A Lightweight Heterogeneous Graph Embedding Framework for Hotspot Detection

Haopeng Yan, Ying Wang, Peng Gao, Fei Yu, Yuzhe Ma, *Member, IEEE*, Xiaoming Xiong, and Shuting Cai, *Member, IEEE*

***Abstract*—Hotspot detection is a crucial step in ensuring the manufacturability of integrated circuits, as it seeks to identify potential defects in the layout. Pattern matching methods have been widely used to accelerate the detection of these defects. However, they often struggle with complicated deviations. Image-based machine learning methods were introduced to confront this challenge, but they often involved distorted information extraction and incurred significant runtime overhead. In this article, we introduce a novel detection framework based on the modified transitive closure graph (MTCG). By applying the concept of MTCG, the layout can be accurately modeled as a heterograph. The embeddings of these heterographs are extracted using an optimized lightweight 3-hop message-passing graph neural network (GNN) and subsequently utilized for classification. Furthermore, a dynamic edge transformation method based on the properties of MTCG is proposed for data augmentation. The proposed method is evaluated with datasets from ICCAD 2012 and ICCAD 2019, demonstrating outstanding performance in recall and false alarm, along with a significantly decreased inference time.**

***Index Terms*—Graph learning, hotspot detection, modified transitive closure graphs (MTCGs).**

I. Introduction

ADVANCES in technology have led to an increase in the complexity and scale of designs, prompting the adoption of various resolution enhancement techniques in the lithography process. Despite the contributions of these techniques, process variations inevitably lead to significant deviations of certain patterns during their transfer to wafers. Pattern-distorted regions, termed lithography hotspots, indicate potential layout issues after physical design. Although lithography simulation is a reliable solution, its time-consuming nature imposes significant limitations on the design process, leading to a longer turn-around time.

Received 17 September 2024; revised 16 December 2024; accepted 8 February 2025. Date of publication 18 February 2025; date of current version 22 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U23A20361, and in part by the Key Area Research and Development Program of Guangdong Province under Grant 2022B0701180001. This article was recommended by Associate Editor L. Behjat. *(Corresponding authors: Peng Gao; Shuting Cai.)*

Haopeng Yan, Ying Wang, Peng Gao, Fei Yu, Xiaoming Xiong, and Shuting Cai are with the School of Integrated Circuits, Guangdong University of Technology, Guangzhou 510006, China (e-mail: 2112204053@mail2.gdut.edu.cn; 2112205302@mail2.gdut.edu.cn; pengao@gdut.edu.cn; yufei@gdut.edu.cn; xmxiong@gdut.edu.cn; shutingcai@gdut.edu.cn).

Yuzhe Ma is with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: yuzhema@hkust-gz.edu.cn).

Digital Object Identifier 10.1109/TCAD.2025.3543436

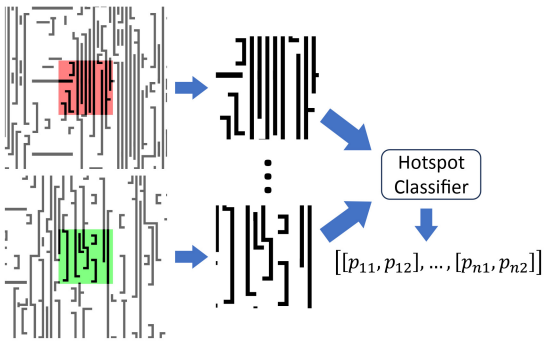

Fig. 1. Clip-level hotspots detection.

To accelerate the lithography process, the hotspot detection technique is introduced. Traditional hotspot detection relied on pattern matching, where patterns within the layout were searched and compared with predefined hotspot patterns. However, this approach was inherently limited to handling fixed patterns. A hotspot detector leveraging modified transitive closure graph (MTCG) [1], [2], was capable of identifying hotspots with rotation, mirroring, and combinations of different hotspot patterns, extracting crucial design rules for hotspot region matching. However, the above pattern matching-based method could not classify patterns that were not present in the hotspot library. For example, the testing dataset of ICCAD 2012 competition [3] contained some hotspot samples that were not included in the given library. ICCAD2019 benchmarks [4] were proposed for conducting evaluations to assess the robustness of the detector against false alarm and its ability to identify samples that deviate from the distribution of the training data. Fig. 1 illustrates the task conducted in this article using the aforementioned datasets.

Machine-learning-based detectors were introduced to solve those pitfalls. A method [5] was proposed to initially cluster hotspot regions based on density and topology types, extract diverse complicated information from these regions using MTCG, and subsequently classify them with a support vector machine. Advancements in computer vision have led to the application of numerous great ideas and techniques to hotspot detection [6], [7], [8], [9]. Unlike previous methods that directly used images as input, a hotspot detector [10], [11] utilized discrete cosine transform (DCT) for feature extraction. With the rise of deep learning, convolutional neural networks (CNNs) enhanced by attention mechanisms have been applied to hotspot detection [12], [13]. The success

of CNNs in this field has further motivated the exploration of more advanced neural architectures [14], [15]. Given the effectiveness of graph neural networks (GNNs), a GNN-based hotspot detector [16] was proposed to leverage the geometric features of polygons. Feature processing is crucial in machine learning methods, especially in hotspot detection. Computer vision-based approaches either converted layouts into images or extracted multiscale feature maps based on process nodes. An existing GNN method decomposed layout polygons into multiple vertically aligned rectangles, using their geometric attributes as node features. Different edge types and features were assigned on the basic of the relationships between the rectangles and the process nodes. However, incorporating various edge and node features can degrade the inference efficiency of the model. Using a large language model (LLM) for hotspot detection was a novel approach. Polygons in a layout were encoded according to their category, position, and shape, transforming the layout into a 1-D textual representation. Subsequently, the LLM was employed to classify the text representation of the layout clip, offering a novel alternative solution [17]. However, similar to other encoding methods, this approach still relied on prior knowledge to handle polygon categories.

Another challenge for machine-learning-based hotspot detectors is the imbalance within datasets, where hotspot regions constitute only a small fraction of the available training data. Training models directly with raw data result in performance degradation or bias toward predicting one class predominantly. Previous studies have typically addressed this issue by oversampling hotspots within the layout. In computer vision, data augmentation methods, such as rotation and mirroring, were commonly used to enhance the original dataset [18]. The method based on MTCG offers a more efficient solution. It can model layouts as heterogeneous directed graphs, incorporating diverse edge types, which facilitates data augmentation by manipulating edge directions and types. This approach enables the reuse of partial information in MTCG during feature extraction, thereby reducing processing time.

Building on previous research, we propose a lightweight GNN-based hotspot detector. This method constructs a heterograph representation of the layout via MTCG, incorporating both topological structures and geometric features without requiring prior knowledge. By effectively integrating information from polygons and white spaces, our method enhances the representation of layouts. In the embedding stage, a three-stage message-passing approach is adopted, ensuring a balance between computational efficiency and model performance. The main contributions of this article are as follows.

1) In contrast to previous single-graph approaches, we propose an end-to-end hotspot detection framework with a dual-graph layout representation based on improved MTCG, facilitating lossless layout clip representation and effective geometric feature extraction.
2) Building on the dual-graph representation, we propose a lightweight incremental GNN to model polygon relationships in the layout, utilizing progressively extracted embeddings to preserve local geometric features.
3) Leveraging MTCG properties, specialized data augmentation techniques are proposed to mitigate data imbalance on benchmarks.
4) Experimental results show that our method achieves higher recall at nearly all false alarm levels while significantly reducing inference time. A detailed analysis further validates the robustness and performance of the approach.

The remainder of this article is organized as follows. Section II introduces the background to hotspot detection, GNN, and MTCG. In Section III, we present a method for transforming layouts into heterographs and extracting graph embeddings. In Section IV, various performance metrics of the model are compared and analyzed. Section V provides a comprehensive summary.

