Interpretable CNN-Based Lithographic Hotspot Detection Through Error Marker Learning

Haoyang Sun, Cong Jiang, Xun Ye, Dan Feng, *Fellow, IEEE*, Benjamin Tan, *Member, IEEE*, Yuzhe Ma, *Member, IEEE*, and Kang Liu, *Member, IEEE*

***Abstract*—As the technology node develops toward its physical limit, lithographic hotspot detection has become increasingly important and ever-challenging in the computer-aided design (CAD) flow. In recent years, convolutional neural networks (CNNs) have achieved great success in hotspot detection. However, the interpretability of their hotspot prediction has yet to be considered. Compared with conventional lithography simulation and pattern matching-based methods, the black-box nature of CNNs wavers their practical applications with confidence. In this article, we propose the first interpretable CNN-based hotspot detector capable of providing high-detection accuracy and reliable explanations for hotspot identification. Specifically, we augment the training dataset with expanded error markers obtained and preprocessed from lithography simulation, which are then learned by an encoder-decoder architecture as intermediate features. We additionally introduce coordinate attention in the encoder to facilitate better-feature extraction. By learning these error markers and part of their surrounding metals as root cause hotspot features, our architecture achieves the highest-hotspot accuracy of 99.78% and the lowest-false positive rate of 5.29% compared to all prior work. Moreover, our method demonstrates the best visual and quantitative interpretability results when applying CNN interpretation methods.**

***Index Terms*—Accuracy, convolutional neural network (CNN), interpretability, lithographic hotspot detection.**

I. Introduction

MACHINE learning (ML) and deep learning (DL) have made unprecedented progress in solving many electronic design automation (EDA) problems, including routability prediction [1], [2], logic synthesis optimization [3], [4], mask synthesis and verification [5], [6], [7], etc., one of these successful applications being lithographic hotspot detection. Lithographic hotspot detection is a critical step in the chip design stage that identifies problematic layouts before manufacturing. As the gap between the optical wavelength of the lithography system and chip feature sizes ever increases, process variations become more sensitive and prone to cause printing defects in the layouts, leading to manufacturing yield loss. Therefore, it is necessary to proactively identify lithographic hotspots in the design stage as early as possible.

Over the years, various methods have been proposed for lithographic hotspot detection, ranging from lithography simulation [8], [9] and pattern matching (PM) [10], [11] to recently studied ML [12], [13] and DL-based solutions [14], [15], [16], [17], [18], [19], [20], [21]. Lithography simulation achieves high accuracy of hotspot detection through mathematical modeling of the lithography process and is deemed the golden solution used in mainstream EDA tools [22]. However, lithography simulation operates at the cost of intensive computation overhead and turnaround time. PM-based approaches take a collection of known hotspot layouts and analyze their feature characteristics; any incoming layouts are scanned over the feature library for hotspot identification. Although PM is fast, it usually fails to recognize hotspot patterns unseen before and raises a high-miss rate. To generalize to a broader range of hotspot layouts than PM solutions, ML and especially DL-based methods, such as convolutional neural networks (CNNs) [14], [15], [16], are now achieving great success by learning the correlations between extracted layout features and their corresponding hotspot and nonhotspot labels from a vast number of layout patterns.

Among these hotspot detection methods, lithography simulation and PM ascertain a hotspot layout by directly or indirectly recognizing their contained defect regions. For instance, lithography simulation directly identifies hotspots by simulating the defect regions with accurate lithography process modeling. Typically, layout masks are partitioned into clips that are subject to optical and resist models to obtain the resist contours [8], [9], where specific layout patterns are revealed to be problematic, e.g., short or open circuits. Based on this, error markers are flagged to indicate the printing defect regions, as shown in Fig. 1. A layout clip is considered a hotspot if the simulated error markers show an intersection with a predefined square Region of Interest (RoI) in the center of the clip [12], [23], [24], [25], [26].

Received 13 February 2024; revised 4 July 2024 and 3 September 2024; accepted 16 September 2024. Date of publication 25 September 2024; date of current version 21 February 2025. The work of Yuzhe Ma was supported in part by the National Natural Science Foundation of China under Grant 62204066. The work of Kang Liu was supported in part by the National Natural Science Foundation of China under Grant 62202190 and Grant U22B2017; in part by the Hubei Natural Science Foundation under Grant 2023AFB237; and in part by the Knowledge Innovation Program of Wuhan-Shuguang. This article was recommended by Associate Editor I. H. R. Jiang. *(Corresponding author: Kang Liu.)*

Haoyang Sun, Cong Jiang, Xun Ye, and Kang Liu are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: hsun2023@hust.edu.cn; jiangconghust@hust.edu.cn; xunye@hust.edu.cn; kangliu@hust.edu.cn).

Dan Feng is with the Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: dfeng@hust.edu.cn).

Benjamin Tan is with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: benjamin.tan1@ucalgary.ca).

Yuzhe Ma is with the with Microelectronics Thrust, Hong Kong University of Science and Technology (Guangzhou), Guangzhou 510530, China (e-mail: yuzhema@hkust-gz.edu.cn).

Digital Object Identifier 10.1109/TCAD.2024.3468016

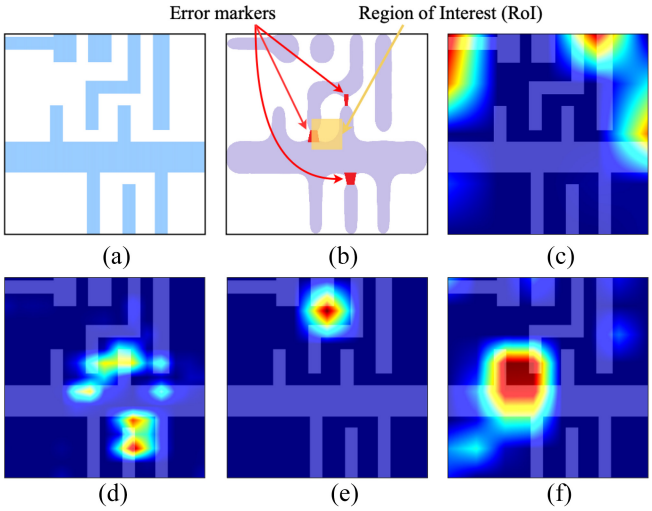


Fig. 1. (a) Layout clip, (b) lithography simulation results, and (c)–(f) grad-CAM explanations of various CNN-based hotspot detectors. Our architecture demonstrates the most accurate interpretation.

However, CNN-based hotspot detectors identify hotspots in a neural network fashion where the neural network learns a transforming function that takes as input a layout and straightforwardly outputs a hotspot or nonhotspot prediction. In other words, compared to lithography simulation and PM that operate with logic and self-explained inference steps, on the one hand, the CNNs perform hotspot detection on the grounds of no explicit evidence as no identification of the actual layout defects is involved; on the other hand, the computation throughout the neural network presents as a black box, as the settings of neural weights manifest no explicit meanings and are configured (i.e., *learned*) without human inspection in the loop.

To build trust in CNN-based hotspot detectors and move toward their practical applications in such a critical field of chip design, transparency of the results obtained from CNN models is required to explain why they predict what they predict. Thus, in this work, we ask the following question regarding lithographic hotspot detection: *Can CNN-based lithographic hotspot detectors reasonably explain their hotspot decisions for a given layout?*

We note that in the context of image classification with CNNs, researchers over the last few years proposed a series of methods [27], [28], [29], [30], [31], [32], [33], [34], [35] to discover the decision-making process beneath the hood of complex neural connections. For instance, the mainstream CNN interpretation methods based on class activation mapping (CAM) [31], [32], [33], [34], [35] obtain a heatmap of the input image and demonstrate which parts significantly impact CNN when assigning a specific label. Such highlighted input regions become the explanation for CNN classification. As exemplified in Fig. 2, the red areas of the heatmaps generated by grad-CAM [32] represent the crucial parts that a CNN looked at to predict a pen or a watch. These areas align with human interpretations of what a pen or watch looks like, indicating this network can accurately and reasonably explain its predictions for the pen and watch classes in this image.

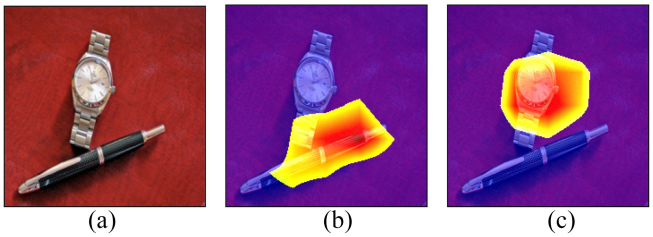


Fig. 2. Grad-CAM explanations of ResNet-50 for (a) input image when predicting, (b) pen class, and (c) watch class.

