

Lithography Hotspot Detection for Complex Non-Manhattan Layouts via Graph Neural Network

Bohao Li^{1, †}, Ranran Liu^{1, †}, Yumeng Liu¹, Cong Jiang², Kang Liu², Bei Yu³, Kun Ren¹, Qi Sun^{1, *}, Cheng Zhuo¹

¹Zhejiang University, Hangzhou, China

²Huazhong University of Science and Technology, Wuhan, China

³The Chinese University of Hong Kong, Hong Kong, China

[†]Equal Contributions; *Corresponding Author: qisunchn@zju.edu.cn

Abstract—Convolutional neural networks (CNNs) have been widely applied in lithography hotspot detection due to their strong feature extraction capability; however, low computational efficiency remains a critical bottleneck. Recently, graph neural networks (GNNs) have emerged as a promising alternative, offering both high inference speed and strong scalability to variable-sized inputs. Nevertheless, existing approaches model layouts by decomposing polygons into rectangles, which introduces redundant boundaries and struggles to handle complex non-Manhattan layouts. In this paper, we propose a novel graph representation that accurately extracts the critical geometric features of non-Manhattan layouts by modeling polygon contours. To capture the long-range interactions induced by optical effects, we introduce a hierarchical message-passing mechanism to encode both local and global layout structures efficiently. Furthermore, building on the graph representation, the clip-level labels of non-hotspots can be transformed into edge-level supervision. Accordingly, we incorporate multiple instance learning (MIL) to leverage the fine-grained supervision from non-hotspot clips, thereby enhancing the ability to distinguish between hotspot and non-hotspot clips. Experiments on industrial non-Manhattan datasets demonstrate that our method yields a 3.6% higher recall, 10.8% fewer false alarms, and a 1.7% increase in F1 score compared with the state-of-the-art (SOTA) methods. The industrial non-Manhattan layout used in this work is available at <https://github.com/yb-hitsz/DATE2026-GNN4LSD>.

I. INTRODUCTION

Lithography hotspots refer to layout regions that may lead to manufacturing defects such as short or open circuits on the wafer. In advanced technology nodes, with the increase in pattern density and the exacerbation of optical proximity effects, the number of lithography hotspots has significantly increased. Therefore, early hotspot detection is critical for lower manufacturing risks and faster time-to-market. Traditional hotspot detection methods rely on lithography simulation, which offers high accuracy but suffers from high computational cost. Alternative approaches based on pattern matching and machine learning have been proposed for efficient hotspot detection. The pattern matching method compares extracted layout clips with a pre-established hotspot pattern library. Constrained by the completeness of the library, pattern matching methods are unable to identify unseen hotspot patterns. In contrast, machine learning-based methods have attracted widespread attention for their strong generalization capabilities [1].

Most machine learning approaches address hotspot detection

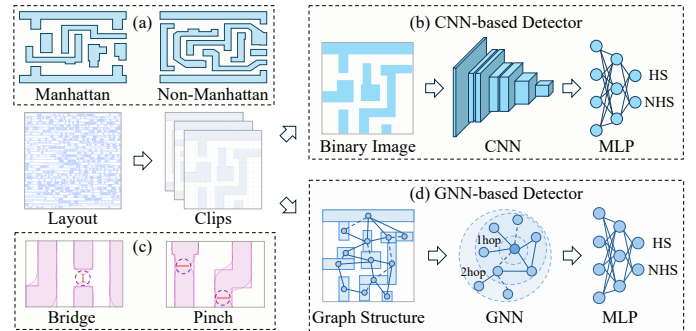


Fig. 1 Illustration of: (a) Manhattan and non-Manhattan layout clips; (b) a CNN-based hotspot detector; (c) two common hotspot types, bridge and pinch; (d) a GNN-based hotspot detector.

from a computer vision perspective, with a workflow comprising two stages, as shown in Fig. 1(b): first, converting GDS layouts into binary images; second, employing neural networks for feature extraction and classification. Specifically, advanced architectures (*e.g.*, CNNs [2], binarized neural networks [3], and Transformer-based models [4]) have been widely adopted to capture layout features. These models have demonstrated exceptional performance, achieving detection accuracies exceeding 98% on the ICCAD-2012 benchmark [5]. However, in image representation, the redundant background information and the feature loss caused by image rasterization collectively limit the potential for further accuracy improvement. Additionally, it also suffers from high computational overhead for feature extraction.

