 2023), tokenization-based approaches with manufacturing constraints (Wu et al., 2025a), and large

vision models for full-chip optimization (Wu et al., 2025b). Diffusion models have also been proposed for ILT tasks, such as HiFiMo (Wang et al., 2025) and BBDM (Jiang et al., 2025). However, these methods only optimize the L2 loss and PVB metric rather than all evaluation objectives. Moreover, their efficiency is still constrained by the number of inference steps required in diffusion. The LithoBench benchmark (Zheng et al., 2023) provides standardized evaluation protocols for comparing ILT methods. Recent hybrid approaches (Yang & Ren, 2024) attempt to combine optimization and learning, though they remain constrained by computational overhead and inability to optimize non-differentiable metrics such as shot count and edge placement error (EPE).

Reinforcement learning has been explored for mask optimization. RL-OPC (Liang et al., 2024a) pioneered the use of deep reinforcement learning in optical proximity correction, demonstrating the capability of RL to directly optimize non-differentiable objectives without requiring differentiable proxies. However, RL-OPC operates within the traditional edge-based OPC framework, performing local adjustments to individual mask features rather than holistic mask generation. This edge-centric approach limits its optimization scope to local neighborhoods and makes it challenging to simultaneously balance conflicting metrics across the entire layout. Furthermore, traditional shot count computation remains a bottleneck for RL-based optimization, as the NP-hard non-overlapping rectangle partitioning problem requires prohibitive computation time. Our work addresses these limitations by formulating generative ILT within a flow-matching framework and introducing an ultra-fast shot count estimation method that enables efficient GRPO-based reinforcement learning fine-tuning. Concurrent work (Yang & Ren, 2026) also applies GRPO to ILT, but uses a WGAN generator refined by a post-hoc numerical solver at inference, whereas our method generates masks directly with one-step rectified flow.

Mask manufacturability considerations, particularly shot count optimization, have been extensively studied in the EDA community (Fujimura et al., 2010; Zable et al., 2010; Chua et al., 2011; Chan et al., 2016; Zhang et al., 2024). These works establish the importance of efficient mask fracturing and shot placement for multi-beam mask writers, motivating the development of fast shot count estimation methods for computational lithography.

A.3. GRPO for Flow-Matching Models

Diffusion and flow-matching models (Ho et al., 2020; Song et al., 2021; Rombach et al., 2022; Lipman et al., 2023) decompose visual generation into iterative denoising processes, significantly advancing visual synthesis and achieving state-of-the-art performance in image and video generation. Inspired by the success of reinforcement learning (RL) in large language models (LLMs), optimization techniques such as PPO (Black et al., 2023) and DPO (Wallace et al., 2024) have been adapted to diffusion models, facilitating preference alignment and enhancing task-specific outcomes. In a similar vein, Flow-GRPO (Liu et al., 2025b) and DanceGRPO (Xue et al., 2025) incorporate GRPO-style policy optimization into flow-matching frameworks by reformulating deterministic ODE sampling as stochastic SDE processes, thereby introducing exploratory noise for group-based policy improvement. More recently, MixGRPO (Li et al., 2025) introduced a hybrid ODE–SDE sampling strategy that enhances training efficiency without compromising generative quality. Concurrently, Flow-CPS (Wang & Yu, 2025) identified a critical limitation in the SDE sampling employed by Flow-GRPO and DanceGRPO—inconsistent noise coefficients across timesteps—which results in residual noise accumulation and imprecise reward estimation. To mitigate this, Flow-CPS proposes a noise-consistent SDE sampling method that improves reward accuracy and accelerates GRPO convergence. In parallel, TempFlowGRPO (He et al., 2025b) and G^2RPO (Zhou et al., 2026) tackle the issues of reward sparsity and inaccuracy arising from assigning a single global reward to multi-step SDE trajectories. Along the line of addressing sparse/ambiguous supervision over multi-step trajectories, E-GRPO (Zhang et al., 2026) identifies that only high-entropy steps contribute to effective exploration, and proposes entropy-aware step consolidation with a multi-step group-normalized advantage to improve learning efficiency. BranchGRPO (Li et al., 2026) reorganizes the rollout process into a branching tree structure, where shared prefixes reduce computational overhead and pruning eliminates low-reward paths and redundant depths. GRPO-Guard (Wang et al., 2026) introduces a regulated clipping mechanism that stabilizes the optimization process and substantially alleviates implicit over-optimization, achieving this without relying on strong KL regularization. Beyond improvements in sampling and credit assignment, recent GRPO-based visual alignment work has begun to explicitly address diversity degradation and reward hacking. DiverseGRPO (Liu et al., 2025a) mitigates mode collapse by adding a distribution-level creativity bonus via semantic grouping and by applying structure-aware regularization that emphasizes early denoising to preserve diversity. GARDO (He et al., 2025a) tackles reward hacking from imperfect proxy rewards through gated, uncertainty-aware regularization with an adaptively updated reference, while further encouraging mode coverage with diversity-aware reward shaping. Multi-GRPO (Lyu et al., 2025) improves sparse-reward credit assignment and multi-objective stability by using tree-based temporal grouping for advantage estimation and reward-wise grouping for decoupled advantage computation before aggregation.

B. Additional Details on Fast Shot Count

We provide a detailed breakdown of the algorithm used for ultra-fast shot count estimation. The method transforms the computationally expensive non-overlapping rectangle partitioning problem into an efficient Set Cover problem solved via Integer Linear Programming (ILP). The overall process is presented in Algorithm 1.

