Masked Layout Modeling Advances Hotspot Detection

Binwu Zhu
Southeast University
bwzhu@seu.edu.cn

Zhengzhou Gu
GWX Technology
kevin@gwxeda.com

Jinbin Deng
GWX Technology
dengjb@gwxeda.com

Yuzhe Ma
HKUST(SZ)
yuzhema@hkust-gz.edu.cn

2025 International Symposium of Electronics Design Automation (ISEDA) | 979-8-3315-3696-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISEDA65950.2025.11101144

***Abstract*—With the rapid advancement of semiconductor technology and the continuous miniaturization of circuit feature sizes, hotspot detection has become an increasingly critical yet challenging task in physical verification workflows. In recent years, numerous deep learning frameworks have been developed to address hotspot detection. However, the performance of these learning-based frameworks is heavily dependent on the quality of the datasets used. However, obtaining a large labeled hotspot dataset with high quality is an extremely time-consuming process. Recently, masked image modeling (MIM) has drawn significant attention for its ability to learn from vast amounts of unlabeled data and has demonstrated effectiveness across a wide range of image tasks. Despite its success, the application of MIM to layout analysis, particularly in the context of semiconductor design, remains largely unexplored. Motivated by the principles of MIM, we propose a transfer learning framework that leverages pretraining through masked layout modeling (MLAM) and subsequently fine-tunes the model on limited labeled hotspot detection datasets. Experimental results on our custom layout datasets demonstrate the effectiveness of our approach.**

***Index Terms*—Hotspot detection, EDA, deep learning**

I. Introduction

As semiconductor technology advances rapidly, the components of integrated circuits are shrinking in size. This trend presents a significant challenge for chip manufacturers, as it becomes increasingly difficult to ensure the printability of layout designs with reduced feature sizes. Consequently, a precise and efficient technique for hotspot detection is essential to accurately find out defects in the layouts.

Broadly speaking, hotspot detection methods fall into three categories: lithography simulation, pattern matching, and machine learning. Lithography simulation is a traditional approach that offers high accuracy but suffers from long processing times. In contrast, pattern matching and machine learning methods provide more efficient hotspot identification. Pattern matching techniques [1]–[4] involve using a library of known hotspot patterns to scan new designs and identify matches as potential hotspots. For example, Wen *et al.* [2] propose a density-based layout encoder that can discriminate between the hotspots and non-hotspots via principal components analysis (PCA). While this method is faster, it struggles to detect new, previously unidentified hotspots, which limits its practical utility.

This work is supported by the Foundation of Southeast University 4006012503.(*Corresponding authors: Binwu Zhu, Zhengzhou Gu*)

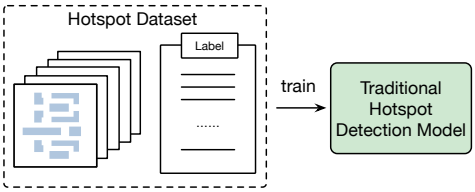

Fig. 1 Traditional training of learning-based hotspot detection model.

Machine learning-based hotspot detection methods [5]–[8], on the other hand, demonstrate strong generalization capabilities and deliver satisfactory results. Particularly, deep learning-based approaches [9]–[14] have shown significant improvements in both accuracy and efficiency. For instance, Yang *et al.* [9] propose to extract layout features with discrete cosine transform and utilize a CNN architecture for hotspot detection. The performance is further improved with the proposed bias learning algorithm because of the imbalanced dataset. Inspired by the object detection problem in computer vision, Chen *et al.* [13] propose a large-scale detector to detect multiple hotspots within large layouts simultaneously. Despite significant advances in learning-based approaches, several critical limitations remain unresolved. A primary challenge stems from the substantial demand for precisely annotated hotspot data to train reliable detection models, as shown in Fig. 1. However, with the shrinking size of technology nodes, manual hotspot annotation becomes an extremely labor-intensive and expertise-dependent process. Current deep learning-based hotspot detection models are usually trained and test on small datasets, e.g., ICCAD 2012 benchmark [15]. Such data scarcity constrains the models' ability to generalize across diverse layout patterns and advanced technology nodes.