To improve efficiency and modeling accuracy, a graph learning-based detection paradigm was first introduced in [6]. By partitioning polygons into rectangle primitives and modeling them as graph nodes, this method achieves an accurate representation of layout topology, as shown in Fig. 1(d). Subsequently, a GNN model customized for this graph representation was developed, effectively leveraging the geometric information for accurate feature embedding. Compared to CNN-based approaches, this method achieves higher accuracy and an impressive inference speedup of over $10 \times$. Further, a transitive closure graph representation was proposed in [7]. By uniformly modeling both layout and space regions, this method captures

geometric information more consistently, thereby achieving better accuracy.

While existing graph-based methods have demonstrated significant potential, they still struggle to handle non-Manhattan layouts. As shown in Fig. 1(a), non-Manhattan layouts refer to designs composed of geometric shapes formed by non-horizontal or vertical line segments. Such layouts have received limited attention in prior studies. Directly extending the partition-based modeling methods to non-Manhattan layouts inevitably introduces numerous irregular geometries (*e.g.*, triangles, parallelograms), which increase the complexity of graph topology and feature design. Moreover, such partitioning breaks the geometric integrity of polygon shapes, making it difficult to model interactions among polygons accurately.

In this work, to achieve accurate representation and efficient processing of non-Manhattan layouts, we propose a partition-free contour-based layout modeling method. Specifically, we abstract polygon edges as graph nodes and introduce three types of graph edges to capture dimensional, contour, and spacing information, thereby accurately preserving the intricate geometric structures in non-Manhattan layouts. Considering that lithography induces polygon interactions over a broad range, we further design a hierarchical message-passing strategy to capture both local and global topological features. Furthermore, based on the proposed graph representation, the clip-level labels of non-hotspots can be refined into strong edge-level supervision signals. Therefore, we integrate a multiple instance learning (MIL) framework to fully exploit edge label information, thereby enhancing the model’s effectiveness in learning layout patterns.

The main contributions of this paper are as follows:

- We propose a novel contour-based representation method for non-Manhattan layouts, which can robustly handle arbitrary geometric shapes while preserving essential topological information through a concise feature design.
- We develop a GNN model that incorporates hierarchical message passing and multiple instance learning, enabling accurate capture of layout topology and hotspot patterns.
- Experimental results demonstrate that our approach outperforms the SOTA method on industrial non-Manhattan datasets, achieving an average improvement of 3.6% in recall, a reduction of 10.8% in false alarms, and an enhancement of 37.8% in inference speed.

The remainder of this paper is organized as follows. Section II introduces the background and problem formulation. Section III provides detailed explanations of our method. Section IV presents the experimental results, followed by a conclusion in Section V.

II. PRELIMINARIES

A. Problem Formulation

To quantitatively evaluate the performance of hotspot detection models, we adopt the following metrics.

Definition 1 (Recall). Recall is the ratio of correctly detected hotspot samples to the total number of hotspot samples. A high recall rate indicates the model’s effectiveness in detecting hotspot samples.

Definition 2 (False Alarm). False alarm (FA) refers to the total number of non-hotspot samples that are incorrectly predicted as hotspots. A low false alarm rate reflects the model’s strong ability to predict non-hotspot samples.

Definition 3 (F1 Score). F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model’s classification performance across both classes.

Based on the above evaluation metrics, we can formally define the GNN-based hotspot detection problem as follows.

Problem 1 (GNN-based Hotspot Detection). Given a set of layout clips, the objective is to exploit their graph-structured representations to train a GNN model that performs binary classification for each clip, indicating whether it contains hotspots, while achieving high recall and low false alarms.

B. Graph Neural Network

GNNs have emerged as an effective approach for representation learning on graph data [8], [9]. Through iterative message passing, GNNs aggregate neighborhood information to update node representations, thereby encoding graph features. A common form of the message-passing process is formulated as:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \mathbf{X}^{(l)}), \quad (1)$$

where $\mathbf{X}^{(l)}$ is the node embedding matrix at the l -th layer, \mathbf{A} is the adjacency matrix encoding the graph connectivity, and \mathbf{D} is the diagonal degree matrix with entries $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The core operation of Equation (1) is the message aggregation process $\mathbf{A} \mathbf{X}^{(l)}$, which propagates information across local neighborhoods. The degree-based normalization $\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$ is introduced to balance the contributions from neighboring nodes and prevent high-degree nodes from dominating the aggregation.