II. Background

A. Problem Formulation

The problem of hotspot detection is defined as follows: identifying and extracting hotspot samples from a given set of layout clips, effectively isolating the specific hotspots within the input clips. The performance of the detector is usually evaluated by recall and false alarm. The definitions of these metrics are as follows.

Definition 1 (Recall): The ratio of correctly predicted hotspots among the set of actual hotspots [3]

$$\text{Recall} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}. \tag{1}$$

Definition 2 (False Alarm): The number of incorrectly predicted nonhotspots [3]

$$\text{FalseAlarm} = \#\text{FP}. \tag{2}$$

B. Graph Neural Network

Hotspot detection is related to the graph classification task, which involves acquiring representations from layouts modeled as heterographs. Heterographs are graphs characterized by different node types and edge types, which facilitate the integration of data with diverse attributes. The message-passing neural network (MPNN) is an efficient type of heterogeneous GNN. leveraging MPNN, the heterogeneous GNN effectively captures and propagates information across the different types of edges and nodes, enabling comprehensive and context-aware learning within heterographs.

MPNN contains a message passing and a readout steps. Fig. 2 illustrates the process of message passing. After the message passing, global features are read out from each type of node. Given a heterograph $G(V, E)$, N_v denotes a subset of nodes which is neighboring to the node v, $N_v \in V$, e_{vw} represents a edge from node w to node v. The message-passing mechanism in MPNNs is performed as follows: after initialization, the information of node v, its neighboring nodes N_v, and the incoming edges of node v are passed into a message function named M_t, represented by

$$m_v^{t+1} = \sum_{w \in N_v} M_t\left(h_v^t, h_w^t, e_{vw}\right) \tag{3}$$

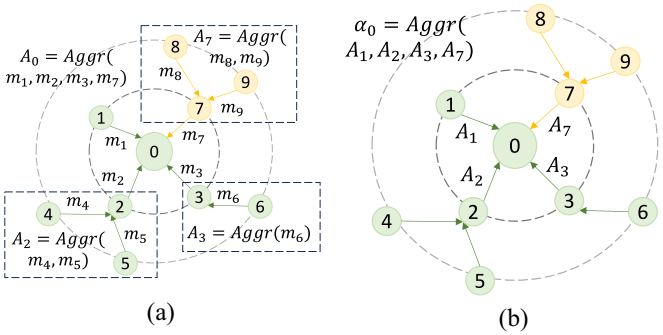

Fig. 2. Example of MPNN involves a message passing across multiple types of nodes and edges. (a) In the 1-hop message passing, nodes 0, 2, 3, and 7 receive messages from their neighbors and aggregate them to generate new features A. (b) In the 2-hop message passing, node 0 receives information from nodes 1, 2, 3, and 7, and aggregates it to produce new features α.

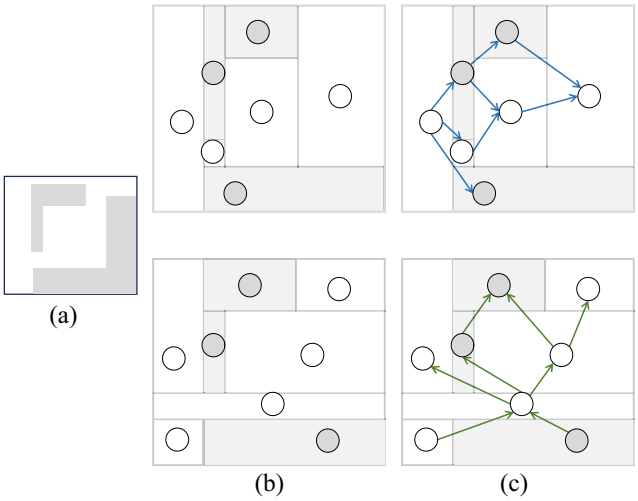

Fig. 3. Transformation between layout clip and MTCG. (a) Original layout clip. (b) Original layout is divided vertically and horizontally into rectangular areas, which can be mapped as nodes. (c) Rectangles are projected vertically and horizontally, forming edges between overlapping neighboring projections. Each layout is represented by a pair of graphs as (c).

where m_v^{t+1} denotes the message generated, and h_v^t represents the hidden state during the t-step of message passing on node v. The result of M_t is then aggregated. The next step involves updating the nodes with a node update function U_t, defined in

$$h_v^{t+1} = U_t\left(h_v^t, m_v^{t+1}\right) \tag{4}$$

which incorporates both the aggregated messages and the current state of the node to generate an updated state representation. The nodes gather and integrate information from their neighbors through several iterations of message generation and updating. After completion of these iterations, a readout function R [19], defined as

$$\hat{y} = R\left(h_v^T | v \in G\right) \tag{5}$$

extracts the feature $\hat{y}$ from the nodes in the graph, where T represents the number of message-passing iterations.

C. Maximum and Attention Aggregation

Maximum aggregation could effectively captures geometric information [20]. For a target node i with neighbors N_i, maximum aggregation could be defined as

$$\text{MAX}(\mathbf{X_i}) = \max(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}) \tag{6}$$

where max is the element-wise maximum operation of each feature vector $\mathbf{x_i}$, $\mathbf{X_i}$ is the features of N_i.

Inspired by [21], we incorporate the self-attention mechanism to aggregate information from neighboring nodes. It computes the hidden states of the node features in

$$F(\mathbf{x_i}) = W^{(2)}\left(\sigma(W^{(1)}\mathbf{x_i} + b^{(1)})\right) + b^{(2)} \tag{7}$$

where $W^{(1)}$ and $W^{(2)}$ are learnable parameters, $b^{(1)}$ and $b^{(2)}$ are biases, and σ is the activation function. Attention scores α for the features of each node are calculated by applying the softmax operation to the hidden states

$$\alpha_i = \frac{\exp(F(\mathbf{x_i}))}{\sum_i \exp(F(\mathbf{x_i}))}. \tag{8}$$

The features of the neighboring nodes are aggregated with attention scores as weights to generate the new feature for node i

$$\text{ATTN}(\mathbf{X_i}) = \sum_{i \in N_i} \alpha_i \mathbf{x_i}. \tag{9}$$

D. Modified Transitive Closure Graph

Layouts can be represented as MTCG [1], an extension of the transitive closure graph (TCG) [22], which preserves the complete geometric information of the original layout. The MTCG method partitions polygons and white spaces within layouts into multiple rectangles, creating a dual-graph that encompasses geometric information for both types of regions. It defines geometric relationships for each rectangle within the divided regions. Fig. 3 illustrates an example of a transformation. Initially, the polygons are partitioned along their vertical or horizontal edges to generate rectangles. When partitioning white spaces within the layout, the process ensures that the partitions respect the boundaries of adjacent polygons and the layout itself.

In partitioned layouts, rectangles can be connected by horizontal or vertical edges. In horizontally connected layouts, the process always starts from the left side and proceeds toward the right side. If the rectangle a shares a vertical border with the rectangle b, a directed edge ab is established to represent the horizontal connection between these two rectangles. In contrast, if there is no shared vertical border between the rectangles, no edges are formed between them. The same procedure is applied for the vertically connected layout. By incorporating these directed graphs, we can obtain the MTCG representation for the layout, encapsulating the geometric information and relationships between the various regions within the layout. MTCG generated graphs are denoted Vc_h, Hc_v, Hc_h, and Vc_v, as shown in Figs. 3(c) and 6(a). In the name Vc_h, the capital letter denotes the direction of the

TABLE I
NODE FEATURES OF HETEROGRAPH

Notation	Definition
w	Width of rectangle
h	Height of rectangle
W	Width of 1-hop neighbor information
H	Height of 1-hop neighbor information
ω	Width of 2-hop neighbor information
η	Height of 2-hop neighbor information
df	The geometric features of the nodes after 2-hop propagation
ar	The geometric features of the nodes after 3-hop propagation
gf	The global numerical and area features of the graph

partition being performed. Specifically, V represents vertical partitioning, while H represents horizontal partitioning. The subscript indicates the type of edges within the partitioned layouts. Similarly, Hc_v, Hc_h, and Vc_v follow the same convention. Using only Hc_h and Vc_v, or Vc_h and Hc_v, is sufficient to characterize the layouts [2].