B.1. Overview

Given a binary mask M of size $N \times N$ ($N = 512$ in our LithoGRPO), we aim to find the minimum number of rectangular shots (which may overlap) that cover all pixels with value 1. The algorithm consists of three main steps:

1. **Maximal Rectangle Generation**: Extract all maximal rectangles from the binary mask in $O(N^2)$ time.
2. **Redundancy Pruning**: Remove rectangles that are fully contained within others, reducing the problem size.
3. **ILP Formulation**: Formulate a Set Cover ILP with scan-line optimization.
4. **ILP Solve**: Solve the ILP task with solver.

B.2. Step 1: Maximal-Rectangle Generation

A *maximal rectangle* is an axis-aligned rectangle that consists exclusively of foreground pixels (1's) and cannot be enlarged in any direction without hitting background (0's). We enumerate all such rectangles with a histogram sweep that scans the mask once:

1. For every row $r \in \{0, \ldots, N-1\}$ maintain a height array $h[c]$ that counts the number of consecutive 1's directly above (and including) pixel (r, c).
2. Treat $h[\cdot]$ as a histogram and, in $O(N)$ time, enumerate *all* maximal rectangles whose bottom edge lies on row r using the classic monotonic-stack algorithm for the "largest rectangle in a histogram" problem. For example, in an 8×8 binary mask, the maximal rectangles for $r \in \{0, 1, 2, 3, 5\}$ are shown in Fig. 11.
3. Append every rectangle found in the second step to the candidate set $\mathcal{R}_{\text{all}}$; let $K_{\text{all}} = |\mathcal{R}_{\text{all}}|$.

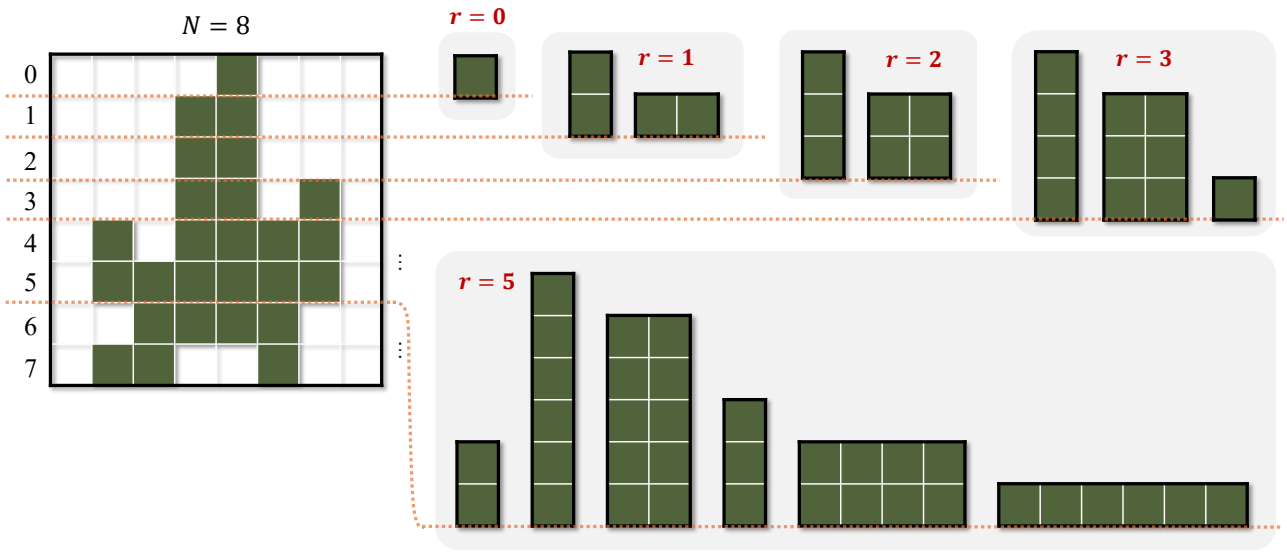

Figure 11. **Visualization of maximal-rectangle generation.** Left: an 8×8 binary mask, where colored (filled) pixels indicate 1's. Right: maximal rectangles enumerated for selected rows $r \in \{0, 1, 2, 3, 5\}$, where each rectangle's bottom edge lies on row r.

Because each of the N rows is processed once, and the per-row work is linear in the image width, the total running time of this stage is $O(N^2)$.

The set $\mathcal{R}_{\text{all}}$ is *complete*: any feasible shot rectangle must be contained in at least one element of $\mathcal{R}_{\text{all}}$. Subsequent stages only remove redundancies.

B.3. Step 2: Redundancy Pruning

The raw candidate set $\mathcal{R}_{\text{all}}$ still contains many rectangles that do not need to appear in the final covering. A rectangle A is redundant if it is fully contained in another rectangle B because any solution that uses A can always replace it by B without increasing the objective.

We employ a Numba-accelerated (Lam et al., 2015) greedy strategy:

1. Sort all rectangles by area in descending order.
2. For every rectangle, test whether it is contained in any rectangle that has already been kept.
3. Append the rectangle to the pruned set $\mathcal{R}_{\text{pruned}}$ iff it is not contained.

After this step, we obtain a pruned set of rectangles, denoted by $\mathcal{R}_{\text{pruned}}$, and define $K = |\mathcal{R}_{\text{pruned}}|$. The procedure needs $\mathcal{O}\big(K_{\text{all}}^2\big)$ comparisons in the worst case, yet it is fast in practice because $K_{\text{all}} \ll N^2$ for real masks. For extremely large instances, one can switch to the $\mathcal{O}(K_{\text{all}} \log^2 K_{\text{all}})$ variant based on a 2-D Binary Indexed Tree, but we found its overhead unnecessary for our datasets.