In contrast to image classification, where human semantic inspection forms the ground truth for CNN explanation, lithography simulation predicts hotspots based on exact defect regions, i.e., error markers (Fig. 1), where hotspots would occur due to surrounding problematic metals; this grants us access to ground truth evidence for hotspot prediction using a CNN-based hotspot detector. However, as far as we know, the interpretability of these CNN-based detection methods has *yet* to be examined. All the prior work developing CNN models for hotspot detection considers solely the prediction accuracy; whether the training process learns actual defect regions of hotspots and how they behave under CNN interpretation methods remains obscure. As shown in Fig. 1, we find that when commonly used CNN interpretation method, such as grad-CAM, is used with SoTA CNN hotspot detectors, their explanations of hotspot prediction significantly deviate from the ground truth error marker and its surrounding metals; this raises questions about the trust and reliability of existing hotspot detectors using CNN architectures.

In light of the fact that in lithography simulation, hotspots are determined by error markers based on certain rules, a well-trained CNN can presumably achieve 100% accuracy in classifying hotspots from error marker inputs. In this work, we seek to mimic lithography simulation in a CNN fashion such that an encoder-decoder architecture learns to transform a layout input to error markers, followed by a prediction network. Intuitively, such target learning propels the CNN to concentrate more on the problematic metals, especially their resulting defect regions, thus enhancing interpretability and detection accuracy. To facilitate hotspot root cause learning, we augment the training dataset, i.e., not only are the layout labels learned, but so are the error markers and their surrounding partial metal areas learned as intermediate representations. We introduce an error marker expansion scheme and attention modules used in the encoder that are necessary for efficient defect representation learning.

We make the following *contributions* in this article.

1) The *first* CNN architecture for lithographic hotspot detection that considers and demonstrates accurate explanations of its prediction results, where an encoder-decoder architecture learns the actual defect area as intermediate representations for hotspot classification.
2) A novel training data augmentation with expanded error markers and formulation of attention modules in the network architecture that facilitate precise defect learning.

3) Experimental results and insights into the prediction accuracy across a range of CNN-based hotspot detectors, where our proposed architecture exhibits the highest-hotspot accuracy (HA) of 99.78% and the lowest-false positive rate (FPR) of 5.29%.
4) Comprehensive evaluation of CNN-based hotspot detectors using SoTA CNN interpretation methods, showing that our architecture demonstrates the best qualitative and quantitative interpretability results.

The remainder of this article is organized as follows. We formulate our problem and briefly introduce the basics of our work in Section II. We provide related works in Section III and detail our proposed methods with architectural and training designs in Section IV. We set up the experiments in Section V, followed by experimental results in Section VI and their implications in Section VII. Section VIII concludes our work.

II. Preliminaries

A. Problem Statement

Generally, lithographic hotspot detection can be defined as a binary classification problem. Its main purpose is to identify whether a layout clip contains lithographic hotspots. When a CNN is used for hotspot detection, its prediction of a layout clip should be interpretable. We thus formulate the CNN-based hotspot detection problem as follows.

Problem (Interpretable CNN-Based Hotspot Detection): Given a set of layout clips containing both hotspots and nonhotspots, along with their corresponding error markers and ground truth labels that are both obtained from lithography simulation, a CNN-based hotspot detector is trained to achieve the following goals:

1) Detection Accuracy: the CNN-based hotspot detector should be able to identify as many real hotspot clips as possible while minimizing the number of actual nonhotspot clips misclassified as hotspots.

2) Interpretability: the CNN architecture should rely on the layout's actual defect regions, as reflected by the simulated error markers, for hotspot prediction; such important input regions affecting CNN prediction can be obtained by applying commonly used CNN interpretation methods.

We detail the evaluation metrics on the accuracy and interpretability of CNN-based hotspot detectors in Section V-D.

B. Convolutional Neural Networks

A CNN generally consists of an input layer, multiple hidden layers, and an output layer nested together. In classification tasks, it takes in an input $\boldsymbol{x} \in \mathbb{R}^N$ and outputs a probability distribution $\boldsymbol{y} \in \mathbb{R}^M$. The class associated with the highest probability $\arg\max_{i\in[1,M]} y_i$ is considered the prediction class of input $\boldsymbol{x}$. A CNN can be defined as a function of $F(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L}$ represents the network's weights in all L layers. The operation of each layer can be expressed by (1). Here, ϕ_l, $\boldsymbol{a}_l$, $\boldsymbol{W}_l$, and $\boldsymbol{b}_l$ are the activation function, the layer output, neural weights and biases of the lth layer, respectively

$$\boldsymbol{a}_l = \phi_l(\boldsymbol{W}_l\boldsymbol{a}_{l-1} + \boldsymbol{b}_l). \tag{1}$$

CNNs use backpropagation algorithms to optimize $\boldsymbol{\theta}$ such that a defined loss $\mathcal{L}$ over the training dataset $\mathcal{D}_{\text{train}}$ is minimized. Specifically, in convolutional layers the neural weights are in the form of a series of convolutional filters.

Despite the great success of CNNs in various tasks, the complex interactions between layers of neurons make it extremely difficult for humans to understand the rationale of its predictions hidden inside the computation flow. The black-box nature of CNNs poses severe risks to their many real-world applications that are risk- or cost-sensitive, such as chip design.

C. CNN Interpretation Methods

Explaining the internal function of CNN's classification behaviors is still under investigation. Many approaches have been proposed to explain CNNs by attempting to identify discriminative regions within an input that are used to predict a specific class. Gradient visualization [27], [28], perturbation-based [29], [30], and CAM-based methods [31], [32], [33], [34], [35] are the three main categories of CNN interpretation methods.

Gradient-Based Methods [27], [28] use the derivative of a target class's prediction score $S(\boldsymbol{x})$ w.r.t. the input $\boldsymbol{x}$ to generate a saliency map $\boldsymbol{M}_{\text{grad}}$, which highlights input pixels with a strong influence on the CNN prediction, as shown in (2). Enhancements [28] to the original gradient-based interpretation are made to sharpen the saliency maps. However, gradient-based methods generally suffer from low quality and noise and are less class-discriminative for CNN interpretation

$$\boldsymbol{M}_{\text{grad}} = \frac{\partial S(\boldsymbol{x})}{\partial \boldsymbol{x}}. \tag{2}$$

Perturbation-Based Methods [29], [30] analyze the alteration in the prediction scores by masking or perturbing the original input to identify class-discriminative regions. These methods are model-agnostic but time-consuming as enormous input perturbations necessitate extensive computations. Additionally, these approaches typically require extra regularization to determine the minimum area to perturb.

LIME [29] is a representative perturbation-based interpretation method that uses simple and interpretable sparse linear models to approximate the local decision surface of a CNN. Inputs are segmented into patches called super-pixels. By turning these super-pixels on and off, perturbed data and CNN prediction scores are obtained. These super-pixels are weighted by perturbation distance and fitted into a linear model, whose coefficients provide an insight into the relative importance of each super-pixel in predicting a specific class.

CAM-Based Methods [31], [32], [33], [34], [35] are now SoTA in CNN interpretation. Feature maps of deeper layers in a CNN are known for capturing higher-level visual constructs of input, and the last convolutional layer typically contains high-level semantics and detailed spatial information. As shown in (3), a weighted sum of these visual patterns of each feature map channel $\boldsymbol{A}_k$ in the last convolutional layer is referred to as the class activation map and usually upsampled to the input size and denoted as $\boldsymbol{M}_{\text{CAM}}$. Such a class activation

map for a particular class indicates the discriminative regions in the input used by CNN for classification

$$\boldsymbol{M}_{\text{CAM}} = \text{Upsample}\left(\text{ReLU}\left(\sum_k \alpha_k \boldsymbol{A}_k\right)\right). \tag{3}$$

Specifically, the weight α_k represents the *importance* of the kth feature map for predicting a target class. Various CAM-based interpretation methods differ primarily on how they determine α_k, either from the gradient information of the prediction score w.r.t. the features maps or prediction score changes when applying these feature maps to inputs (i.e., gradient-free).

Grad-CAM [32] uses a channel-wise global average of the gradients w.r.t. the feature maps in the last convolutional layer to represent the importance for a target class. A weighted average of the gradients is used in grad-CAM++ [33], which considers cases where multiple occurrences of an object occur and yield better localization. Unlike previous methods that only consider global information of each feature map, LayerCAM [35] uses pixel-wise gradients to highlight the different importance of each location in the feature maps, which retains fine-grained details of the target class.

In gradient-free approaches [34], the global contribution of corresponding input features is used instead of the local sensitivity measurement (i.e., gradient). In score-CAM [34], feature maps are upsampled and masked on the original input, and the changes in prediction score compared with the original input are obtained as the coefficient α_k.