III. MTCG-GNN-BASED HOTSPOT DETECTOR

This section details the proposed graph embedding framework and the data augmentation strategy. The overall process is as follows.

1) MTCGs are extracted from layouts and converted into heterographs. To incorporate more geometric information, we introduce boundary and polygon nodes into these dual-graphs. The widths and heights of rectangles are used as features for the corresponding nodes.
2) These heterographs are updated with a lightweight 3-hop GNN to extract local features. Combined with the global features derived from the MTCGs, the layout embeddings are generated.
3) Building on the properties of the directed and heterogeneous nature of the MTCG representation, we propose a data augmentation method specifically designed for layout representations in MTCG.

A. Represent Layouts With Heterographs

As illustrated in Fig. 1, the layouts provided by the benchmarks of ICCAD 2012 and ICCAD 2019 consist of the core area where the hotspot may potentially occur along with the surrounding context. Polygons within the core area serve as the fundamental criterion to assess the presence of a hotspot in the layouts. In the proposed method, Vc_h and Hc_v are utilized to represent the original layouts. Their edge types are orthogonal to the partitioning direction, allowing a better representation of the geometric relationships within the layouts. In the case of partitioned figures, each rectangle is represented as a node in the heterograph. The properties associated with the rectangles within the subgraph, such as width and height shown in Table I, are extracted and utilized as the features for the corresponding node in the heterograph.

Fig. 3 shows the transformation from dual-graph to heterograph, the rectangles generated from polygons in the layout are labeled V^b, while those from white spaces are labeled as V^s. The directed connections between V^b and V^s are denoted by E, where the superscript indicates the type of edge. In this case, E^h represents horizontal connections, while E^v is used to represent vertical connections. To achieve a comprehensive integration of information from the dual-graph, the polygon nodes V^p and the boundary nodes V^d are introduced. The purpose of V^p is to record the connection relationship of V^b from the dual-graph that originally belongs to the same polygon through the aggressive connection E^p. The nodes of MTCGs may lack incoming edges to connect with other nodes, particularly along the boundaries of the MTCGs. To facilitate synchronous update of the features for these nodes during the message-passing step and identify the nodes near the boundary, V^d is introduced. The widths and heights of V^d are initialized to 0, while features of V^p can be omitted. Inspired by the density-based features presented in [23], for each graph, the geometric features of the nodes are extracted, the total area of the block and space nodes is calculated, along with the combined area of both types and the ratios of area of each type to the total area. Regarding the node count features, including the number of block and space nodes, the total number of nodes, and the ratios of each type to the total node count. The features for the number of horizontal and vertical connections are also extracted in a similar manner. These three types of features are concatenated to form the global feature representation of the graph, denoted as gf, which is defined as follows:

$$\begin{aligned} es &= \#E^h + \#E^v \\ ns &= \#V^b + \#V^s \\ ec &= \left[\#E^h, \#E^v, es, \frac{\#E^h}{es}, \frac{\#E^v}{es}\right] \\ nc &= \left[\#V^b, \#V^s, ns, \frac{\#V^b}{ns}, \frac{\#V^s}{ns}\right] \\ ac &= \left[A\left(V^b\right), A\left(V^s\right), A(\text{clip}), \frac{A\left(V^b\right)}{A(\text{clip})}, \frac{A(V^s)}{A(\text{clip})}\right] \\ gf &= ec \,||\, nc \,||\, ac \end{aligned} \tag{10}$$

where # represents the number of edges or nodes, A indicates the area of the rectangle corresponding to the node, and $A(\text{clip})$ denotes the area of the entire layout clip. The above graphs and global features collectively form the initial heterograph representation of the layout.

B. Graph Embedding

The preceding steps yield heterograph representations of the layouts, consisting of four types of nodes and three types of edges. Among them, V^b, V^s, and V^d possess initial features w and h, which denote the width and height of the nodes corresponding to the original components, respectively. However, these features alone are insufficient, as lithography hotspots often result from interactions between adjacent polygons. Therefore, we propose extracting three-stage neighbor information along different edge types to capture node neighbor information more comprehensively, as illustrated in Fig. 4.

1-hop neighbor information involves fusing the features of node i and its neighbors N_i. If $e_{ij} \in E^h$ and $j \in N_i$, the widths of node i and node j are concatenated and passed through a shared MLP to generate a new feature W, which is stored on the

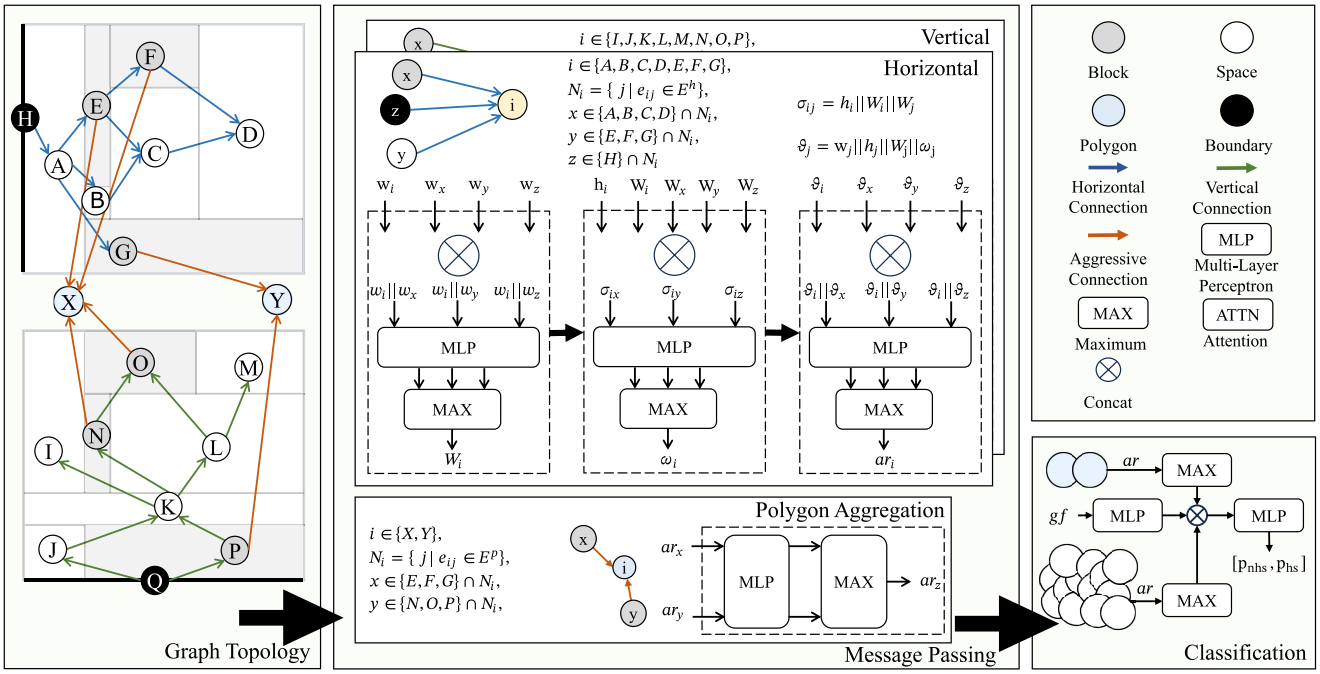

Fig. 4. Overview of the proposed framework. The update mechanisms for horizontal and vertical edges in the figure are similar. Horizontal edges are updated based on w, modifying the attributes W, ω, ar, while vertical edges are updated based on h, modifying the attributes H, η, ar.

edge e_{ij}. These edge features W are subsequently aggregated to node i using (9). For $e_{ij} \in E^v$ and $j \in N_i$, height information is used to generate the new feature H through a similar process.