As a promising solution to these challenges, self-supervised learning (SSL) paradigms [16] have emerged as powerful alternatives that autonomously derive supervisory signals from intrinsic data structures. Recent breakthroughs demonstrate SSL's dual capability in mitigating data dependency while learning transferable representations for dense prediction tasks. A particularly successful instantiation is masked signal modeling, which operates through a reconstruction mechanism: strategically masking portions of input data and training models to recover the masked content. This approach has

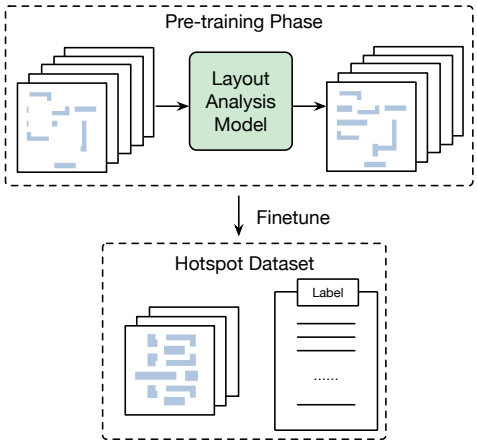

Fig. 2 Our proposed transfer learning-based layout analysis model.

fundamentally transformed natural language processing and computer vision, where architectures like BERT [17] and ViT [18] have established new paradigms for pre-training large-scale models on unannotated corpora followed by task-specific fine-tuning.

Drawing inspiration from the self-supervised learning paradigm, we propose a transfer learning-based hotspot detection framework as shown in Fig. 2. Our approach begins with the creation of a substantial dataset of layout designs, from which we randomly mask a portion of the layout tiles. During the pre-training phase, our designed layout analysis model (LAM) is asked to reconstruct the original patterns from partially masked layouts. Such a pre-training process allows the LAM to utilize a large amount of masked layout data for thorough model pre-training without annotation requirement. Besides, it also enables the model to capture both missing geometric features and spatial dependencies through layout reconstruction tasks. After pre-training, we can fine-tune the pre-trained LAM with limited labeled hotspot dataset while achieving satisfactory performance.

The main contributions of this paper are listed as follows:

- We propose a self-supervised learning scheme, masked layout modeling, which does not require any human-labeled data.
- We design a layout analysis model, which is equipped with strong abilities to capture geometric features and spatial dependencies of layout patterns.
- We create a large layout dataset using layout generation techniques, which solves the data-hungry issue.
- Experimental results show that our model achieves remarkable performance on hotspot detection task.

II. Preliminaries

In this section, we will introduce some preliminary knowledge related to this work.

A. Hotspot Detection

In chip manufacturing, designed layout patterns are transferred onto silicon wafers through the lithographic process. However, this process is subject to numerous variations, which can reduce manufacturing yield by causing potential open-circuit or short-circuit failures. Regions of layout patterns that are particularly sensitive to these lithographic process variations are referred to as hotspot regions. To enhance manufacturing yield, it is essential to develop an efficient and accurate hotspot detection technique to pinpoint defect-prone areas in layouts. An effective hotspot detector should maximize the accurate identification of hotspots. To evaluate the performance, we define the following metrics.

Accuracy.. The ratio between the number of correctly detected hotspots and the number of ground truth hotspots.

False Alarm.. The number of non-hotspots that are predicted as hotspots by the classifier.

With the evaluation metrics defined above, we formulate the hotspot detection problem as follows:

Problem 1 (Hotspot Detection)**.** *Given a collection of clips containing hotspot and non-hotspot layout patterns, the objective of hotspot detection is to train a detector to locate and classify all hotspots and non-hotspots, such that the detection accuracy is maximized and the false alarm is minimized*

B. Masked Image Modeling

Masked image modeling (MIM), as a self-supervised learning paradigm, reconstructs semantic representations from partially corrupted visual inputs. Originating from the masked language modeling (MLM) framework in natural language processing, MIM initially struggled to gain prominence in computer vision. Early explorations like Denoising Autoencoders (DAE) [19] conceptualized image masking as a form of structural noise, while context encoder [20] pioneered region-level reconstruction by predicting pixel values in large masked rectangular regions.

