Hotspot Detection via Multi-task Learning and Transformer Encoder

Binwu Zhu[1], Ran Chen[1], Xinyun Zhang[1], Fan Yang[2], Xuan Zeng[2], Bei Yu[1], Martin D.F. Wong[1]
[1]The Chinese University of Hong Kong [2]Fudan University

***Abstract*—With the rapid development of semiconductors and the continuous scaling-down of circuit feature size, hotspot detection has become much more challenging and crucial as a critical step in the physical verification flow. In recent years, advanced deep learning techniques have spawned many frameworks for hotspot detection. However, most existing hotspot detectors can only detect defects arising in the central region of small clips, making the whole detection process time-consuming on large layouts. Some advanced hotspot detectors can detect multiple hotspots in a large area but need to propose potential defect regions, and a refinement step is required to locate the hotspot precisely. To simplify the procedure of multi-stage detectors, an end-to-end single-stage hotspot detector is proposed to identify hotspots on large scales without refining potential regions. Besides, multiple tasks are developed to learn various pattern topological features. Also, a feature aggregation module based on Transformer Encoder is designed to globally capture the relationship between different features, further enhancing the feature representation ability. Experimental results show that our proposed framework achieves higher accuracy over prior methods with faster inference speed.**

I. Introduction

As semiconductor technology develops rapidly, the size of integrated circuit components is becoming much smaller. This poses a challenge for chip manufacturers since it is much more difficult to ensure the printability of layout designs due to shrinking feature sizes. Therefore, a precise and efficient hotspot detection technique is crucial to help locate the defect position of a given layout.

In a nutshell, the hotspot detection methods can be divided into three categories, lithography simulation, pattern matching, and machine learning. Lithography simulation is a common method that can achieve high accuracy but suffers from runtime overhead issues. In contrast to lithography simulation, pattern matching and machine learning methods can identify the hotspots efficiently. Pattern matching methods [1]–[4] take a collection of hotspot layout patterns and use them to scan over new designs to identify any matched patterns as hotspots. Although pattern matching overcomes the time-consuming drawback, it fails to detect unknown hotspots, which are limited in real scenarios.

Different from pattern matching, hotspot detection methods based on machine learning [5]–[10] show strong generalization abilities and achieve acceptable performance. Especially the approaches [11]–[17] built upon deep learning techniques get significant improvements on both accuracy and efficiency. For example, Yang *et al.* [11] design a convolutional neural network (CNN) model to detect hotspots in clip-wise and

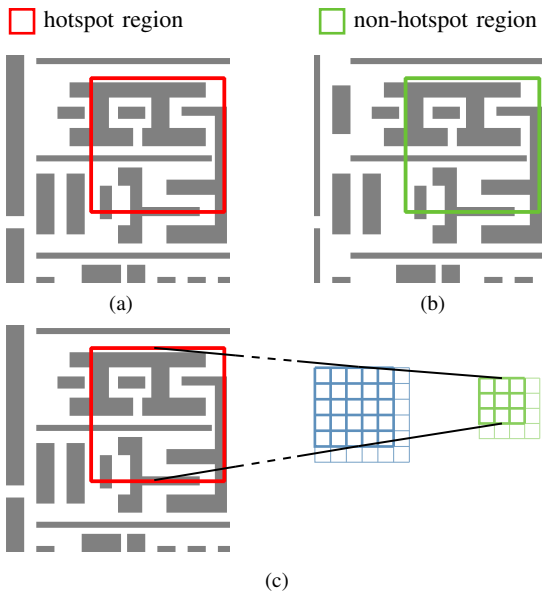


Fig. 1 The comparison between two regions with the same layout patterns. (a) Hotspot region. (b) Non-hotspot region. (c) Locality inductive bias of CNN.

successfully address the issue arising from the imbalanced data distribution.

However, there are still some issues with previous methods. (1) Most deep learning-based frameworks are designed to detect whether there is a hotspot at the center of an input clip. It would take a long time to detect the whole layout design. Although the method proposed in [16] can detect multiple defects in large scales, some redundant designs may affect the efficiency. To be specific, the whole framework is a two-stage detector that requires a region proposal network as the first stage to select the potential defect regions. A refinement module is designed to process these candidate defect regions for more accurate results (2) Chen *et al*. [16] trains a framework to locate hotspots with rectangle bounding boxes. However, the underlying information, e.g., corner information, is not utilized as supervision for model training. The underlying information is helpful for detection tasks in some aspects. For example, corner information contributes to improve the localization accuracy [18], while center information performs better on detecting small targets [19]. (3) We observe some outlier situations where two regions sharing the same layout patterns may have different simulation results,