In graph-based hotspot detection, edge features often encode critical geometric information. Therefore, GNN models that support joint updating of both node and edge embeddings are essential. However, in classic GNNs, edge features are utilized indirectly, as they are transformed into weights for node feature aggregation and remain fixed during graph convolution operations. Recent work proposed an edge-featured GNN that updates the node and edge features simultaneously [6]. The neighborhood aggregation for node v_i is presented in Equation (2), where the neighbor node feature $\mathbf{h}_{v_j}^{(l)}$ is concatenated with the corresponding edge feature $\mathbf{h}_{e_{i,j}}^{(l)}$ and aggregated to obtain the neighborhood representation $\mathbf{g}_{v_i}^{(l+1)}$. The edge embedding update for $e_{i,j}$ is described in Equation (3): the endpoint neighborhood representations $\mathbf{g}_{v_i}^{(l+1)}$ and $\mathbf{g}_{v_j}^{(l+1)}$ are concatenated and fused with the original edge feature $\mathbf{h}_{e_{i,j}}^{(l)}$ to generate the updated edge feature $\mathbf{h}_{e_{i,j}}^{(l+1)}$.

$$\mathbf{g}_{v_i}^{(l+1)} = \sum_{v_j \in \mathcal{N}_{v_i}} \text{MLP} \left(\text{CAT} \left(\mathbf{h}_{v_j}^{(l)}, \mathbf{h}_{e_{i,j}}^{(l)} \right) \right), \quad (2)$$

$$\mathbf{h}_{e_{i,j}}^{(l+1)} = \text{MLP}_v \left(\text{CAT} \left(\mathbf{g}_{v_i}^{(l+1)}, \mathbf{g}_{v_j}^{(l+1)} \right) \right) + \text{MLP}_e \left(\mathbf{h}_{e_{i,j}}^{(l)} \right), \quad (3)$$

where \mathcal{N} , \mathbf{h} , and $\text{CAT}(\cdot, \cdot)$ denote the set of neighbor nodes, the node or edge feature vector, and the concatenation operation, respectively.

III. HOTSPOT DETECTION OF NON-MANHATTAN LAYOUT

In this section, we present our proposed GNN-based hotspot detection method. We first describe how non-Manhattan layouts are modeled as graphs to capture shape features and positional relationships. Subsequently, we detail the GNN architecture and training strategy designed to achieve high recall, few false alarms, and good scalability.

A. Graph Representation

Prior graph-based hotspot detection methods partition polygons into rectangles to process Manhattan layouts, which struggle to represent intricate non-Manhattan structure and their complex spatial relationships. To address this limitation, we propose a fine-grained graph modeling method that captures geometry by explicitly describing polygon contours. For clarity, we term the edges of layout polygons “polygon edges” and the edges in the graph structure “graph edges”.

Specifically, each polygon edge is modeled as a node in the graph. For both Manhattan and non-Manhattan layouts, polygon edges differ only in orientation; thus, the proposed method is capable of uniformly handling patterns of any structural complexity. To establish node connectivity, we first define their neighboring regions. Each polygon edge is extended by distances w and h along the radial and normal directions, thereby defining a bounding box, as illustrated in Fig. 2(b). w and h are process-dependent parameters that characterize the spatial impact range of interference and diffraction effects in the lithography process. Any polygon edge that intersects or is contained within this region is considered adjacent to the central polygon edge, indicating a potential connection. Notably, the central polygon edge is not connected to all polygon edges within the rectangle. To reduce redundant graph edges, we impose a set of constraints. For instance, connections between perpendicular polygon edges are prohibited, and in each direction, only the nearest polygon edges are connected. These constraints ensure that only critical structures are preserved, thereby improving the efficiency of graph learning.

The resulting graph edges are classified into three types based on the differences in the geometric spatial relationships they reflect, as shown in Fig. 2(d)-(f):

- NeighborEdge connects intersecting polygon edges, capturing the local contour shapes of the polygon;
- LayoutEdge connects polygon edges with a layout region in between, characterizing the dimensional features within the same polygon;
- SpaceEdge connects two polygon edges with a space region in between, describing the relative positional relationships between different polygons.

For each node, we assign three features to characterize the geometric properties of the corresponding polygon edge, including its length and its orientation angle relative to the horizontal axis. In addition, the number of polygon edges intersecting the bounding box of the given polygon edge (as shown in Fig. 2(b)) is included as an extra feature to quantify the complexity of the neighborhood geometry.