B.4. Step 3: ILP Formulation with Scan-Line Constraint Generation

Let $x_i \in \{0, 1\}$ indicate whether pruned rectangle $R_i \in \mathcal{R}_{\text{pruned}}$ is selected. Let $M : \{0, \dots, H-1\} \times \{0, \dots, W-1\} \to \{0, 1\}$ be the binary target mask, where $M(p) = 1$ indicates that pixel p belongs to the target shape (i.e., is to be covered).

The covering ILP is

$$\text{minimise} \quad \sum_{i=1}^{K} x_i \tag{16}$$

$$\text{subject to} \quad \sum_{i:\ p \in R_i} x_i \ \geq 1, \qquad \forall\, p \text{ with } M(p) = 1, \tag{17}$$

where $K = |\mathcal{R}_{\text{pruned}}|$. Creating one constraint per target pixel would yield up to $\Theta(N^2)$ constraints, which is computationally prohibitive. We therefore apply a scan-line reduction:

1. For each row r, compute the run-length encoding (RLE) of $M(r, \cdot)$, i.e., the set of maximal contiguous foreground segments (s, e) such that $M(r, c) = 1$ for all $c \in [s, e]$. For example, if $M(r, \cdot) = 0011110011100$, then the RLE segments are $(s, e) = (2, 5)$ and $(8, 10)$ (0-indexed).
2. For each row r collect the rectangles that intersect it, $\mathcal{R}_r = \{R_i \mid R_i \cap \text{row } r \neq \varnothing\}$.
3. Collect the *critical* x-coordinates in row r at which coverage can change: the left/right x-boundaries of all rectangles in $\mathcal{R}_r$, together with the endpoints s, e of the foreground RLE segments in row r. For example, with RLE segments $(2, 5)$ and $(8, 10)$ and rectangles spanning $[1, 6)$, $[4, 9)$, and $[8, 12)$ on row r, we collect $\{1, 6, 4, 9, 8, 12\} \cup \{2, 5, 8, 10\}$.
4. Sort and deduplicate the critical coordinates. Consecutive pairs define intervals on which the set of covering rectangles is constant. For the example above, sorting yields $\{1, 2, 4, 5, 6, 8, 9, 10, 12\}$ and thus the intervals $[1, 2), [2, 4), [4, 5), \dots, [10, 12)$.
5. For every interval that overlaps foreground pixels, insert one covering constraint using any representative column c inside that interval: $\sum_{i:\ c \in R_i} x_i \geq 1$. For example, if an interval is $[4, 5)$ and on row r it is covered by exactly R_1 and R_2, we may choose $c = 4$ and add $x_1 + x_2 \geq 1$.

This replaces per-pixel covering constraints with per-interval constraints without changing the objective or the feasible set: within a fixed row r, the set $\{i \mid c \in R_i\}$ is constant for all columns c inside the same interval between consecutive critical x-coordinates, so a single constraint at any representative column c enforces coverage for every foreground pixel in that interval.

For a fixed row r, we collect $\Theta(k_r + t_r)$ critical x-coordinates (rectangle boundaries and RLE endpoints) and sort them, which costs $\mathcal{O}\big((k_r + t_r)\log(k_r + t_r)\big)$ time. The sorted coordinates induce $\Theta(k_r + t_r)$ intervals; to generate one covering constraint per foreground-overlapping interval we test membership against the k_r rectangles intersecting the row, costing $\mathcal{O}(k_r)$ per interval and thus $\mathcal{O}\big(k_r(k_r + t_r)\big)$ per row. Summing the per-row costs over $r = 1, \ldots, N$ yields

$$\sum_{r=1}^{N} \mathcal{O}\Big((k_r + t_r)\log(k_r + t_r) + k_r(k_r + t_r)\Big) = \sum_{r=1}^{N} \mathcal{O}\big(k_r^2 + k_r \log(k_r + t_r)\big),$$

where the last step groups lower-order terms for readability, and k_r is the number of pruned rectangles intersecting row r and t_r is the number of foreground RLE runs in that row (for the example above, $k_r = 3$ and $t_r = 2$). In the worst case $k_r = \Theta(K)$ for all rows, yielding the bound $\mathcal{O}(NK^2 + NK\log K)$. In practice, masks exhibit spatial locality so $k_r \ll K$ for most rows; consequently the quadratic term $\sum_r k_r^2$ is much smaller than the worst-case NK^2, and the runtime is often dominated by sorting critical points, i.e., $\sum_r \mathcal{O}\big(k_r \log(k_r + t_r)\big)$.

B.5. Step 4: ILP Solve

The final ILP has K binary variables and C covering constraints. Let $\bar{k}$ denote the average number of rectangles covering a foreground interval (equivalently, the average number of nonzeros per covering constraint). Due to limited overlap, $\bar{k}$ is small in practice, so the constraint matrix is sparse. Consequently, the resulting ILP instances are typically solved quickly by a modern solver. We formulate the ILP using the PuLP package and solve it with the CBC solver.

