Location is All You Need: Efficient Lithographic Hotspot Detection Using Only Polygon Locations

Yujia Wang*, Jiaxing Wang*, Dan Feng*, Yuzhe Ma† and Kang Liu*
*Huazhong University of Science and Technology, †Hong Kong University of Science and Technology (Guangzhou)
*{yujiawang, wangjiaxing, dfeng, kangliu}@hust.edu.cn, †yuzhema@hkust-gz.edu.cn

***Abstract*—With integrated circuits at advanced technology nodes shrinking in feature size, lithographic hotspot detection has become increasingly important. Deep learning, especially convolutional neural networks (CNNs) and graph neural networks (GNNs) have recently succeeded in lithographic hotspot detection, where layout patterns, represented as images or graph features, are classified into hotspots and non-hotspots. However, with increasingly sophisticated CNN architectural designs, CNN-based hotspot detection requires excessive training and inference costs with expanding model sizes but only marginally improves detection accuracy. Existing GNN-based hotspot detector requires more intuitive and efficient layout graph feature representation. Driven by the understanding that lithographic hotspots result from complex interactions among metal polygons through the light system, we propose the absolute and relative locations of metal polygons are all we need to detect hotspots of a layout clip. We propose a novel layout graph feature representation for hotspot detection where the coordinates of each polygon and the distances between them are taken as node and edge features, respectively. We design an advanced GNN architecture using graph attention and different feature update functions for different edge types of polygons. Our experimental results demonstrate that our GNN hotspot detector achieves the highest hotspot accuracy and the lowest false alarm on different datasets. Notably, we employ one-third of the graph features of the previous GNN hotspot detector and achieve higher accuracy. We outperform all CNN hotspot detectors with higher accuracy, up to $32\times$ speed up in inference time, and $64\times$ reduction in model size.**

***Index Terms*—lithography, hotspot detection, GNN, polygon, location**

I. Introduction

As transistors' feature sizes continue to shrink in integrated circuits, even minor process variations in optical lithography can lead to printing defects of certain sensitive layout patterns. These printing defects, known as *lithographic hotspots*, have a profound impact on manufacturing yield and must be addressed as early as possible in the design stage.

In traditional physical design, lithography simulation [1], as depicted in Fig. 1, is used to detect lithographic hotspots. While this method is accurate, it is computationally intensive and time-consuming, making it almost impractical to use in the early physical design loop for iterative optimization of defect layouts. Pattern matching (PM)-based approaches [2]–[4] are proposed to accelerate the hotspot detection process by collecting the feature characteristics of a series of known hotspot patterns in a library, and any incoming layouts are compared against this feature library for hotspot detection. While they achieve a much faster turnaround time than lithography simulation, they cannot detect hotspot patterns that have not seen before. To address the generalization issues of PM solutions, machine learning (ML) [5]–[8] and deep learning (DL) [9]–[12] methods have evolved over the past ten years for improved generalization and, thus, higher detection accuracy than PM and preserve the advantage of fast detection speed compared to lithography simulation. ML- and DL-based hotspot detectors learn a mapping function from layout features to their ground-truth lithography labels from massive training samples, and generalize to unseen layouts after deployment.

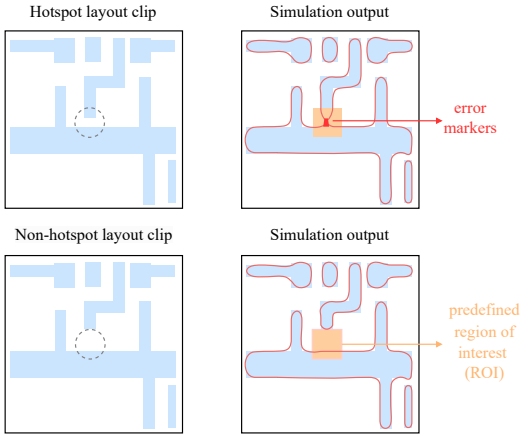


Fig. 1: Layout clips and lithography simulation results. Error markers overlapping with the ROI indicate lithographic hotspots. Minor variations in the polygon distance (circled area in the layout clips) can result in different lithography results.

