You Only Need Non-hotspot: An Unsupervised Training-Free Method for Layout Hotspot Detection

SILIN CHEN, School of Integrated Circuits, Nanjing University, Suzhou, China
KANGJIAN DI, School of Integrated Circuits, Nanjing University, Suzhou, China
YIBO HUANG, School of Integrated Circuits, Nanjing University, Suzhou, China
BINWU ZHU, Southeast University, Nanjing, China
NINGMU ZOU*, School of Integrated Circuits, Nanjing University, Suzhou, China and Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou, China

Recent advances in deep learning-based layout hotspot detection have made remarkable progress in identifying potential defect patterns at early design stages. However, most existing methods rely on supervised learning, which requires manual identification of pre-defined hotspots and leads to considerable labeling effort. Moreover, design houses often struggle to obtain a sufficient number of labeled hotspot samples, limiting the applicability and scalability of such methods. In this article, we introduce a novel approach, termed you only need non-hotspot (YONN), which to the best of our knowledge, is the first unsupervised and training-free framework for layout hotspot detection. The proposed method mitigates the dependence on labeled hotspot data by leveraging memorized prototypes and a query-based inference mechanism. Specifically, YONN employs a CNN-based prototype generation network to extract multi-scale, fine-grained representations of layouts. During inference, a combination of shape-aware and topology-aware query mechanisms facilitates precise pixel-wise matching between test layout and memorized prototypes. To further enhance YONN's efficiency and scalability, we propose a prototype sampling strategy that integrates density-based clustering techniques, significantly reducing the scale of the prototypes. Experimental results indicate that YONN achieves performance within 10% of leading state-of-the-art supervised learning methods, despite operating in a fully unsupervised setting without access to hotspot data. As an optional extension, YONN surpasses existing state-of-the-art approaches using only 30% hotspot labels. Notably, YONN is a training-free framework that enables on-the-fly adaptation by directly incorporating novel samples into the prototype bank, thereby supporting efficient and scalable learning within design for manufacturability workflows.

CCS Concepts: • **Hardware → Design for manufacturability**; **Methodologies for EDA**; • **Computing methodologies → Artificial intelligence**.

Additional Key Words and Phrases: Lithographic hotspot detection, deep learning, unsupervised learning

*Corresponding author: nzou@nju.edu.cn

Authors' Contact Information: Silin Chen, School of Integrated Circuits, Nanjing University, Suzhou, Jiangsu, China; e-mail: silin.chen@smail.nju.edu.cn; Kangjian Di, School of Integrated Circuits, Nanjing University, Suzhou, Jiangsu, China; e-mail: kangjiandi@smail.nju.edu.cn; Yibo Huang, School of Integrated Circuits, Nanjing University, Suzhou, Jiangsu, China; e-mail: yibohuang@smail.nju.edu.cn; Binwu Zhu, Southeast University, Nanjing, Jiangsu, China; e-mail: bwzhu@seu.edu.cn; Ningmu Zou, School of Integrated Circuits, Nanjing University, Suzhou, Jiangsu, China and Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou, Jiangsu, China; e-mail: nzou@nju.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1557-7309/2025/10-ART
https://doi.org/10.1145/3771767

Just Accepted

1 Introduction

The continuous scaling of transistor feature sizes and the increasing complexity of integrated circuits have introduced significant challenges in Design for Manufacturability (DFM). Among various DFM tasks, lithographic hotspot detection plays a critical role in identifying potentially defective layout patterns at the early stages of the design flow. Traditional approaches to hotspot detection primarily rely on lithography simulations, pattern matching techniques, and more recently, machine learning (ML)-based methods [54]. Lithography simulation methods aim to accurately model the photolithographic process using detailed mathematical and physical representations, enabling precise identification of lithographic hotspots during design validation [31]. However, full-chip lithography simulations are computationally intensive and require extensive knowledge of process and design rules, making them less practical for modern and rapidly evolving technology nodes [34, 38]. In contrast, pattern matching techniques [4, 19, 50, 57] attempt to detect hotspots by comparing layout regions against a pre-defined library of known hotspot patterns. Although these methods are computationally more efficient, they suffer from key limitations: constructing a comprehensive and representative hotspot library is inherently challenging, and the efficacy of pattern matching depends heavily on the quality of feature selection and matching strategies employed [20].

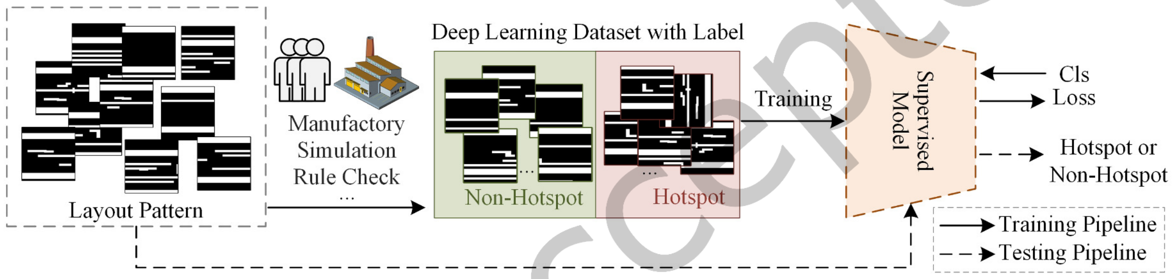

Fig. 1. Illustration of the current DL-based methods. To develop an effective detection model, a labeled dataset comprising both hotspot and non-hotspot instances must first be constructed.

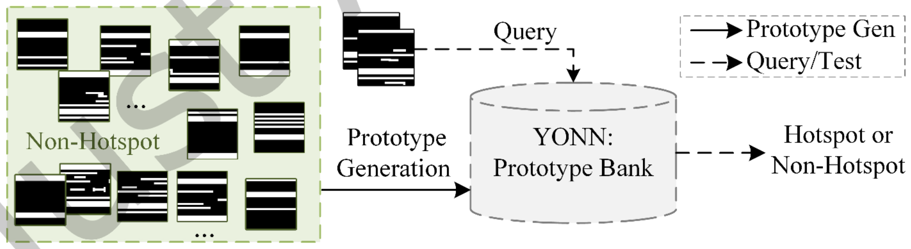

Fig. 2. Illustration of our method, which operates without requiring any labeled hotspot patterns and training.

In recent years, ML and deep learning (DL) techniques have been increasingly used in DFM applications [6, 48, 60], demonstrating significant advancements in layout hotspot detection [9, 18, 22, 51, 59, 61]. Current DL-based approaches to layout analysis typically convert layout patterns into representations suitable for specific neural network architectures. A common choice is to render the layout into images and apply convolutional neural networks (CNNs) for feature extraction [5, 14, 15, 21, 28, 33, 45]. However, due to the high spatial resolution of the layout pattern (e.g. 1 nm), the resulting image representations are extremely large (e.g. 4800×4800). To

Just Accepted

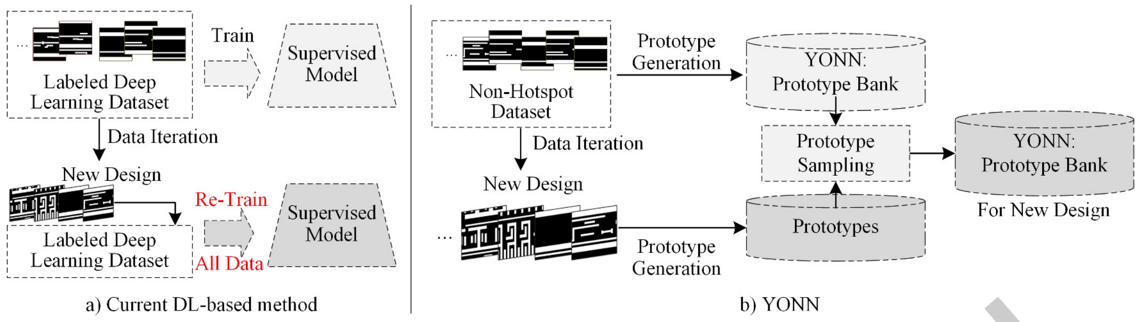

Fig. 3. Comparison between existing DL-based method and our method for handling new design updates.

Just Accepted

accommodate the input constraints of CNNs, existing methods often resize these images, resulting in a significant loss of critical spatial detail. To address this challenge, several methods adopt the squish pattern representation [16] to capture the layout topology, integrating it with CNN-based methods [11, 26, 55, 56]. The squish pattern mitigates the sparsity inherent in traditional image-based representations, significantly reducing the input resolution and information loss. In parallel, other research efforts have explored graph-based representations of layouts, enabling the application of graph neural networks (GNNs) to capture both the structural and relational aspects of layout pattern polygons [10, 23, 39, 52]. However, these methods necessitate the decomposition of polygons into multiple rectangular components, which increases computational complexity. With the rapid advancements in language models, a novel approach [9, 49] has emerged that represents layout information as sequences of language tokens. In this setting, polygons are encoded by category, position, and shape, and a language model is used to classify the resulting sequence representation of a layout clip. While this offers a promising alternative, it is resource-intensive and depends on prior knowledge for polygon categories[52]. Accordingly, this article employs the adaptive squish pattern as a foundational component in the construction of the prototype bank.

Moreover, all the methods mentioned above rely on supervised learning to model a decision boundary that distinguishes between hotspot and non-hotspot samples. As shown in Figure 1, constructing an effective training dataset is crucial for developing a robust classification model. This dataset must encompass both hotspot and non-hotspot patterns to ensure comprehensive learning. However, in practical design workflows, only a limited subset of layout hotspots, which are directly correlated with yield degradation, are reported back to design houses after fabrication. This delayed feedback significantly increases the time required to build a representative and high quality training dataset. While rule-based checking and lithography simulation can identify certain hotspot patterns, the number of such patterns remains significantly smaller than that of non-hotspot samples. This significant class imbalance poses a challenge for model training, often leading to biased or suboptimal learning results [15, 40]. Furthermore, as products are frequently iterated and updated within design houses, the corresponding layout files are continuously revised. In most existing hotspot detection, as shown in Figure 3(a), each new iteration requires a new model to be retrained from scratch. At the early stage of a new technology node, the availability of labeled hotspot and non-hotspot data is often severely limited[7]. Given the continuous accumulation of data and repeated cycles of product iterations, these methods become increasingly inefficient and unsustainable over time.