The advent of vision transformers reinvigorated MIM research through NLP-inspired architectures. iGPT [21] introduced pixel clustering for masked content classification, whereas ViT [18] explored patch-level mean color prediction as a pretext task. BEiT [22] advanced this direction by employing discrete variational autoencoders (dVAE) [23] to tokenize pixels into learnable visual codes. A pivotal breakthrough emerged with MAE [24], which demonstrated that reconstructing raw pixels from heavily masked inputs (up to 75%) could yield semantically meaningful representations. MAE's asymmetric encoder-decoder design, featuring a streamlined decoder, significantly improved computational efficiency. This innovation was further refined in SimMIM [25], where decoder complexity was reduced to a single linear layer without performance degradation. Despite these advancements, existing MIM methodologies remain predominantly validated on natural image datasets.

C. Transfer Learning

Transfer learning is a machine learning paradigm that leverages knowledge gained from source domain to improve the performance of a model on a related but distinct target

domain. Unlike traditional methods that train models from scratch, transfer learning aims to mitigate data scarcity or computational inefficiency by reusing pre-trained models or learned features. This approach is particularly valuable when the target domain has limited labeled data or requires rapid deployment. The core idea revolves around identifying transferable patterns, representations, or parameters that generalize across domains. Common techniques include fine-tuning pre-trained neural networks, feature extraction, and domain adaptation. Transfer learning has become foundational in fields like natural language processing and computer vision, where it significantly reduces training costs while maintaining competitive performance. Transfer learning has also been widely used in the EDA area. For example, [26] proposes a transfer learning-based framework for transistor sizing that transfers knowledge in different circuits. In the case of timing prediction, since collecting the timing data requires running a time-consuming tool-chain, Zhang *et al.* [27] propose a transfer learning framework that leverages abundant data from preceding technology nodes to enhance learning on the target technology node.

III. Algorithms

In this section, we provide a detailed explanation of the architecture of our layout analysis model (LAM).

A. Overview

The proposed LAM adopts the typical structure of MIM (Masked Image Modeling) models, incorporating an encoder-decoder architecture. As our framework is based on transfer learning, the pre-training and fine-tuning tasks differ, as illustrated in Fig. 2.

Pre-training Phase.. In the pre-training phase, we employ the proposed masked layout modeling method to pre-train the LAM. The masking strategy (Section III-B) obscures specific portions of the input patches, while the LAM encoder (Section III-C) maps these input patches into a latent space. The LAM decoder (Section III-D), in turn, aims to reconstruct the original layout patterns. This training approach enables the model to effectively learn the spatial features of the layout. The reconstruction target guides the pre-training process through back-propagation, ensuring the model captures meaningful layout representations.

Fine-tuning Phase.. Since the ultimate goal of our framework is hotspot detection, we design an additional decoder specifically to accomplish this task using the encoded representations from the LAM encoder. Leveraging the strong feature-capturing capabilities of the pre-trained LAM encoder, the hotspot detection decoder is designed to be lightweight. This lightweight structure allows the model to be fine-tuned efficiently on a limited labeled hotspot dataset while still achieving satisfactory performance.

B. Masking Strategy

Several random masking methods have been proposed in related previous works: (1) Context encoder [20] introduces a central region masking strategy; (2) BEiT [22] proposes a more complex block-wise masking approach; (3) more recent methods, such as MAE [24] and SimMIM [25], first divide an image into several uniform patches and then adopt a uniformly random masking strategy at the patch level, experimenting with different masked patch sizes and masking ratios. Most random masking schemes are patch-based, as operating on a patch-by-patch basis is more convenient. Each patch is either fully visible or fully masked. Prior works demonstrate that uniformly random sampling with a suitable masking ratio effectively eliminates redundancy, creating a self-supervisory task that cannot be easily solved through extrapolation from visible neighboring patches. In this work, we also adopt the random patch masking approach as shown in Fig. 3 for its simplicity and effectiveness.

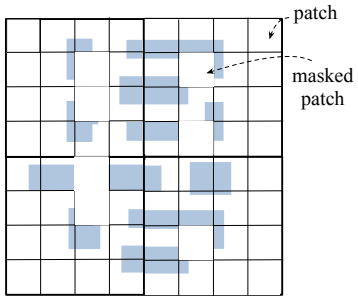

Fig. 3 A layout is divided into uniform patches, which are randomly masked.