Thanks to the expressive capability of neural networks, a line of DL solutions apply convolutional neural networks (CNNs) for feature extraction and binary classification, where layouts are directly represented as images and taken as inputs by a CNN. State-of-the-art (SoTA) CNN-based hotspot detectors use sophisticated designs such as Discrete Cosine Transform (DCT) [9], binarized neural networks (BNNs) [11], Inception blocks with attention modules [10], and neural architecture search (NAS)-based architectures [13] to extract more distinctive features, resulting in enhanced classification accuracy. However, as the CNN models become increasingly complex, their training and inference costs and the resulting model sizes escalate significantly. Yet, the corresponding rise in detection

Kang Liu is partly supported by National Natural Science Foundation of China No. 62202190, Hubei Natural Science Foundation No. 2023AFB237, and the Knowledge Innovation Program of Wuhan-Shuguang. Yuzhe Ma is supported in part by National Natural Science Foundation of China No. 62204066. Kang Liu is the corresponding author.

978-3-9826741-0-0/DATE25/© 2025 EDAA
Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.

accuracy is not commensurate and marginally improves. Instead of using CNNs for hotspot detection, recently, a graph neural network (GNN)-based hotspot detector [12] is applied to layout clips represented as graphs comprising nodes and edges. These extracted layout graph features are further processed by a GNN for binary classification, which yields improved computational efficiency and comparable accuracy than CNN methods.

It is worth noting that lithographic hotspots are not random occurrences; they emerge from the intricate interactions among metal polygons of a layout through the light system during lithography, where the shapes and locations of metal polygons play a crucial role. For instance, corner-to-corner distances, positions of jogs, and line-ends can all contribute to the emergence of hotspots [14]. Even minor variations, such as the distance between metal polygons, as depicted in Fig. 1, can lead to strikingly different lithography results. However, as critical features for differentiating hotspots from non-hotspots, these variations in polygon shapes and locations pose a significant challenge to learn by end-to-end CNNs solely based on training loss optimization. In [12], only part of this critical location information is extracted, along with several features that are not directly related to lithographic hotspots.

In this work, inspired by the fundamental causes of lithographic hotspots where shapes and locations of metal polygons and their relative positions determine the hotspot nature of a layout, we leverage their graph representations and a GNN for lithographic hotspot detection. We view the layout as a graph with different polygons as the nodes and their interactions as the edges. Such a graph representation with shape and location information of all the metal polygons preserves the complete geometric information of the layout design, encapsulating all the necessary details for hotspot detection. Specifically, we break down complex polygon shapes horizontally into separate rectangles and use their coordinates and the distances between them as node and edge features, respectively, by which polygon locations are preserved and their shapes are implied. We design a GNN architecture using different feature update functions for different edge types as well as graph attention for more efficient feature extraction. Compared to the previous GNN-based method [12], we achieve higher detection accuracy with one-third of the graph features and reduced model size. We also outperform prior CNN-based methods with higher accuracy, smaller model size, and much faster inference time by one order of magnitude. Our main contributions are listed below:

- A novel layout graph feature representation using only polygon locations for lithographic hotspot detection where coordinates of partitioned rectangles and their distances are extracted as node and edge features.
- An advanced GNN architecture using graph attention and different feature update functions for different edge types of polygons.
- Experimental results and insights into our GNN hotspot detector that achieves the highest accuracy, the smallest model size, and a highly efficient inference cost compared to all prior CNN and GNN solutions.
- A thorough comparative study that explores various layout graph features and network components on the accuracy of GNN hotspot detectors.

II. Preliminaries and Problem Formulation

A. Graph Neural Networks

GNN is a type of neural network that processes graph-structured data $G = (V, E)$, where V are the graph nodes and E are the graph edges. A GNN typically consists of several message-passing layers and one readout layer. In the k-th message-passing layer, each node $v \in V$ combines message information with each neighbor node $w \in N(v)$ with node feature h_w^k and each edge $e_{v,w}$ with edge feature $h_{e_{v,w}}^k$, using a message function ψ_v^k. The neighbor messages are aggregated as m_v^{k+1} using operation $\square$, based on which the node v updates its node feature h_v^k through a node feature update function ϕ_v^k, as shown in Equation 1 and Equation 2.

$$m_v^{k+1} = \square_{w \in N(v)} \psi_v^k(h_v^k, h_w^k, h_{e_{v,w}}^k) \tag{1}$$

$$h_v^{k+1} = \phi_v^k(h_v^k, m_v^{k+1}) \tag{2}$$

Similarly, the edge feature $h_{e_{v,w}}^k$ of edge $e_{v,w}$ can be updated as expressed in Equation 3 and Equation 4.

$$m_{e_{v,w}}^{k+1} = \psi_e^k(h_v^k, h_w^k, h_{e_{v,w}}^k) \tag{3}$$

$$h_{e_{v,w}}^{k+1} = \phi_e^k(h_{e_{v,w}}^k, m_{e_{v,w}}^{k+1}) \tag{4}$$

Typically, ψ_v^k, ϕ_v^k, ψ_e^k, and ϕ_e^k are multi-layer perceptions (MLP) that share cross nodes and edges at the same layer, and aggregation operation $\square$ is usually sum, mean, or max.