To represent the relative positional relationships between polygon edges, we employ a three-dimensional feature vector

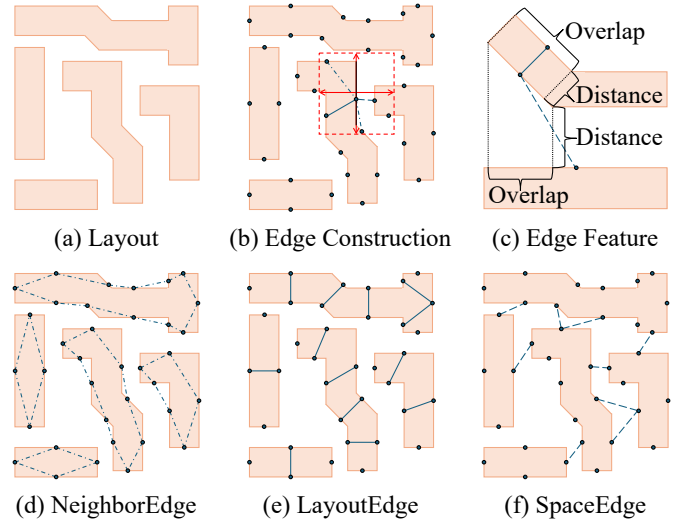


Fig. 2 Illustration of our graph representation method.

for each SpaceEdge and LayoutEdge, as shown in Fig. 2(c). First, and most importantly, the distance between two polygon edges is used to represent their relative position in the normal direction. Specially, for non-parallel polygon edges, this is given by the minimum distance between any two points on the two edges. Second, we extract the overlapping length between two polygon edges, defined as the projection length of one edge onto the other, to describe their positional relationship in the radial direction. Finally, the angular difference between the two polygon edges is used as the third edge feature. For NeighborEdge, the edge features are initialized as a zero vector, since it is solely designed to represent the connectivity between adjacent polygon edges to capture the polygon contour. With all edge and node features defined, the layout graph is complete. All length-related features are normalized using max normalization, while angle-related features are normalized using sin-cos transformation (*i.e.*, $(\sin(\theta), \cos(\theta))$) to improve training stability.

In summary, the proposed layout modeling method captures critical dimension and contour information through a concise feature design based solely on length and angle, effectively representing the key geometric characteristics of non-Manhattan layouts and producing compact, learnable representations for subsequent GNN models.

B. GNN Architecture

In the proposed graph representation, the key geometric information is primarily encoded in the edge features; hence, the GNN must fully exploit these features to capture hotspot patterns. In existing GNNs, edge features are either treated merely as weights for node feature aggregation or updated through complex feature fusion mechanisms. In contrast, we regard nodes and edges as graph instances of equal importance and apply a unified feature update strategy, thereby enabling effective joint learning of node and edge representations. Our message passing process follows a three-stage procedure.

First, for the target node v_i , the features of its connected edges are aggregated separately according to their edge types.

In our graph construction, the set of edge types \mathcal{R} contains three types, r_S, r_L, r_N , corresponding to SpaceEdge, LayoutEdge, and NeighborEdge. These edge types result in three aggregated vectors, as shown in Equation (4):

$$\mathbf{m}_{v_i}^{(l),r} = \sum_{j \in \mathcal{N}_{v_i}^r} \mathbf{W}_r^{(l)} \mathbf{h}_{e_{i,j}}^{(l)}, \quad (4)$$

where \mathcal{N}_{v_i} represents the set of neighbor nodes of v_i . $\mathbf{h}_{v_i}^{(l)}$, $\mathbf{h}_{e_{i,j}}^{(l)}$ denote the node and edge embeddings on the l -th layer. \mathbf{m}_{v_i} represents the feature vector aggregated from the neighborhood of v_i . \mathbf{W} denotes the learnable weight matrix.