C. Encoder

The encoder's primary objective involves learning latent feature representations from masked image patches, which are subsequently decoded to reconstruct original pixel signals in occluded regions. Contemporary architectures predominantly employ Transformer networks [28], which have demonstrated superior performance across vision-language tasks through their global contextual modeling capabilities.

Inspired by the success of Transformer, our LAM encoder is designed based on the vision Transformer (ViT) [18] block, which is illustrated in Fig. 4. It processes visual inputs by first decomposing images of dimension $H \times W \times C$ into $P \times P$ patches, generating a sequence of $N = \frac{HW}{P^2}$ token embeddings. This transformation begins with linear projection layers that map flattened patches into an embedding space $\boldsymbol{X}_E \in \mathbb{R}^{N \times N_H}$, effectively converting spatial information into sequential representations. To preserve positional relationships, spatial encoding is subsequently incorporated through additive positional embeddings $\boldsymbol{E}_P \in \mathbb{R}^{N \times N_H}$, resulting in position-aware inputs $\boldsymbol{X}_P = \boldsymbol{X}_E + \boldsymbol{E}_P$.

The multi-head self-attention mechanism then processes these enhanced embeddings through parallel attention operations. Initially, the input features are projected into query/key/-value subspaces using learnable parameter matrices:

$$\boldsymbol{Q}\ \boldsymbol{K}\ \boldsymbol{V} = \boldsymbol{X}_P[\boldsymbol{W}^Q\ \boldsymbol{W}^K\ \boldsymbol{W}^V] \quad (\boldsymbol{W}^{Q/K/V} \in \mathbb{R}^{N_H \times d_k}) \tag{1}$$

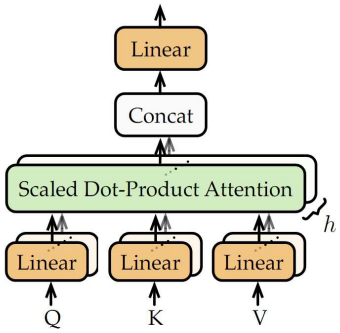

Fig. 4 The vision Transformer block.

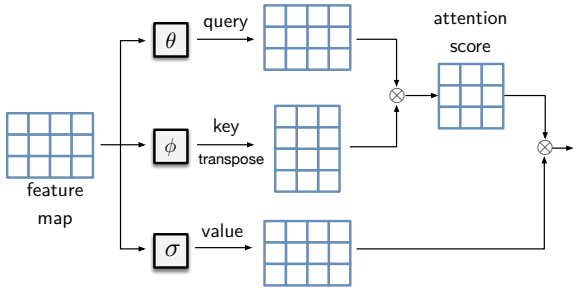

Fig. 5 The self-attention mechanism.

Each attention head independently computes scaled dot-product attention as illustrated in Fig. 5:

$$\boldsymbol{H}_i = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{W}_i^Q(\boldsymbol{K}\boldsymbol{W}_i^K)^\top}{\sqrt{d_k}}\right)\boldsymbol{V}\boldsymbol{W}_i^V \tag{2}$$

where $\boldsymbol{W}_i^Q\ \boldsymbol{W}_i^K\ \boldsymbol{W}_i^V \in \mathbb{R}^{d_k \times d_k/h}$ represent head-specific projection matrices. These parallel computations enable the model to capture diverse interaction patterns across different feature subspaces.

The final multi-head output is obtained by concatenating all attention heads and applying dimensional reduction:

$$\text{MultiHead}(\boldsymbol{Q}\ \boldsymbol{K}\ \boldsymbol{V}) = \text{Concat}(\boldsymbol{H}_1 \qquad \boldsymbol{H}_h)\boldsymbol{W}^O \tag{3}$$

where $\boldsymbol{W}^O \in \mathbb{R}^{d_k \times N_H}$ serves as the output projection matrix. This hierarchical design achieves three critical objectives: diversified feature learning through orthogonal subspace projections, adaptive attention focusing via position-sensitive similarity weighting, and efficient dimension management through controlled parameter growth.