In the readout layer, all the node features h_v^K from the last message-passing layer K combine to provide a global graph representation y using a readout function for downstream tasks.

B. Problem Formulation

Generally, lithographic hotspot detection is a binary classification problem in which a hotspot detector predicts layout clips into hotspots and non-hotspots. In this work, we use a GNN-based hotspot detector. We use the following metrics to evaluate the detection accuracy and define our problem as follows.

Definition 1: Hotspot Accuracy (HA)—The ratio of real hotspot clips that are successfully detected as hotspot.

Definition 2: False Alarm (FA)—The number of actual non-hotspot clips that are misclassified as hotspot.

Problem 1: Hotspot Detection—Given a set of layout clips with verified hotspot/non-hotspot labels, we train a GNN-based hotspot detector that maximizes HA and minimizes FA.

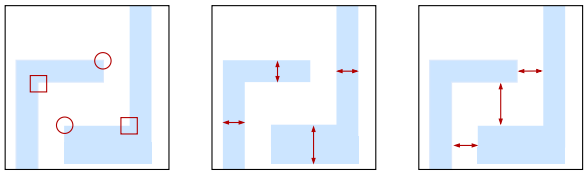
III. Proposed Method

A. Motivation and Overall Framework

Several types of layout geometric information highly correlate with the occurrences of lithographic hotspots [7]. As shown in Fig. 2, we list three major types of geometric features of metal polygons affecting layout printability.

(1) **Corner**, including convex and concave corners as marked by circles and squares respectively in Fig. 2a. A particular corner may interact with its nearby patterns and cause a hotspot.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c)

Fig. 2: Major types of layout geometric information related to lithographic hotspots, including (a) corner, (b) width, and (c) distance of metal polygons.

(2) **Width** is the distance between two parallel borders of a polygon as shown in Fig. 2b. The smaller the width, the easier it is to cause pinching and an open circuit.

(3) **Distance** measures the gap between borders of two adjacent polygons as shown in Fig. 2c. The smaller the distance, the easier it is to cause bridging and a short circuit.

Essentially, width contains the shape dimension of part of the complex metal polygon, and distance reflects the relative locations and, thus, the interactions between two adjacent metal polygons. The corner information encodes both the shape and location information of a complex metal as well as its internal connectivity. Inspired by these findings, we propose a layout graph construction scheme that encodes the location and shape information of all the metal polygons and their connectivity for further GNN classification, which we consider as critical information for hotspot detection. We illustrate our layout graph construction and feature representation scheme in Fig. 3 and summarize the GNN hotspot detection process as follows.

Step 1: Polygon decomposition. We decompose complex polygons in the layout clip horizontally into simple rectangles.

Step 2: Graph construction. We represent partitioned rectangles as the graph nodes. Two types of edges are built, one is the internal edges between connected rectangles from the same polygon, and the other is the external edges between rectangles from separate polygons that have a distance smaller than a threshold, as shown in Fig. 4.

Step 3: Feature representation. We take the coordinates of rectangles as node features. External edges use the distance as edge features and internal edges have a distance of 0.

Step 4: GNN classification. We take the feature representations of layout graphs as inputs to our GNN hotspot detector for binary classification of hotspots and non-hotspots.

B. Layout Graph Construction

We describe our layout graph construction and feature representation scheme in details as follows.

Polygon decomposition. Layout clips consist of complex polygon shapes, and their interactions precipitate lithographic hotspots. However, considering each polygon as one single graph node loses critical information for hotspot detection, as interactions exist internally within a metal polygon. For instance, the corner information of a polygon involves both horizontal and vertical parts of the same metal and contains rich information about local printability. In addition, metal polygons have drastically different shapes and expand over irregular areas within the layout clip; it is difficult to describe the polygon location and shape in a simple and unified format. Therefore, we propose to decompose each polygon into horizontally separate rectangles, and each rectangle is a graph node.