2-hop neighbor information involves the fusion of the features of node i and its neighbors N_i based on the 1-hop neighbor information. The transformation process is similar to that of 1-hop neighbor information, with w and h replaced by W and H. In addition, for $e_{ij} \in E^h$ and $j \in N_i$, the h feature of the node i is included in the calculations. For $e_{ij} \in E^v$ and $j \in N_i$, the w feature of node i is incorporated. By applying (6) to the edge features, the 2-hop neighbor information ω and η is generated.

3-hop neighbor information concatenates the node features w, h, W, and ω of node i and node $j \in N_i$ into vectors if $e_{ij} \in E^h$, while the node features w, h, H, and η are combined into vectors if $e_{ij} \in E^v$, which are then transformed into hidden states with an MLP. Subsequently, the hidden states of node i and node j are concatenated and passed through another MLP to determine their relationship. The new feature for node i is derived using (6) on the generated hidden states.

The update process for three-stages message passing is shown in Algorithm 1. In the heterograph, updates for 1-hop and 2-hop involve the V^b and V^s, and the node features are updated along the horizontal and vertical connections. Using all node features in message passing can increase model inference time and hinder the ability of the model to learn key features. Therefore, in the 1-hop stage, only the width or height features of node i and its neighbors N_i are incorporated into message passing. The 1×1 features of node i and N_i are paired to generate higher-dimensional hidden states. After computing the hidden states of node i and its neighbors N_i, the information is aggregated. To maximize the retention of geometric information from node i and its neighbors, attention-weighted aggregation is introduced as the aggregation function in the 1-hop stage, ensuring that all neighbor information contributes to the computation of new features. This serves as the foundation for subsequent feature learning. Based on the features generated in the 1-hop stage, the 2-hop stage further updates the features by incorporating additional information. For example, if $e_{ij} \in E^h$ and $j \in N_i$, h contributes to the update by being concatenated with W_i and W_j, whereas if $e_{ij} \in E^v$, w is used in the update. After these two stages, the features are expanded to higher dimensions. To further explore the relationships between nodes, a 3-hop message-passing mechanism is proposed. It is divided into two steps: updating the information of nodes and passing information between the node and its neighbors. After the two-stage incremental feature update, different MLPs are used to convert the concatenated geometric feature vectors in V^s and V^b into hidden states df representing the complete relationship between nodes and their neighbors in higher dimensions. Similarly, to learn more accurate internode relationships, the generated high-dimensional features are propagated between nodes once more, resulting in the generation of features ar. Global max pooling is used to transfer the ar features of V^b to their corresponding V^p. Polygon features ar^p, representing the geometric information of all polygons in the layout, are generated by applying the global maximum pooling to ar of all polygon nodes V^p. For V^s, which do not correspond to polygons in the layout, global maximum pooling is applied directly to ar to generate the features ar^s. These features, combined with the global features gf illustrated in (10), form the graph embedding of original layout.

In Fig. 5(a) and (b), we use principal component analysis (PCA) to reduce the dimensionality of the embeddings learned by the CNN method and the proposed GNN method for

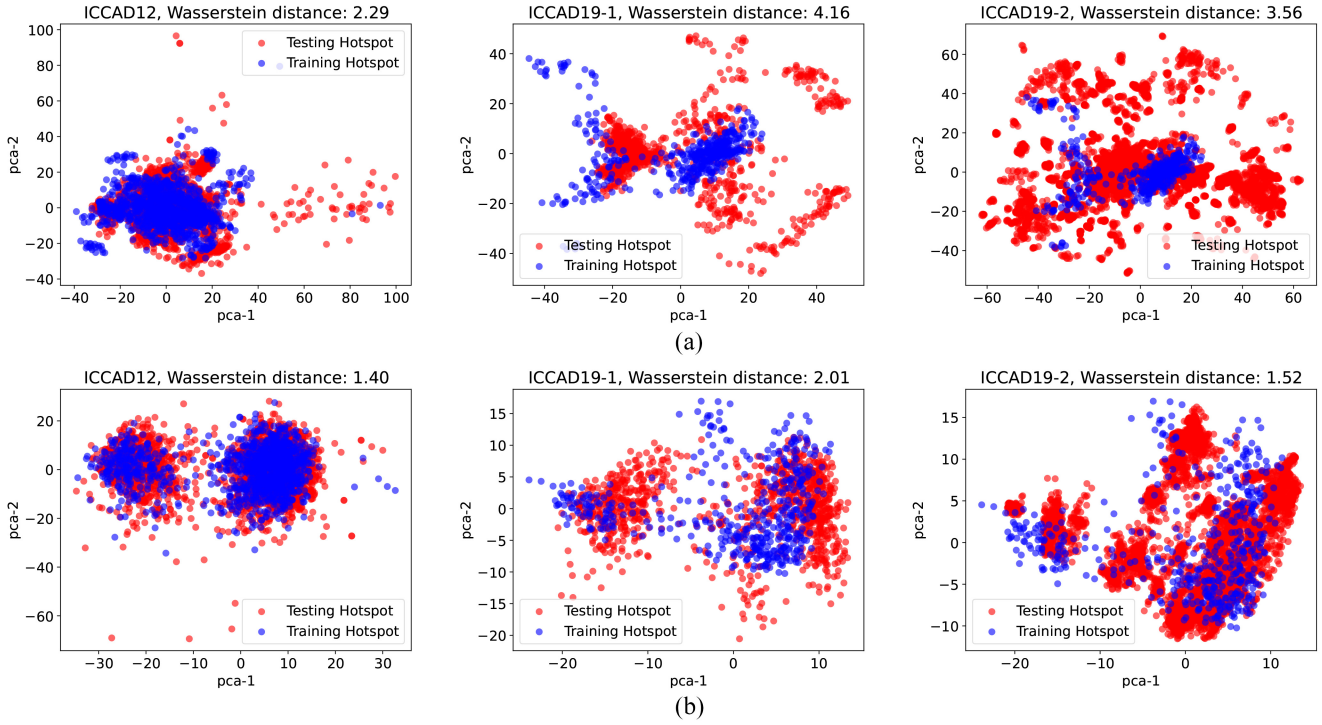

Fig. 5. Embeddings of hotspots obtained from the trained model on the training set and testing set, visualized after applying PCA. The Wasserstein distance is employed to quantify the difference between the distributions of embeddings. (a) Embedding from the CNN model [24]. (b) Embedding from our method.

visualization. We employ the Wasserstein distance [25] to quantify the distributional differences of hotspots between the training and testing data

$$W_p(\mu,\nu) = \inf_{\substack{X\sim\mu \\ Y\sim\nu}} \mathbb{E}\big(\|X-Y\|^p\big)^{\frac{1}{p}}, \quad p \geq 1 \tag{11}$$

where the infimum is taken over pairs of 2-D embeddings X and Y marginally distributed as μ and ν, respectively. The CNN method can gather embeddings more densely, aligning the distribution of the training and test data in ICCAD 2012-28 more effectively. However, for ICCAD 2019-1, which requires robust generalization, the CNN method maps the training data into a relatively constrained space, failing to achieve satisfactory results for test data that deviate from the training distribution. For simplicity, only the clip2 test case from ICCAD2019-2 was used for visualization. The ICCAD 2019-2 dataset tests the model's ability to detect fine-grained differences accurately. The CNN method, constrained by information loss during feature generation, demonstrates limited capability in effectively capturing these distinctions. With the improved MTCG representation and the proposed GNN structure, our method effectively captures the geometric features of layouts without loss. It disperses layouts with different characteristics into distinct regions in the embedding space rather than clustering them together, thereby significantly enhancing detection accuracy and generalization on more challenging datasets.