D. Decoder

The decoder resamples latent vectors back into the original image space. As suggested in previous studies [24], [25], a lightweight decoder design is recommended to reduce training time. In our task, we also observe that a lightweight decoder not only decreases computational complexity but also enhances the encoder's ability to learn more generalizable representations. These representations can then be efficiently processed and interpreted by the decoder.

The LAM decoder is designed based on the widely used Feature Pyramid Network (FPN) architecture [29], which is significantly more lightweight compared to the LAM encoder. FPN is a neural network architecture specifically designed to improve multi-scale feature representation by leveraging the hierarchical nature of convolutional neural networks. The shallow layers of FPN capture high-resolution, low-level details, while the deeper layers extract low-resolution, high-level semantic features. Additionally, FPN introduces a top-down pathway to propagate high-level semantic information from deeper layers to shallower ones. This is further enhanced by lateral connections that merge top-down features with corresponding bottom-up layers, creating a more comprehensive multi-scale representation.

The output of the LAM decoder is the reconstructed layout. The pre-training loss is calculated based on the difference between the raw layout and the reconstructed layout, as defined below:

$$\mathcal{L}^{pretrain} = \|\boldsymbol{f}(\overline{\boldsymbol{X}}) - \boldsymbol{X}\|_2^2 \tag{4}$$

where $\overline{\boldsymbol{X}}$ represents the masked layout pattern and $\boldsymbol{X}$ indicates the original layout pattern. $\boldsymbol{f}(\overline{\boldsymbol{X}})$ stands for the output of our model. Therefore, our pre-training process does not require any human-labeled data.

E. Fine-tuning

Algorithm 1 LAM Fine-tuning

Input: Pre-trained LAM encoder $\boldsymbol{P}_{encoder}$;
Output: Fine-tuned hotspot detection decoder $\boldsymbol{P}_{hsd}$
1: Fix $\boldsymbol{P}_{encoder}$;
2: $\boldsymbol{P}_{hsd} \leftarrow$ Initialize HSD decoder parameter;
3: **for** $t = 1$ to $Epoch$ **do**
4: Sample a train set from dataset $\mathbb{D}_{\mathbb{M}}$;
5: Train $\boldsymbol{P}_{hsd}$ to mimimize L_{HSD};
6: **end for**

In the fine-tuning phase, we transfer the pre-trained model to hotspot detections tasks by freezing the parameters of the pre-trained LAM encoder, and then we fine-tune a hotspot detection decoder.

Our decoder employs a customized architecture to determine the presence of hotspot regions within layout patterns. It comprises four sequentially connected convolutional layers that process feature maps from the preceding LAM encoder stage. This configuration enables patch-level hotspot classification according to the spatial partitioning scheme defined in Section III-B.

The decoder generates an $N \times N$ probability matrix where each scalar value $p_{xy} \in [0\ 1]$ quantifies the local likelihood of hotspot occurrence. To optimize the detector's parameters, we implement a binary cross-entropy loss function formulated as:

$$\mathcal{L}^{\text{HSD}}(\boldsymbol{X}) = \sum_{x=1}^{N}\sum_{y=1}^{N}[-p_{xy}\log\hat{p}_{xy} - (1-pxy)\log(1-\hat{p}_{xy})] \tag{5}$$

where $p_{xy} \in 0\ 1$ denotes the ground truth label for position $(x\ y)$, and $\hat{p}_{xy}$ represents the predicted probability of hotspot presence at that location.

IV. Experiments

A. Experimental Settings

In existing hotspot detection research, the ICCAD-2012 benchmark [15] has been extensively adopted as a standard layout dataset. However, being primarily designed for academic research, this dataset contains a limited number of layout patterns. Machine learning models trained on such restricted data typically exhibit limited generalization capacity when deployed in practical industrial applications, often resulting in suboptimal performance.

To address this data scarcity, GAN-OPC [30] introduced a layout synthesis methodology capable of generating approximately 4,000 artificial metal-layer tiles. While representing a significant improvement, this scale remains insufficient for the training requirements of our framework. To enhance model generalization, we require substantially more diverse layout patterns for effective training of our Lithography-Aware Model (LAM).