B.6. Computational Complexity

Let N denote the image size ($H = W = N$). Let K_{all} be the number of locally maximal rectangles produced in Step 1, and let $K \le K_{\text{all}}$ be the number remaining after redundancy pruning. For each row r, let k_r be the number of pruned rectangles intersecting row r, and let t_r be the number of foreground RLE runs in that row. Note that $\sum_{r=1}^{N} k_r \le KN$ (each rectangle intersects at most N rows) and $\sum_{r=1}^{N} t_r \le N\lceil N/2 \rceil = \mathcal{O}(N^2)$.

1. **Maximal-rectangle enumeration.** We compute the maximal rectangles in $\mathcal{O}(N^2)$ time:

$$T_1 = \mathcal{O}(N^2).$$

2. **Redundancy pruning.** Sorting by area and greedy pairwise containment tests yield

$$T_2 = \mathcal{O}(K_{\text{all}}^2).$$

3. **Scan-line constraint generation.** For each row r, we collect $\Theta(k_r + t_r)$ critical x-coordinates and sort them in $\mathcal{O}((k_r + t_r)\log(k_r + t_r))$ time. These points define $\Theta(k_r + t_r)$ intervals. For every interval overlapping foreground pixels, we add one covering constraint; in the simplest implementation, building each constraint scans the k_r rectangles intersecting the row, giving $\mathcal{O}(k_r(k_r + t_r))$ time per row. Summing over rows,

$$T_3 = \sum_{r=1}^{N} \mathcal{O}\Big((k_r + t_r)\log(k_r + t_r) + k_r(k_r + t_r)\Big).$$

   Using $k_r \le K$ and $t_r \le \lceil N/2 \rceil$ for all r gives the coarse worst-case bound

$$T_3 = \mathcal{O}\big(N(K + N)\log(K + N) + NK^2 + N^2K\big).$$

   In particular, when $K \ge N$ this simplifies to $T_3 = \mathcal{O}(NK^2)$.

Algorithm 1 Fast Shot Count Estimation

Require: Binary mask $M \in \{0,1\}^{N\times N}$
Ensure: Minimum overlapping shot count S
FastShotCount(M):
Step 1: Maximal-Rectangle Generation
$\mathcal{R}_{\text{all}} \leftarrow$ MAXIMALRECTANGLES(M)
Step 2: Redundancy Pruning
$\mathcal{R}_{\text{pruned}} \leftarrow$ PRUNECONTAINED$(\mathcal{R}_{\text{all}})$
$K \leftarrow |\mathcal{R}_{\text{pruned}}|$
Step 3: ILP Formulation with Scan-Line Constraints
create binary vars $x_1, \ldots, x_K$, objective $\min \sum_i x_i$
runs $\leftarrow$ RUNLENGTHENCODE(M)
for $r \leftarrow 0$ **to** $N-1$ **do**
  $\mathcal{R}_r \leftarrow \{i \in \{1,\ldots,K\} \mid R_i$ intersects row $r\}$
  $C \leftarrow$ CRITICALPOINTS$(\mathcal{R}_r, \text{runs}[r])$
  sort and deduplicate C ascending
  if $|C| < 2$ **then**
    continue
  end if
  for $j \leftarrow 1$ **to** $|C|-1$ **do**
    $x_L \leftarrow C_j, \quad x_R \leftarrow C_{j+1}$
    if OVERLAPSFOREGROUND$(\text{runs}[r], [x_L, x_R))$ **then**
      $c \leftarrow x_L$
      $\mathcal{I} \leftarrow \{i \in \mathcal{R}_r \mid c \in R_i\}$
      if $\mathcal{I} = \varnothing$ **then**
        return infeasible
      end if
      add constraint $\sum_{i\in\mathcal{I}} x_i \geq 1$
    end if
  end for
end for
Step 4: ILP Solve
solution $\leftarrow$ SOLVEILP()
$S \leftarrow \sum_{i=1}^{K} \text{solution}(x_i)$
return S

Therefore,

$$T = \mathcal{O}\big(N^2 + K_{\text{all}}^2 + NK^2\big).$$

In typical masks, spatial locality makes k_r small for most rows, so the observed runtime is closer to $\sum_r \mathcal{O}((k_r + t_r)\log(k_r + t_r))$ plus a modest scanning cost. By contrast, a naive per-pixel approach has $\Theta(|\Omega|)$ constraints and can take $\Theta(|\Omega|K)$ time just to construct them (worst-case $\Theta(N^2K)$), whereas our scan-line method constructs far fewer constraints.

B.7. Traditional Shot Count Analysis

According to the practical implementation in LithoBench (Zheng et al., 2023), the shot count is computed using the `adaptive-boxes` (AdaBox) package, which greedily decomposes the foreground pixels into a set of axis-aligned rectangles.

Given a binary mask, AdaBox first represents all foreground pixels as a point set and then iteratively generates rectangles until all points are covered. In each iteration, it samples several seed points from the currently uncovered set, grows a candidate rectangle around each seed by expanding along the seed's row/column, selects the candidate with the largest area, and finally marks all points inside the chosen rectangle as covered. The number of selected rectangles is returned as the shot

count.