Graph construction. After decomposing metal polygons to obtain nodes of layout graphs, we add connections between interacted rectangles as graph edges. We define two types of edges between partitioned rectangles, namely (1) **internal edges** and (2) **external edges**. We first place edges between connected rectangles from the same polygon due to their intrinsic interactions presented as corners (Fig. 4b). We additionally add external edges between adjacent polygons when their projections overlap (grey area in Fig. 4d), and their distance is smaller than a predefined threshold t. For instance, in Fig. 4c, rectangles with distance d_1 and d_2 (smaller than t) are added with external edges, and rectangles with distance d_3 (larger than t) have no edge connection. External edges indicate interactions between rectangles from different polygons. In our layout graph construction, we set t to 65 nm, which is half of the pitch size of our dataset.

Feature representation. With nodes and edges constructed for the layout graph, we define node and edge features to represent the location and shape information of partitioned rectangles and their interactions for hotspot detection. We propose using the vertex coordinates of each rectangle in the layout plane as node features, which can precisely reflect the location and shape information. The smaller the distance between rectangles with external edges, the more likely it is to cause a hotspot. So we take the distance as external edge features. Internal edges have a distance of 0. Specifically, we define the following node and edge feature representations of our layout graph: **(1) node feature: coordinates** of the top-left and bottom-right vertices of the partitioned rectangles, which directly provide their absolute locations and, at the same time, imply their shape dimensions, e.g., (x_1, y_1) and (x_2, y_2) for R_1 in Fig. 4d; **(2) edge feature: distance** between two rectangles with external edges, which measures the smallest gap between their closest parallel borders, e.g., d in Fig. 4d.

C. GNN Architecture for Hotspot Detection

Our GNN architecture for hotspot detection consists of graph feature update as well as readout and classification, as shown in Fig. 5. Similar to convolutional layers in CNNs that extract image features, message-passing layers in GNNs extract graph features from input node and edge features. In each message-passing layer, each node aggregates information from its neighbor nodes and associated edges to capture the local graph structure and updates its feature representation. By stacking multiple message-passing layers, information propagates between nodes in the graph. After the k-th message-passing layer, each node will contain information from its neighbors as far as k-hops away. Taking into account the number of rectangles originating from the same polygon that naturally become neighbors and closely interact, as well as the nearby rectangles coming from different polygons within a certain distance, we stack four message-passing layers in our GNN architecture that is adequate to capture the local optical

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.

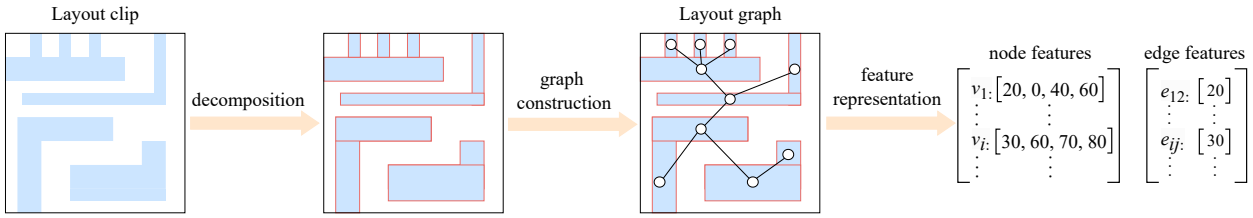


Fig. 3: Layout graph construction and feature representation.

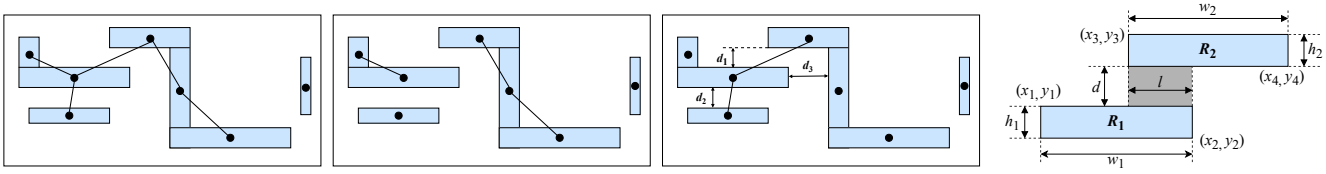


Fig. 4: Building (a) layout graph with (b) internal and (c) external edges; (d) geometric information of two adjacent rectangles.