In this article, to overcome the practical limitations associated with existing lithographic hotspot detection methods, we propose a novel hotspot detection framework, termed you only need non-hotspot (YONN). In contrast to prior methods, YONN can do without hotspot-labeled samples during model development, relying exclusively on non-hotspot data to achieve accurate and robust hotspot detection. To better align with practical deployment scenarios where a limited number of hotspot labels may become available, we incorporate an optional refinement that leverages these labels to further enhance detection performance. Furthermore, YONN supports on-the-fly adaptation during design iterations, as it operates without any training requirements. As shown in Figure 2, YONN introduces a memorized prototype bank and a query-based inference mechanism to enable unsupervised, training-free hotspot detection. At its core, YONN employs an adaptive squish pattern representation to model layout patterns, coupled with a fine-grained prototype generation network for extracting multi-level prototype features. During inference, a shape-aware query and a topology-aware query are used to evaluate the test sample against the memorized prototype bank, enabling reliable classification of hotspots and non-hotspots. To mitigate storage overhead and computational complexity, we introduce a prototype sampling strategy based on clustering techniques, which significantly reduces the size of the prototype bank while maintaining competitive detection performance. Furthermore, as shown in Figure 3(b), YONN accommodates design revisions efficiently. When encountering new design updates, it simply extracts prototypes from updated non-hotspot samples and integrates them into the existing prototype bank, thus ensuring scalable and adaptive hotspot detection throughout the design process. While our primary formulation assumes no access to labeled hotspot data, YONN supports an optional refinement for scenarios with a limited hotspot labeling. Specifically, YONN constructs a supplementary hotspot prototype bank and integrates its scores with those from the non-hotspot bank to further boost performance. The main contributions of this article are summarized as follows:

- To the best of our knowledge, YONN is the first approach to tackle layout hotspot detection in an unsupervised and training-free manner. In addition, it explicitly addresses the challenge of adapting the hotspot detector in response to iterative design modifications. Without requiring any pre-labeled hotspot data, YONN demonstrates a preliminary capability to distinguish between hotspot and non-hotspot patterns. When a limited amount of hotspot data is available, YONN further enhances its discrimination performance, achieving more accurate classification between hotspot and non-hotspot samples.
- We introduce a memorized prototype bank to store representative feature embeddings of patterns. To enhance granularity during the subsequent query phase, we introduce a fine-grained prototype generation network that preserves multi-scale and pixel-wise prototype representations, thereby improving the precision and robustness of the detection process.
- To further improve the reliability of the query stage, we propose two query strategies, which complement the shape-aware query with a topology-aware query operation. It is particularly effective in improving the detection accuracy of challenging hotspot instances.
- To improve scalability and reduce computational overhead, we propose a clustering-based prototype sampling strategy that reduces the size of the prototype bank by up to 5×, while maintaining competitive detection performance.

The remainder of this article is structured as follows. Section 2 introduces the foundational concepts of unsupervised hotspot detection and presents the layout representation employed by YONN. Section 3 describes the proposed algorithm, including the procedures for prototype generation and query formulation. Section 4 reports the experimental results obtained from three benchmark datasets, and Section 5 concludes the article.

2 Preliminary

2.1 Problem Formulation

The lithographic process uses a photomask to precisely transfer circuit layout patterns onto semiconductor wafers. However, certain regions within the layout, known as lithographic hotspots, are highly sensitive to process variations that can significantly affect manufacturing yield. [10]. Therefore, the accurate identification and effective mitigation of these hotspots are essential to enhance design robustness and ensure manufacturability.

Just Accepted

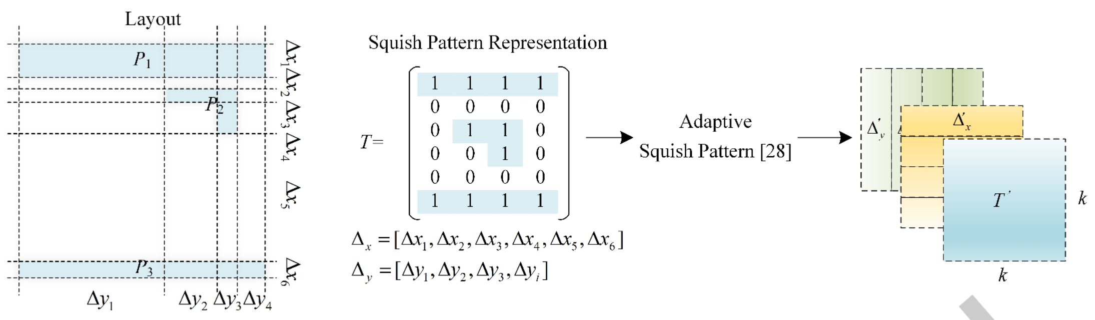
Layout
P_1
P_2
P_3
Δx_1 Δx_2 Δx_3 Δx_4 Δx_5 Δx_6
Δy_1 Δy_2 Δy_3 Δy_4
Squish Pattern Representation
$T=\begin{bmatrix}1&1&1&1\\0&0&0&0\\0&1&1&0\\0&0&1&0\\0&0&0&0\\1&1&1&1\end{bmatrix}$
$\Delta_x=[\Delta x_1,\Delta x_2,\Delta x_3,\Delta x_4,\Delta x_5,\Delta x_6]$
$\Delta_y=[\Delta y_1,\Delta y_2,\Delta y_3,\Delta y_i]$
Adaptive Squish Pattern [28]
Δ_y'
Δ_x'
T'
k
k

Fig. 4. Illustration of adaptive squish pattern representation.

Just Accepted

Hotspot detection is typically formulated as a binary classification problem that aims to distinguish between hotspot and non-hotspot layout clips. In the early pre-silicon design stages, verified hotspot labels are often unavailable, whereas large volumes of non-hotspot data can be collected efficiently. Furthermore, rapid design iterations and engineering change orders (ECOs) frequently produce a wide array of new layout variants without corresponding hotspot labels. This leads to a scenario where hotspot feedback is both limited and delayed, thereby motivating the development of detection methods that can learn effectively using only non-hotspot data. In this article, we propose a novel unsupervised hotspot detection framework that performs hotspot classification using only non-hotspot data. Compared to supervised approaches, the proposed method significantly reduces data requirements and offers greater flexibility for adaptation to future design updates. The following definitions and metrics are used to evaluate the performance of an unsupervised hotspot detector.

Definition 1 (Accuracy). Accuracy (ACC) is defined as the proportion of correctly classified hotspot instances to the total number of ground truth hotspot instances. Formally,

$$ACC = \frac{|TP|}{|TP| + |FN|}, \tag{1}$$

where $|TP|$ denotes the number of true positives (correctly identified hotspots), and $|FN|$ denotes the number of false negatives (missed hotspots).

Definition 2 (False Alarm Rate). False Alarm Rate (FAR) refers to the proportion of non-hotspot instances incorrectly classified as hotspots. It is computed as

$$FAR = \frac{|FP|}{|FP| + |TN|}, \tag{2}$$

where $|FP|$ denotes the number of false positives (non-hotspots classified as hotspots), and $|TN|$ denotes the number of true negatives (correctly identified non-hotspots).

With these metrics, we formulate the unsupervised hotspot detection problem as follows:

Problem 1 (Unsupervised Hotspot Detection). Given a dataset composed primarily or exclusively of layout clips containing non-hotspot patterns, our objective is to develop a detector that maximizes detection accuracy while minimizing the false alarm rate, despite having no or very few confirmed hotspot examples for training.

2.2 Layout Representation

During the design phase, layout files are typically composed of a series of polygons. However, large-scale layouts tend to be spatially sparse, resulting in significant computational overhead and an increased risk of overfitting when processed by DL-based models. To address these challenges, the squish pattern layout representation [16],

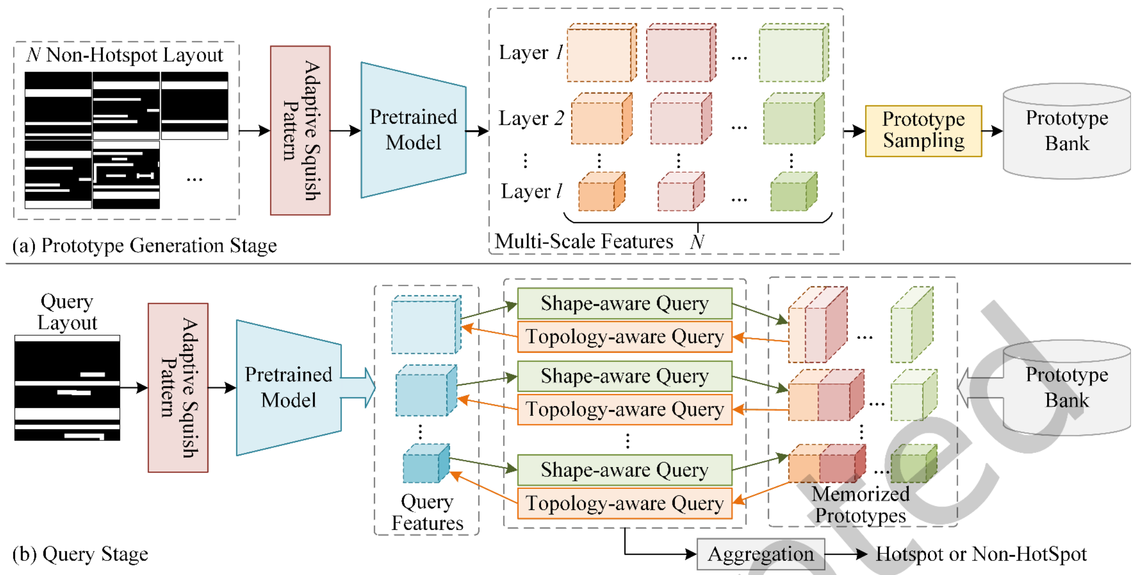

Fig. 5. Overview of YONN, consisting of the prototype generation stage and query stage.

Just Accepted

illustrated in Figure 4, transforms the layout into a topological matrix T along with associated geometric vectors Δ_x and Δ_y. This representation reduces computational complexity while preserving critical design features. Specifically, the squish pattern discretizes the layout space by meshing along polygon edges, producing a binary topological matrix in which each element indicates the presence or absence of a polygonal feature. Simultaneously, the physical distances between adjacent sweep lines in both the x- and y-directions are encoded into the geometric vectors. The resulting topological matrices and geometric vectors are variable in length, depending on layout complexity. To enable compatibility with standard DL-based models, which typically require fixed-size inputs, we adopt the adaptive squish pattern strategy proposed in [55]. This approach normalizes the representation by padding the geometric vectors, thereby converting variable-size patterns into fixed-dimension square matrices. This facilitates seamless integration with convolutional and transformer-based neural network architectures while maintaining fidelity to the original layout information.

3 Algorithms

3.1 Overview

As shown in Figure 5, the YONN architecture is specifically developed to enable unsupervised, training-free hotspot detection via a two-stage process: prototype generation and query evaluation. In the prototype generation stage, the model receives a dataset consisting of N non-hotspot layout patterns. These input patterns are first converted into adaptive squish representations to ensure consistent formatting, denoted as $\{X_n\}_{n=1}^{N} \in \mathbb{R}^{N\times3\times k\times k}$, where k represents the resolution of the squish-transformed pattern. YONN then employs a frozen, pre-trained convolutional neural network $\mathcal{F}(\cdot)$ to extract hierarchical features from the inputs. The resulting feature maps at layer l are expressed as $F^l \in \mathbb{R}^{N\times C_l\times W_l\times H_l}$, where C_l, W_l, and H_l correspond to the number of channels, width, and height of the feature maps at the l-th layer, respectively. Consistent with prior findings[27, 35] that subtle geometric modifications can determine hotspot presence, YONN retains all pixel-level features in the prototype bank to comprehensively represent normal patterns and account for minor variations. To reduce

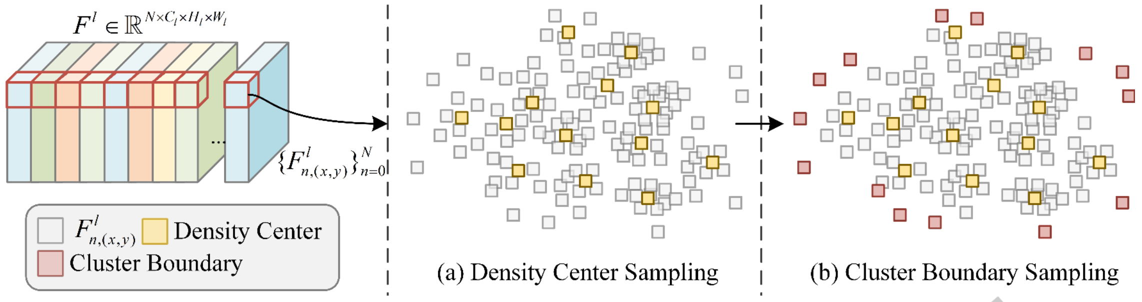
$F^l \in \mathbb{R}^{N \times C_l \times H_l \times W_l}$
$\{F^l_{n,(x,y)}\}^N_{n=0}$
$F^l_{n,(x,y)}$
Density Center
Cluster Boundary
(a) Density Center Sampling
(b) Cluster Boundary Sampling

Fig. 6. Illustration prototype sampling strategy.

storage and computational overhead during the query stage, a prototype sampling strategy is employed. This strategy selectively stores a subset of features $P^l \in \mathbb{R}^{K \times C_l \times W_l \times H_l}$ from each layer into a prototype bank, controlled by a sampling parameter K.