This procedure is slow in our setting, mainly due to repeated point-set scans. Evaluating a single candidate rectangle involves filtering the uncovered point array to collect points on the seed's row/column (followed by sorting and boundary extraction), which costs $\Theta(M)$ time up to logarithmic factors, where M is the number of foreground pixels. After a rectangle is selected, AdaBox updates the global assignment by testing all points against the rectangle bounds, incurring another $\Theta(M)$ full pass. Let S denote the number of random seeds evaluated per iteration; LithoBench uses $S = 4$. Let R be the number of rectangles produced by the greedy procedure, i.e., the resulting (reported) shot count. A coarse runtime estimate is therefore

$$T_{\text{AdaBox}} \approx \Theta\big(R\,(S\,M + M)\big) = \Theta(RSM),$$

again up to sorting-related logarithmic factors. Since $M \le N^2$, this implies a worst-case bound $T_{\text{AdaBox}} = \mathcal{O}(RSN^2)$ when the foreground occupies a constant fraction of an $N \times N$ mask. Empirically, our masks are often fragmented and AdaBox typically outputs $R = \mathcal{O}(10^2\text{–}10^3)$ rectangles, so the repeated $\Theta(M)$ scans dominate and can lead to minute-level runtimes on 512×512 instances even with the small constant $S = 4$.

In contrast, our fast shot count formulates the problem as an Integer Linear Program (ILP) over a carefully chosen set of candidate rectangles. A naive approach would enumerate all possible axis-aligned rectangles contained in the foreground region, yielding $\mathcal{O}(N^4)$ candidates in the worst case, which is far too many for practical ILP solving. However, we observe that when rectangles are allowed to overlap, any feasible cover using a non-maximal rectangle can be replaced by one using a maximal rectangle that contains it, without increasing the objective. This dominance property allows us to restrict the candidate set to *maximal rectangles* only, i.e., rectangles that cannot be extended in any direction while remaining inside the foreground. Using a standard histogram-based technique, we enumerate all maximal rectangles in $\mathcal{O}(N^2)$ time: for each row, we maintain the height of consecutive foreground pixels above it, then apply a monotonic stack to extract all locally maximal rectangles ending at that row. This typically produces $K = \mathcal{O}(10^3\text{–}10^4)$ candidates.

To further reduce the ILP size, we employ scan-line aggregation: instead of generating one covering constraint per foreground pixel, we observe that pixels sharing the same row and covered by identical sets of candidate rectangles can be merged into a single constraint. This reduces the number of constraints from $\mathcal{O}(M)$ to $\mathcal{O}(K)$ in practice. The resulting ILP has K binary variables and only a few thousand constraints, which modern solvers such as CBC handle in sub-second time.

Therefore, the speedup primarily comes from (1) restricting candidates to maximal rectangles via the overlap-dominance argument, (2) avoiding R rounds of point-set filtering and global re-scans, and (3) replacing per-pixel constraints with aggregated scan-line constraints, yielding a compact optimization problem solvable in a single pass.

Additionally, due to its greedy nature with random seed sampling, AdaBox produces non-deterministic results: running the same mask multiple times may yield different shot counts. In contrast, our ILP-based formulation is fully deterministic: given the same mask, it always returns the same optimal shot count, which is desirable for reliable evaluation and reproducibility.

C. Log-Probability Computation

For GRPO optimization, we compute the log-probability of a sampled trajectory as the sum of per-step transition log-probabilities, following Flow-GRPO (Liu et al., 2025b):

$$\log \pi_\theta(\mathbf{x}) = \sum_t \log p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t). \tag{18}$$

Each transition follows a Gaussian distribution:

$$\log p(\mathbf{x}_{t+\Delta t}|\mathbf{x}_t) \propto -\frac{\|\mathbf{x}_{t+\Delta t} - \boldsymbol{\mu}_t\|^2}{2\sigma_t^2 \Delta t}, \tag{19}$$

where $\boldsymbol{\mu}_t = \mathbf{x}_t + \big[\mathbf{v}_\theta(\mathbf{x}_t, t) + \frac{\sigma_t^2}{2t}(\mathbf{x}_t + (1-t)\mathbf{v}_\theta(\mathbf{x}_t, t))\big]\Delta t$ is the deterministic component in Eq. 15, and we assume pixel-wise independence for computational efficiency.

Although the colored noise used in our SDE sampler introduces spatial correlations between pixels, we approximate the log-probability using an independent Gaussian. This approximation is valid for GRPO because only the probability ratio $\pi_\theta/\pi_{\text{ref}}$ is used, and both policies share the same noise generation process, causing the approximation error to largely cancel out.

D. Model Architecture

Overall Structure. Our model adopts a symmetric encoder–decoder architecture summarized in Table 4. Each downsampling stage halves the spatial resolution and doubles the channel dimension, while the decoder mirrors this hierarchy with upsampling and skip connections. All convolutions use kernel size 3×3, stride 1, and padding 1. Group normalization (8 groups) and SiLU activation follow each convolution. The network takes a two-channel input (layout and mask) and outputs a single-channel prediction. The base channel width is 64, and it doubles after each down-sampling stage, reaching 1024 channels at the bottleneck.

Table 4. **Model architecture.** The network takes a **two-channel input**, where the first channel is the noisy sample x_t and the second is the conditional target (e.g., mask or layout). Each DownLayer and UpLayer is a residual block with two 3×3 convolutions, GroupNorm (8 groups), and SiLU activation. Skip connections concatenate the encoder and decoder features of matching resolutions.