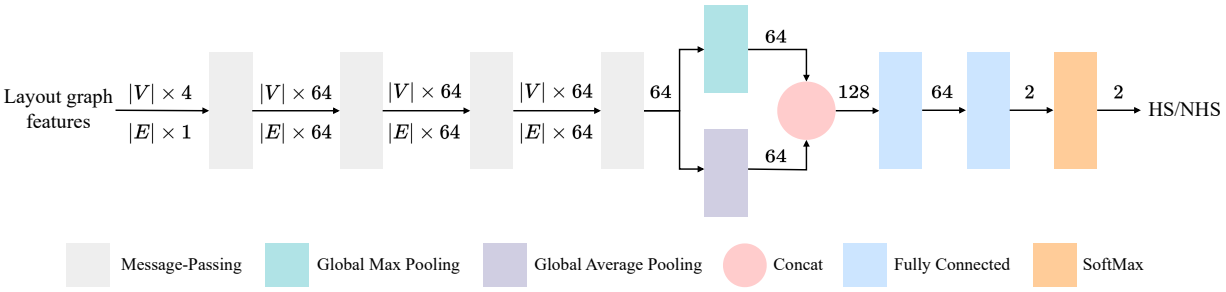


Fig. 5: GNN Architecture of our proposed hotspot detector.

proximity effect and polygon interactions. We also empirically find that using four message-passing layers results in the best detection accuracy.

Edge feature update. To distill more efficient feature characteristics of the interactions, i.e., edges, between graph nodes, we apply different update functions for internal and external edges. Specifically, in the k-th message-passing layer, we update the edge features $h_{e_{v,w}}$ as shown in Equation 5 and Equation 6.

$$m_{e_{v,w}}^{k+1} = U^k(h_v^k) \oplus U^k(h_w^k) \tag{5}$$

$$h_{e_{v,w}}^{k+1} = H_r^k(h_{e_{v,w}}^k \oplus m_{e_{v,w}}^{k+1}) \tag{6}$$

Here, h_v^k and h_w^k represent the features of nodes v and w. $\oplus$ is concatenation. U^k and H_r^k are one-layer MLPs. r denotes the type of $e_{v,w}$ that belongs to either internal or external edge.

Node feature update. Analogously, we update node features as expressed in Equation 7 and Equation 8.

$$m_{v,r}^{k+1} = \frac{1}{|N(v,r)|} \sum_{w \in N(v,r)} \alpha_{v,w} S_r^k(U^k(h_w^k) \oplus h_{e_{v,w}}^{k+1}) \tag{7}$$

$$h_v^{k+1} = h_v^k + \sum_r m_{v,r}^{k+1} \tag{8}$$

Here, $N(v,r)$ is the set of neighbor nodes connected with node v with either internal or external edges. S_r^k is the edge-dependant node feature update function using a two-layer MLP.

Inspired by Graph Attention Networks (GAT) [15], we introduce graph attention in our architecture to learn the different importance of neighbor nodes that contribute differently to the update of node features. The attention coefficients $\alpha_{v,w}$ are computed as follows in Equation 9.

$$\alpha_{v,w} = \frac{\exp(a^{\mathrm{T}}\mathrm{LeakyReLU}(h_{e_{v,w}}^{k+1}))}{\sum_{i \in N(v)} \exp(a^{\mathrm{T}}\mathrm{LeakyReLU}(h_{e_{v,i}}^{k+1}))} \tag{9}$$

Here, a is a learnable vector controlling the attention that node v receives from its neighbors.

Readout and Classification. After layout graph feature extraction by all K message-passing layers, the aggregated feature representations of all the metal polygons and their interactions can be obtained as shown in Equation 10.

$$y = GMP(\{h_v^K \mid v \in V\}) \oplus GAP(\{h_v^K \mid v \in V\}) \tag{10}$$

Here, GMP and GAP denote global max and global average pooling, respectively, which concatenate to provide a global graph representation y that is further classified by a two-layer MLP into hotspots and non-hotspots with improved accuracy.

We compare the differences between our GNN hotspot detector and prior GNN method [12] in Section V-D.