III. Related Work

Lithographic Hotspot Detection: Various methods have been proposed for detecting lithographic hotspots during the design phase to improve manufacturability. These methods include lithography simulation [8], [9], which is highly accurate but time-consuming, PM-based techniques [10], [11] that are faster than simulation, and ML solutions [12], [13] that offer satisfactory accuracy and improved generalization than PM. Recently, DL has been applied for hotspot detection [14], [15], [16], [17], [18], where deep neural networks analyze the extracted layout features or entire layout clips for hotspot prediction, producing higher accuracy and generalization than ML. These DL techniques include CNNs [14], [15], [16] (we use as baselines in this article), graph neural networks (GNNs) [17], and neural architecture search (NAS)-based networks [18]. We note that more sophisticated architectures like GNN or NAS still require more efficient and reliable interpretation methods. The abovementioned work performs lithographic hotspot detection on a single layout clip; other work detects hotspots from an entire layout and combines localization and detection [19], [20], [21]. Training efficiency is also studied via adaptive training data sampling [36], [37]. Another line of work, from a perspective different than detection accuracy and training efficiency, investigates the security and robustness of DNNs in lithographic hotspot detection [25], [26], [38], [39], where attacks and defenses on adversarial and backdoored layouts are studied. Our work differs from previous studies and considers the interpretability of the underlying CNNs used for lithographic hotspot detection for the first time.

CNN Interpretation Methods: Numerous methods have been studied to explain how CNNs make decisions. Early work on gradient-based approaches [27], [28] generates saliency maps specific to a class through gradient backpropagation, which suffers from noise effects. Perturbation-based approaches [29], [30] examine how small changes to the input patches affect the model's behavior, which helps identify critical input features for a target prediction. However, this method is time-consuming as it needs to examine numerous combinations of input perturbations. In this article, we adopt the widely used CAM-based methods [31], [32], [33], [34], [35] for CNN interpretation, which are SoTA and have proven their efficiency and reliability in the computer vision community.

IV. Proposed Method

A. Overview

The shapes and positions of adjacent metals, such as corner-to-corner distances, jogs, and line-end positions, cause complex light interactions during lithography, resulting in printing defects. Learning from these crucial defect regions, instead of directly from layout clips, will improve the interpretability and accuracy of a CNN-based hotspot detector. However, as far as we know, all the prior work on CNN-based hotspot detection neglects such important defect information, only using this (obtained from lithography simulation) for ground truth labeling in dataset preparation and excluding it from the training process. Thus, in this work, we propose an interpretable end-to-end CNN-based hotspot detector that learns to locate defect regions for high accuracy and improved interpretability when means of CNN interpretation are applied.

Inspired by the lithography simulation flow where error markers are simulated for subsequent layout ground truth labeling, we resemble this process in a neural network fashion where an image-to-image architecture is designed to learn the error markers from inputs of layout clips. A following classification component is used to predict hotspots from the learned defect maps, as shown in Fig. 3. Instead of naively representing the exact error markers as intermediate layer outputs, we expand the defect areas as the training target by a fixed amount to ease the learning of their shapes and locations. We also include coordinate attention blocks during layout feature extraction, which is found to promote interpretability. We describe our proposed method in detail below.

B. Training Data Augmentation

Compared to existing work on CNN hotspot detection, we augment the training data with lithography error markers, in addition to their layout clips and corresponding hotspot/nonhotspot labels. We aim to learn the actual defect areas for each clip during training as intermediate representations of the CNN, which first requires a screening process of all the simulated error markers. We only consider error markers that highly overlap with the RoI as the exact defects,

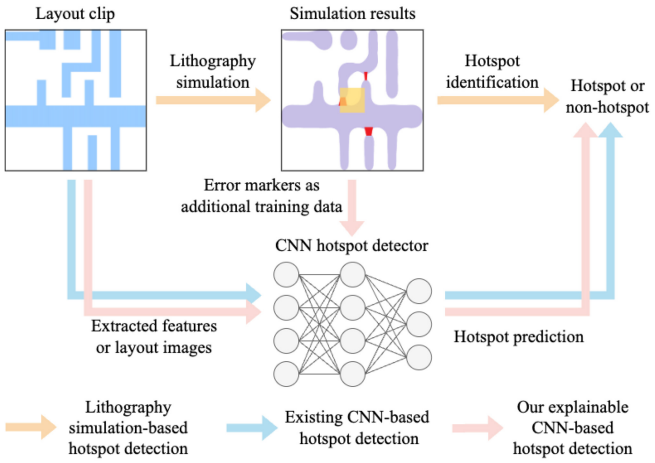


Fig. 3. Illustration of our explainable CNN-based hotspot detection, compared with lithography simulation-based and existing CNN-based hotspot detection.

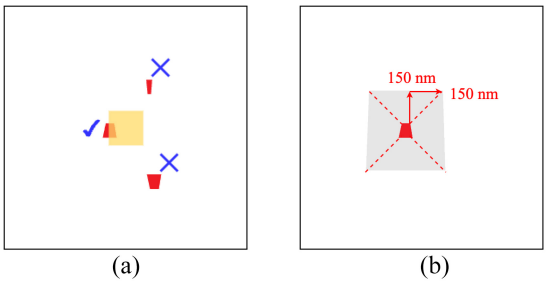


Fig. 4. Screening and expansion of error markers. (a) We preserve error markers (in red) that have an intersection with the RoI (in yellow) over a predefined threshold as the actual defect areas and screen out the rest; we differentiate them with checks and crosses. (b) We preprocess the preserved error markers by stretching their vertices outwards by 150 nm.

which have full neighboring information when being lithography simulated. It is not the case for error markers outside the RoI. Thus, for hotspots, we preserve error markers that intersect with the RoI over 30% as in [25] and [26], and screen out the rest. For nonhotspots that have no printing defects w.r.t. RoI, an empty layout clip is included as the learning target.

The preserved error markers denote the actual defect regions where the hotspot is made, whereas its surrounding metals are the neighboring patterns that cause the hotspot. Thus, an interpretable CNN-based hotspot detector is supposed to focus on the actual defect as well as its ambit for hotspot prediction. As a result, we expand the error markers by a fixed amount as the learning target so that the defect region and surrounding metal areas are both emphasized in training using different weights. Specifically, we stretch out the corners of the error marker by extending their horizontal and vertical coordinates far from the original with 150 nm (we use layout clips of size 1110 nm $\times$ 1110 nm), as shown in Fig. 4(b). This error marker expansion roughly aligns with the typical lithographic ambit of $\sim$400 nm [40], and we empirically find to have the best learning and interpretability results.

Our training data augmentation requires little additional effort, given that these error markers are pre-existing results after lithography simulation.

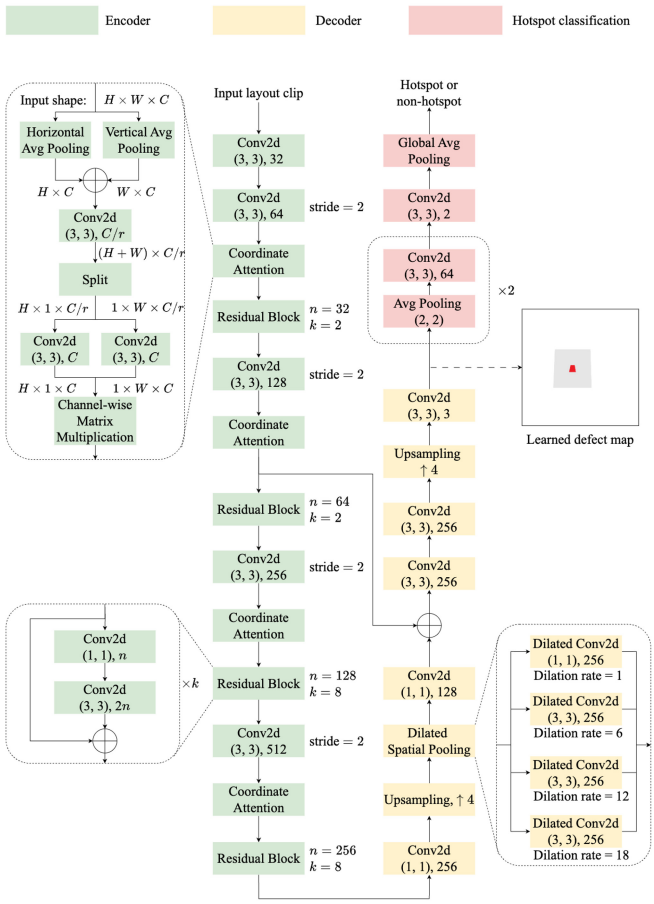


Fig. 5. Architecture of our proposed CNN-based hotspot detector.

C. Network Architecture

Our CNN hotspot detector, as shown in Fig. 5, consists of an encoder-decoder network that learns the actual defect locations from layout clips and a following fully convolutional network that predicts hotspots and nonhotspots from this intermediate defect information.

For layout feature encoding, we acquire wisdom from the computer vision community, where the YOLO algorithms [41] are popular for real-time object detection due to their compelling feature extraction abilities and efficiency with less parameter training. We adapt part of the Darknet-53 network from YOLOv3 [41] to encode a compressed latent from layout clips. We add residual connections within the encoder, integrating high-level and low-level features for improved representation learning.