C. Data Augmentation

ICCAD 2012 [3] and ICCAD 2019 [4] both suffer from the issue of class imbalance. The training data predominantly consists of nonhotspot areas, while hotspot areas represent a minority within the datasets. In machine learning methodologies, an imbalanced dataset may lead models to exhibit a tendency to predict input data as the majority class. Therefore, the introduction of data augmentation techniques is necessary to mitigate this imbalance by increasing the number of samples from the minority class. As mentioned in Section II-D, two dual-graphs can be generated from the same layout. While Vc_h and Hc_v represent the original layout, Hc_h and Vc_v provide an alternative representation to augment the minority class. Inspired by data augmentation techniques in computer vision, such as rotating or mirroring images [26], similar approaches can be applied to the proposed heterograph transformation method to generate additional heterograph representations from the existing two dual-graphs. This is achieved by flipping the direction of the edges, swapping the type of edges, and simultaneously employing them. It can be applied to both Hc_v and Vc_h, as well as Hc_h and Vc_v. The results of data augmentation are shown in Fig. 6, where different graph representations of the same layout are generated.

The proposed data augmentation method extracts different heterograph representations from the original layout, rather than generating heterographs from rotated or mirrored layouts. This approach not only effectively utilizes existing graph information, reducing the time required to generate additional data, but also leverages the two pairs of dual-graphs generated by MTCG, improving the diversity of graph representations. In addition to employing the aforementioned data augmentation techniques, oversampling has been introduced to equalize the number of hotspots and nonhotspots in the training dataset.

Algorithm 1: Update Feature

Data: Graph $G(V, E)$; features of node i: w_i, h_i; neighbor nodes of i: N_i
Result: Updated node features ar

$S \leftarrow V^b \cup V^s$;
for $i \in S$ **do**
  for $j \in N_i$ **do**
    if $e_{ij} \in E^h$ **then**
      $W_i \leftarrow ATTN(F([w_i, w_j]), W_i)$;
    else if $e_{ij} \in E^v$ **then**
      $H_i \leftarrow ATTN(F([h_i, h_j]), H_i)$;
for $i \in S$ **do**
  for $j \in N_i$ **do**
    if $e_{ij} \in E^h$ **then**
      $\omega_i \leftarrow MAX(F([h_i, W_i, W_j]), \omega_i)$;
    else if $e_{ij} \in E^v$ **then**
      $\eta_i \leftarrow MAX(F([w_i, H_i, H_j]), \eta_i)$;
for $i \in S$ **do**
  for $j \in N_i$ **do**
    if $e_{ij} \in E^h$ **then**
      $df_i \leftarrow F([w_i, h_i, W_i, \omega_i])$;
      $df_j \leftarrow F([w_j, h_j, W_j, \omega_j])$;
    else if $e_{ij} \in E^v$ **then**
      $df_i \leftarrow F([w_i, h_i, H_i, \eta_i])$;
      $df_j \leftarrow F([w_j, h_j, H_j, \eta_j])$;
    $ar_i \leftarrow MAX(F([df_i, df_j]), ar_i)$;
for $i \in V^p$ **do**
  for $j \in N_i$ **do**
    $ar_i \leftarrow MAX(ar_j, ar_i)$;

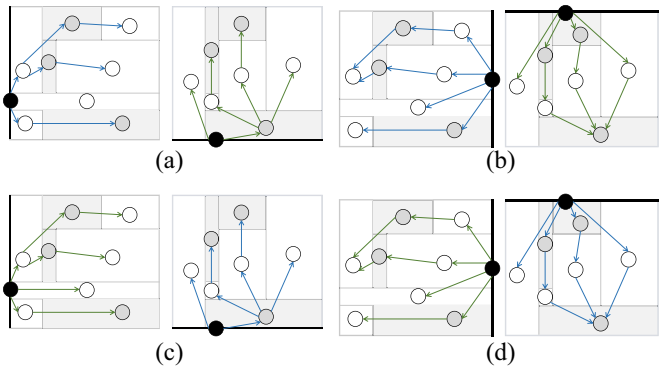

Fig. 6. Data augmentation with MTCG. (a) Other pair of dual-graphs extracted by MTCG, Hc_h, and Vc_v. (b) Flipping directed edges of Hc_h and Vc_v. (c) Modifying the edge types of Hc_h and Vc_v. (d) Modifying both the direction and edge types of Hc_h and Vc_v. For simplicity, the polygon nodes are omitted, as they share the same connectivity as the original data.

IV. Experiment

The proposed method is tested with the ICCAD 2012 [3] and ICCAD 2019 [4] benchmarks. One case at the 32-nm technology node and four cases at the 28-nm technology node are included in the ICCAD 2012 benchmark. To fully evaluate the performance of the model, all 28-nm cases from ICCAD 2012 are consolidated into one, denoted as ICCAD 2012-28. The ICCAD 2019 benchmark comprises one training dataset and two testing datasets. Benchmark statistics, including the number of hotspot regions (#HS) and nonhotspot regions (#NHS), are described in Table II. Conversion of layout files into the required heterographs is facilitated by a C++ program with the libtorch library. The GNN model is implemented with Pytorch [27] and DGL [28]. Feature extraction, model training, and evaluation are conducted on a Linux machine with a 2.4-GHz CPU and an Nvidia A100 40-GB graphic card.

TABLE II
Benchmark Information

Benchmark	Training Set		Testing Set	
	#HS	#NHS	#HS	#NHS
ICCAD 2012-28	1204	17096	2524	13503
ICCAD 2019-1	467	17785	1001	14621
ICCAD 2019-2			64310	65523

TABLE III
Hotspot Detection Result Comparison

Dataset	Method	Recall (%)	# FA	FA (%)	Time (s)
ICCAD 2012-28	TCAD'21 [29]	**99.4**	2660	19.7	4.4^a
	DATE'22 [16]	98.4	1731	12.8	3.2^b
	TCAD'22 [13]	98.8	1099	6.5	19.3^b
	DATE'23 [15]	96.2	880	9.7	7.5^b
	DAC'24 [17]	98.5	1132	8.4	9.9^a
	Ours	99.0	**783**	**5.8**	$\mathbf{2.1^a}$
ICCAD 2019-1	TCAD'21 [29]	80.9	**365**	**2.5**	3.8^a
	TCAD'22 [13]	69.1	443	3.0	9.3^b
	DATE'23 [15]	**91.6**	1257	8.6	8.3^b
	Ours	91.3	1222	8.4	$\mathbf{2.6^a}$
ICCAD 2019-2	TCAD'21 [29]	89.8	54974	83.9	31.4^a
	TCAD'22 [13]	**95.2**	54391	83.0	112.4^b
	DATE'23 [15]	90.5	54973	83.9	44.9^b
	Ours	94.6	**53579**	**81.8**	$\mathbf{29.8^a}$

*The testing results for Recall and FA of the literature methods are derived from the data reported in their respective papers.
[a]Inference time measured on the Nvidia A100 platform.
[b]Inference time reported in their respective papers.