Stage	Layer(s)	Channels	Output Size
Input	Conv2d, 3×3, padding=1	2→64	256×256
Down1	2×DownLayer; MaxPool(2×2)	64→128	128×128
Down2	2×DownLayer; MaxPool(2×2)	128→256	64×64
Down3	2×DownLayer; MaxPool(2×2)	256→512	32×32
Down4	2×DownLayer; MaxPool(2×2)	512→1024	16×16
Middle	MiddleLayer	1024→1024	16×16
Up1	Upsample($\times 2$); 2×UpLayer	2048→512	32×32
Up2	Upsample($\times 2$); 2×UpLayer	1024→256	64×64
Up3	Upsample($\times 2$); 2×UpLayer	512→128	128×128
Up4	Upsample($\times 2$); 2×UpLayer	256→64	256×256
Output	Conv2d, 1×1	64→1	256×256

Layer Modules. Each DownLayer consists of two 3×3 convolutions (stride 1, padding 1), each followed by GroupNorm (8 groups) and SiLU activation. A residual shortcut adds the input to the output; when channel dimensions differ, a 1×1 convolution aligns them. A linear projection of the time embedding ($t_{\text{emb}} \in \mathbb{R}^{B \times d}$, where each of the B samples has a d-dimensional time embedding) is added to the feature maps before the first convolution. DownLayers are applied before each 2×2 max-pooling operation in the encoder.

The UpLayer mirrors the DownLayer structure. Each block first upsamples features by a factor of 2 using nearest-neighbor interpolation, then applies two 3×3 convolutions with GroupNorm and SiLU activations, plus the same residual connection and time-embedding injection as in the encoder.

The MiddleLayer has the same residual structure as the DownLayer but operates at the bottleneck resolution without pooling or upsampling. It refines the deepest latent features before entering the decoder.

Normalization and Activation. All normalization layers use GroupNorm with eight groups, and all nonlinearities employ the SiLU activation function.

Time Embeddings. Time embeddings are encoded using sinusoidal positional encodings and projected through a learnable linear layer to match the feature-map dimensionality. The resulting embedding is added element-wise to feature maps at each block, allowing the network to incorporate temporal information during training.

E. Detailed Experimental Settings

We report the training and inference settings in our experiments in Table 5. For reinforcement learning fine-tuning (RLFT), the model is optimized for 1000 iterations rather than epochs. Each iteration uses 10 images as input, and for every image, $G = 6$ exploratory samples are generated, forming a mini-batch of 60 samples for one gradient update.

For learning rate scheduling, different strategies are employed for each stage: (1) during pretraining, we use a StepLR scheduler that decays the learning rate by a factor of 0.1 halfway through training; (2) during RLFT, we apply a warm-up with cosine decay scheduler, where the first 5% of iterations are used for linear warm-up and the learning rate gradually decreases to 5% of the maximum; (3) no learning rate scheduling is applied during SFT.

Table 5. **Training and inference settings.** Each column corresponds to a dataset and training stages (Pretrain, SFT, and RLFT). For RLFT, the batch size denotes the total number of samples per update (10 images × 6 generations). Rows such as λ_{PVB} and λ_{L2} are specific to SFT stages only, described in Eq. 11. A dash indicates that the parameter is not applicable.

Hyperparameter	MetalSet			ViaSet			StdContact	
	Pretrain	SFT	RLFT	Pretrain	SFT	RLFT	SFT	RLFT
Epochs	50	25	-	10	1	-	100	-
Batch size	12	12	60	12	12	60	12	60
Learning rate	1e-4	2e-5	2e-6	1e-4	2e-5	2e-6	2e-5	2e-6
λ_{PVB}	–	0.248	–	–	2e-5	–	2e-5	–
λ_{L2}	–	0.002	–	–	2e-5	–	2e-5	–
λ_{flow}	$1 - \lambda_{\text{PVB}} - \lambda_{\text{L2}}$							

F. Supplementary Results

F.1. Noise Type and Level on ViaSet

Similar to Fig. 8, we also report the training status with different noise types and levels in the RLFT stage. The results are shown in Fig. 12. Overall, the conclusions are consistent with the observations on the MetalSet experiments: the *colored* noise with $a = 0.1$ still demonstrates more stable and generally better performance. Although the case with $a = 0.5$ yields a slightly higher reward in this particular setting, it fails to achieve optimal and stable results across all datasets and also exhibits slower convergence.

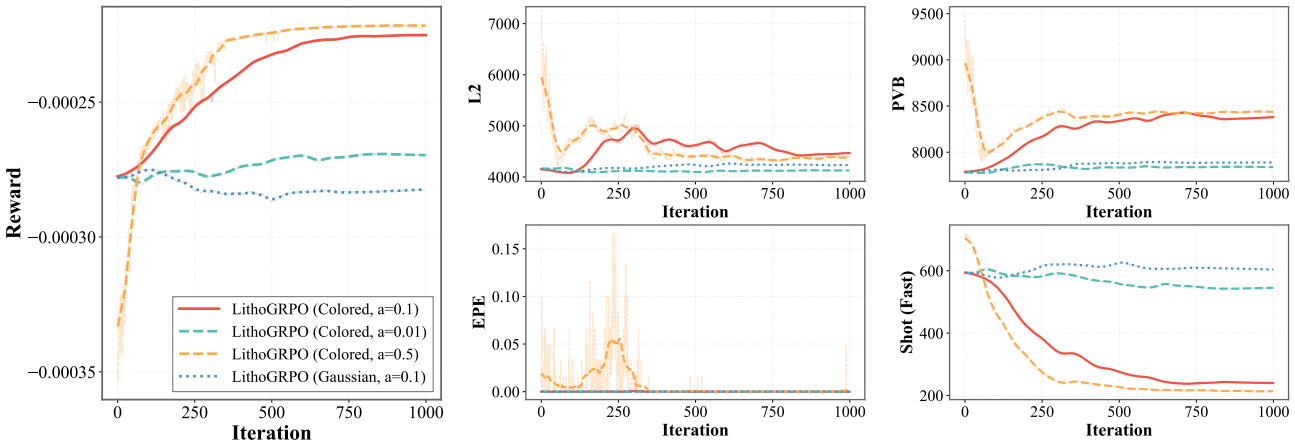

Figure 12. **Noise configurations in SDE (ViaSet).** We compare noise types (Colored vs. Gaussian) and noise levels ($a \in \{0.01, 0.1, 0.5\}$) for LithoGRPO.