On the decoder side, we adopt the decoder network introduced in DeepLabv3+ [42] that extends its predecessors in semantic segmentation and constructs the expanded error markers, where core defect pixels and their expanded areas are segmented from the background. We adhere to the conventions in semantic segmentation, where the output of the decoder learns a probability map. In our case, a three-channel defect map is generated, and each channel corresponds to the probabilities of input pixels that belong to the original error marker, its expanded area, and the layout background, respectively. The convolutional layers of the encoder preserve

much spatial information of the layout clips, and the decoder leverages intermediate representations from the encoder to refine the boundaries of the learned defect regions using a shortcut. To further capture information on a larger scale of the feature maps, spatial pyramid pooling with dilated convolution is applied. Specifically, the encoder learns compressed feature maps with dimensions that are 1/16 in both vertical and horizontal dimensions of the input layout clip, followed by the decoder that transforms such extracted feature maps back to the original input size as learned defect maps.

Learned defect regions and their surroundings retain rich spatial information crucial for hotspot prediction. Thus, we use fully convolutional layers along with global pooling for classification. The last convolution consists of two filters that each extract hotspot and nonhotspot information.

We note that the sole inputs to our CNN hotspot detector are the layout clips, and the preprocessed error markers, as part of the training data, are intermediate learning objectives during the training phase for learned defect map generation and are no longer required for inference.

D. Coordinate Attention

Unlike general image classification, subtle nm-level variations in layout clips can drastically alter their printability, i.e., hotspot or nonhotspot labels, by varying polygon spaces, widths, corner locations, or jogs [24]; this suggests more fine-grained capture of spatial information in a larger receptive field (involving multiple polygons) is required for hotspot detection.

In CNNs, attention mechanisms [43], [44] are widely used to enhance neural network performance by mimicking human perception and paying more attention to the critical information within a receptive field rather than analyzing without discrimination. For instance, in CNN hotspot detection, CBAM attention [43] is used to improve detection accuracy [16]. In this work, we adopt the coordinate attention mechanism [44], which can capture more information from the global spatial plane than CBAM that primarily analyzes local features. Coordinate attention comprises two stages: 1) coordinate information embedding and 2) coordinate attention generation.

Coordinate Information Embedding: Given an intermediate feature representation $\boldsymbol{X} \in \mathbb{R}^{H\times W\times C}$ of a neural network, coordinate attention first embeds spatial information along horizontal and vertical directions separately with average pooling, such that

$$\boldsymbol{Z}_1^H(i,c) = \frac{1}{W}\sum_{j=1}^{W}\boldsymbol{X}(i,j,c) \tag{4}$$

$$\boldsymbol{Z}_1^W(j,c) = \frac{1}{H}\sum_{i=1}^{H}\boldsymbol{X}(i,j,c). \tag{5}$$

i, j, and c are vertical, horizontal, and channel axial indices. $\boldsymbol{Z}_1^H \in \mathbb{R}^{H\times C}$ and $\boldsymbol{Z}_1^W \in \mathbb{R}^{W\times C}$ combine to aggregate spatial information more delicately with direction awareness.

Coordinate Attention Generation: $\boldsymbol{Z}_1^H$ and $\boldsymbol{Z}_1^W$ are further concatenated and processed by 1-D convolutional layers g_1 with (C/r) filters followed by an activation function σ_1 (e.g., ReLU), such that attention information of $\boldsymbol{X}$ is learned and extracted as denoted by $\boldsymbol{Z}_2 \in \mathbb{R}^{(H+W)\times(C/r)}$

$$\boldsymbol{Z}_2 = \sigma_1\big(g_1\big(\text{Concat}\big(\boldsymbol{Z}_1^H, \boldsymbol{Z}_1^W\big)\big)\big). \tag{6}$$

Here, r is the reduction ratio for controlling the number of convolutional filters of g_1, which varies the learning capacity and computational cost.

To generate attention maps for feature representation $\boldsymbol{X}$ in separate horizontal and vertical dimensions with the same number of channels C, condensed attention information $\boldsymbol{Z}_2$ splits into $\boldsymbol{Z}_2^H \in \mathbb{R}^{H\times(C/r)}$ and $\boldsymbol{Z}_2^W \in \mathbb{R}^{W\times(C/r)}$. 1-D convolutions g_2^H and g_2^W, each with C filters, decode horizontal and vertical attention into C channels, represented by $\boldsymbol{Z}^H \in \mathbb{R}^{H\times C}$ and $\boldsymbol{Z}^W \in \mathbb{R}^{W\times C}$, respectively. Here, σ_2 is sigmoid activation

$$\boldsymbol{Z}^H = \sigma_2\big(g_2^H\big(\boldsymbol{Z}_2^H\big)\big) \tag{7}$$

$$\boldsymbol{Z}^W = \sigma_2\big(g_2^W\big(\boldsymbol{Z}_2^W\big)\big). \tag{8}$$

Thus, the output $\boldsymbol{Y}$ of the attention block is calculated as follows, where each element of the feature representation $\boldsymbol{X}$ is weighted by its corresponding horizontal and vertical attention $\boldsymbol{Z}^H$ and $\boldsymbol{Z}^W$ in the spatial plane:

$$\boldsymbol{Y}(i,j,c) = \boldsymbol{X}(i,j,c) \times \boldsymbol{Z}^H(i,c) \times \boldsymbol{Z}^W(j,c). \tag{9}$$

Specifically, we add coordinate attention in the encoder after each convolutional block that reduces feature map sizes (i.e., convolutional layers with stride $= 2$) for more efficient feature extraction.

E. Network Training

In hotspot detection, we are primarily in favor of high HA as well as overall accuracy. In light of the imbalanced dataset where hotspot samples are usually fewer than nonhotspots, various training strategies [14], [25], [26], [39] are proposed for more accurate hotspot feature learning. For instance, biased learning is introduced in [14] to shift the boundary of the hotspot learning target to increase HA. A class weight parameter used in [25], [26], and [39] weighs the loss terms between nonhotspots and hotspots in the loss function, causing the network to pay more attention to samples from the under-represented class (i.e., hotspots). In this work, we use focal loss [45] in training on top of the class weight, which helps the model focus more on hotspot samples and those hard to learn (i.e., with high-training loss) in both classes. In addition to the binary classification of hotspots and nonhotspots, our architecture also learns a three-class segmentation map of the defect areas as described in Section IV-C. Therefore, we also apply focal loss in learning the defect maps.

Considering an N-category classification problem, $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are the ground truth and predicted probability vectors of a given input, and y_n and $\hat{y}_n$ are their nth element, respectively. We express focal loss FL as shown in (10), where δ_n is the class weight for balancing the loss terms of different classes and $\gamma > 0$ is the modulating factor that relatively increases the losses for predictions far from the ground truth, i.e., hard-to-learn samples. Specifically, in hotspot/nonhotspot classification ($N = 2$), the hotspot has a larger δ than the nonhotspot in the classification loss, as denoted by $\text{FL}_{\text{hotspot}}$. In defect map

segmentation ($N = 3$), the focal loss $\mathrm{FL}_{\mathrm{defect}}$ is summed up over all the pixels of the defect map

$$\mathrm{FL}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{n=1}^{N} \delta_n \left(1 - \hat{y}_n\right)^{\gamma} y_n \log\left(\hat{y}_n\right). \tag{10}$$

The total training loss $\mathcal{L}$ of our CNN hotspot detector for a given layout clip is shown in (11). Here, β_1 and β_2 are scalars balancing the loss terms for hotspot classification and defect map learning, respectively

$$\mathcal{L} = \beta_1 \mathrm{FL}_{\mathrm{hotspot}} + \beta_2 \mathrm{FL}_{\mathrm{defect}}. \tag{11}$$

V. Experimental Setup

A. Layout Dataset

We evaluate baseline CNN hotspot detectors and our architecture on the same dataset used in prior work [24], [25], [26], [39], which is prepared based on the freely available 45-nm FreePDK [46]. We notice other datasets used for hotspot detection, such as `Industry1-3` and `ICCAD-2012`, which provide layouts and their labels but without corresponding error markers required by our architecture. We cannot perform full lithography simulation on these datasets for error markers due to the unavailability of their proprietary PDK parameters. On our dataset, we use Calibre [22] to perform lithography simulation for error markers and obtain ground truth labels for each layout clip. We prepared and collected 19 9561 training hotspots, 24 8519 training nonhotspots, 888 validation hotspots, and 15 405 validation nonhotspots.

All the layout clips and error markers are in GDSII format. To facilitate the training of CNN hotspot detectors, we convert GDSII files to images with a size of 1110×1110. Metal polygons in layout clips are indicated with a pixel intensity of 1, and the rest background areas are 0-valued. As intermediate training objectives, each pixel of the defect maps with expanded error markers is represented by a 3-D one-hot vector, denoting it belongs to either the error markers, the expanded areas, or the layout background. For end-to-end training models, we downsize layout clips and defect maps from 1110×1110 to 256×256 to reduce computation cost.

B. Baseline CNN Hotspot Detectors

We compare our architecture with three SoTA CNN-based hotspot detectors from TCAD'19 [14], TCAD'21 [15], and TCAD'22 [16]. TCAD'19 performs discrete cosine transform (DCT) on the layout clips for frequency feature extraction and uses a CNN comprising four convolutional layers and two fully connected layers for hotspot detection. TCAD'21 proposes a two-branch binary neural network (BNN) where each branch consists of 8 and 10 convolutional layers grouped in 3 and 4 residual blocks, respectively. Predictions from these two branches combine to perform hotspot detection using a meta-classifier. TCAD'22 integrates CBAM attention into five inception blocks for improved feature extraction and classification.