Next, the aggregated features of the three edge types are concatenated with the features of node v_i and mapped to the updated node representation via an MLP parameterized by \mathbf{W}_u , as shown in Equations (5) and (6):

$$\mathbf{m}_{v_i}^{(l)} = \text{CAT}(\{\mathbf{m}_{v_i}^{(l),r} | r \in \mathcal{R}\}), \quad (5)$$

$$\mathbf{h}_{v_i}^{(l+1)} = \sigma(\mathbf{W}_u \cdot \text{CAT}(\mathbf{m}_{v_i}^{(l)}, \mathbf{h}_{v_i}^{(l)})). \quad (6)$$

Finally, for edge $e_{i,j}$, the edge features and the features of its two connected nodes are concatenated and transformed into the updated edge embedding via another MLP parameterized by \mathbf{W}_v , as shown in Equation (7):

$$\mathbf{h}_{e_{i,j}}^{(l+1)} = \sigma(\mathbf{W}_v \cdot \text{CAT}(\mathbf{h}_{v_i}^{(l+1)}, \mathbf{h}_{v_j}^{(l+1)}, \mathbf{h}_{e_{i,j}}^{(l)})). \quad (7)$$

Through the process above, edge features can effectively encode the local geometric structures. However, owing to the fine-grained nature of our graph representation, the spatial scale of the receptive field in a single convolution operation is quite limited. This becomes problematic in lithography, where optical proximity effects give rise to long-range interactions among spatially distant patterns, making it necessary for the GNN to capture long-range dependencies. Considering that the number of nodes within the receptive field grows rapidly as the neighborhood radius increases, simply stacking more GNN layers is computationally inefficient.

Inspired by the inherent hierarchical structure of layouts (*i.e.*, polygon edges are contained within polygons, which in turn form a layout), we adopt a hierarchical feature aggregation scheme to expand the receptive field while enhancing computational efficiency. Specifically, the aggregation is performed in a bottom-up order across two levels, followed by a feature fusion step. First, at the node level, we apply self-attention pooling to aggregate the node features within each polygon, thereby capturing the polygon shape and producing a polygon-level embedding. Second, at the polygon level, each polygon is abstracted as a hypernode, and a high-level polygon graph is constructed based on the inter-polygon connections defined by SpaceEdges. On this polygon graph, we perform message passing using the aforementioned graph convolution operation to obtain the polygon embeddings that model the long-range interactions. Finally, the polygon embedding is concatenated with the embeddings of all NeighborEdges and LayoutEdges within the polygon, thereby explicitly injecting long-range geometric information for more expressive representations.

C. Multiple Instance Learning

While the proposed message-passing process learns edge-level representations that accurately encode the layout geometry, capturing hotspot patterns embedded within the layout topology remains challenging, as hotspot detection is inherently a weakly supervised task with inexact labels. Previous graph-based methods typically apply global max pooling to generate a graph-level representation, which is then fed into an MLP for binary classification. Since max pooling only retains salient features while losing critical structural details, the resulting embedding sacrifices interpretability.

In the proposed graph representation, SpaceEdge and LayoutEdge are designed to extract the geometric features from regions that may generate bridge and pinch hotspots. Consequently, for non-hotspot clips, where no bridge or pinch hotspots exist, all SpaceEdges and LayoutEdges can be labeled as non-hotspot. Therefore, the supervision is refined from graph-level to edge-level. For hotspot clips, however, supervision remains at the graph-level. Notably, due to the inherent class imbalance issue in hotspot detection, the abundance of precisely labeled non-hotspot clips provides strong edge-level supervision, which facilitates the learning of discriminative features between hotspot and non-hotspot patterns, thereby improving generalization and interpretability.

Based on these analyses, we introduce the following MIL strategy. First, the edge embeddings of SpaceEdge and LayoutEdge are multiplied by a learnable vector to obtain anomaly scores for each edge in the clip. Then, max pooling is applied to select the highest score, representing the anomaly score of the entire clip. We adopt an MIL ranking loss to enforce that the anomaly score of a hotspot clip should be higher than the score of a non-hotspot clip by at least a margin m . Let s_h and s_n denote the anomaly score of the hotspot and non-hotspot clip, respectively. The ranking loss is formulated as:

$$\mathcal{L}_{rank} = \frac{1}{|\mathcal{H}||\mathcal{N}|} \sum_{h \in \mathcal{H}} \sum_{n \in \mathcal{N}} \max(0, m - s_h + s_n), \quad (8)$$

where h and n denote hotspot and non-hotspot clip. \mathcal{H} and \mathcal{N} represent the sets of hotspot and non-hotspot clips, respectively. While the ranking loss offers edge-level supervision, we further incorporate a conventional classifier with focal loss to leverage graph-level labels, thereby improving training stability and mitigating overfitting. The total loss function is defined as a weighted combination of the two losses.