Following the methodology outlined in [30], we consequently developed GenLay - a new synthetic layout dataset comprising 12,000 distinct patterns. Fabricated for the 32nm technology node, GenLay ensures strict compliance with 32nm design rules through its generation algorithm while maintaining pattern diversity.

The dataset is partitioned into three subsets: 70% for pre-training the LAM framework, 20% for fine-tuning the hotspot detection module, and 10% reserved for performance evaluation. For hotspot annotation, we employ advanced lithography simulation tools [31] to identify sensitive lithographic regions. This process involves: (1) Generating optimized photomasks through multi-objective optimization, (2) Performing edge placement error (EPE) analysis across critical sample points, and (3) Designating regions with EPE violations exceeding process specifications as confirmed hotspots.

B. Pre-training and Fine-tuning Results

The GenLay dataset we develop enables the LAM to learn from diverse layout patterns and acquire the knowledge necessary for precise downstream layout tasks. While generating this training data may require extra effort, it is crucial for training a robust and effective model. As depicted in Fig. 6, during the training phase, LAM optimizes its parameters through an iterative process, progressively enhancing its ability to accurately interpret layouts. This phase is also vital for fine-tuning the model, enabling it to generalize well to new, unseen data, with the performance changes illustrated in Fig. 7.

C. Hotspot Detection Results Comparison

TABLE I presents the comparative evaluation results of our proposed method against leading hotspot detection frameworks. The experimental data highlight LAM's exceptional detection capabilities, achieving a mean accuracy of 92.10%,

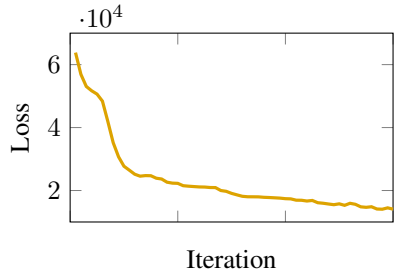

Fig. 6 Loss curve of pre-training phase.

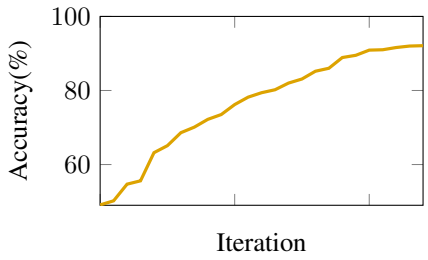

Fig. 7 Accuracy curve of hotspot detection fine-tuning phase.

which surpasses ICCAD'21 (88.91%) [32], DAC'19 (86.76%) [13], and TCAD'19 (83.63%) [33] by substantial margins.

Our framework significantly excels in reducing false alarms compared to existing methods. The system achieves false positive rates that are 53.8%, 33.6%, and 11.4% lower than those reported for TCAD'19 [33], DAC'19 [13], and ICCAD'21 [32], respectively, demonstrating enhanced operational reliability.

Regarding computational efficiency, LAM completes hotspot detection for each layout in an average time of 0.26 seconds. Although slightly slower than ICCAD'21, which records a time of 0.12 seconds [32], our solution remains highly competitive. It supports a multi-task architecture that allows concurrent hotspot detection and mask optimization. Despite the complexity of this design compared to ICCAD'21's single-task approach, the modest time difference of 0.14 seconds represents a reasonable compromise, especially considering LAM's substantial improvements in detection accuracy and false alarm reduction.

TABLE I
Comparison with other SOTA methods.

	Accuracy (%)	**False Alarm**	**Runtime (s)**
TCAD'19	83.63	7094	0.40
DAC'19	86.76	4931	0.28
ICCAD'21	88.91	3692	**0.12**
Ours	**92.10**	**3272**	0.26

D. Ablation Study

We conduct ablation studies to evaluate the impact of different masking ratios during the pre-training stage of our model. Identifying an optimal mask ratio is crucial as it enhances LAM's ability to learn and predict the masked parts effectively. We assessed the effects of 30%, 40%, 50%, 60%, and 70%