C. CNN Interpretation Methods

We apply five interpretation methods for the interpretability analysis of CNN-based hotspot detectors, including both perturbation-based method LIME and CAM-based methods grad-CAM, grad-CAM++, score-CAM, and LayerCAM. We implement LIME with the lime package[1] and limit the number of perturbed samples to 500 for balanced speed and accuracy when regressing the linear model. We use the tf-keras-vis library[2] to implement CAM-based interpretation, where feature maps from the last convolutional layer are used for CNN interpretation.

D. Evaluation Metrics

We evaluate hotspot detection accuracy and interpretability performance of CNN-based hotspot detectors using the following metrics.

Hotspot Detection: We measure hotspot detection performance using 1) HA and 2) FPR.

1) *HA:* The percentage of ground truth hotspot clips correctly classified as hotspots. HA represents the capability of the hotspot detector to catch the real hotspots for further mask optimization.
2) *FPR:* The percentage of ground truth nonhotspot clips misclassified as hotspots. FPR quantifies the unnecessary and wasted effort for relithography simulation when a benign nonhotspot is flagged as a hotspot.

Interpretability: We analyze the interpretability of CNN-based hotspot detectors with both qualitative visualizations and quantitative metrics.

1) *Visual Explanation:* We visualize the CNN explanation, denoted by $\boldsymbol{E}$, by element-wisely multiplying the importance map $\boldsymbol{M}$ provided by a CNN interpretation method [e.g., $\boldsymbol{M}_{\mathrm{CAM}}$ from CAM-based interpretation in (3)] with the original layout clip $\boldsymbol{x}$. Specifically, we set 20% of the pixels with the lowest significance in the importance map to 0 before calculating the visual explanation for noise impact alleviation [34]

$$\boldsymbol{E} = \boldsymbol{M} \odot \boldsymbol{x}. \tag{12}$$

The interpretability of a CNN correlates strongly with its decision confidence, as better-interpretation hints at more accurate feature representation and, consequently, higher-prediction scores for a target class. Therefore, we qualitatively evaluate the interpretability of a CNN hotspot detector using 1) an average drop in scores (ADSs) and 2) an increase in scores (ISs), as used in prior work [33], [34]. We also calculate 3) the intersection of the union between the importance map and hotspot root cause areas, which, in our case, are defect maps with error markers and surrounding areas.

1) *ADSs:* A CNN with ideal interpretability is assumed to have a visual explanation containing all the essential information required to predict a particular class. If the CNN results in a lower-prediction score with its visual explanation as input instead of the original input, it

[1] https://lime-ml.readthedocs.io/en/latest/index.html
[2] https://keisen.github.io/tf-keras-vis-docs/

indicates that the CNN has missed out on some crucial input information for classification. Therefore, a CNN hotspot detector with good interpretability should exhibit a minimal reduction, if there is one, in its prediction scores when provided with an input of its explanation. From this perspective, we calculate the average drop in CNN prediction scores for the hotspot class over real hotspot clips

$$\text{ADS} = \frac{1}{T}\sum_{t=1}^{T}\frac{\max\left(0, y_t^{hs} - o_t^{hs}\right)}{y_t^{hs}}. \tag{13}$$

Here, y_t^{hs} and o_t^{hs} are the confidence scores for predicting the hotspot class with inputs of the original layout clip $\boldsymbol{x}_t$ and the visual explanation $\boldsymbol{E}_t$ (provided by a CNN interpretation method), respectively. T is the total number of ground truth hotspot clips in the validation dataset.

2) *ISs:* Irrelevant information embedded in the input can hurt the CNN prediction. A CNN with good interpretability is likely to exclude this irrelevant information in its explanations and, consequently, increase the prediction scores of the target class when the explanation is taken as input. Thus, complementary to ADS, we calculate the ratio of hotspot clips that increase the CNN prediction scores for the hotspot class when the explanation is inputted instead of the original layout clip

$$\text{IS} = \sum_{t=1}^{T}\left(\frac{1_{y_t^{hs} < o_t^{hs}}}{T}\right). \tag{14}$$

Here, $1_{\text{condition}}$ is an indicator function that returns 1 if the condition is met or otherwise 0.

3) *Intersection of Union (IoU):* An interpretable CNN hotspot detector should focus on the hotspot root cause areas for hotspot prediction, i.e., error markers and their surroundings as represented by the defect maps (with expanded error markers). Therefore, we calculate the intersection of the union between the importance maps of the CNN and ground truth defect maps

$$\text{IoU} = \frac{1}{T}\sum_{t=1}^{T}\frac{\boldsymbol{B}_t \cap \boldsymbol{M}_t}{\boldsymbol{B}_t \cup \boldsymbol{M}_t}. \tag{15}$$

Here, $\boldsymbol{M}_t$ is the importance map for layout clip $\boldsymbol{x}_t$, and $\boldsymbol{B}_t$ is its defect map.

E. Experimental Platform and Training Hyperparameters

We conduct experiments on Ubuntu 20.04 with an Intel Xeon W-3335 CPU and NVIDIA GeForce RTX 3090 GPU. We implement our codes in Python 3.9 and TensorFlow 2.11. We use Adam optimizer in training with an initial learning rate of 0.001, which is reduced by 1/3 with patience of 5 epochs upon loss plateaus. We set the batch size to 48 and limit the training process to 100 epochs.

In our training loss, as defined in (10) and (11), we set the modulating factor γ to 2; we assign the class weight δ to 0.8 and 0.2 for hotspots and nonhotspots in $\text{FL}_{\text{hotspot}}$ and 0.4, 0.2, and 0.2 for error markers, expanded areas, and the background in $\text{FL}_{\text{defect}}$. β_1 and β_2 are both 1. In training baseline hotspot detectors, we use class weights of 10 and 1 for hotspots and nonhotspots in their loss functions.

To stabilize training and interpretation results of our hotspot detector, we also add a regularizer to the last convolutional layer, where we maximize the largest elements of the feature map for the hotspot channel after normalization.

TABLE I
HA, FPR, AND INFERENCE TIME PER LAYOUT CLIP OF VARIOUS CNN-BASED HOTSPOT DETECTORS COMPARED WITH LITHOGRAPHY SIMULATION

		HA	FPR	Inference
Lithography Simulation		Ground Truth		3779.0 ms
CNN-based Hotspot Detectors	TCAD'19	96.70%	10.79%	4.03 ms
	TCAD'21	98.79%	6.00%	1.34 ms
	TCAD'22	99.09%	6.74%	2.53 ms
	Ours	**99.78%**	**5.29%**	4.69 ms

VI. Experimental Results

A. Accuracy and Interpretability Analysis

Accuracy: We show in Table I the detection accuracy of our CNN hotspot detector compared with three baseline architectures, i.e., TCAD'19, TCAD'21, and TCAD'22, where our architecture achieves the highest HA of 99.78% and the lowest FPR of 5.29%. We observe that end-to-end learning outperforms learning from hand-designed features, as evidenced by the improved HA and FPR of TCAD'21, TCAD'22, and our architecture than TCAD'19, where the former learns from entire layout clips for feature extraction, while the latter uses layout's DCT coefficients as fixed input features. Further, our architecture stands out among end-to-end learning methods, as we guide the CNN specifically to learn hotspot root cause features (i.e., defect maps) instead of untargeted feature extraction used in TCAD'21 and TCAD'22.

We also report the inference time required by various CNN-based hotspot detectors and lithography simulation in classifying one layout clip, as shown in Table I. On average, lithography simulation requires 3779 ms to classify one layout clip, whereas CNN-based hotspot detectors are much faster by three orders of magnitude, requiring 1.34–4.69 ms. We find that TCAD'21 using a BNN is the most time-efficient due to its binarized weights and reduced computational complexity, costing 1.34 ms per clip. Our architecture requires 4.69 ms for one inference run of a layout clip. Despite the extra time cost, we achieve the highest-detection accuracy among all CNN-based solutions.