During the query stage, as shown in Figure 5(b), a query layout X_q is processed through the same CNN $\mathcal{F}(\cdot)$ to obtain hierarchical features $\{Q^l\}^L_{l=1}$, where $Q^l \in \mathbb{R}^{C_l \times W_l \times H_l}$ corresponds to the feature representation at layer l. Similar to prototype generation, pixel-level features from the query are compared with those in the prototype bank. A set of hierarchical hotspot likelihood maps $\{M^l\}^L_{l=1}$ is then computed using a combination of shape-aware and topology-aware query mechanisms that compare Q^l against the stored prototypes P^l across all layers and spatial positions. The final hotspot decision is made by aggregating the multi-layer hotspot maps using an aggregation function $\text{Agg}(\cdot)$. A query layout is classified as a hotspot if the aggregated score exceeds a predefined hotspot threshold th:

$$\text{Agg}(\{M^l\}^L_{l=1}) \geq th, \tag{3}$$

where L is the total number of CNN layers used for evaluation. This approach enables YONN to construct a prototype-based decision boundary that effectively separates hotspot and non-hotspot patterns without requiring supervised training.

3.2 Prototype Sampling Strategy

As shown in Figure 5(b), a key criterion in determining whether a query layout is a hotspot is the distance between its feature distribution and the feature distribution in the prototype bank. Intuitively, a prototype bank that captures a more comprehensive and representative distribution is expected to get higher detection accuracy during the query stage. However, modern chip layouts in advanced design environments contain a large number of complex and densely packed polygons. This results in an extremely large and diverse set of non-hotspot layout patterns. Consequently, maintaining prototypes for the full range of non-hotspot patterns incurs prohibitive storage costs, while the computational burden of query time similarity evaluations becomes impractical.

To address this challenge, we propose a prototype sampling strategy that effectively reduces the scale of the prototype bank while preserving the representational prototypes. Specifically, as shown in Figure 6, given the feature maps F^l extracted from the l-th layer of the pre-trained network $\mathcal{F}(\cdot)$, we perform pixel-wise sampling to generate refined prototypes. Let the features of N layouts at spatial coordinates (x, y) be represented as $\{F^l_{n,(x,y)}\}^N_{n=1}$. To extract representative prototypes, we employ the HDBSCAN clustering algorithm [3] on these pixel-wise features. HDBSCAN is particularly well-suited to this task due to its ability to discover clusters of varying densities without requiring the number of clusters to be specified a priori. Let $S^l_{(x,y)}$ denote the set of all clusters of HDBSCAN, $s_i \in S^l_{(x,y)}$ denote the i-th cluster identified by HDBSCAN. For each cluster s_i, we compute

Just Accepted

the representative center prototype $p_i \in P^l_{(x,y)}$ from its high-density cluster as follows:

$$p_i = \underset{F \in s_i}{\arg\max}\Big(\sum_{F' \in s_i} \mathrm{Sim}(F, F')\Big), \tag{4}$$

where $P^l_{(x,y)}$ is the collected center prototypes after density center sampling and $\mathrm{Sim}(\cdot,\cdot)$ is the similarity function defined as:

$$\mathrm{Sim}(\mathrm{X}, \mathrm{Y}) = \frac{\mathrm{X} \cdot (\mathrm{Y})^{\mathrm{T}}}{\|\mathrm{X}\| \cdot \|\mathrm{Y}\|}. \tag{5}$$

Furthermore, to account for edge cases where the distribution of non-hotspot features is highly sparse or exhibits substantial heterogeneity, we incorporate a fallback mechanism that guarantees prototype coverage. Specifically, when HDBSCAN fails to identify valid clusters at a given spatial coordinate (i.e., $|S^l_{(x,y)}| = 0$), the algorithm defaults to a global feature aggregation strategy. In this case, all pixel features $\{F^l_{n,(x,y)}\}_{n=1}^N$ are aggregated, and the representative prototype is selected using the same similarity-based criterion defined in Equation (5). This ensures that representative prototypes are still retained from the entire pixel-wise feature, preventing information loss in under-represented regions of the feature space.

As shown in Figure 2(a), selecting only the densest center from each cluster as a representative prototype may lead to an overly compact decision boundary. This compactness poses a risk of misclassification, especially in scenarios where inter-cluster variance is minimal and peripheral prototypical features are excluded from representation. Hence, to address this limitation and enhance the completeness of the prototype representation, we design a secondary sampling mechanism that samples additional prototypes from cluster boundaries. Specifically, given the set of initial prototypes $P^l_{(x,y)}$, we iteratively identify the pixel-wise features that are furthest from the current prototypes and add them to the prototype bank. The additional prototypes p' are computed as:

$$p' = \underset{F \in \overline{P^l_{(x,y)}}}{\arg\min}\Big(\sum_{F' \in P^l_{(x,y)}} \mathrm{Sim}(F, F')\Big), \tag{6}$$

where $\overline{P^l_{(x,y)}}$ denotes the set of candidate features not already included in the center prototype bank. This process continues until the prototype bank reaches the desired size K. This auxiliary sampling process effectively expands the decision boundary coverage and improves the YONN's ability to discriminate between borderline hotspot and non-hotspot patterns. The detail of the prototype sampling process is in Algorithm 1.

3.3 Query Mechanisms

In the query stage, we utilize the prototype bank to construct a robust decision boundary. Given the hierarchical query features $\{Q^l\}_{l=1}^L$ extracted by the frozen CNN $\mathcal{F}(\cdot)$, and the locally stored high-quality prototype bank $\{P^l\}_{l=1}^L$, we perform a layer-wise and pixel-wise similarity comparison to classify the query layouts. Specifically, during the design phase, preserving shape information and layout topology is critical for accurately identifying potential layout hotspots under specific design rules [48, 53]. Each prototype feature stored at a given coordinate in the prototype bank retains the shape characteristics within its receptive field. To effectively exploit this, we propose a shape-aware query mechanism that matches the query vectors at each position of the query features to their corresponding prototypes based on local feature similarity. Furthermore, YONN incorporates a topology-aware query mechanism to improve the detection of topology hotspots. In this framework, the features of the four-neighbor prototypes are utilized to match against the corresponding four-neighbor search regions within the query features, thereby enabling robust topology-aware classification at each query location.

Just Accepted

Algorithm 1 Prototype Sampling Process

```
Require: The features $\{F^l_{n,(x,y)}\}^N_{n=1}$ of $N$ non-hotspot layouts at $(x,y)$, the desired size $K$ of final prototype bank;
Ensure: The collected prototypes bank $P^l_{(x,y)}$;
1: Initialize $P^l_{(x,y)} \leftarrow \{\}$;
2: $S^l_{(x,y)} \leftarrow HDBSCAN(\{F^l_{n,(x,y)}\}^N_{n=1})$;
3: if $|S^l_{(x,y)}| = 0$ then
4:     $S^l_{(x,y)} \leftarrow \{F^l_{n,(x,y)}\}^N_{n=1}$;
5: end if
6: for each $s_i$ in $S^l_{(x,y)}$ do
7:     $p_i \leftarrow$ Computing the representative prototype in $s_i$ by Equation(4)(5);
8:     $P^l_{(x,y)} \leftarrow P^l_{(x,y)} \cup \{p_i\}$;
9: end for
10: while $|P^l_{(x,y)}| < K$ do
11:     $p' \leftarrow$ Computing the boundary prototype by Equation(6);
12:     $P^l_{(x,y)} \leftarrow P^l_{(x,y)} \cup \{p'\}$;
13: end while
14: return $P^l_{(x,y)}$.
```

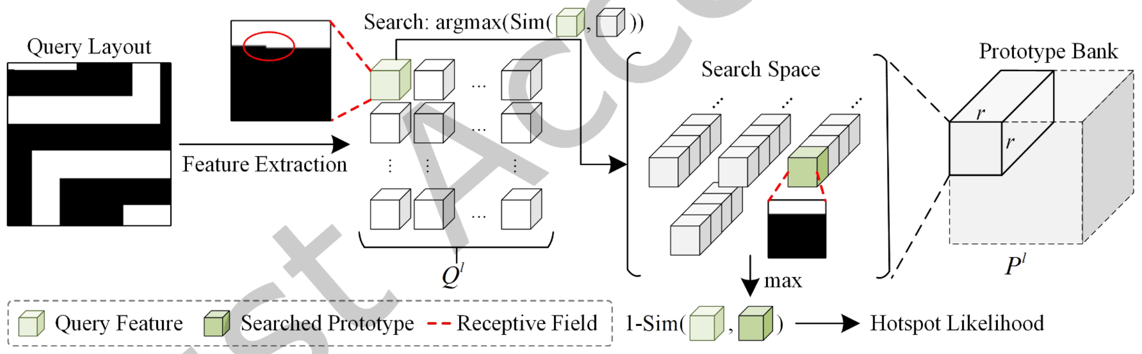

Fig. 7. Illustration of shape-aware query mechanism.