IV. Comparative Study of Layout Graph Features on GNN Hotspot Detection

In addition to the coordinates and distance of metal polygons that we use as layout features for GNN hotspot detection,

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.

various other layout information potentially affects lithographic hotspots. To provide a thorough evaluation of various layout features on detection accuracy, we conduct a comparative study that enumerates all the layout features we see related to hotspots and provides their graph representations categorized as node and edge features. We provide experimental results of using these layout features for hotspot detection later in Section V.

A. Additional Node Features of Layout Graph

Width and height. The width and height of rectangles encode their shape information, e.g., (w_1, h_1) of R_1 in Fig. 4d.

Weight. Different rectangles have different influences, i.e., weights, in affecting lithography result of a layout clip based on their locations. As lithography simulation examines printing defects within the predefined ROI area centered in the layout clip, the closer a rectangle is to the layout center, the more important it is and, thus, the higher its weight. Similar feature representation using polygon weights is used in prior work [14].

B. Additional Edge Features of Layout Graph

The overlap projection of two adjacent polygons reflects their interactions and can be represented as edge features of a layout graph for hotspot detection.

Offset. Offset reflects the relative location of two adjacent rectangles, denoted as the coordinate shift of the top-left vertices of two rectangles, e.g., $(x_1 - x_3, y_1 - y_3)$ in Fig. 4d.

Length. The length of the overlap projection also measures the relative location of two rectangles, e.g., l in Fig. 4d.

Orientation. Orientation indicates the horizontal or vertical direction of the overlap projection. Horizontal and vertical hotspots occur when lithography illumination is asymmetric [16]; such orientation feature is used in [5], [12].

V. Experimental Results and Discussion

A. Datasets and Baselines

We evaluate our proposed GNN-based hotspot detector on two commonly used datasets, one is the smaller ICCAD 2012 [17], and the other is the augmented dataset in VTS 2018 [14] that contains a massive number of layout clips and used by other works [18]–[21]. Details of these two datasets are listed in Table I. Layout clips in ICCAD 2012 are partitioned with a resolution of 1200 nm $\times$ 1200 nm, and VTS 2018 layout clips have a resolution of 1110 nm $\times$ 1110 nm. When represented as images for CNN-based hotspot detection, layout metal areas have pixel values of 1, and the background is 0.

We compare our location-based GNN hotspot detector with three SoTA DL-based solutions, including TCAD'19 [9] and TCAD'22 [10] based on CNNs and DATE'22 [12] based on GNN. TCAD'19 computes DCT coefficients of layout clips as inputs to a CNN with four convolutional layers and two fully connected layers. TCAD'22 uses five Inception blocks with CBAM attention and three fully connected layers for more distinctive feature extraction. DATE'22 extracts five node features and four edge features for GNN-based hotspot detection.

We train our models for 20 epochs with a batch size of 128 and an initial learning rate of 0.001 using the Adam optimizer for cross-entropy loss optimization. We implement our codes in Python 3.10 and PyTorch 1.12.0 on servers with an Intel Xeon W-3455 CPU and NVIDIA GeForce RTX 4090 GPU.

B. Performance of DL-based Hotspot Detection

We show in Table I the detection accuracy and inference time per layout clip of various DL-based hotspot detectors on two datasets and their model sizes.

Accuracy. Our GNN achieves the highest HA and the lowest FA on both ICCAD 2012 and VTS 2018 datasets. Specifically, we see more advantage in detection accuracy of our GNN than other DL models on the larger and much more difficult VTS 2018 dataset. The other GNN architecture, DATE'22, also outperforms two CNNs on VTS 2018. TCAD'22 generally performs better than TCAD'19 on both datasets.

Inference time. Our GNN requires significantly less inference time than CNNs with up to $31.8\times$ and $29.2\times$ speed up on two datasets, respectively, when compared to the slowest TCAD'19. Generally, GNNs are much faster than CNNs by one order of magnitude. Our GNN takes slightly longer but comparable inference time than DATE'22 that also uses a GNN.

Model size. Our GNN has the smallest model size compared to all three baselines, and only takes approximately $1/2.5$ and $1/64$ of the sizes of TCAD'19 and TCAD'22, respectively.

Discussion. TCAD'19 achieves less accurate detection results than others as its use of DCT coefficients in frequency domain loses critical spacial information of layout clips, which is crucial for hotspot detection. TCAD'22 outperforms TCAD'19 on both datasets due to its more complex network architecture that includes attention modules and Inception blocks, which help CNNs focus on more important and distinctive features between hotspots and non-hotspots. In contrast to CNN hotspot detectors that extract layout features from entire layout images, GNN solutions directly extract layout information pertaining to lithographic hotspots as graph features for classification. By extracting the most comprehensive and critical information of layout clips using polygon locations, our GNN achieves the highest detection accuracy than all prior methods.