IV. EXPERIMENTS

To thoroughly investigate the effectiveness of our method, we evaluate its performance on open-source and industrial benchmarks and compare its performance with several representative hotspot detection methods.

A. Benchmarks and Baselines

We use industrial metal layer layouts from full-chip non-Manhattan designs as the original data, and perform lithography simulation and lithography rule checking using Calibre to obtain accurate labels. The layouts are partitioned into $4 \mu\text{m} \times 4 \mu\text{m}$ clips to construct the industrial datasets. Clips that

TABLE I Benchmark Statistics.

Benchmark	Training Set		Testing Set	
	#Non-hotspot	#Hotspot	#Non-hotspot	#Hotspot
ICCAD 2019-1	17758	467	14621	1001
ICCAD 2019-2	17758	467	65523	64310
Industry-1	4000	2000	20000	9820
Industry-2	5000	2000	10000	4423
Industry-3	10000	2000	20000	9231
Industry-4	10000	1500	20000	4459

contain either pinch or bridge hotspots are labeled as hotspot clips. The detailed statistics for these datasets are summarized in TABLE I, where #NHS and #HS denote the number of non-hotspot and hotspot clips. In addition to the industrial datasets, to further evaluate the performance, we also adopt ICCAD 2019 [10], a challenging Manhattan layout benchmark.

We compare our methods with SOTA methods and other representative baselines, including both CNN-based and GNN-based approaches:

- TCAD’19 [11], TCAD’21 [3], TCAD’22 [12], DATE’23 [13]: CNN-based hotspot detectors with various performance enhancement techniques, *e.g.*, binarization, attention mechanism, and neural architecture search.
- TCAD’20 [14]: hotspot detection via region-based CNN.
- DATE’22 [6], TCAD’25 [7]: GNN-based hotspot detector tailored for Manhattan layout.
- TCAD’24 [15], TODAES’25 [16]: CNN-based interpretable detector with hotspot marker supervision.

We evaluate the TCAD’19, TCAD’20, DATE’22, TCAD’24, and TODAES’25 baselines using the official implementations, obtained from public releases or from the authors, whereas the performance results for TCAD’21, TCAD’22, DATE’23, and TCAD’25 in TABLE II were cited from [7].

B. Experiments Settings

We implement our method using the Pytorch Geometric library [17], and the experiments are performed on a Linux server with Intel Xeon 8375C CPUs and NVIDIA RTX A6000 GPUs.

In the graph construction process, the bounding box of each polygon edge is obtained by extending the edge along the radial and normal directions by 0.05 μm and 0.10 μm for the ICCAD 2019 benchmark, and 0.10 μm and 0.20 μm for the industrial datasets.

For model training, we employ a GNN model with two graph convolutional layers and a hidden size of 32. The initial learning rate is set to 0.001 and decays by a factor of 0.5 every 50 epochs. ReLU activation, a dropout rate of 0.2, and the AdamW optimizer are used in all experiments. The “Time” in all tables refers to the model inference time. In TABLE II, the batch size for our method is set to 64, the same as in DATE’22, whereas in TABLE III, all methods use a batch size of 1 for fair comparison. To mitigate the class imbalance problem in the training data of the ICCAD 2019 benchmark, the hotspot clips are repeated four times.

C. Hotspot Detection Performance

We first evaluate the proposed method using the ICCAD 2019 benchmark, with results shown in TABLE II. The ICCAD

TABLE II Performance Comparison of Hotspot Detection Methods on the ICCAD 2019 Benchmark. Recall: %, F1: %, Time: Seconds.

Benchmark	Baseline	Recall	FA	F1	Time
ICCAD 2019-1	TCAD’21 [3]	80.9	365	74.4	3.8
	DATE’22 [6]	86.4	1470	51.9	2.3
	TCAD’22 [12]	69.1	443	64.8	9.3
	DATE’23 [13]	91.6	1257	57.8	8.3
	TCAD’25 [7]	91.3	1222	58.3	2.6
	Ours	93.2	1171	60.1	3.7
ICCAD 2019-2	TCAD’21 [3]	89.8	54974	65.2	31.4
	DATE’22 [6]	85.8	52831	64.0	31.6
	TCAD’22 [12]	95.2	54391	68.1	112.4
	DATE’23 [13]	90.5	54973	65.6	44.9
	TCAD’25 [7]	94.6	53579	68.1	29.8
	Ours	95.3	52680	68.8	46.7