Just Accepted

3.3.1 *Shape-aware Query.* As shown in Figure 7, given a query vector $Q^l_{(x,y)}$ with coordinates (x, y) in the l-th layer query feature and the corresponding prototype feature P^l at the same layer, the shape-aware query mechanism aims to identify the most similar prototype and subsequently compute the hotspot likelihood map. Notably, recognizing that layout patterns are location-independent, we introduce a radius r to expand the search space during the query process. This expansion mitigates potential mismatches caused by location-fixed prototype comparisons. The shape-aware search space $\Omega^{P^l}_{(x,y)}$ is mathematically defined as follows:

$$\Omega^{P^l}_{(x,y)} = \{p_i \in P^l_{(x+a,y+b)} \mid a \le r, b \le r\}, \tag{7}$$

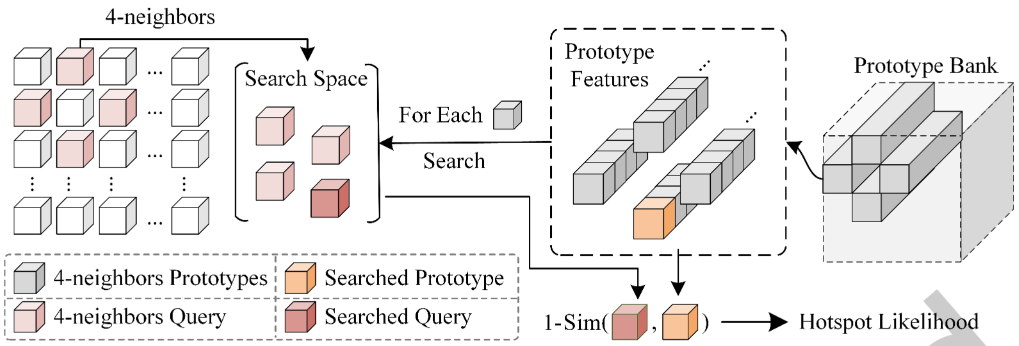

Fig. 8. Illustration of topology-aware query mechanism.

where $a, b \in \mathbb{Z}$ and p_i denotes the i-th prototype in $P^l_{(x+a,y+b)}$. The most similar prototype to $Q^l_{(x,y)}$ is selected based on a similarity computation, and the corresponding hotspot likelihood score is computed as:

$$M^l_{s,(x,y)} = 1 - \max_{p \in \Omega^{P^l}_{(x,y)}} (\mathrm{Sim}(Q^l_{(x,y)}, p)), \tag{8}$$

where $M^l_{s,(x,y)}$ represents the hotspot likelihood score at position (x, y) in the shape-aware likelihood map M^l_s. As shown in Figure 7, the receptive fields of the query vector and the corresponding searched prototypes allow for the recognition of small shape deformations, thereby enhancing robustness to layout variations.

3.3.2 Topology-aware Query. In the prototype search space, the vectors retrieved by the shape-aware query consider only the legality constraints associated with the query's shape, without incorporating information regarding the local topological context. To address this limitation, YONN introduces a topology-aware query mechanism. This mechanism defines a set of four-neighbor prototypes $\mathcal{N}^{P^l}_4(x, y)$ centered at the (x, y) coordinate within the prototype bank:

$$\mathcal{N}^{P^l}_4(x, y) = \{p_i \in P^l_{(x+a,y+b)} \mid |a| \leq 1, |b| \leq 1\}. \tag{9}$$

Subsequently, an inverse search procedure is conducted, mapping from prototype features back to query features, to identify the vector within the four-neighborhood of a given query vector that exhibits the highest similarity to the prototype. The local search space is formally defined as:

$$\Omega^{Q^l}_{(x,y)} = \{Q^l_{(x+a,y+b)} \mid |a| \leq 1, |b| \leq 1\}. \tag{10}$$

This inverse search operation enforces local topological consistency by validating whether the query vector aligns with its neighboring prototypes. Based on the best-matching neighbor, the hotspot likelihood score is then computed as:

$$M^l_{t,(x,y)} = \min_{Q^l_{(x,y)} \in \Omega^{Q^l}_{(x,y)}} (1 - \max_{p \in \mathcal{N}^{P^l}_4(x,y)} (\mathrm{Sim}(Q^l_{(x,y)}, p))), \tag{11}$$

where $M^l_{t,(x,y)}$ denotes the topology-aware hotspot likelihood at location (x, y) within the likelihood map M^l_t.

3.3.3 Aggregation. To improve hotspot detection accuracy, we integrate the shape-aware hotspot likelihood map (M^l_s) and the topology-aware hotspot likelihood map (M^l_t) to generate the final hotspot likelihood map (M^l) at the l-th layer. This integration is performed via a weighted summation, defined as:

$$M^l = \lambda M^l_s + (1 - \lambda) M^l_t. \tag{12}$$

Just Accepted

Algorithm 2 Query Process

```
Require: The test layout $X_q$, the frozen CNN $\mathcal{F}(\cdot)$, the stored prototype bank $P$;
Ensure: The final hotspot likelihood score $M_{final}$;
 1: $\{Q^l\}_{l=1}^L \leftarrow \mathcal{F}(X_q)$, The query feature extraction;
 2: Initialize hotspot likelihood map $M \leftarrow \{\}$;
 3: for $l \in [1, L)$ do
 4:     $M_s^l \leftarrow$ ShapeQuery($Q^l$,$P^l$);
 5:     $M_t^l \leftarrow$ TopologyQuery($Q^l$,$P^l$);
 6:     $M^l \leftarrow$ Computing the $l$-th hotspot likelihood map by Equation(12);
 7:     $M \leftarrow M \cup \{M^l\}$;
 8: end for
 9: $M_{final} \leftarrow$ Computing the final hotspot likelihood score by Equation(13);
10: return $M_{final}$
ShapeQuery($Q^l$,$P^l$):
11: $H^l, W^l \leftarrow$ Height and width of the $Q^l$;
12: for $(x, y) \in [0, H^l) \times [0, W^l)$ do
13:     $\Omega_{(x,y)}^{P^l} \leftarrow$ Computing the search space by Equation(7);
14:     $M_{s,(x,y)}^l \leftarrow$ Computing the hotspot likelihood maps on $(x, y)$ by Equation(8);
15: end for
16: return $M_s^l$
TopologyQuery($Q^l$,$P^l$):
17: $H^l, W^l \leftarrow$ Height and width of the $Q^l$;
18: for $(x, y) \in [0, H^l) \times [0, W^l)$ do
19:     $\mathcal{N}_4^{P^l}(x, y) \leftarrow$ Computing the four-neighbor prototypes by Equation(9);
20:     $\Omega_{(x,y)}^{Q^l} \leftarrow$ Computing the search space by Equation(10);
21:     $M_{t,(x,y)}^l \leftarrow$ Computing the hotspot likelihood maps on $(x, y)$ by Equation(11);
22: end for
23: return $M_t^l$
```

Just Accepted

The set of hotspot likelihood maps $\{M^l\}_{l=1}^L$, collected across multiple layers with varying spatial resolutions, is subsequently up-sampled to a common resolution. Following this, the up-sampled maps are aggregated by element-wise summation. Motivated by industrial anomaly detection [36, 62], we apply a Gaussian blur filter to the aggregated hotspot likelihood map to further smooth the prediction. The final hotspot likelihood score, M_{final}, is obtained by extracting the global maximum value from the blurred map. The overall process is mathematically formulated as:

$$M_{final} = \max_{x \in H^0, y \in W^0} (\mathcal{G}_\sigma(\sum_{l=1}^{L} \text{UpSample}_{(\text{H}^0,\text{W}^0)}(M^l))), \tag{13}$$

where $\mathcal{G}_\sigma(\cdot)$ denotes the application of a Gaussian blur with kernel standard deviation σ, and $\text{UpSample}_{(\text{H}^0,\text{W}^0)}(\cdot)$ represents the up-sampling operation to the target resolution (H^0, W^0).

3.3.4 Optional refinement under a few available hotspots. In practical scenarios, entirely disregarding hotspot samples can limit the model's ability to capture discriminative features, potentially degrading detection performance. Thus, YONN can be refined by incorporating a few available hotspot samples. A prototype bank is constructed from few hotspot samples using the same methodology as for non-hotspots. During inference, the query sample first estimates its likelihood of being a hotspot using the non-hotspot prototype bank. Subsequently, the same inference procedure is applied using the hotspot prototype bank. The final hotspot probability $\bar{M}$ is computed by aggregating the two likelihoods as follows:

$$\bar{M} = \alpha M_{final}^{nh} + (1 - \alpha)(1 - M_{final}^{h}). \tag{14}$$

where α is a balancing weight between the two queries, M_{final}^{nh} denotes the inference result from the non-hotspot prototype bank, and M_{final}^{h} denotes the result from the hotspot prototype bank.

3.4 On-the-fly Update

In modern design houses, frequent product updates and iterative design modifications often result in the degradation or failure of previously trained hotspot detectors. As shown in Figure 3, adapting these detectors to new designs typically requires mixing historical and newly collected data followed by full model retraining. However, as the volume of layout data continues to grow, this retraining approach becomes increasingly impractical due to the associated computational cost and data management complexity. In contrast, YONN adapts to new designs through simple expansion of the prototype bank, thereby offering a scalable and efficient solution for hotspot detection across evolving design iterations. Specifically, given a new set of layout data $\{X_n^{\text{new}}\}_{n=1}^{N}$, YONN extracts new features F^{new} using the same prototype generation network $\mathcal{F}(\cdot)$. To address potential concerns regarding the accumulation of redundant prototypes and the scalability of the query process, YONN incorporates the prototype sampling strategy during each update. Specifically, when new non-hotspot features are extracted and merged into the existing prototype bank $F'^{\text{new},l} \leftarrow P^l \cup F^{\text{new},l}$, the combined bank undergoes the refined prototype resampling process by using Algorithm 1, with $F'^{\text{new},l}$ as the input. This ensures that only high-density core representations and diverse boundary variants are retained, effectively eliminating redundant or overlapping prototypes. As a result, the size of the prototype bank remains compact and representative, even after multiple design updates.

4 Experiments

All experiments were performed using the PyTorch framework and the NVIDIA CUDA toolkit on a high-performance computing system equipped with Intel Xeon Platinum 8462Y+ 128-core processors and a single NVIDIA H800 GPU featuring 80 GB of memory. This setup provided the computational resources necessary for efficient model training and evaluation.

4.1 Datasets

This article utilizes three benchmark datasets that include both metal layer and via layer layouts. The details of these benchmarks are summarized as follows:

- The ICCAD2012 benchmark[43] consists of four layout datasets. The first dataset, ICCAD2012-1, is derived from a 32 nm process with a single metal layer, while ICCAD2012-2, ICCAD2012-3, ICCAD2012-4, and ICCAD2012-5 are based on a 28 nm process. Consistent with the methodology described in [14], each dataset is divided into training and testing subsets. Each layout clip spans a fixed dimension of 4.8 μm $\times$ 4.8 μm. For clips labeled as hotspots, a central region measuring 1.2 μm $\times$ 1.2 μm is explicitly defined as the hotspot area.

Just Accepted

Table 1. Statistics of the Benchmark.

	ICCAD2012					ICCAD2016	ICCAD2020			
	-1	-2	-3	-4	-5		Via-1	Via-2	Via-3	Via-4
Training #HS	99	176	909	98	26	1300	3430	1029	614	39
Training #NHS	340	5285	4643	4452	2716	9935	10290	11319	19034	23010
Testing #HS	226	499	1835	189	42	1301	2267	724	432	26
Testing #NHS	319	4143	3484	3370	2110	2484	6878	7489	12614	15313

- The ICCAD2016 benchmark[42] has been adapted to comply with design rules relevant to extreme ultraviolet (EUV) lithography for metal layers. Lithographic hotspot simulation and analysis annotate hotspot locations directly within the layout files. For classification, hotspot samples were randomly cropped from the labeled hotspot regions in ICCAD2016, while non-hotspot samples were randomly cropped from regions outside these hotspot areas. Each cropped layout clip had a resolution of 512×512 pixels. Furthermore, we introduce an enhanced data augmentation pipeline that allows multiple hotspot instances to be included within a single cropped layout image, thereby increasing complexity and the difficulty of the classification task.
- The ICCAD2020 benchmark[15] provides layout clips that include via patterns. In this study, we utilize four datasets (Via 1 to Via 4) extracted from the benchmark. The technology node of the benchmark is below 45 nanometers. Each individual dataset within the ICCAD2020 benchmark exhibits varying densities of via patterns. For experimental purposes, each dataset is partitioned into training and testing subsets. Further details on the ICCAD2020 benchmark can be found in [15].