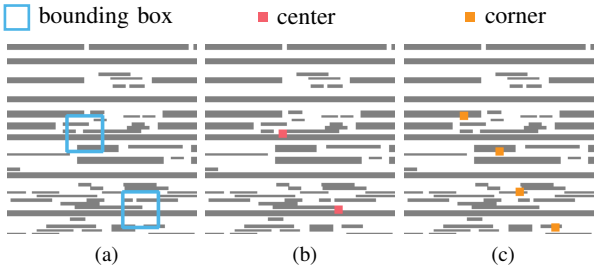


Fig. 2 The visualization for three different representations to indicate hotspot regions. (a) The bounding box of the hotspot region. (b) Center of the hotspot region. (c) Top-left and bottom-right corner of the hotspot region.

i.e., one region is identified as a hotspot while the other is not. An example is shown in Fig. 1(a) and Fig. 1(b). This phenomenon indicates that we can not judge whether a hotspot exists only by focusing on the local layout patterns, instead, context information plays an important role. However, CNNs, commonly adopted by previous methods, are infeasible to capture the long-range dependencies due to the locality inductive bias. Only neighboring pixels are taken into consideration when performing convolution operation, leading to a limited receptive field for each position, as illustrated in Fig. 1(c). Therefore, frameworks solely based on CNNs have limitations on hotspot detection.

To address these issues, we propose a single-stage detection framework that is more efficient than [16] by eliminating the region proposal stage. We design two modules called center head and corner head to learn the underlying representations of the hotspots. These two modules act as auxiliary to help produce more accurate bounding boxes, which are used to indicate hotspot regions. A visualized example of different representations is shown in Fig. 2. Our motivation is that by jointly training the hotspot detector to learn different but related tasks, the knowledge learned from one task can be leveraged by others. This process simulates human perception and helps improve the overall performance. In addition, a feature aggregation module based on Transformer Encoder [20] is designed to augment the feature representation ability by modeling the dependencies between each feature with others. With this module, the issue of the CNNs, which can only attend to local features, is mitigated. The main contributions of this paper are listed as follows:

- We propose a single-stage detector skipping the region proposal stage, which can effectively detect the hotspots.
- We build up center head and corner head to detect the center and corner points of the hotspot regions.
- We design a feature aggregation module and a feature sampling strategy to enrich the feature representation. The sampling strategy is adopted to save the computation cost of the aggregation operation.
- Experimental results show that our model achieves high detection accuracy and speed over prior state-of-the-art models.

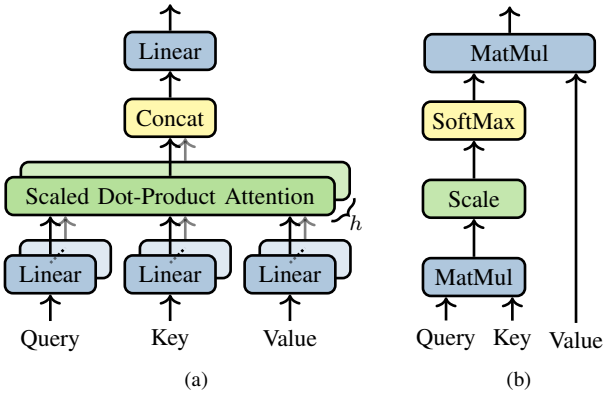


Fig. 3 Two basic modules for Transformer architecture. (a) Multi-Head Attention. (b) Scaled Dot-Product Attention.

The rest of this paper is organized as follows. Section II introduces terminologies and evaluation metrics related to this work. Section III discusses each component of the proposed hotspot detector. Section IV describes the implementation details including the loss function design. Section V shows the experimental comparison results with state-of-the-art, followed by the conclusion in Section VI.

II. Preliminaries

In this section, we will introduce the problem formulation and some preliminary knowledge related to this work.

A. Problem Formulation

In chip manufacturing, designed layout patterns are transferred onto silicon wafers through a technique called lithographic process. However, lithographic process involves many variations, and some patterns are sensitive to these variations, which may reduce the manufacturing yield due to potential open or short circuit failures. Layout patterns that are sensitive to lithographic process variations are defined as *hotspots*.

A high-performance hotspot detector should correctly detect as many hotspots as possible and avoid mistaking non-hotspot patterns for hotspot patterns. To evaluate the performance, we define the following metrics.

Definition 1 (Accuracy)**.** *The ratio between the number of correctly detected hotspots and the number of groundtruth hotspots.*

Definition 2 (False Alarm)**.** *The number of non-hotspots that are predicted as hotspots by the classifier.*

With the evaluation metrics defined above, we formulate the hotspot detection problem as follows:

Problem 1 (Hotspot Detection)**.** *Given a collection of clips containing hotspot and non-hotspot layout patterns, the objective of hotspot detection is to train a detector to locate and classify all hotspots and non-hotspots, such that the detection accuracy is maximized and the false alarm is minimized.*

B. Transformer and Multi-Head Attention

Recently, Transformer [20] has made much progress in many sequence-to-sequence tasks [21]–[23]. Transformer consists of two modules, Encoder and Decoder. Both the Encoder and the Decoder are built on a core mechanism called Multi-Head Attention, which allows the model to attend to information at different positions globally [20]. The architecture of Multi-Head Attention is illustrated in Fig. 3(a) and it is formulated as:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}\left(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_h\right) \boldsymbol{W}^O, \quad (1)$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n \times d_m}$ are the input matrices. In Transformer [20], $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are called *query*, *key* and *value* separately. n is sequence length and d_m is the dimension for each element of the sequence. $\boldsymbol{H}_i, i \in \{1, 2, \ldots, h\}$ is the output of a single scaled dot-product attention head as shown in Fig. 3(b) and h is the number of heads.