Interpretability: We provide visual explanations of our CNN hotspot detector and three baseline models using various CNN interpretation methods, namely, LIME, grad-CAM, grad-CAM++, score-CAM, and LayerCAM in Fig. 6. Our CNN model demonstrates the most consistent visual explanations across all interpretation methods, and the highlighted regions highly overlap with ground truth defect maps. TCAD'19 tends to focus on layout corners or boundaries in CAM-based methods where no actual error marker occurs. TCAD'21

TABLE II
INTERPRETABILITY ANALYSIS (%) OF VARIOUS CNN-BASED HOTSPOT DETECTORS USING DIFFERENT CNN INTERPRETATION METHODS

	LIME			Grad-CAM			Grad-CAM++			Score-CAM			LayerCAM		
	ADS	IS	IoU	ADS	IS	IoU	ADS	IS	IoU	ADS	IS	IoU	ADS	IS	IoU
TCAD'19	**13.97**	9.0	7.91	226.83	1.01	0.61	309.19	0.56	0.47	169.91	0.79	7.66	233.7	0.45	1.40
TCAD'21	97.53	0.0	6.89	170.12	0.0	25.57	163.30	0.11	23.50	145.69	0.23	24.78	152.18	0.11	42.74
TCAD'22	50.66	0.0	7.12	168.41	0.0	12.52	153.56	0.53	24.36	75.64	1.76	43.73	108.40	0.12	53.21
Ours	26.28	**38.0**	**7.97**	**11.46**	**37.50**	**53.86**	**11.42**	**37.47**	**53.38**	**8.83**	**55.97**	**44.52**	**11.28**	**38.24**	**54.04**

TABLE III
ACCURACY, INTERPETABILITY (USING GRAD-CAM), TRAINING EPOCHS, AND INFERENCE TIME PER LAYOUT CLIP OF VARIOUS NETWORKS WITH DIFFERENT COMPONENT COMBINATIONS. E-M: ERROR MARKERS

Network Component Combinations	Learning E-M	HA	FPR	ADS	IS	IoU	Training Epochs	Inference
Encoder w/o Attention	No	98.89%	6.32%	104.42%	0.0%	0.10%	84	**2.72 ms**
Encoder w/ CBAM Attention	No	99.19%	6.64%	102.72%	0.12%	0.06%	81	2.75 ms
Encoder w/ Coordinate Attention	No	99.60%	6.56%	106.44%	0.0%	0.23%	85	2.93 ms
Encoder w/ Coordinate Attention + Decoder	Yes	**99.78%**	**5.29%**	**11.46%**	**37.50%**	**53.86%**	**79**	4.69 ms

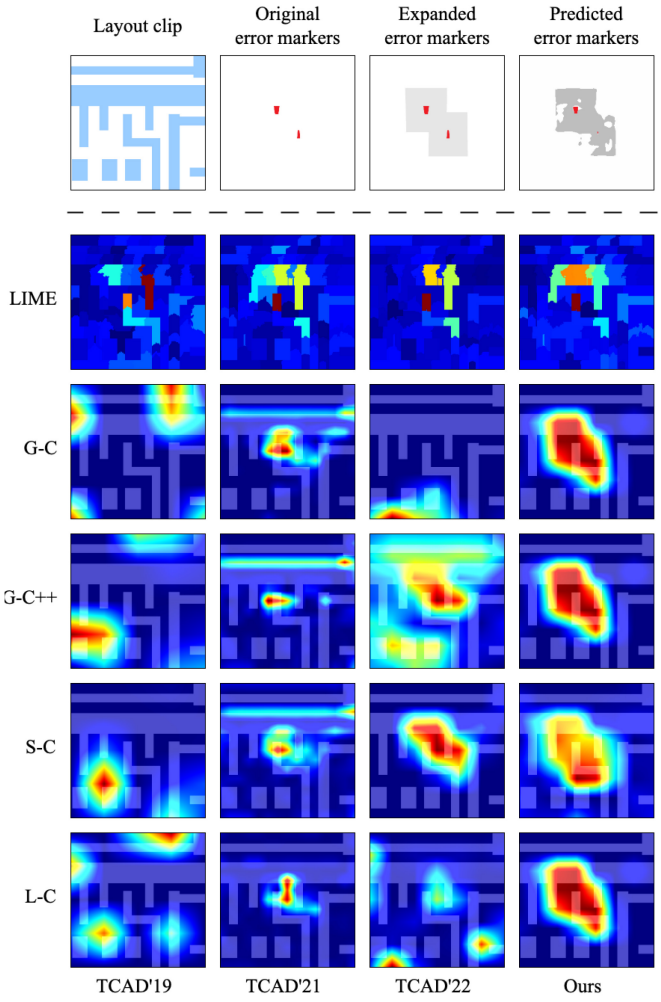


Fig. 6. Visual explanations of various network architectures with different CNN interpretation methods. G-C: Grad-CAM, G-C++: Grad-CAM++, S-C: Score-CAM, and L-C: LayerCAM.

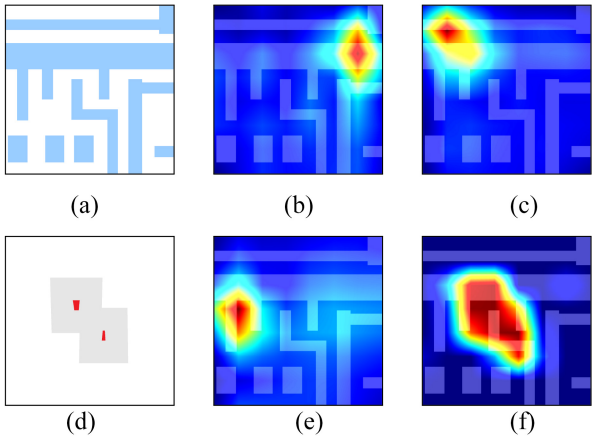


Fig. 7. (a) Layout clip, (d) expanded error markers, and (b), (c), (e), and (f) grad-CAM explanations of various networks with different component combinations. E-M: Error marker and C-A: Coordinate attention.

occasionally relies on metal boundaries (grad-CAM, grad-CAM++, and score-CAM) that have nothing to do with the defects. TCAD'22 using score-CAM provides a visual explanation similar to ours but performs worse using other CAM-based interpretation methods; for instance, layout corners and boundaries are looked at for hotspot prediction. All these models produce similar LIME-based visual explanations that the highlighted regions are small and close to the defect areas but are not accurate enough.

Our qualitative interpretability analysis aligns with the visual explanations. The ADS (the lower, the better) and IS and IoU (the higher, the better) scores of our CNN hotspot detector and three baselines using various interpretation methods are shown in Table II. Our CNN architecture exhibits the best interpretability across all CAM-based methods regarding all metrics. Overall, it is the best in LIME interpretation by achieving the highest IS and IoU scores and the second-to-best-ADS score. TCAD'19 is generally less interpretable than TCAD'21 and TCAD'22 in CAM-based interpretation, as reflected by its highest ADS and lowest-IoU scores.

B. *Effects of Coordinate Attention*

We evaluate the use of attention modules in the encoder for hotspot feature extraction, as presented in Table III and Fig. 7.

TABLE IV
ACCURACY AND INTERPRETABILITY (USING GRAD-CAM) OF OUR ARCHITECTURE THAT LEARNS WITH ORIGINAL OR EXPANDED ERROR MARKERS

	HA	FPR	ADS	IS	IoU
Original error marker	95.97%	6.7%	130.09%	0.22%	10.90%
Expanded error marker	**99.78%**	**5.29%**	**11.46%**	**37.50%**	**53.86%**

Our findings show that incorporating either CBAM or coordinate attention can increase HA, with coordinate attention yielding the best HA (Table III). However, in contrast to detection accuracy, attention hardly improves interpretability, as seen from the equally high ADS and low IS and IoU scores of encoders with or without attention (Table III). Fig. 7 further illustrates that the highlighted regions in visual explanations all deviate from the actual defect areas, regardless of using attention. Such lack of improvement in interpretability when using attention in CNN reveals the challenge of achieving high interpretability for end-to-end learning, as no guarantees are made on learning genuine features for classification. In terms of training and inference cost, each encoder and attention combination requires similar training epochs before convergence and similar inference runtime for each layout clip, as shown by the first three rows in Table III, when no error markers are learned without a decoder.

C. Effects of Learning Error Markers

We further evaluate the detection accuracy and interpretability of learning the error markers as intermediate features on top of using coordinate attention in the encoder. Our CNN architecture that uses a decoder and learns the ground truth defect maps as hotspot features significantly improves interpretability, as visualized in Fig. 7 and detailed in Table III. In Fig. 7, the highlighted regions in the visual explanation of our CNN align closely with actual defect areas, and in Table III, the ADS, IS, and IoU scores are dramatically improved. In addition, our architecture also achieves the highest HA and lowest FPA compared to all prior architectures with or without attention.

By adding an additional decoder for error marker learning, our architecture requires slightly larger inference time than other networks using only encoders and attention modules. However, this extra learning target accelerates the optimization process as reflected by the fewer training epochs required for training loss convergence, as shown in Table III.

D. Effects of Error Marker Expansion

We demonstrate the use of error marker expansion in our CNN hotspot detector and its effect on the detection accuracy and interpretability in Table IV and Fig. 8. We show that learning the proportionally expanded error markers, instead of the original ones, vastly enhances detection accuracy and interpretability. We also find that learning the minuscule original error markers poses significant challenges for the CNN, as shown by the largely noisy learned defect maps in Fig. 8 and as reflected by its low-detection accuracy and poor interpretability performance in Table IV.