4.2 Implementation Details

In this article, YONN employs a uniform transformation of layout clips with varying resolutions, converting them into a squish pattern representation of size $3 \times 128 \times 128$ using the adaptive squish pattern method described in [55]. In the prototype generation stage, the pre-trained model leverages Wide-ResNet101[58] as a fixed prototype extractor, which is a widely adopted architecture in DL. To refine the prototypes, YONN utilizes feature maps from three layers (L1, L2, L3) of the Wide-ResNet101, thereby preserving low-level geometric details while capturing high-level semantic representations. The batch size for the prototype generation is 4. In the query stage, the batch size is 256. For YONN, the prototype sample size is set to $K = 60$ to balance accuracy and storage, the query balancing parameter is $\lambda = 0.5$, the hotspot aggregation weight is $\alpha = 0.3$, and the shape-aware search-space parameter is set to $r = 5$.

To evaluate the effect of incorporating a small number of hotspot samples, we adopt a purely random hotspot sampling strategy consistent with [7], ensuring comparability with prior work. For this setting, three independent random sampling experiments are conducted, and the corresponding mean and standard deviation of the results are reported.

4.3 Results Comparison on ICCAD2012 Benchmark

To evaluate the performance of the proposed method, we conduct a comprehensive comparison using the ICCAD2012 benchmark. As shown in Table 2, the method proposed by TODAES’2019 achieves a perfect ACC of 100.00% on ICCAD2012-1 but exhibits relatively high FAR across the other datasets. Specifically, TODAES’2019 demonstrates an average FAR of 29.88% on ICCAD2012-1, and the FAR ranges from 2.04% to 46.67% for the remaining datasets. Meanwhile, the method in DAC’2021 shows good performance with an ACC of 99.40% on ICCAD2012-2 and a significantly reduced FAR of 2.50%. However, its performance drops on ICCAD2012-3, where

Just Accepted

Table 2. Comparison on ICCAD2012 benchmark with the state-of-the-art models (%).

Method	Usage of hotspots	ICCAD2012-1		ICCAD2012-2		ICCAD2012-3		ICCAD2012-4		ICCAD2012-5		Time(s)
		ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR	
TODAES'2019[18]	100%	100.00	29.88	99.00	3.48	98.06	19.52	98.31	6.41	95.12	2.04	330.00
DAC'2021[44]		-	-	99.40	2.50	98.10	46.67	99.40	9.97	97.60	1.61	-
TCAD'2022[14]		99.56	73.85	98.20	1.59	99.90	12.66	94.18	4.93	95.24	1.14	92.99
IWAPS'2022[25]		99.60	37.64	98.20	2.82	98.10	6.71	93.90	10.65	98.60	11.75	-
TODAES'2025[47]		-	-	99.40	3.69	97.70	4.18	93.90	1.57	98.60	2.94	26.10
Ours		99.56	4.70	99.60	1.23	99.18	3.59	98.41	1.45	97.62	1.09	31.12
TCAD'2020[7]	10%	-	-	96.91	23.36	97.77	5.45	54.35	13.47	25.85	2.80	-
	30%	-	-	96.95	7.36	97.83	5.06	72.54	8.37	40.49	2.04	
	50%	-	-	97.99	7.41	97.83	5.42	81.69	7.60	88.29	5.07	
Ours	0%	90.27	15.67	90.32	10.86	90.24	12.60	92.06	11.04	95.24	10.33	15.49
	30%	98.67 (±1.17)	7.63 (±1.27)	98.06 (±0.81)	2.77 (±1.06)	98.02 (±0.54)	4.38 (±0.50)	95.42 (±0.81)	3.01 (±1.03)	94.44 (±3.63)	1.96 (±0.52)	20.72

Table 3. Comparison on ICCAD2016 benchmark with the state-of-the-art unsupervised method in deep learning (%).

	ICML'2018[37]	CVPR'2022[36]	CVPR'2023[30]	WACV'2024[2]	ECCV'2024[24]	Ours
ACC	7.84	30.13	17.06	4.77	40.05	88.09
FAR	72.14	81.24	60.06	41.22	39.94	15.90

Just Accepted

it achieves only 98.10% ACC and a high FAR of 46.67%. The TODAES'2025 shows the best performance with the ACC of 99.40% on ICCAD2012-1 and maintaining relatively low FAR across different datasets. Notably, all of these methods operate in a fully supervised setting, requiring extensive hotspot annotations. In contrast, YONN without hotspot samples (usage of hotspots 0%) achieves competitive accuracy, ranging from 90.24% to 95.24%, with FAR between 10.33% and 15.67%, representing a favorable ACC–FAR trade-off compared to methods such as IWAPS'2022 and TCAD'2022, which exhibit higher FAR despite good ACC. We further compare with TCAD'2020, which employs a semi-supervised strategy. Its performance improves as more labeled data are used, but only becomes comparable to state-of-the-art fully supervised methods when trained with 50% of the available data. Remarkably, YONN, when utilizing only 30% of the hotspot samples, achieves ACC and FAR competitive with the best supervised approaches.

In addition to detection accuracy and FAR, inference efficiency is an important consideration for practical deployment. As shown in the Time(s) column of Table 2, YONN demonstrates substantial runtime advantages. For instance, in the fully unsupervised setting (0% hotspots), YONN requires only 15.49 seconds on the ICCAD2012 benchmark, compared to 330.00 seconds for TODAES'2019 and 92.99 seconds for DAC'2021. Even with 30% hotspot usage, YONN completes inference in 20.72 seconds, remaining faster than most supervised baselines.

Figure 9 illustrates the variation of ACC and FAR on ICCAD2012-2, -3, and -4 as the proportion of hotspot samples increases. With very few hotspot samples, YONN's performance improves rapidly as more samples are added, with larger standard deviations in early stages reflecting the impact of random sampling. Nonetheless, the standard deviation remains within 2% overall. At approximately 30% hotspot coverage, YONN reaches performance comparable to the TODAES'2025.

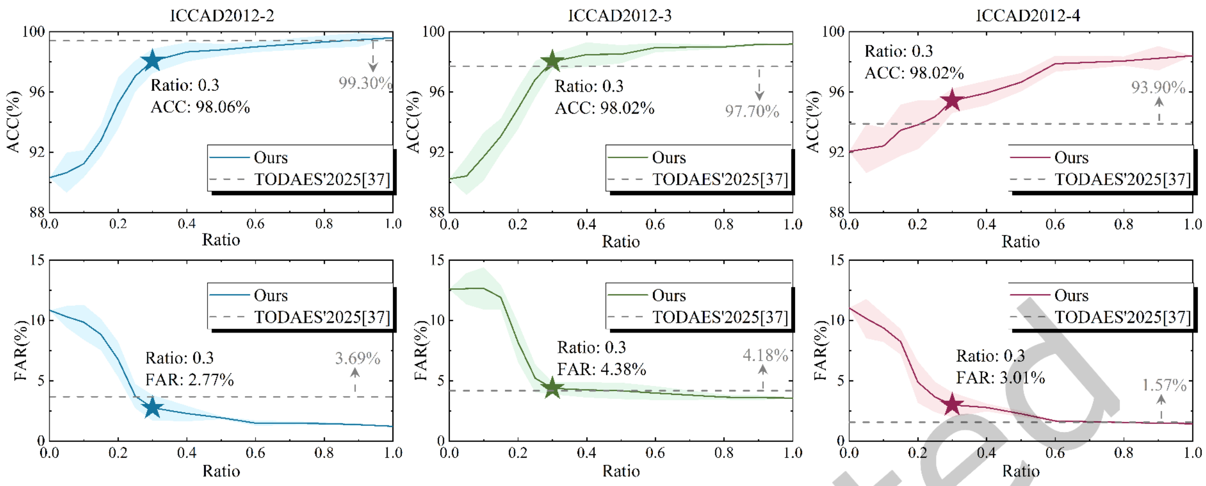

Fig. 9. Variation of ACC and FAR on ICCAD2012-2, -3, and -4 as the proportion of hotspot samples increases.

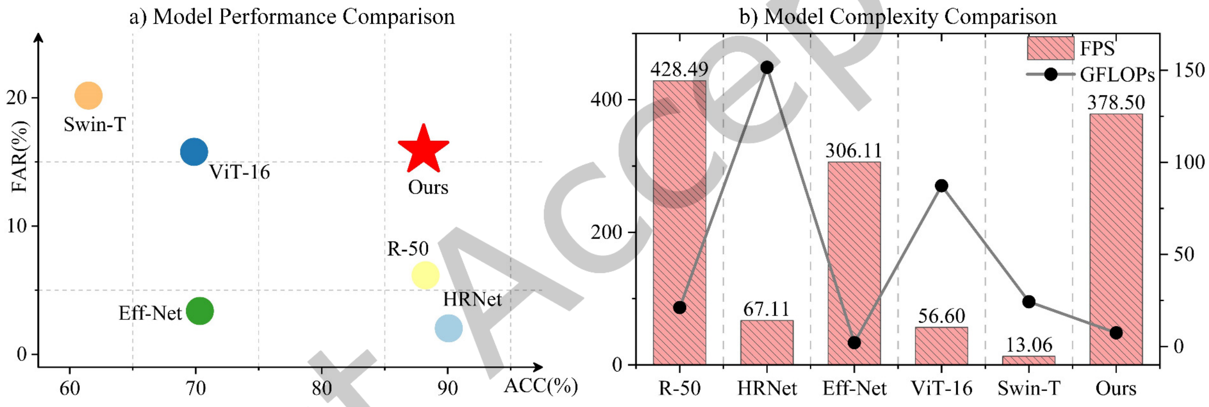

Fig. 10. Comparison of model performance and complexity on ICCAD2016.

Just Accepted

4.4 Results Comparison on ICCAD2016 Benchmark

To evaluate the effectiveness of YONN, we benchmarked its performance against several widely adopted DL-based architectures on the ICCAD2016 dataset, including ResNet-50[17] (R-50), HRNet[46], EfficientNet[41] (Eff-Net), Vision Transformer[12] (ViT-16), and Swin Transformer[29] (Swin-T). The evaluation metrics include ACC, FAR, inference speed in frames per second (FPS), and computational complexity measured by GFLOPs. As shown in Figure 10(a), YONN demonstrates competitive accuracy, achieving an ACC of 88.09%, which is second only to HRNet's top performance of 90.09%. YONN outperforms other models such as ViT-16 (69.87%) and Swin-T (61.49%) by a substantial margin. Regarding the FAR metric, YONN shows a higher FAR compared to R-50, HRNet, and EfficientNet, yet it remains lower than that of Swin-T and ViT-16. Specifically, YONN achieves a FAR of 15.91%.

Regarding computational efficiency, YONN demonstrates a balance between accuracy and resource requirements. Figure 10(b) compares all models in terms of FPS and GFLOPs. YONN achieves an inference speed of

Table 4. Comparison on ICCAD2020 benchmark with the state-of-the-art models (%).