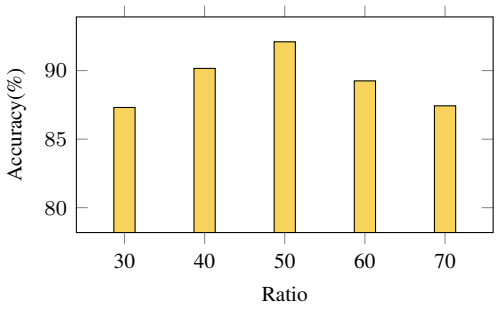

Fig. 8 Ablation study results of different masking ratios.

masking ratios on the performance of hotspot detection tasks. The experimental results, depicted in Fig. 8, reveal that a 50% masking ratio yields the best performance.

We observed that a lower masking ratio, although resulting in high accuracy during the layout reconstruction in the pre-training phase, leads to insufficient generalization capabilities. On the other hand, an excessively high masking ratio compromises the model's accuracy in reconstructing layouts, thereby hindering its ability to accurately interpret layout geometry information.

V. Conclusion

In this work, we introduce a transfer learning framework based on masked layout modeling (MLAM) for hotspot detection in semiconductor layouts. By leveraging MLAM, we successfully pre-trained our LAM to learn crucial geometric features and spatial dependencies without extensive manual annotation. Experimental results show that our model outperforms existing hotspot detection methods. Additionally, the adaptability and scalability of our approach suggest potential applications beyond hotspot detection in semiconductor manufacturing. Overall, this research advances the use of machine learning in EDA and lays the groundwork for further exploration of self-supervised learning paradigms in specialized fields.

References

[1] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *Proc. DAC*, 2013.

[2] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE TCAD*, vol. 33, no. 11, pp. 1671–1680, 2014.

[3] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," in *Proc. DAC*, 2012.

[4] S. S.-E. Tseng, W.-C. Chang, I. H.-R. Jiang, J. Zhu, and J. P. Shiely, "Efficient search of layout hotspot patterns for matching SEM images using multilevel pixelation," in *Proc. SPIE*, vol. 10961, 2019.

[5] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. DAC*, 2013.

[6] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng, "Sparse VLSI layout feature extraction: A dictionary learning approach," in *Proc. ISVLSI*, 2018.

[7] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan, "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *JM3*, vol. 14, no. 1, p. 011003, 2015.

[8] H. Zhang, B. Yu, and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *Proc. ICCAD*, 2016.

[9] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proc. DAC*, 2017.

[10] H. Yang, S. Li, C. Tabery, B. Lin, and B. Yu, "Bridging the gap between layout pattern sampling and hotspot detection via batch active learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, pp. 1–1, 08 2020.

[11] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," in *Proc. ICCAD*, 2020.

[12] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. ASPDAC*, 2019.

[13] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *Proc. DAC*, 2019.

[14] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network," in *Proc. DAC*, 2019.

[15] A. J. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, 2012.

[16] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," 2021. [Online]. Available: https://arxiv.org/abs/2011.00362

[17] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*.

[19] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[20] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CVPR*, 2016.

[21] M. Chen, A. Radford, J. Wu, H. Jun, P. Dhariwal, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *ICML*, 2020.

[22] H. Bao, L. Dong, and F. Wei, "Beit: Bert pre-training of image transformers," *ICLR*, 2022.

[23] A. Vahdat, E. Andriyash, and W. G. Macready, "Dvae: Discrete variational autoencoders with relaxed boltzmann priors," 2018. [Online]. Available: https://arxiv.org/abs/1805.07445

[24] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022.

[25] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, "Simmim: A simple framework for masked image modeling," in *CVPR*, 2022.

[26] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. DAC*, 2020.

[27] X. Zhang, B. Zhu, F. Liu, Z. Wang, P. Xu, H. Xu, and B. Yu, "Disentangle, align and generalize: Learning a timing predictor from different technology nodes," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[29] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[30] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Y. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE TCAD*, 2020.

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

[32] B. Zhu, R. Chen, X. Zhang, F. Yang, X. Zeng, B. Yu, and M. D. Wong, "Hotspot detection via multi-task learning and transformer encoder," in *Proc. ICCAD*, 2021.

[33] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, vol. 38, no. 6, pp. 1175–1187, 2019.