Besides the accuracy advantage, GNN's use of layout graph features largely reduces input size, obviating the need for complex feature extraction architectures like CNNs. Thus, it results in more efficient inference costs and smaller model sizes.

C. Layout Graph Features on GNN Detection Accuracy

We show in Table II our comparative study on the detection accuracy of GNN-based hotspot detectors using different combinations of node and edge features. We mainly group the layout graph features into two categories, one provides location information of metal polygons using coordinates with shape implied, and the other represents only shape information with width and height. Additional node features include rectangle weights, and additional edge features include offset, distance, length, and orientation, as we described in Section IV.

We see far better detection accuracy when using polygon location than shape information for GNN hotspot detection. Specifically, using coordinates and distance as node and edge features results in the highest HA on both datasets. With more features added to the nodes and edges, overall accuracy does not

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.

TABLE I: Layout statistics and hotspot accuracy (%), false alarm, inference time (μs) per layout clip, and model size of various hotspot detectors. I.T.: inference time. Note that TCAD'22 already applies down-sampling on layout inputs for computational efficiency. *(We report the original accuracy numbers as claimed in the baseline papers on ICCAD 2012 dataset.)*

Dataset	Training		Validation		TCAD'19 (2.108 MB)			TCAD'22 (53.235 MB)			DATE'22 (0.840 MB)			Ours (0.835 MB)		
	#HS	#NHS	#HS	#NHS	HA	FA	I.T.	HA	FA	I.T.	HA	FA	I.T.	HA	FA	I.T.
ICCAD 2012	1204	17096	2524	13503	98.40	3535	2819.37	98.77	2510	1244.16	98.42	1731	**43.99**	**98.97**	**1068**	88.60
VTS 2018	250509	268466	999	19001	93.69	1174	2135.70	95.60	1041	1306.12	96.40	1026	**64.50**	**98.50**	**984**	73.05

TABLE II: GNN hotspot detection accuracy with different combinations of node and edge features of layout graph. *(Coord: coordinate, Wgt: weight, W: width, H: height, Ofst: offset, Dist: distance, Len: length, Orient: orientation.)*

Polygon	Node	Edge	ICCAD 2012		VTS 2018	
			HA	FA	HA	FA
Location (w/ shape implied)	Coord	-	98.02	**632**	97.40	1164
	Coord	Ofst	98.49	763	98.00	1235
	Coord	Dist	**98.97**	1068	**98.50**	984
	Coord	Dist, Len	98.61	869	97.90	993
	Coord	Dist, Len, Orient	98.42	850	97.50	**921**
	Coord, Wgt	Dist, Len, Orient	98.57	952	98.00	988
Shape	W, H	-	92.47	830	90.59	2928
	W, H	Ofst	90.1	636	94.60	1489
	W, H	Dist	93.38	804	92.89	2777
	W, H	Dist, Len	92.79	726	92.39	2745
	W, H	Dist, Len, Orient	95.01	979	90.59	2720
	W, H, Wgt	Dist, Len, Orient	95.37	1078	91.59	1371

increase. When using shape information for hotspot detection on VTS 2018, the combination of width, height and offset provides overall the highest detection accuracy, whereas using only node features of width and height has the lowest accuracy. We find that adding rectangle weights to node features does not necessarily improve detection accuracy.

Discussion. Coordinate and distance as node and edge features denote the absolute and relative locations of metal polygons on the layout clip and imply their shapes, which retain all the geometric information of a layout clip. In contrast, width and height only provide the shape of partitioned rectangles, the precise shape and location of original complex polygons are not accurately described. As a result, GNN hotspot detection using polygon locations obtains higher detection accuracy than using polygon shapes. Due to GNN's limited learning capacity, adding more features to the layout graph does not necessarily increase or even hurt accuracy. Assigning weights to rectangles does not improve detection accuracy when the graph features include the coordinates, as such weight information is embedded in the rectangle locations and can be properly implied.

D. Comparisons Between DATE'22 and Our Method

We summarize the differences of our GNN hotspot detector and DATE'22 in terms of layout graph features and network architectures in Table III. One key difference is that DATE'22 uses shape information (width and height) with relative locations as graph features, whereas ours use the precise locations (coordinate) for hotspot detection, which contains more comprehensive information of layout clips. We additionally use

TABLE III: Comparisons between DATE'22 and our method.