Method	Usage of hotspots	Via-1		Via-2		Via-3		Via-4	
		ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR
TCAD'2020[21]	100%	89.85	27.42	73.00	16.32	73.38	27.00	73.08	99.84
TCAD'2022[15]		93.42	23.10	86.32	14.69	88.20	16.69	80.77	1.00
DATE'2022[39]		95.41	25.04	90.33	20.12	89.81	15.28	84.62	2.61
DAC'2024[9]		95.76	24.75	91.30	15.50	91.43	12.98	92.31	1.99
TCAD'2025[8]		95.56	23.80	90.74	17.67	89.81	14.27	88.46	2.34
Ours		96.25	19.21	92.40	14.54	92.36	13.66	88.46	2.04
CVPR'2022[36]	0%	39.35	64.45	49.31	47.96	12.96	55.42	30.77	77.86
ECCV'2024[24]		43.54	62.71	29.28	74.94	25.69	52.58	34.62	59.58
Ours	0%	86.68	21.05	87.02	23.52	82.41	25.10	84.62	10.75
	30%	94.28 (±1.44)	20.35 (±1.61)	91.11 (±0.89)	16.82 (±1.15)	90.20 (±1.40)	14.18 (±0.67)	87.18 (±5.88)	3.20 (±0.85)

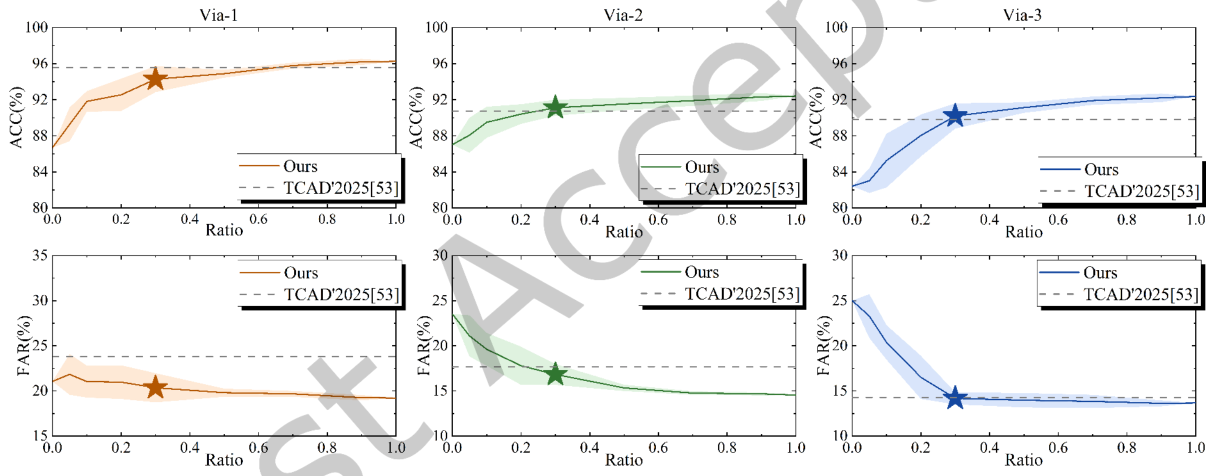

Fig. 11. Variation of ACC and FAR on Via-1, -2, and -3 as the proportion of hotspot samples increases.

Just Accepted

378.5 FPS with a computational cost of only 7.46 GFLOPs, indicating its efficiency for real-time deployment scenarios. In contrast, HRNet, while exhibiting slightly higher accuracy, operates at a lower FPS (67.11) and incurs a significantly higher computational cost (151.64 GFLOPs). Similarly, R-50 and ViT-16 exhibit higher GFLOPs and lower FPS than YONN, further highlighting the efficiency advantage of our approach.

We additionally compared YONN with state-of-the-art unsupervised DL methods, including ICML'2018, CVPR'2022, CVPR'2023, WACV'2024, and ECCV'2024. As shown in Table 3, the results demonstrate that existing unsupervised approaches struggle to achieve robust hotspot detection, whereas YONN consistently delivers higher detection accuracy and a lower FAR than these baselines.

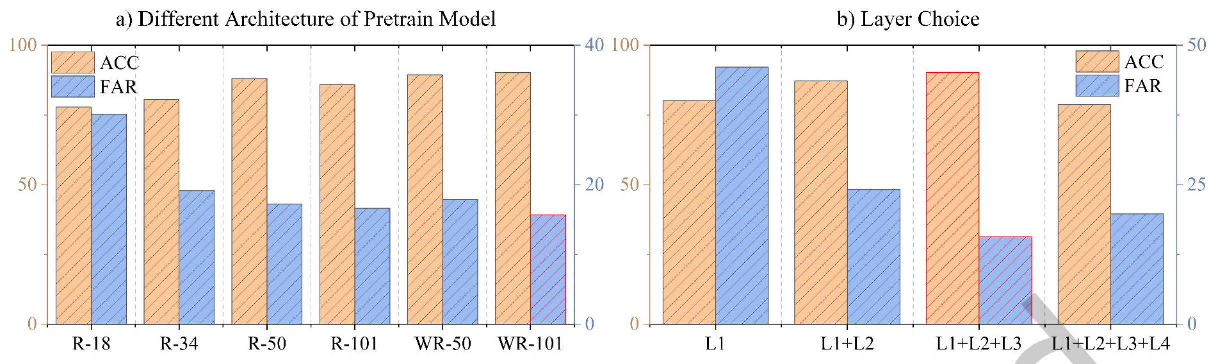

Fig. 12. Illustration of the effects of different architecture in $\mathcal{F}(\cdot)$.

4.5 Results Comparison on ICCAD2020 Benchmark

We present a detailed comparison of our method with state-of-the-art models on the ICCAD2020 benchmark. As shown in Table 4, YONN demonstrates the best performance across all via datasets with the usage of 100% hotspots samples. Moreover, in the fully unsupervised setting, YONN achieves competitive performance across all via datasets, attaining accuracies of 86.68%, 87.02%, 82.41%, and 84.62% for Via-1, Via-2, Via-3, and Via-4, respectively, with corresponding FAR values of 21.06%, 23.52%, 25.10%, and 10.75%. We additionally compared state-of-the-art unsupervised detection methods in deep learning, including CVPR'2022 and ECCV'2024. In the unsupervised setting, these methods fail to provide effective hotspot detection on via data, exhibiting extremely low accuracy and excessively high FAR. Notably, with the usage of only 30% hotspot samples, YONN's performance approaches that of the latest supervised method (TCAD'2025), highlighting its effectiveness and adaptability. Figure 11 shows how ACC and FAR change as the proportion of hotspot samples increases in the Via-1, 2, and 3 datasets. Figure 11 further illustrates the variation in ACC and FAR as the proportion of hotspot samples increases for Via-1, Via-2, and Via-3.

4.6 Ablation Study

4.6.1 Impact of Different Architecture for Prototype Generation Network $\mathcal{F}(\cdot)$. To validate the effectiveness of Wide-ResNet101 (WR-101) as a prototype generative network, we conducted a comparative study across various ResNet-based CNNs. Specifically, we evaluated ResNet18 (R-18), ResNet34 (R-34), ResNet50 (R-50), ResNet101 (R-101), Wide-ResNet50 (WR-50), and WR-101 on the ICCAD2012-1 dataset, as shown in Figure 12(a). The results indicate that increasing network depth generally leads to improved ACC and reduced FAR, with Wide-ResNet architectures consistently outperforming standard ResNet. Furthermore, in the YONN framework, we preserve features extracted from multiple layers to enhance prototype refinement. Figure 12(b) presents an analysis of the impact of selecting different layers for prototypes. The original ResNet-based architecture comprises five layers, where deeper layers provide lower-resolution feature maps with richer semantic information. Our experiments indicate that while incorporating features from an increasing number of layers initially improves performance, inclusion of features from the fourth layer (L4) leads to a noticeable decline. This degradation is attributed to the predominance of geometric properties over semantic information in layout design, which are less effectively captured by lower-resolution, higher-level features. Consequently, YONN optimally retains features from the first through third layers (L1, L2, L3).

Just Accepted

Table 5. Impact of the prototype sampling size K on performance. "w/o. Sampling" indicates the results obtained when prototype sampling is not applied. PSS refers to the prototype storage space. The units for ACC and FAR are percentage points (%).

w/o. Sampling			K=40			K=50			K=60			K=70		
ACC	FAR	PSS	ACC	FAR	PSS	ACC	FAR	PSS	ACC	FAR	PSS	ACC	FAR	PSS
91.15	14.73	638M	88.50	16.30	76M	89.38	15.99	94M	90.27	15.67	113M	89.82	15.05	132M

Table 6. Impact of the different clustering strategies. PG-Time refers to the prototype generation time. The units for ACC and FAR are percentage points (%).

KMeans[32]			DBSCAN[13]			OPTICS[1]			HDBSCAN[3]		
ACC	FAR	PG-Time	ACC	FAR	PG-Time	ACC	FAR	PG-Time	ACC	FAR	PG-Time
88.94	17.55	55.96(s)	88.05	16.93	56.71(s)	89.82	15.99	56.67(s)	90.27	15.67	23.38(s)

Just Accepted

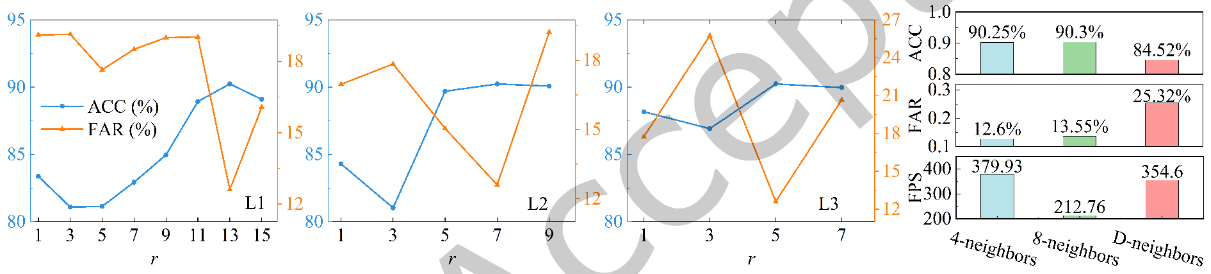

Fig. 13. Impact of the hyperparameter in query space on ICCAD2012-3.

4.6.2 Impact of Prototype Sampling Size K. We investigate the impact of the prototype sampling size K for YONN performance on ICCAD2012-1, as detailed in Table 5. The "w/o. Sampling" setting represents the configuration where all available prototypes are retained without any sampling, yielding the highest ACC of 91.15% and the lowest FAR of 14.73%. However, this comes at a substantial prototype storage space (PSS) cost of 638MB, which is impractical for resource-constrained deployment scenarios. By contrast, applying prototype sampling with varying values of K significantly reduces the storage cost. For instance, setting $K = 40$ results in a compression of PSS to just 76MB—an 88.1% reduction compared to the baseline. As K increases from 40 to 70, both ACC and FAR improve gradually, demonstrating the trade-off between model compactness and detection performance. At $K = 60$, the model achieves a balance with an ACC of 90.26%, a FAR of 15.67%, and a moderate PSS of 113MB.

4.6.3 Impact of Different Clustering Strategies. In this study, we evaluate the influence of various clustering strategies on the performance and efficiency of our prototype generation process. Specifically, we compare four widely adopted clustering algorithms, KMeans [32], DBSCAN [13], OPTICS [1], and HDBSCAN [3], to investigate their effects on ACC, FAR, and prototype generation time (PG-Time). Table 6 summarizes the experimental results. HDBSCAN demonstrates superior performance by achieving the highest ACC of 90.27% and the lowest FAR of 15.67%. Notably, it also requires the shortest prototype generation time of 23.38 seconds, indicating enhanced computational efficiency compared to other methods.