2019-1 testing set contains hotspot samples that are significantly different from those in the training set, which aims to evaluate the model’s generalization capability to identify unseen hotspot patterns. Our method achieves higher recall and lower FA than other graph-based methods (*i.e.*, DATE’22 and TCAD’25). However, due to the introduction of hierarchical message passing to enhance the limited receptive field, our model exhibits slower inference speed than these methods. Compared to the image-based methods, although our method exhibits higher FA than TCAD’21 and TCAD’22, it achieves the highest recall, indicating stronger generalization capability in detecting unseen hotspots. The ICCAD 2019-2 testing set contains numerous highly similar hotspot/non-hotspot pairs, aiming to assess the model’s ability to distinguish hard-to-classify (HTC) patterns. On the ICCAD 2019-2 benchmark, our method outperforms the baselines in both recall and FA, highlighting its superior capability in classifying HTC patterns.

We further evaluate our method on industrial non-Manhattan datasets, as shown in TABLE III. Since existing graph-based methods cannot handle non-Manhattan layouts, we only compare our approach against image-based methods. For each industrial dataset, our method consistently surpasses or matches the SOTA performance. On average over the four datasets, our method achieves a recall of 95.0%, FA of 375.8, F1-score of 95.3%, and inference time of 458.0 seconds, surpassing all baselines in each metric. Notably, both TCAD’24 and TODAES’25 employ hotspot markers as strong supervision signals during training, whereas our method only relies on clip-level labels, yet still achieves superior detection performance.

D. Ablation Study

We conduct ablation studies on the Industry-1 dataset to verify the contributions of the key components in our method, with the results summarized in Fig. 3. Here, “Full” denotes the complete method; “SE”, “LE”, and “NE” represent the removal of SpaceEdge, LayoutEdge, and NeighborEdge from the proposed graph representation, respectively; “HMP” denotes the removal of hierarchical message passing, while “MIL” means replacing the MIL-based classifier with a conventional one, in

TABLE III Performance Comparison of Hotspot Detection Methods on Industrial Non-Manhattan Datasets. Recall: %, F1: %, Time: Seconds.

Benchmark	Baseline	Recall	FA	F1	Time
Industry-1	TCAD'19 [11]	91.4	4845	75.9	5670.4
	TCAD'20 [14]	91.2	919	90.9	2324.1
	TCAD'24 [15]	92.5	1936	87.2	2620.2
	TODAES'25 [16]	93.7	512	94.2	865.6
	Ours	94.2	514	94.5	517.5
Industry-2	TCAD'19 [11]	96.2	1151	86.6	1622.3
	TCAD'20 [14]	97.9	368	94.9	942.0
	TCAD'24 [15]	96.8	145	96.8	1256.0
	TODAES'25 [16]	98.5	54	98.6	443.5
	Ours	99.4	64	99.0	274.1
Industry-3	TCAD'19 [11]	70.2	2437	71.4	5284.4
	TCAD'20 [14]	88.7	3245	79.3	2411.2
	TCAD'24 [15]	76.3	522	83.9	2565.3
	TODAES'25 [16]	80.2	989	84.0	907.2
	Ours	91.3	802	91.3	568.6
Industry-4	TCAD'19 [11]	64.6	2062	61.3	4148.0
	TCAD'20 [14]	96.2	882	89.1	1579.6
	TCAD'24 [15]	87.4	1314	80.6	2145.1
	TODAES'25 [16]	93.1	130	95.0	727.3
	Ours	95.2	123	96.2	471.6
Average	TCAD'19 [11]	80.6	2623.8	73.8	4181.3
	TCAD'20 [14]	93.5	1353.5	88.6	1814.2
	TCAD'24 [15]	88.3	979.3	87.1	2146.7
	TODAES'25 [16]	91.4	421.3	93.0	735.9
	Ours	95.0	375.8	95.3	458.0

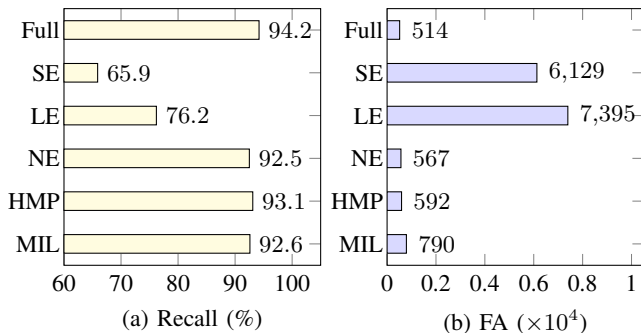


Fig. 3 Ablation study on the Industry-1 dataset.

which graph-level embeddings are obtained via max pooling and then classified using a two-layer MLP.