F.2. Fast Shot Count Estimation

We employ a fast shot count estimator to accelerate the reward computation during RLFT. Here we analyze the accuracy and reliability of this estimator compared to the ground-truth shot count computed by the full algorithm.

We evaluate the fast estimator on both the StdMetal and StdContact benchmarks generated by LithoGRPO (RLFT). We fit a linear regression between the fast estimator and the exact shot count. The residual errors exhibit a near-zero mean ($\mu = -0.28\%$) with a standard deviation of $\sigma = 4.80\%$, and 95% of all errors fall within $\pm 9.54\%$. These results are summarized in Fig. 13(a, b).

More importantly, for reinforcement learning, the ranking consistency of rewards matters more than their absolute accuracy, since policy gradient methods depend primarily on relative comparisons among samples within each batch. Furthermore, as the reward is normalized within each batch during training, any systematic bias or scale discrepancy in the estimator is effectively eliminated, leaving only the relative ordering to influence gradient updates. As shown in Fig. 13(c), the fast

estimator achieves a Spearman correlation of $\rho = 0.994$ and a Kendall's $\tau = 0.936$, indicating near-perfect preservation of the relative ordering. This strong rank correlation ensures that the RL training signal remains reliable.

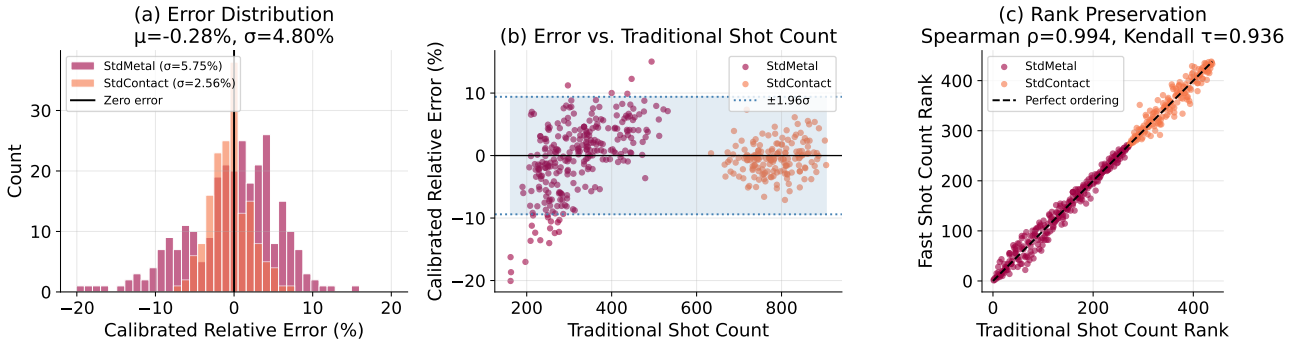

Figure 13. **Analysis of the fast shot count estimator.** (a) Distribution of calibrated relative errors across StdMetal and StdContact benchmarks. (b) Calibrated error as a function of traditional shot count, with 95% confidence bounds shown. (c) Rank preservation between the fast estimator and traditional shot counts, demonstrating high correlation (Spearman $\rho = 0.994$, Kendall $\tau = 0.936$).

F.3. Sensitivity Analysis of Metric Weights

To evaluate the robustness of our reward weighting scheme, we conduct sensitivity analysis by varying the relative weights of fidelity metrics (L2, PVB) and manufacturability metrics (EPE, Shot), as shown in Fig. 14. We evaluate five configurations: balanced (1:1:1:1), fidelity-emphasized (2:2:1:1 and 4:4:1:1), and manufacturability-emphasized (1:1:2:2 and 1:1:4:4).

The results reveal expected trade-off behaviors: increasing fidelity weights leads to improved L2 and EPE scores but degrades PVB and Shot performance, while emphasizing manufacturability weights yields the opposite effect. This trade-off scales proportionally with the weight magnitude: configurations with 4× weights show more pronounced metric shifts than those with 2× weights. Notably, despite these variations, all configurations exhibit stable optimization trajectories without training collapse or divergence. This demonstrates that LithoGRPO is robust to weight perturbations, allowing practitioners to adjust weights according to application-specific priorities without risking training instability.

F.4. Visualization on ViaSet

We visualize a representative case from the ViaSet test set in comparison with the baselines and our LithoGRPO in Fig. 15. Our method shows competitive results.