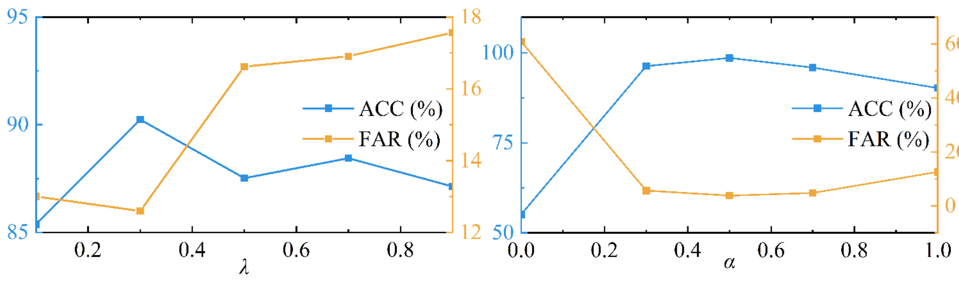

Fig. 14. Impact of λ and α.

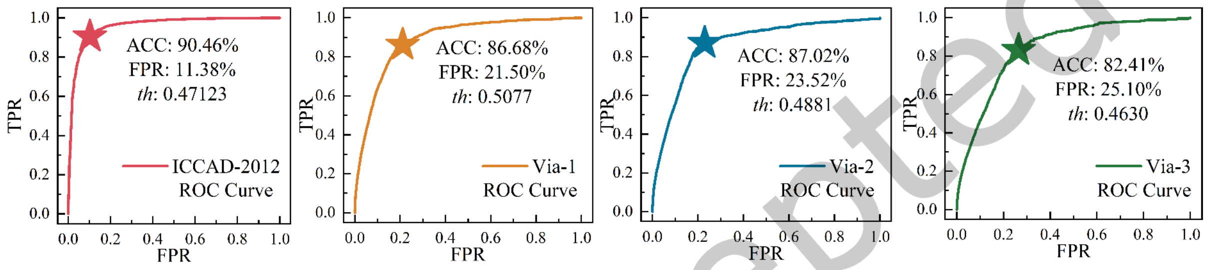

Fig. 15. Receiver Operating Characteristic (ROC) curves of YONN evaluated across multiple benchmark datasets, illustrating the trade-off between true positive rate and false positive rate for different operating thresholds.

Just Accepted

4.6.4 Impact of the Hyperparameter in Query Space. To evaluate the influence of the shape-aware search space parameter r, we conduct an ablation study on three representative layers (L1–L3) of YONN. As shown in Figure 13, increasing the search radius generally enhances ACC while reducing the FAR. For L1, optimal performance is observed when $r = 13$, whereas for L2 and L3, the best results are attained at $r = 5$. These differences reflect the varying representation of the layout features across layers. Notably, an excessively large search space in lower-level layers can degrade performance due to the introduction of noisy or irrelevant structural information, which may impair the effectiveness of shape-aware query.

In addition, we also explore the impact of different neighborhood configurations on model performance, including 4-neighborhood, 8-neighborhood, and D-neighborhood strategies. As shown in Figure 13, the 4-neighborhood and 8-neighborhood models yield comparable ACC (90.25% and 90.30%, respectively) and low FAR (12.60% and 13.55%), while maintaining high runtime efficiency with inference speeds of 379.93 FPS and 212.76 FPS. By contrast, the D-neighborhood configuration exhibits a noticeable drop in ACC (84.5%) and a significant increase in FAR (25.32%), along with reduced runtime performance (354.60 FPS). These results suggest that while expanding the search range may offer marginal gains in accuracy for some layers, overly broad search spaces introduce noise and degrade model performance.

4.6.5 Impact of the Hyperparameter λ and α. The weight λ in Eq. (12) balances shape-aware and topology-aware queries. As shown in Figure 14, larger λ favors shape similarity, improving ACC but raising FAR, while smaller λ emphasizes topology, reducing FAR but missing shape-sensitive hotspots. Across datasets, $\lambda = 0.5$ yields the best trade-off. The parameter α in Eq. (14) balances predictions from non-hotspot and hotspot prototype banks when few hotspot samples are available. Higher α benefits when hotspot coverage is sparse; lower α is preferable with richer hotspot data. The optimal value $\alpha = 0.3$ consistently achieves a favorable ACC–FAR balance.

Table 7. Effectiveness of query mechanisms on ICCAD2012-1. (%).

w./ Shape-aware Query		w./ Topology-aware Query		YONN	
ACC	FAR	ACC	FAR	ACC	FAR
88.50	18.18	86.73	17.87	90.27	15.67

Table 8. Performance on ICCAD2012 for the update(%).

Base Dataset	ICCAD2012-1		ICCAD2012-2		ICCAD2012-3		ICCAD2012-4		ICCAD2012-5	
for Prototype Generation	ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR
1	90.27	15.67	-	-	-	-	-	-	-	-
1+2	89.38	16.30	89.38	11.05	-	-	-	-	-	-
1+2+3	89.82	15.67	90.18	11.92	89.37	14.18	-	-	-	-
1+2+3+4	90.27	15.05	89.98	10.89	89.92	12.60	91.01	11.04	-	-
1+2+3+4+5	88.94	16.61	89.78	10.89	90.19	13.35	90.48	12.88	92.86	10.05

4.6.6 Impact of the threshold th for hotspot classification. The selection of the decision threshold th plays a crucial role in determining the trade-off between detection accuracy and false alarm rate in hotspot classification. It is important to emphasize that this thresholding step is not part of the training phase; instead, it is applied exclusively during the query stage of testing to decide whether a given sample is classified as a hotspot or a non-hotspot.

To determine the optimal threshold, we adopt a Receiver Operating Characteristic (ROC) curve analysis, which plots the true positive rate (equivalent to ACC in hotspot detection) against the false positive rate (equivalent to FAR) for varying threshold values. As illustrated in Figure 15, the optimal operating point is selected to achieve a balanced compromise between high ACC and low FAR. Across the ICCAD2012 and ICCAD2020 benchmarks, this strategy consistently yields optimal thresholds in a narrow range between 0.4630 and 0.5077. The stability of these optimal thresholds across different datasets indicates that the YONN framework is robust to threshold selection. Consequently, in practical deployment scenarios, a default threshold of $th = 0.5$ can be adopted without dataset-specific tuning, simplifying implementation while preserving detection performance.

4.6.7 Effectiveness of Query Mechanisms. To evaluate the contribution of individual query mechanisms in YONN, we conducted an ablation study focusing on the shape-aware and topology-aware query components. Table 7 summarizes the ACC and FAR achieved when incorporating each query type independently, as well as the overall performance of YONN. When using only the shape-aware query module, the model achieves an ACC of 88.50% and a FAR of 18.18%. In contrast, the topology-aware query mechanism yields an ACC of 86.73% with a slightly improved FAR of 17.87%. These results indicate that both query types contribute positively to model performance, with the shape-aware mechanism being more effective in enhancing accuracy and the topology-aware variant offering a modest reduction in false acceptances. The YONN, which integrates both query strategies within a unified framework, significantly outperforms each configuration. It achieves the highest ACC of 90.27% and the lowest FAR of 15.67%.

Just Accepted

Table 9. Performance on ICCAD2020 for the update(%).

Base Dataset	Via-1		Via-2		Via-3		Via-4	
for Prototype Generation	ACC	FAR	ACC	FAR	ACC	FAR	ACC	FAR
1	86.68	21.05	-	-	-	-	-	-
1+2	86.46	22.08	86.74	23.38	-	-	-	-
1+2+3	86.90	20.94	87.15	23.62	81.94	26.26	-	-
1+2+3+4	86.41	21.33	86.05	23.75	82.18	25.11	84.62	11.11

4.7 Performance for On-the-fly Update

To evaluate the adaptability and performance of our framework under incremental data inclusion, we conducted a progressive update experiment on ICCAD2012 and ICCAD2020.

As shown in Table 8, initially, only ICCAD2012-1 is used for prototype generation, yielding an ACC of 90.27% and a FAR of 15.67%. As additional datasets are integrated in sequence (ICCAD2012-2 through ICCAD2012-5), both ACC and FAR are observed for each combination. The inclusion of ICCAD2012-2 (1+2) leads to a slight reduction in ACC for ICCAD2012-1 (89.38%), but a marked improvement for ICCAD2012-2 (89.38%) with a reduced FAR of 11.05%. Upon the addition of ICCAD2012-3 (1+2+3), ACC across all three datasets stabilizes above 89%, demonstrating the robustness of the model during incremental learning. The integration of ICCAD2012-4 (1+2+3+4) results in a further improvement, achieving over 91% ACC on ICCAD2012-4 and maintaining stable performance across previous datasets. Finally, with the full inclusion of ICCAD2012-5 (1+2+3+4+5), the YONN reaches its highest ACC of 92.86% on ICCAD2012-5, along with the lowest observed FAR of 10.05%.

As shown in Table 9, starting with Via-1 alone, YONN achieves an ACC of 86.68% and a FAR of 21.05%. Incorporating Via-2 (1+2) slightly decreases ACC on Via-1 (86.46%) while marginally improving Via-2 (86.74%) with a FAR of 23.38%. The addition of Via-3 (1+2+3) maintains stable ACC on the first two datasets and yields 81.94% ACC on Via-3, albeit with a higher FAR of 26.26%. With the full integration of Via-4 (1+2+3+4), ACC on Via-4 reaches 84.62% with a FAR of 11.11%, while earlier datasets maintain performance close to their initial values.

These results indicate that the proposed YONN effectively accommodates new design data while maintaining generalization across earlier distributions.

5 Conclusion

This paper introduces YONN, an unsupervised hotspot detection algorithm that utilizes only non-hotspot samples and requires no training. Specifically, YONN is based on a novel paradigm involving prototype-based decision boundary and query inference for hotspot detection. Initially, YONN employs a frozen CNN to extract multi-scale, fine-grained prototype features. Then, a cluster-boundary based sampling method is proposed to reduce the size of the prototype bank while preserving competitive performance. During the query phase, YONN performs both shape-aware and topology-aware queries based on layout features. The shape-aware query assesses the hotspot likelihood in local features, while the topology-aware query compares features within the neighbors of the layout. Since YONN is training-free, it can adaptively update the prototype bank for new products or designs. Moreover, when a limited number of hotspot samples is available, YONN can be augmented with a supplementary hotspot prototype bank, further improving classification accuracy. Extensive experiments demonstrate that YONN's performance is within 10% of fully supervised approaches, and with only 30% of hotspot-labeled data, it surpasses

Just Accepted

state-of-the-art methods. Future work will focus on extending YONN's applicability across advanced process nodes and improving computational efficiency.

Acknowledgement

Silin Chen was supported in part by Postgraduate Research Practice Innovation Program of Jiangsu Province (Grant Number: KYCX25_0394). Ningmu Zou was supported in part by the National Natural Science Foundation of China (62341408).