GNN	Features & Components	DATE'22	Ours
Node	Witdh and height	✓	
	Coordinates		✓
	Min distance of external edges	✓	
	# of internal/external edges	✓	
Edge	Distance	✓	✓
	Orientation	✓	
	Coordinates of overlap/junction	✓	
Architecture	Edge-dependent node update function	✓	✓
	Edge-dependent edge update function		✓
	Graph attention		✓
	Readout global max pooling	✓	✓
	Readout global average pooling		✓

TABLE IV: Detection accuracy using combinations of graph features and GNN architectures from DATE'22 and our method.

Graph features	Architecture	ICCAD 2012		VTS 2018	
		HA	FA	HA	FA
DATE'22	DATE'22	98.42	1731	96.40	1026
DATE'22	Ours	98.65	1464	96.60	1137
Ours	DATE'22	98.93	**817**	97.40	**940**
Ours	Ours	**98.97**	1068	**98.50**	984

separate update functions for internal and external edge features. We introduce graph attention to focus on more important neighboring nodes and edges, as well as global average pooling along with global max pooling in the readout layer for more efficient layout feature extraction.

We perform an ablation study using layout graph features and network architectures from DATE'22 and our method and report their detection accuracy in Table IV. We find that our graph features and architecture combine to yield the best overall detection accuracy with the highest HA. Despite using fewer layout graph features than DATE'22 for hotspot detection, we achieve large enhancements in both HA and FA using either architecture from DATE'22 or our method. The simpler GNN architecture from DATE'22 and our more concise yet critical graph features result in less FA but inferior HA, especially when evaluated on the more comprehensive VTS 2018 dataset.

VI. Conclusion

In this work, we propose a novel GNN-based lithographic hotspot detector that uses polygon locations. Compared to all prior DL-based solutions, it has the highest detection accuracy, the smallest model size, and highly efficient inference costs.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] C. A. Mack, "Thirty years of lithography simulation," in *Optical Microlithography XVIII*, vol. 5754. SPIE, 2005, pp. 1–12.

[2] 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 *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–6.

[3] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1671–1680, 2014.

[4] 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 *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 1167–1172.

[5] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan, "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *2009 IEEE international conference on IC design and technology*. IEEE, 2009, pp. 219–222.

[6] H. Zhang, B. Yu, and E. F. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.

[7] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1621–1634, 2011.

[8] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on adaboost classifier and simplified feature extraction," in *Design-Process-Technology Co-optimization for Manufacturability IX*, vol. 9427. SPIE, 2015, pp. 201–211.

[9] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 6, pp. 1175–1187, 2019.

[10] H. Geng, H. Yang, L. Zhang, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2685–2698, 2022.

[11] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network ensemble," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, 2021.

[12] S. Sun, Y. Jiang, F. Yang, B. Yu, and X. Zeng, "Efficient hotspot detection via graph neural network," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1233–1238.

[13] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via neural architecture search," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 6, pp. 1–16, 2022.

[14] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 2018, pp. 1–6.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[16] J. W. Park, A. Torres, and X. Song, "Litho-aware machine learning for hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1510–1514, 2017.

[17] J. A. Torres, "Iccad-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 349–350.

[18] K. Liu, B. Tan, R. Karri, and S. Garg, "Poisoning the (data) well in ml-based cad: A case study of hiding lithographic hotspots," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 306–309.

[19] Z. Cheng and K. Behdinan, "Deep learning hotspots detection with generative adversarial network-based data augmentation," *Journal of Micro/Nanopatterning, Materials, and Metrology*, vol. 21, no. 2, pp. 024 201–024 201, 2022.

[20] K. Liu, B. Tan, R. Karri, and S. Garg, "Training data poisoning in ml-cad: Backdooring dl-based lithographic hotspot detectors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1244–1257, 2020.

[21] T. Zhang, H. Yang, K. Liu, and Z. Xie, "Apple: An explainer of ml predictions on circuit layout at the circuit-element level," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 374–379.

Authorized licensed use limited to: The Hong Kong University of Science and Technology (Guangzhou). Downloaded on August 25,2025 at 06:10:54 UTC from IEEE Xplore. Restrictions apply.