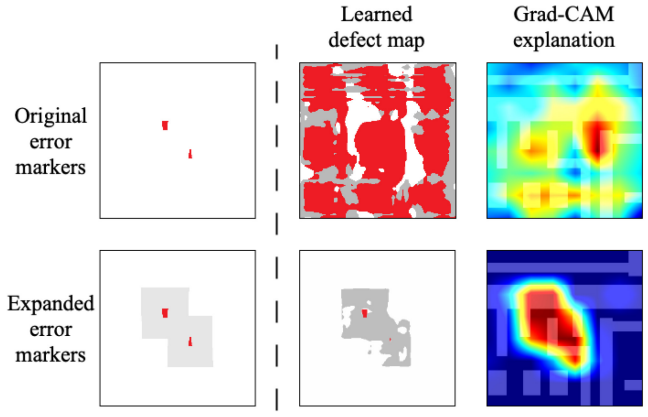


Fig. 8. Learned defect maps and grad-CAM explanations of our architecture that learns with original or expanded error markers.

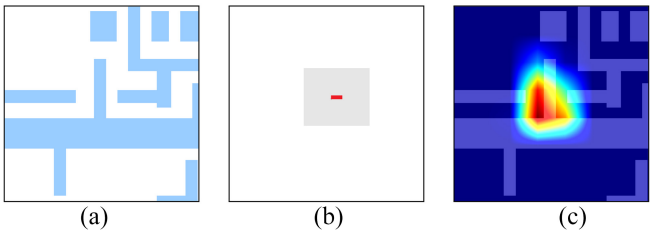


Fig. 9. (a) Exemplar hotspot layout clip mispredicted as a nonhotspot, (b) expanded error marker, and (c) grad-CAM explanation. E-M: Error marker and G-C: Grad-CAM.

E. Identifying Mispredicted Hotspots

We showcase how we can further enhance our CNN model by using our reliable visual explanation to identify misclassified hotspots in addition to our already high HA (99.78%). Fig. 9(a) shows a hotspot clip misclassified as a nonhotspot by our CNN. Although our CNN's focus is on the correct layout areas [Fig. 9(c), as confirmed by the ground truth error marker in Fig. 9(b)], the classification component still falsely predicts it as a nonhotspot. However, our visual explanation using grad-CAM [Fig. 9(c)] contradicts the prediction result, suggesting there might be a missed catch (i.e., an actual hotspot), and it is necessary to conduct further lithography simulation for verification.

VII. DISCUSSION

A. Accuracy and Interpretability of Various Architectures

We compare the detection accuracy and interpretability between our architecture and the baselines in Tables I and II and Fig. 6, where our CNN achieves higher HA and much lower FPA and the best interpretability. Such improvement is due to two main factors: 1) in contrast to end-to-end learning

that performs feature extraction solely guided by classification loss or using manually designed features for classification, our CNN particularly learns root cause hotspot features, i.e., the error markers and their surroundings, at intermediate layers, in addition to training with conventional classification loss, which inherently improves HA and interpretability and 2) when designing the learning target of nonhotspot features, we use a blank defect map free of error markers. It helps to significantly deepen the gap between hotspot and nonhotspot features, further reducing FPA.

As shown by Fig. 6, visual explanations of TCAD'19 are less optimal than the other architectures, primarily due to its use of the frequency components (only the lower part) of layout images as input features for hotspot detection. It loses partial layout information and, more importantly, destroys the spatial characteristics crucial for hotspot identification.

TCAD'21 partially relies on irrelevant input features, such as long metal boundaries, for hotspot detection, as shown in Fig. 6, which possibly results from the binarized weights of the BNN and leads to more sensitive responses from the binarized boundary areas in the layout. In addition, binarized convolutional filters weaken the ability of more sophisticated hotspot feature learning than floating-point weights.

TCAD'22 uses the CBAM attention and a triple loss, achieving good detection accuracy. The attention mechanism helps CNN focus on more important features that minimize the optimization loss, and the triple loss includes additional distance losses between in-class hotspot/nonhotspot features in addition to the classification loss.

B. Accuracy and Interpretability of Using Attentions

As shown in Table III, coordinate attention is more effective than CBAM in hotspot detection. We attribute this to the fact that, in the spatial plane, coordinate attention combines local and global information of the feature maps other than the local analysis (via convolution) in CBAM, thus resulting in a more accurate feature extraction of the interaction of metal polygons. Such metal polygons can occupy a larger receptive field than typical convolutional filter sizes.

Specifically, CBAM computes spatial attention by convolving local information with a limited filter size of the feature maps. In contrast, coordinate attention exploits global information via two parallel 1-D global poolings in horizontal and vertical directions, capturing long-range dependencies in the entire spatial plane. This way, direction-aware and position-sensitive information is preserved for coordinate attention.

However, despite enhancing detection accuracy, using attention does not necessarily increase interpretability, as reflected in Table III and Fig. 7. Attention mechanisms help capture essential information that efficiently differentiates hotspots and nonhotspots, thus improving detection accuracy. However, there is no guarantee these features extracted by the CNN and attention are genuine hotspot features, and they can be far from the root cause of printing defects. As a result, guided feature learning for hotspot root causes is necessary to improve interpretability as used in our architecture.

C. Error Marker Preprocessing and Learning Effect

We see a significant increase in detection accuracy and interpretability when learning expanded error markers instead of the original ones at the intermediate layer, as seen in Table IV and Fig. 8. Learning the original size of error markers (each occupying $\sim$2.5% area of the layout clip) poses significant challenges to the CNN and results in massive noise in the learned feature maps (Fig. 8). This less effective feature extraction severely hampers the classification accuracy and overall interpretability of the CNN.

On the other hand, proportionally expanded error markers cover a larger area without changing the original positions of the error markers and provide much more accessible and learnable patterns for CNN. Furthermore, the expanded regions involve partial surrounding metals whose interaction is the root cause of hotspots in the lithography system. Such expanded learning targets help CNN focus more on the metals around defects and their spatial relations, improving detection accuracy and interoperability.

D. Training and Inference Cost of Learning Error Markers

As shown by Table III, our CNN hotspot detector requires, on average, 4.69 ms for classifying one layout clip, compared to 2.72–2.93 ms required by other network architectures. We attribute this larger inference runtime to the additional decoder we use in the architecture. Our vastly improved interpretability and higher accuracy compensate for this extra inference cost. We reckon this is a worthwhile and necessary cost, especially when CNN interpretability is paramount to critical tasks, such as layout hotspot detection. Moreover, compared with the average runtime of 3779 ms for simulating one layout clip using conventional lithography simulation (Table I), CNN inference runtime, in general, is negligible.

Despite the slightly longer inference time than other CNN hotspot detectors, our architecture requires fewer training epochs before convergence (Table III). We conjecture that our targeted training, which learns preprocessed error markers as intermediate representations, adds to the conventional classification loss and regulates the optimization process to a faster and more directional path.

E. Explanation of CNN-Based Hotspot Detectors

Due to CNN's extremely hyperdimensional operation space and highly nonlinear nature, there is a lack of rigorous and well-formulated physical or mathematical explanation thus far for the rationale of CNN's computation flow. However, instead of using an analytical way to interpret the computation paradigm of the hidden layers of CNN, in image classification tasks, intuitive forms are available to provide a visual explanation at the input's semantic level by exploring the correlations between CNN's specific input and target output. Various CNN interpretation methods, such as LIME, CAM, and their variants all fall in this category and are widely used to provide explanations for CNN's image classification results in the input domain. Humans compare these visual explanations in the inputs with common sense or ground truth semantic

features, e.g., "pen" and "watch" as illustrated in Fig. 2, to decide if a CNN is "interpretable."

In the context of CNN-based lithographic hotspot detection where layout clips are classified as images, the error markers and their surrounding metals are the root cause areas that result in lithographic hotspots, and hence are the most important regions within a layout input that an interpretable CNN should rely on for hotspot prediction. As a result, when visual explanations provided by CNN interpretation methods, such as grad-CAM align with simulated error markers and their neighboring areas, we believe the underlying hotspot detection CNN has used the genuine layout features for a hotspot prediction and is thus interpretable. In our CNN-based hotspot detector, we particularly learn these actual hotspot-causing areas as intermediate feature representations and based on which predict a hotspot. Our hotspot detector is interpretable by its architectural design and has been empirically verified by our experimental results in Section VI.

F. Use of Interpretable CNN-Based Hotspot Detectors

In chip design flow, full layouts or their partitioned clips are subject to lithography simulation to ascertain the presence of lithographic hotspots. However, conventional lithography simulation is extremely computationally intensive, where in our experiments, it takes an average of 3779 ms to verify one layout clip. This is prohibitive to use in the early physical design stage, especially when iterations of hotspot detection and mask optimization are involved and a fast turnaround time is highly anticipated for feedback. In this context, PM, ML, and DL-based solutions have evolved over the years to expedite the hotspot detection process. However, despite the high-detection accuracy of existing CNN-based hotspot detectors, they suffer from interpretability issues as examined by our visual and qualitative experimental results in Section VI, where they ground their hotspot prediction on hotspot-irrelevant layout areas, e.g., layout corners or boundaries. This raises severe concerns about the reliability and trustworthiness of their lithography prediction results in practical applications. Our CNN-based hotspot detector, on the one hand, focuses on genuine hotspot features for prediction, representing high interpretability and credibility; on the other hand, its predicted defect maps can provide approximate input regions causing lithographic hotspots, thereby guiding the mask optimization process with fast defect localization.