For graph structure, removing SpaceEdge or LayoutEdge leads to severe performance degradation, showing their importance in encoding key spacing information. Removing NeighborEdge results in a 2.9% reduction in recall, implying that polygon contour information, though less critical, remains beneficial. For the GNN architecture, discarding either HMP or MIL lowers recall to 92.5% and 89.6%, respectively, and increases false alarms, highlighting the significance of both hierarchical message passing and multiple instance learning.

V. CONCLUSION

In this paper, we address the challenge of accurate and efficient hotspot detection for complex non-Manhattan layouts. We present a general graph representation method and a hierarchical graph convolution model to encode and capture non-Manhattan geometries. In addition, a multiple instance learning strategy is adopted to enhance generalization. Experimental results on industry datasets demonstrate the effectiveness of our framework, suggesting that graph neural networks offer a promising direction for modeling non-Manhattan layouts. Future research could further exploit graph-based representations, with potential extensions to curvilinear layout modeling.

ACKNOWLEDGMENT

This project was supported by NSFC (Grant No. W2412034 and U25A20485).

REFERENCES

- [1] Q. Jin, Y. Liu, Y. Jiang, Q. Sun, and C. Zhuo, "Unitho: A unified multi-task framework for computational lithography," *arXiv preprint arXiv:2511.10255*, 2025.
- [2] 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, pp. 1–8.
- [3] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network ensemble," *IEEE TCAD*, vol. 40, no. 7, pp. 1476–1488, 2020.
- [4] 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, pp. 1–8.
- [5] J. A. Torres, "Iccad-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, 2012, pp. 349–350.
- [6] S. Sun, Y. Jiang, F. Yang, B. Yu, and X. Zeng, "Efficient hotspot detection via graph neural network," in *Proc. DATE*, 2022, pp. 1233–1238.
- [7] H. Yan, Y. Wang, P. Gao, F. Yu, Y. Ma, X. Xiong, and S. Cai, "A lightweight heterogeneous graph embedding framework for hotspot detection," *IEEE TCAD*, 2025.
- [8] B. Li, S. Wang, T. Chen, Q. Sun, and C. Zhuo, "Efficient subgraph matching framework for fast subcircuit identification," in *Proc. MLCAD*, 2024, pp. 1–7.
- [9] B. Li, Q. Peng, C. Wang, T. Ni, T. Chen, Q. Sun, and C. Zhuo, "H3match: A hybrid heterogeneous hypergraph matching method for subcircuit identification," in *Proc. DAC*, 2025, pp. 1–7.
- [10] G. R. Reddy, K. Madkour, and Y. Makris, "Machine learning-based hotspot detection: Fallacies, pitfalls and marching orders," in *Proc. ICCAD*, 2019, pp. 1–8.
- [11] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, vol. 38, no. 6, pp. 1175–1187, 2019.
- [12] H. Geng, H. Yang, L. Zhang, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," *IEEE TCAD*, vol. 41, no. 8, pp. 2685–2698, 2021.
- [13] Z. Chen, F. Yang, L. Shang, and X. Zeng, "Automated and agile design of layout hotspot detector via neural architecture search," in *Proc. DATE*, 2023, pp. 1–6.
- [14] R. Chen, W. Zhong, H. Yang, H. Geng, F. Yang, X. Zeng, and B. Yu, "Faster region-based hotspot detection," *IEEE TCAD*, vol. 41, no. 3, pp. 669–680, 2020.
- [15] H. Sun, C. Jiang, X. Ye, D. Feng, B. Tan, Y. Ma, and K. Liu, "Interpretable cnn-based lithographic hotspot detection through error marker learning," *IEEE TCAD*, 2024.
- [16] C. Jiang, H. Sun, D. Feng, Z. Xie, B. Tan, and K. Liu, "Lithoexp: Explainable two-stage cnn-based lithographic hotspot detection with layout defect localization," *ACM TODAES*, vol. 30, no. 3, pp. 1–25, 2025.
- [17] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.