References

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record* 28, 2 (1999), 49–60.
[2] Kilian Batzner, Lars Heckler, and Rebecca König. 2024. Efficientad: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 128–138.
[3] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 160–172.
[4] Wei-Chun Chang and Iris Hui-Ru Jiang. 2019. iClaire: A fast and general layout pattern classification algorithm with clip shifting and centroid recreation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 8 (2019), 1662–1673.
[5] Ran Chen, Wei Zhong, Haoyu Yang, Hao Geng, Fan Yang, Xuan Zeng, and Bei Yu. 2020. Faster Region-Based Hotspot Detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 3 (2020), 669–680.
[6] Tinghuan Chen, Grace Li Zhang, Bei Yu, Bing Li, and Ulf Schlichtmann. 2022. Machine learning in advanced IC design: A methodological survey. *IEEE Design & Test* 40, 1 (2022), 17–33.
[7] Ying Chen, Yibo Lin, Tianyang Gai, Yajuan Su, Yayi Wei, and David Z Pan. 2020. Semisupervised Hotspot Detection With Self-Paced Multitask Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 7 (2020), 1511–1523.
[8] Yuyang Chen, Qi Sun, Su Zheng, Xinyun Zhang, Bei Yu, and Hao Geng. 2025. HyDAS: Hybrid Domain Deformed Attention for Selective Hotspot Detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).
[9] Yuyang Chen, Yiwen Wu, Jingya Wang, Tao Wu, Xuming He, Jingyi Yu, and Hao Geng. 2024. LLM-HD: Layout Language Model for Hotspot Detection with GDS Semantic Encoding. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.
[10] Zihao Chen, Fan Yang, Li Shang, and Xuan Zeng. 2023. Automated and Agile Design of Layout Hotspot Detector via Neural Architecture Search. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.
[11] Jian Cui, Jian Zhang, and Xuexiang Wang. 2023. Lithographic Hotspot Detection Using Adaptive Squish Pattern Sampling Combined with Faster R-CNN. In *2023 IEEE 15th International Conference on ASIC (ASICON)*. IEEE, 1–4.
[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, G Heigold, S Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.
[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, Vol. 96. 226–231.
[14] Tianyang Gai, Tong Qu, Shuhan Wang, Xiaojing Su, Renren Xu, Yun Wang, Jing Xue, Yajuan Su, Yayi Wei, and Tianchun Ye. 2021. Flexible hotspot detection based on fully convolutional network with transfer learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2021), 4626–4638.
[15] Hao Geng, Haoyu Yang, Lu Zhang, Fan Yang, Xuan Zeng, and Bei Yu. 2022. Hotspot Detection via Attention-Based Deep Layout Metric Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 8 (2022), 2685–2698.
[16] Frank E Gennari and Ya-Chieh Lai. 2014. Topology design using squish patterns. US Patent 8,832,621.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[18] Xu He, Yu Deng, Shizhe Zhou, Rui Li, Yao Wang, and Yang Guo. 2019. Lithography hotspot detection with FFT-based feature extraction and imbalanced learning rate. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 25, 2 (2019), 1–21.
[19] Xu He, Yao Wang, Zhiyong Fu, Yipei Wang, and Yang Guo. 2023. A General Layout Pattern Clustering Using Geometric Matching-based Clip Relocation and Lower-bound Aided Optimization. *ACM Transactions on Design Automation of Electronic Systems* 28, 6 (2023), 1–23.
[20] Cong Jiang, Haoyang Sun, Dan Feng, Zhiyao Xie, Benjamin Tan, and Kang Liu. 2025. LithoExp: Explainable Two-stage CNN-based Lithographic Hotspot Detection with Layout Defect Localization. *ACM Transactions on Design Automation of Electronic Systems* 30, 3 (2025), 1–25.
[21] Yiyang Jiang, Fan Yang, Bei Yu, Dian Zhou, and Xuan Zeng. 2020. Efficient layout hotspot detection via binarized residual neural network ensemble. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 7 (2020), 1476–1488.

Just Accepted

[22] Yiyang Jiang, Fan Yang, Bei Yu, Dian Zhou, and Xuan Zeng. 2022. Efficient layout hotspot detection via neural architecture search. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 27, 6 (2022), 1–16.

[23] Andrew B Kahng, Chul-Hong Park, and Xu Xu. 2008. Fast dual-graph-based hotspot filtering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 9 (2008), 1635–1642.

[24] Joo Chan Lee, Taejune Kim, Eunbyung Park, Simon S Woo, and Jong Hwan Ko. 2024. Continuous memory representation for anomaly detection. In *European Conference on Computer Vision*. Springer, 438–454.

[25] Mu Lin, Fanwenqing Zeng, and Yijiang Shen. 2022. Lithography hotspot detection with ResNet network. In *2022 International Workshop on Advanced Patterning Solutions (IWAPS)*. IEEE, 1–4.

[26] Tsung-Wei Lin, Chun Yen Liu, Hung-Yu Lin, Mang-Shiun Chiang, Jason Sweis, Philippe Hurat, Chun Yen Liao, Chun-Sheng Wu, Chao-Yi Huang, and Ya-Chieh Lai. 2022. OPC optimization and hotspot detection using pattern diffing. In *DTCO and Computational Patterning*, Vol. 12052. SPIE, 326–332.

[27] Kang Liu, Benjamin Tan, Gaurav Rajavendra Reddy, Siddharth Garg, Yiorgos Makris, and Ramesh Karri. 2020. Bias busters: Robustifying DL-based lithographic hotspot detectors against backdooring attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 10 (2020), 2077–2089.

[28] Kang Liu, Haoyu Yang, Yuzhe Ma, Benjamin Tan, Bei Yu, Evangeline FY Young, Ramesh Karri, and Siddharth Garg. 2020. Adversarial perturbation attacks on ML-based CAD: A case study on CNN-based lithographic hotspot detection. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 25, 5 (2020), 1–31.

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[30] Zhikang Liu, Yiming Zhou, Yuansheng Xu, and Zilei Wang. 2023. Simplenet: A simple network for image anomaly detection and localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20402–20411.

[31] Chris A Mack. 2005. Thirty years of lithography simulation. In *Optical Microlithography XVIII*, Vol. 5754. SPIE, 1–12.

[32] James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Vol. 5. University of California press, 281–298.

[33] Jingyu Pan, Chen-Chia Chang, Zhiyao Xie, Jiang Hu, and Yiran Chen. 2022. Robustify ML-based lithography hotspot detectors. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–7.

[34] Jingyu Pan, Xuezhong Lin, Jinming Xu, Yiran Chen, and Cheng Zhuo. 2023. Lithography hotspot detection based on heterogeneous federated learning with local adaptation and feature selection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 5 (2023), 1484–1496.

[35] Gaurav Rajavendra Reddy, Constantinos Xanthopoulos, and Yiorgos Makris. 2018. Enhanced hotspot detection through synthetic pattern generation and design of experiments. In *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 1–6.

[36] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. 2022. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14318–14328.

[37] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*. PMLR, 4393–4402.

[38] Haoyang Sun, Cong Jiang, Xun Ye, Dan Feng, Benjamin Tan, Yuzhe Ma, and Kang Liu. 2024. Interpretable CNN-Based Lithographic Hotspot Detection Through Error Marker Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2024).

[39] Shuyuan Sun, Yiyang Jiang, Fan Yang, Bei Yu, and Xuan Zeng. 2022. Efficient hotspot detection via graph neural network. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1233–1238.

[40] Shuyuan Sun, Yiyang Jiang, Fan Yang, and Xuan Zeng. 2022. Adversarial Sample Generation for Lithography Hotspot Detection. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 3503–3506.

[41] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[42] Rasit O Topaloglu. 2016. ICCAD-2016 CAD contest in pattern classification for integrated circuit design space analysis and benchmark suite. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–4.

[43] J Andres Torres. 2012. ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *Proceedings of the International Conference on Computer-Aided Design*. 349–350.

[44] Bingshu Wang, Lanfan Jiang, Wenxing Zhu, Longkun Guo, Jianli Chen, and Yao-Wen Chang. 2021. Two-stage neural network classifier for the data imbalance problem with application to hotspot detection. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 175–180.

[45] Chun Wang, Yi Fang, and Sihai Zhang. 2024. Feature Fusion based Hotspot Detection with R-EfficientNet. In *Proceedings of the Great Lakes Symposium on VLSI 2024*. 446–451.

[46] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. 2020. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine*

Just Accepted

intelligence 43, 10 (2020), 3349–3364.

[47] Ying Wang, Haopeng Yan, Yiwen Zhang, Peng Gao, Fei Yu, Xiaoming Xiong, and Shuting Cai. 2025. PSCaps: High-Performance Pose-Sensitive Layout Hotspot Detector based on CapsNet. *ACM Transactions on Design Automation of Electronic Systems* (2025).

[48] Zixiao Wang, Yunheng Shen, Wenqian Zhao, Yang Bai, Guojin Chen, Farzan Farnia, and Bei Yu. 2023. Diffpattern: Layout pattern generation via discrete diffusion. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[49] Liangjian Wen, Yi Zhu, Lei Ye, Guojin Chen, Bei Yu, Jianzhuang Liu, and Chunjing Xu. 2022. Layoutransformer: Generating layout patterns with transformer via sequential pattern modeling. In *Proceedings of the 41st IEEE/ACM international conference on computer-aided design*. 1–9.

[50] Wan-Yu Wen, Jin-Cheng Li, Sheng-Yuan Lin, Jing-Yi Chen, and Shih-Chieh Chang. 2014. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 11 (2014), 1671–1680.

[51] Yifeng Xiao, Miaodi Su, Haoyu Yang, Jianli Chen, Jun Yu, and Bei Yu. 2021. Low-cost lithography hotspot detection with active entropy sampling and model calibration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 907–912.

[52] Haopeng Yan, Ying Wang, Peng Gao, Fei Yu, Yuzhe Ma, Xiaoming Xiong, and Shuting Cai. 2025. A Lightweight Heterogeneous Graph Embedding Framework for Hotspot Detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[53] Haoyu Yang, Shuhe Li, Wen Chen, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2021. DeePattern: Layout Pattern Generation With Transforming Convolutional Auto-Encoder. *IEEE Transactions on Semiconductor Manufacturing* 35, 1 (2021), 67–77.

[54] Haoyu Yang, Yajun Lin, Bei Yu, and Evangeline FY Young. 2017. Lithography hotspot detection: From shallow to deep learning. In *2017 30th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 233–238.

[55] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2019. Detecting multi-layer layout hotspots with adaptive squish patterns. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 299–304.

[56] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2019. Hotspot detection using squish-net. In *Design-Process-Technology Co-optimization for Manufacturability XIII*, Vol. 10962. SPIE, 188–195.

[57] Yen-Ting Yu, Ya-Chung Chan, Subarna Sinha, Iris Hui-Ru Jiang, and Charles Chiang. 2012. Accurate process-hotspot detection using critical design rule extraction. In *Proceedings of the 49th Annual Design Automation Conference*. 1167–1172.

[58] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. In *Procedings of the British Machine Vision Conference 2016*. British Machine Vision Association, 87–1.

[59] Hang Zhang, Bei Yu, and Evangeline FY Young. 2016. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[60] Guanglei Zhou, Bhargav Korrapati, Gaurav Rajavendra Reddy, Jiang Hu, Yiran Chen, and Dipto G Thakurta. 2024. PatternPaint: Generating Layout Patterns Using Generative AI and Inpainting Techniques. *arXiv preprint arXiv:2409.01348* (2024).

[61] Binwu Zhu, Su Zheng, Yuzhe Ma, Bei Yu, and Martin Wong. 2025. Bridging Hotspot Detection and Mask Optimization via Domain-Crossing Masked Layout Modeling. *ACM Transactions on Design Automation of Electronic Systems* (2025).

[62] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. 2022. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European Conference on Computer Vision*. Springer, 392–408.

Received 26 May 2025; revised 14 August 2025; accepted 27 September 2025

Just Accepted