A. Result Comparison

The performance of the proposed method and several state-of-the-art approaches are summarized in Table III, outlining the experimental results across the ICCAD 2012-28 and ICCAD 2019 benchmarks. In the results comparison, we present four key metrics: 1) Recall (%), which represents the probability of correctly detecting hotspots in the testing data; 2) #FA, the number of nonhotspot samples in the testing data incorrectly identified as hotspots; 3) FA (%), the ratio of incorrectly identified nonhotspot samples to the total number of nonhotspot samples; and 4) Time (s), the inference time of the model on the entire testing dataset. For the ICCAD 2012-28 benchmark, comparative results are provided for the graph-based method DATE'22 [16], attention-based method TCAD'22 [13], binary residual network TCAD'21 [29], neural architecture search method DATE'23 [15], and LLM-based method DAC'24 [17]. For the ICCAD 2019 benchmark, comparative results from DATE'23, TCAD'22, and TCAD'21 are included, DATE'22 and DAC'24 did not provide test results on the ICCAD 2019 dataset. The proposed method

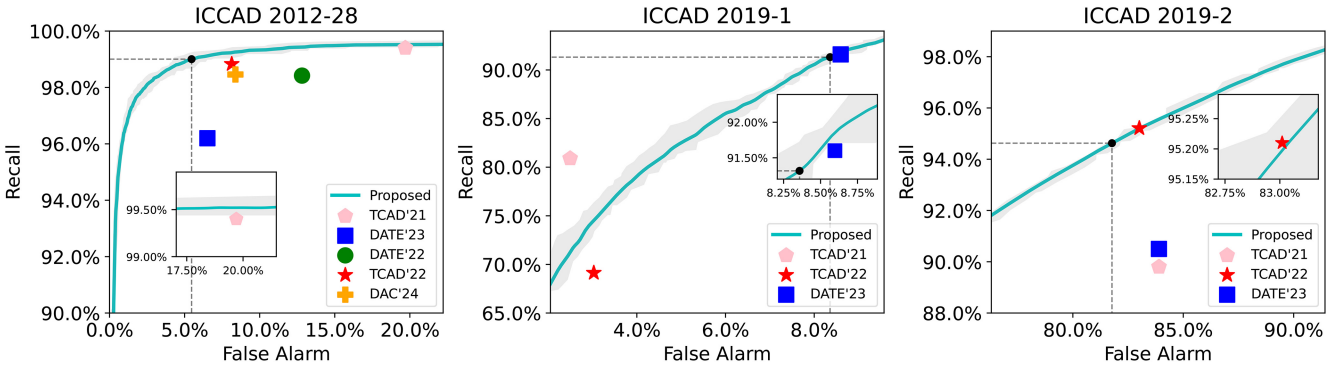

Fig. 7. Adjusting the classification threshold allows obtaining the recall of the model at different false alarm for intuitive comparison of different methods. The black dashed lines intersect at the recall and false alarm depicted in Table III. The gray area delineates the boundary encompassing multiple training results.

achieves 99.0% recall in the ICCAD 2012-28 dataset while maintaining the lowest false alarm among the compared methods, at 783. For the ICCAD 2019 dataset, the results in ICCAD 2019-1 indicate that the proposed method outperforms TCAD'22 and is comparable to DATE'23, while achieving the shortest inference time. In ICCAD 2019-2, the proposed method achieves a performance comparable to TCAD'22, with high recall at relatively low false alarms.

As different studies report varying recall and false alarm rates, comparing model performance becomes less intuitive, we adjust the classification threshold to achieve consistent recall performance across varying false alarm rates. By adjusting thresholds to generate different performance points, False Alarm–Recall curves are fitted to these points, and the performance of models from previous studies are plotted on the same figure. Visual comparison of recall at equal false alarm rates can be made by drawing lines perpendicular to the x-axis passing through the performance points of previous models. Hotspot detection requires maximizing recall while minimizing false alarm. In other words, the closer the points are to the top-left corner, the more effective the method. As shown in Fig. 7, the curve of the proposed method is positioned to the upper left of most points compared to the other methods, indicating that the proposed method demonstrates relatively superior performance in various false alarm levels.

The proposed method consistently achieves a higher recall in the ICCAD 2012-28 dataset at the same false alarm level compared to the previous methods. Compared to the previous graph learning method DATE'22, a recall improvement of more than 1% is observed at the same false alarm level. This improvement can be attributed to the more accurate use of both local and global features in the proposed method. The introduction of diverse types of nodes, such as block and space nodes, facilitates a more effective representation of features across various local regions. The proposed GNN model processes different features, such as node width and height, according to edge types, enabling more precise capture of local features. Additionally, the proposed global features facilitate more refined clustering of the layout, thereby enhancing the generalization of the model. The ICCAD 2019-1 dataset primarily tests the accuracy of the model to predict truly never-seen-before (TNSB) data [4]. The performance of the proposed method on this dataset is comparable to that of TCAD'22 and DATE'23 but slightly weaker than that of TCAD'21. The ICCAD 2019-2 dataset primarily tests the ability of the model to classify layouts with densely distributed state spaces [4]. The proposed method outperformed both TCAD'21 and DATE'23. Although the performance points of TCAD'22 are located slightly to the upper left of the cyan average performance curve, they fall within the gray area that represents the error range, indicating that the performance of TCAD'22 is comparable to the proposed method. This can largely be attributed to the proposed 3-hop GNN, which accurately captures local features, while the multilevel aggregation does not significantly impact the quality of the final learned graph embedding.

B. Runtime Analysis

Most of the compared works used inference time as a metric to evaluate model performance. Similarly, the inference time is utilized as an indicator of the performance of the proposed model in Table III. To further analyze the runtime composition of the proposed method, this section presents the time consumption of each step in the end-to-end process, from GDS and OAS file input to the determination of whether each clip is a lithography hotspot. It comprises several parts: generating the MTCG for each layout and extracting information, converting the extracted information into heterographs, loading the heterographs into memory, and the inference time.

The runtime analysis is shown in Fig. 8. The process of generating MTCG and extracting adjacency matrices and node features of heterographs is implemented in C++ with multithreaded optimization, resulting in relatively low time consumption: approximately 1.9 s for ICCAD 2012, 2.1 s for ICCAD 2019-1, and 22.3 s for ICCAD 2019-2. This aligns with the trend of linear growth relative to dataset size. DGL APIs are invoked in the Build stage to generate heterographs using this information, a process that can be parallelized. After generation, the heterographs are merged into larger heterographs in the Merge stage for parallel evaluation. In addition to the aforementioned time overhead, we measure the total inference time required to process all test data that have been loaded into memory. Observably, the runtime primarily revolves around invoking DGL APIs to generate heterographs. To further reduce the end-to-end runtime, benefiting from

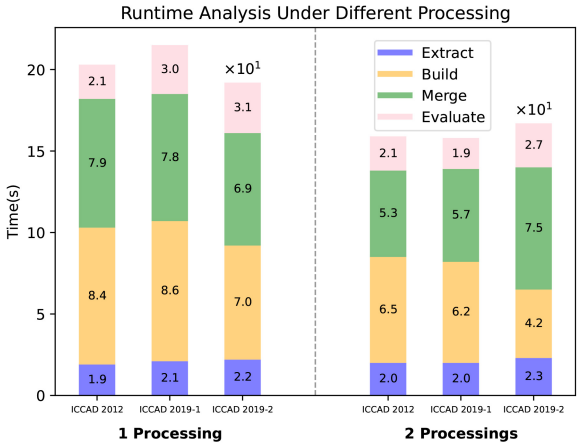

Fig. 8. Analysis of runtime on the testing benchmarks. The runtime on the larger ICCAD 2019-2 dataset is approximately 10× that of the other datasets.

the lightweight nature of the model, multiple models can be instantiated simultaneously on the experimental GPU. Leveraging multiprocessing for heterograph generation and model inference can further optimize the end-to-end runtime. In the multiprocessing runtime, the maximum time taken by each stage in each process is selected as the runtime for that stage.