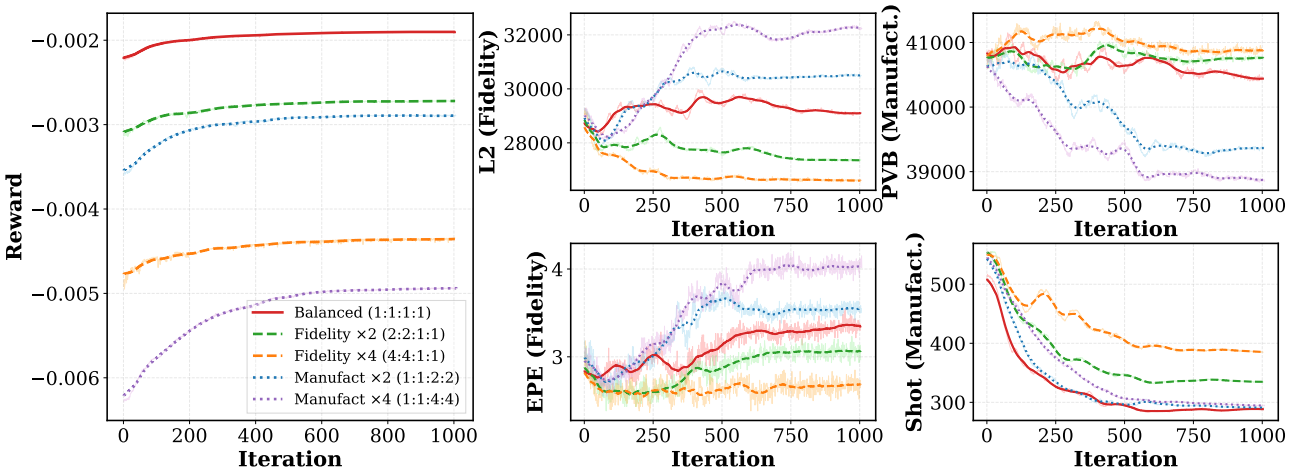

Figure 14. **Sensitivity analysis of reward metric weights across different configurations on MetalSet.** Increasing fidelity weights (L2, EPE) improves fidelity metrics at the cost of manufacturability (PVB, Shot), and vice versa. Despite 4× weight variations, all configurations maintain stable training.

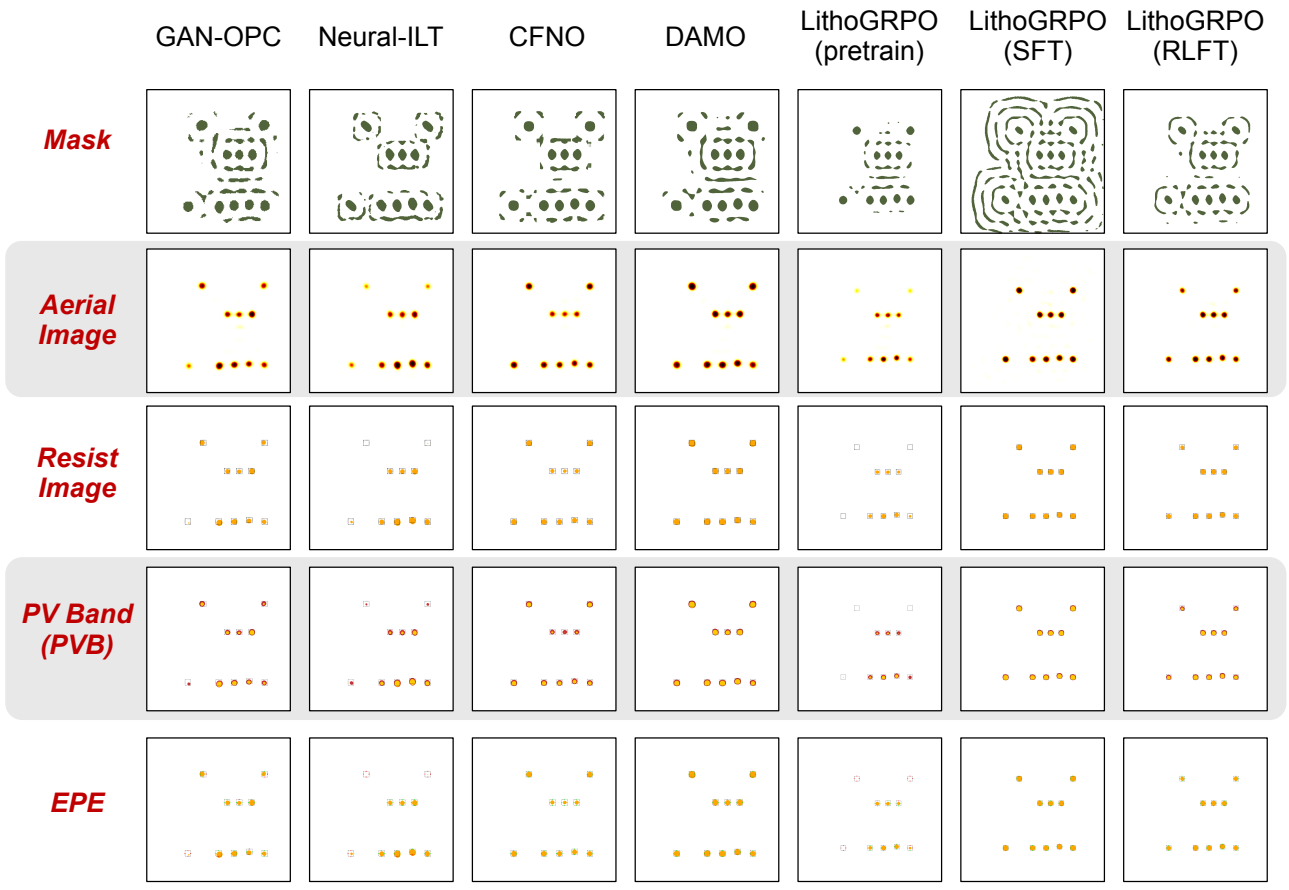

Figure 15. **Visualization of ILT results on the ViaSet case.**