G. Learning Resist Contours for Interpretable CNN-Based Hotspot Detection

In our design of the interpretable CNN-based hotspot detector, we abstract the lithography simulation process and learn its resulting error markers as the genuine hotspot features, which, in practice, directly determine the hotspot/nonhotspot nature of the layout clips. Generally, neural networks mimicking the lithography simulation pipeline can be used in an interpretable way for hotspot detection. For instance, the generative adversarial network (GAN) is explored in [7] to learn the resist contours of the via layer contact patterns, generating an oval shape for each clip. However, current GAN techniques are incompetent in generating much more complex metal layer shapes, which we evaluate in our work. Precise resist pattern generation for critical areas is necessary, where slight variations in the resist patterns can lead to drastically different lithography results, i.e., hotspots and nonhotspots. With more advanced image generation tools, using resist patterns in interpretable CNN hotspot detection is a promising future direction that merits further exploration.

VIII. Conclusion

In this article, we presented the first interpretable CNN-based lithographic hotspot detector that achieved the highest-detection accuracy and best interpretability. We augmented the training data with expanded error markers obtained and preprocessed from lithography simulation. We used an encoder-decoder architecture with coordinate attention to learn such expanded error markers at intermediate layers, allowing our architecture to focus on genuine hotspot features. We achieved the highest HA of 99.78% and the lowest FPR of 5.29%, with the best visual and quantitative results on interpretability compared to all prior work.

References

[1] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability optimization for industrial designs at sub-14nm process nodes using machine learning," in *Proc. ACM Int. Symp. Phys. Design*, 2017, pp. 15–21.

[2] Z. Xie et al., "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.

[3] W. Haaswijk et al., "Deep learning for logic optimization algorithms," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–4.

[4] K. Zhu, M. Liu, H. Chen, Z. Zhao, and D. Z. Pan, "Exploring logic optimizations with reinforcement learning and graph convolutional network," in *Proc. ACM/IEEE Workshop Mach. Learn. CAD*, 2020, pp. 145–150.

[5] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. 55th Annu. Design Autom. Conf.*, 2018, pp. 1–6.

[6] Y. Watanabe, T. Kimura, T. Matsunawa, and S. Nojima, "Accurate lithography simulation model based on convolutional neural networks," in *Proc. 30th SPIE Opt. Microlithogr.*, 2017, pp. 137–145.

[7] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, pp. 1–6.

[8] C. A. Mack, "Thirty years of lithography simulation," in *Proc. 18th Opt. SPIE Microlithogr.*, 2005, pp. 1–12.

[9] A. Erdmann, T. Fühner, F. Shao, and P. Evanschitzky, "Lithography simulation: Modeling techniques and selected applications," in *Proc. 2nd SPIE Model. Aspects Opt. Metrol.*, 2009, pp. 13–29.

[10] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 11, pp. 1671–1680, Nov. 2014.

[11] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2006, pp. 625–632.

[12] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. 50th Annu. Design Autom. Conf.*, 2013, pp. 1–6.

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

[14] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1175–1187, Jun. 2019.

[15] Y. Jiang, 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.

[16] 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.

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

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

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

[20] 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.

[21] T. Gai et al., "Flexible hotspot detection based on fully convolutional network with transfer learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4626–4638, Nov. 2021.

[22] M. Graphics. "Calibre LFD." 2019. [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/design-for-manufacturing/calibre-lfd/

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

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

[25] K. Liu, B. Tan, R. Karri, and S. Garg, "Poisoning the (data) well in ML-based CAD: A case study of hiding lithographic hotspots," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2020, pp. 306–309.

[26] K. Liu, B. Tan, G. R. Reddy, S. Garg, Y. Makris, and R. Karri, "Bias busters: Robustifying DL-based lithographic hotspot detectors against backdooring attacks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 10, pp. 2077–2089, Oct. 2021.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.

[28] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise," 2017, *arXiv:1706.03825*.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 1135–1144.

[30] V. Petsiuk, A. Das, and K. Saenko, "RISE: Randomized input sampling for explanation of black-box models," 2018, *arXiv:1806.07421*.

[31] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2921–2929.

[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 618–626.

[33] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2018, pp. 839–847.

[34] H. Wang et al., "Score-CAM: Score-weighted visual explanations for convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 24–25.

[35] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, and Y. Wei, "LayerCAM: Exploring hierarchical class activation maps for localization," *IEEE Trans. Image Process.*, vol. 30, pp. 5875–5888, 2021.

[36] H. Yang, S. Li, C. Tabery, B. Lin, and B. Yu, "Bridging the gap between layout pattern sampling and hotspot detection via batch active learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 7, pp. 1464–1475, Jul. 2021.

[37] Y. Xiao, M. Su, H. Yang, J. Chen, J. Yu, and B. Yu, "Low-cost lithography hotspot detection with active entropy sampling and model calibration," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 907–912.

[38] K. Liu et al., "Adversarial perturbation attacks on ML-based CAD: A case study on CNN-based lithographic hotspot detection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 5, pp. 1–31, 2020.

[39] K. Liu, B. Tan, R. Karri, and S. Garg, "Training data poisoning in ML-CAD: Backdooring DL-based lithographic hotspot detectors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1244–1257, Jun. 2021.

[40] J.-Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Detecting context sensitive hot spots in standard cell libraries," in *Proc. 3rd SPIE Design Manuf. Design-Process Integr.*, 2009, pp. 312–320.

[41] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[42] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[43] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[44] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13713–13722.

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[46] "FreePDK45: Contents—NCSU EDA Wiki." Accessed: Feb. 13, 2024. [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK45:Contents

Haoyang Sun received the B.E degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2023, where he is currently pursuing the M.E. degree in computer science.

His research interests include explainable AI and EDA.

Dr. Sun received the Outstanding Undergraduate Thesis Award.

Cong Jiang received the B.E. degree in process equipment and control engineering from the North University of China, Taiyuan, China, in 2019, and the M.E. degree in control engineering from Shandong University, Jinan, China, in 2022. He is currently pursuing the Ph.D. degree in computer science with the Huazhong University of Science and Technology, Wuhan, China.

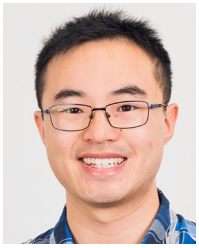
His research interests include machine learning security and machine learning for electronic design automation.

Xun Ye is currently pursuing the bachelor's degree in computer science with the Huazhong University of Science and Technology, Wuhan, China.

His research interests include machine learning for EDA and interpretable deep learning.

Dan Feng (Fellow, IEEE) received the B.E., M.E., and Ph.D. degrees in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991, 1994, and 1997, respectively.

She is a Professor and a Director of the Data Storage System Division, Wuhan National Laboratory for Optoelectronics, HUST, where she is also the Dean of the School of Computer Science and Technology. She has over 100 publications in journals and international conferences, including FAST, USENIX ATC, ICDCS, HPDC, SC, ICS, and IPDPS. Her research interests include computer architecture, massive storage systems, parallel file systems, disk arrays, and solid-state disks.



Benjamin Tan (Member, IEEE) received the B.E. (Hons.) degree in computer systems engineering and the Ph.D. degree from the University of Auckland, Auckland, New Zealand, in 2014 and 2019, respectively.

He was a Professional Teaching Fellow with the Department of Electrical and Computer Engineering, University of Auckland in 2018. From 2019 to 2021, he was with New York University, Brooklyn, NY, USA, where he was a Postdoctoral Associate and then a Research Assistant Professor affiliated with the NYU Center for Cybersecurity. He is currently an Assistant Professor with the University of Calgary, Calgary, AB, Canada. His research interests include computer engineering, hardware security, and electronic design automation.

Dr. Tan is a member of the ACM.

Yuzhe Ma (Member, IEEE) received the B.E. degree from the Department of Microelectronics, Sun Yat-sen University, Guangzhou, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2020.

He is currently an Assistant Professor of Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. His research interests include agile VLSI design methodologies, machine learning-assisted VLSI design, and hardware-friendly machine learning.

Dr. Ma received Best Paper Awards from MLCAD 2023, ICCAD 2021, ASPDAC 2021, ICTAI 2019, and Best Paper Award Nomination from ASPDAC 2019.

Kang Liu (Member, IEEE) received the Ph.D. degree in electrical engineering from New York University, Brooklyn, NY, USA, in 2021.

From 2015 to 2016, he was a Design Engineer with Evertz Microsystems Ltd., Burlington, ON, Canada. Since 2021, he has been a Research Associate Professor with the School of Computer Science, Huazhong University of Science and Technology, Wuhan, China. His research interests include machine learning security and machine learning for electronic design automation.

Dr. Liu has been a reviewer for several IEEE and ACM journals and conferences. He serves on the technical program committee for several top conferences in machine learning and design automation.