C. Ablation Study

To more accurately evaluate and verify the robustness of the proposed method, ablation experiments are conducted from two perspectives: 1) model structure and 2) data augmentation methods. The ablation experiments for the model structure are further divided into those that target the GNN and those that focus on the features received by the classifier. Ablation experiments are conducted, as illustrated in Fig. 9, to validate the proposed model. These experiments remove specific components to assess their impact. Compared to the proposed method, Fig. 9(a) uses only 2-hop message passing, omitting the step of transmitting *df* between the nodes and their neighbors. In Fig. 9(b), *gf* of the graph is removed to verify its effectiveness. Fig. 9(c) combines Fig. 9(a) and (b), removing both 3-hop message passing and *gf* to investigate the mutually reinforcing relationship between these two types of features. Fig. 9(d)–(f) removes different parts of the global graph features individually to compare their effectiveness. The ablation experiments described above are referred to as remove 3-hop (RT), remove global features (RG), remove both 3-hop and global features (RTG), remove polygon features (RP), remove space features (RS), and remove both polygon and space features (RPS), respectively.

In addition to the aforementioned ablation experiments on the proposed method, the effectiveness of the proposed data augmentation methods is further validated. The data augmentation methods used are categorized into three types: 1) multiple dual-graphs; 2) modification of the heterograph structure; and 3) oversampling. The corresponding ablation settings for these three methods are as follows: 1) using Hc_v and Vc_h dual-graph as the original data, also Hc_h and Vc_v dual-graph as the augmented data, and oversampling applied

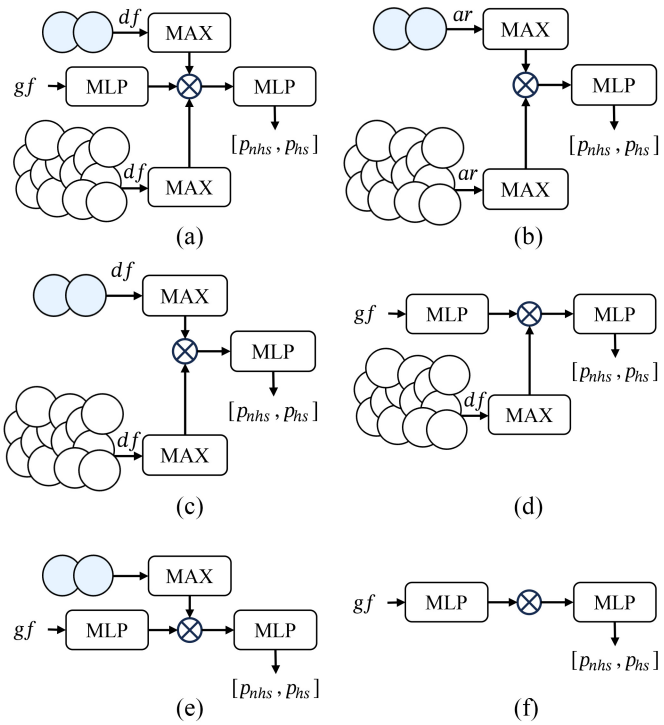

Fig. 9. Ablation experiment setup. (a) 3-hop message passing is removed, and *df* is used as the higher-order node feature. (b) Global graph features *gf* are removed. (c) Both 3-hop message passing and *gf* are removed. (d) Polygon-type nodes are removed. (e) Space-type nodes are removed. (f) Only *gf* is used.

as augmentation techniques on both dual-graphs; 2) only the dual-graph Hc_v and Vc_h are used as training data, with edge direction reversal, edge type modification, and oversampling as augmentation techniques; and 3) using Hc_v and Vc_h dual-graph with only oversampling. The results from these three approaches demonstrate that the proposed data augmentation methods can effectively address data imbalance issues and enhance the generalization capabilities of the model. The ablation experiments described above are referred to as both pair and oversampling (BPO), single pair and full augmentation (SPA), and single pair and oversampling (SPO), respectively.

To directly compare the proposed method with the results from the ablation experiments, the classification thresholds of methods are adjusted to fix the recall. The false alarm and inference times are then compared, as presented in Table IV. The results indicate that in all ablation settings, the false alarm increases, demonstrating the effectiveness of each component of the proposed method.

When the 3-hop convolution is removed in RT, there is a significant increase in false alarms, underscoring the importance of the 3-hop convolution. However, it is also time-consuming, with the 3-hop convolution accounting for 23%–42% of the inference time. After the removal of global features in RG, the impact on the false alarm for ICCAD 2012-28 and ICCAD 2019-2 is smaller than in RT, but the is opposite to ICCAD 2019-1. This suggests that the introduction of global features enhances the generalization ability of model without significantly affecting inference time. The removal of both 3-hop convolution and global features in RTG has an even more pronounced impact on performance. RP, which does not pass

TABLE IV
EXPERIMENT RESULTS OF ABLATION

Benchmarks	ICCAD 2012				ICCAD 2019-1				ICCAD 2019-2			
	Recall(%)	FA(%)	#FA	Time(s)	Recall(%)	FA(%)	#FA	Time(s)	Recall(%)	FA(%)	#FA	Time(s)
Origin	99.0	5.8	783	2.1	91.3	8.3	1222	2.6	94.6	81.8	53579	29.8
RT		14.6	1966	1.6		9.4	1380	1.5		85.6	56109	16.8
RG		8.4	1131	2.3		9.7	1415	2.5		84.2	55193	28.6
RTG		11.8	1598	1.2		10.9	1588	1.2		86.0	56328	15.4
RP		5.9	798	2.3		8.7	1268	2.5		85.1	55797	31.2
RS		12.1	1629	2.4		10.8	1587	2.6		84.7	55499	29.3
RPS		79.1	10678	<0.1		76.7	11219	<0.1		95.3	62443	<0.1
BPO		13.4	1817	2.6		9.0	1309	2.5		82.2	53844	31.5
SPA		6.4	865	2.6		8.5	1246	2.6		84.0	55032	29.2
SPO		8.7	1175	2.4		8.8	1287	2.9		82.6	54138	29.3

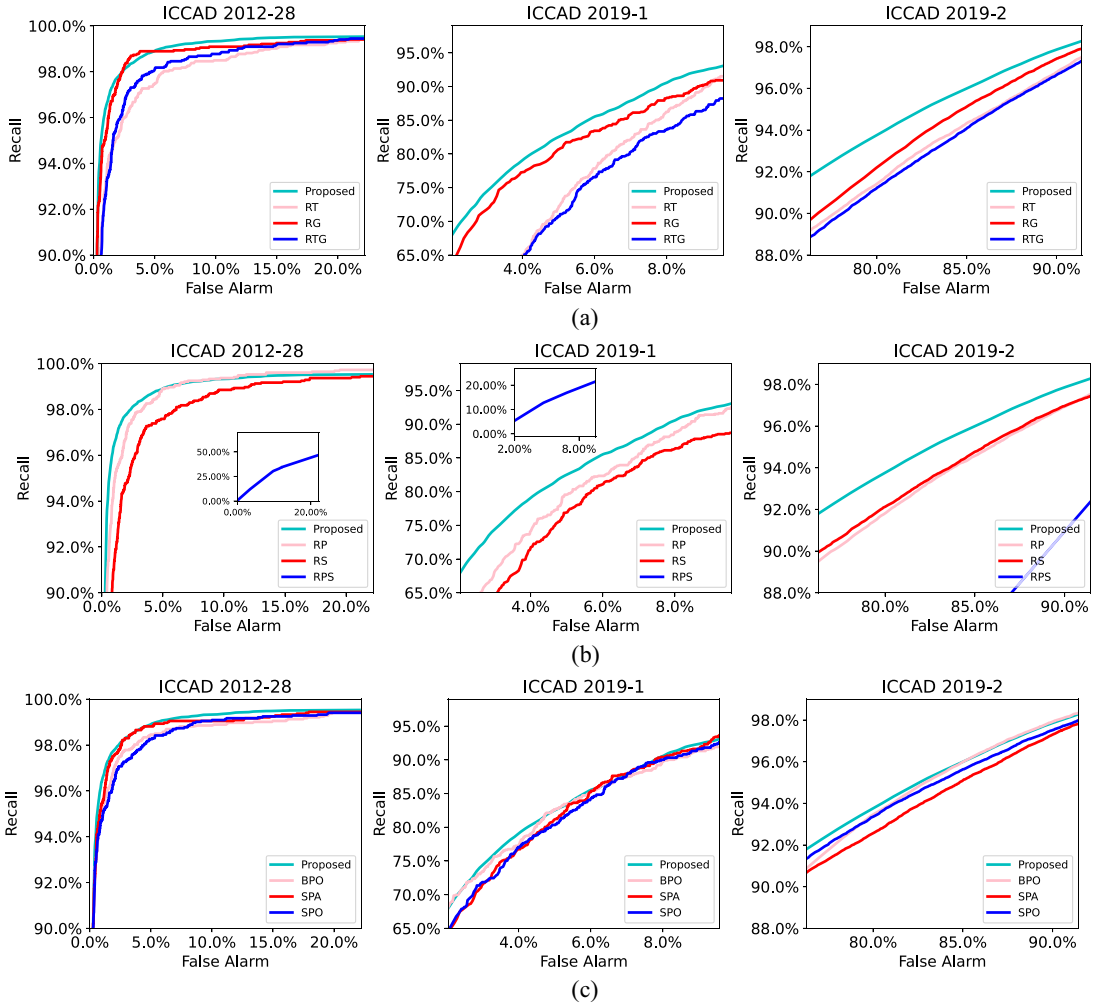

Fig. 10. Ablation experiment result. (a) Ablation experiments on the GNN architecture. (b) Ablation experiments on the classifier. (c) Ablation experiments on training data.

polygon node features to the classifier, has little effect on the results for ICCAD 2012 and ICCAD 2019-1 but significantly affects ICCAD 2019-2, where the model needs to identify minor layout differences accurately. The results of RS show that the features of space nodes also have a substantial impact on the performance of model. If only global features are used, the false alarm rises to an unacceptable level, further proving the effectiveness of the proposed graph embedding model.

In the ablation experiments for the proposed data augmentation method, BPO achieves a balance between hotspot and nonhotspot data in the training set by incorporating both Hc_h and Vc_v dual-graph and applying oversampling. SPA uses Vc_h and Hc_v dual-graph as training data, employing edge direction reversal, edge type modification, and oversampling for data augmentation. SPO only utilizes Vc_h and Hc_v dual-graph, with oversampling as the augmentation method. Fig. 10 presents the False Alarm–Recall curves for different ablation settings in various datasets. Fig. 10(a) illustrates the ablation experiments on the GNN architecture. With the exception of RG on ICCAD 2012-28, which partially overlaps with the

proposed method in the upper left corner, all other curves lie below it, further validating the effectiveness of the proposed GNN structure. Fig. 10(b) shows the ablation experiments for the features received by the classifier. Similarly, the majority of ablation experiment results fall below the curve of the proposed method. The results shown in Fig. 10(c) are also similar. For the ICCAD 2012 dataset, the comparison between the origin method and BPO, as well as between SPA and SPO, indicates that the proposed methods of edge reversal and edge type modification significantly impact the trained model. For the ICCAD 2019-1 test data, the effect of the training data on model performance is not substantial. However, for the ICCAD 2019-2 test data, using both Vc_h and Hc_v dual-graph alongside Hc_h and Vc_v dual-graph as training data more accurately captures the subtle differences in the layout.

V. Conclusion

This article introduces a method utilizing MTCG that converts layouts into heterographs, while simultaneously extracting global features to enhance model generalization. The resulting heterographs are processed with an efficient incremental message-passing GNN to learn embeddings and perform classification. In addition, data augmentation techniques leveraging the MTCG structure are proposed to mitigate data imbalance issues within the benchmarks. The experimental results demonstrate the superior performance of our method on the ICCAD 2012 benchmark, achieving 99.0% recall with only 783 false alarms. We further evaluate the generalization of our detector by the ICCAD 2019 benchmark, yielding satisfactory results. Moreover, our method significantly reduces inference times on both benchmarks.

References

[1] 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 Design Autom. Conf.*, 2012, pp. 1163–1168.

[2] Y.-T. Yu, I. H.-R. Jiang, Y. Zhang, and C. Chiang, "DRC-based hotspot detection considering edge tolerance and incomplete specification," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 101–107.

[3] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2012, pp. 349–350.

[4] G. R. Reddy, K. Madkour, and Y. Makris, "Machine learning-based Hotspot detection: Fallacies, pitfalls and marching orders," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2019, pp. 1–8.

[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," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 460–470, Mar. 2015.

[6] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan, "Semisupervised hotspot detection with self-paced multitask learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 7, pp. 1511–1523, Jul. 2020.

[7] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. 24th Asia South Pac. Design Autom. Conf.*, 2019, pp. 299–304.

[8] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[9] J. Chen, Y. Lin, Y. Guo, M. Zhang, M. B. Alawieh, and D. Z. Pan, "Lithography hotspot detection using a double inception module architecture," *J. Micro/Nanolithogr., MEMS, MOEMS*, vol. 18, no. 1, 2019, Art. no. 13507.

[10] X. He, Y. Deng, S. Zhou, R. Li, Y. Wang, and Y. Guo, "Lithography hotspot detection with FFT-based feature extraction and imbalanced learning rate," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 2, pp. 1–21, 2019.

[11] 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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1175–1187, Jun. 2019.

[12] B. Zhu et al., "Hotspot detection via multi-task learning and transformer encoder," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2021, pp. 1–8.

[13] H. Geng, H. Yang, L. Zhang, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2685–2698, Aug. 2022.

[14] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via neural architecture search," *ACM Trans. Design Autom. of Electron. Syst.* , vol. 27, no. 6, pp. 1–16, 2022.

[15] Z. Chen, F. Yang, L. Shang, and X. Zeng, "Automated and agile design of layout hotspot detector via neural architecture search," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2023, pp. 1–6.

[16] S. Sun, Y. Jiang, F. Yang, B. Yu, and X. Zeng, "Efficient hotspot detection via graph neural network," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2022, pp. 1233–1238.

[17] Y. Chen et al., "LLM-HD: Layout language model for hotspot detection with GDS semantic encoding," in *Proc. 61st ACM/IEEE Design Autom. Conf.*, 2024, pp. 1–6.

[18] S. Connor and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[19] J. Zhou et al., "Graph neural networks: A review of methods and applications," 2021, *arXiv:1812.08434*.

[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019, *arXiv:1810.00826*.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018, *arXiv:1710.10903*.

[22] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for non-slicing floorplans," in *Proc. 38th Design Autom. Conf.*, 2001, pp. 764–769.

[23] H. Yang, Y. Lin, B. Yu, and E. F. Y. Young, "Lithography hotspot detection: From shallow to deep learning," in *Proc. 30th IEEE Int. Syst.-Chip Conf. (SOCC)*, 2017, pp. 233–238.

[24] V. Borisov and J. Scheible, "Lithography hotspots detection using deep learning," in *Proc. 15th Int. Conf. Synthesis, Model., Anal. Simul. Methods Appl. Circuit Design (SMACD)*, 2018, pp. 145–148.

[25] V. M. Panaretos and Y. Zemel, "Statistical aspects of Wasserstein distances," *Annu. Rev. Statist. Appl.*, vol. 6, no. 1, pp. 405–431, 2019.

[26] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: A deep learning approach," *J. Micro/Nanolithogr., MEMS, MOEMS*, vol. 16, no. 3, 2017, Art. no. 33504.

[27] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," 2019, *arXiv:1912.01703*.

[28] M. Wang et al., "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2020, *arXiv:1909.01315*.

[29] Y. Jiang, F. Yang, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network ensemble," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 7, pp. 1476–1488, Jul. 2021.