To illustrate the dimension of $\boldsymbol{H}_i$ and $\boldsymbol{W}^O$, we first give the formulation of $\boldsymbol{H}_i$ as follows:

$$\begin{aligned} \boldsymbol{H}_i &= \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right) \\ &= \text{softmax}\left[\frac{\boldsymbol{Q}\boldsymbol{W}_i^Q \left(\boldsymbol{K}\boldsymbol{W}_i^K\right)^\top}{\sqrt{d_k}}\right] \boldsymbol{V}\boldsymbol{W}_i^V. \end{aligned} \quad (2)$$

For each attention head, the original input $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are projected into different subspaces via projection matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K \in \mathbb{R}^{d_m \times d_k}, \boldsymbol{W}_i^V \in \mathbb{R}^{d_m \times d_v}$ so that different heads deal with different input to learn richer information [20]. The attention head then computes the similarity between projected *query* and *key* via scaled dot-product and a softmax function is then applied to obtain the weights on projected *value*.

The Multi-Head Attention formulation in Equation (1) concatenates all the outputs $\boldsymbol{H}_i \in \mathbb{R}^{n \times d_v}, i \in \{1, 2, \ldots, h\}$ from different heads and then reduce the high dimension feature to low dimension via matrix $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d_m}$.

III. Hotspot Detection Architecture

The proposed architecture overview of our framework is illustrated in Fig. 4. (1) The first part is a backbone made up of ResNet-50 and Feature Pyramid Network. With the backbone, we can transfer the input clip to high dimensional features, and multiple level feature maps are generated for subsequent detection. (2) To conduct the hotspot detection task, we predefine dense regions on the output feature map of the backbone. The classification head performs label prediction to judge whether a region is a hotspot region or not. The localization head adjusts the predefined regions to fit the groundtruth hotspot regions better. (3) Corner head and center head receive the output from the backbone and identify the corners and centers of input clips. (4) The feature aggregation module is used to learn a rich hierarchy of associative features across different positions in the localization and classification head separately. The feature selection module selects informative features and filters out unimportant ones to save the computation cost.

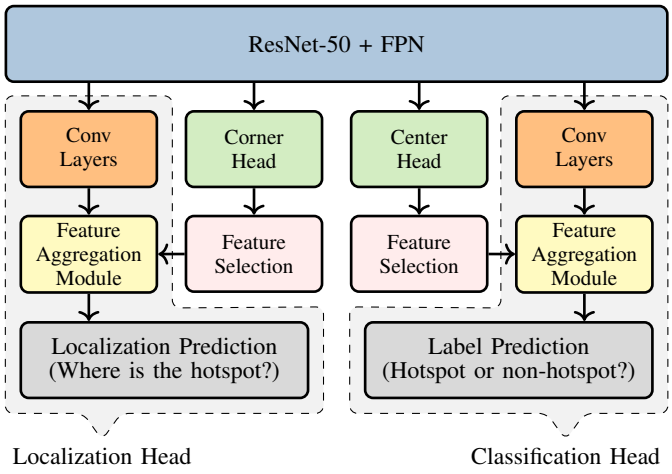
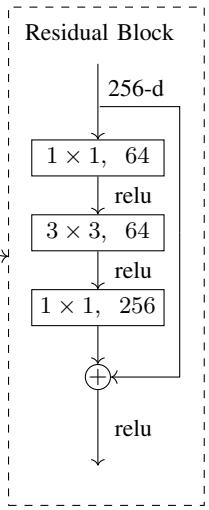


Fig. 4 The architecture of the proposed hotspot detector.

TABLE I ResNet-50 Architecture

layer name	ResNet-50
conv1	7×7, 64, stride 2
conv2_x	3×3 max pool, stride 2
	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$



A. Backbone

Feature extraction is a process that identifies important features from input clips, which makes it easier for later hotspot detection. The deep convolutional neural network is a powerful tool for extracting effective features.

ResNet-50. Different from the previous works like [11], [16], our work does not concentrate on the design of the convolutional neural network for feature extraction. Instead, we adopt ResNet-50 [24] as the feature extractor, which proves to have prominent feature extraction ability according to much work like [24]–[26]. The architecture of ResNet-50 is listed in TABLE. I.

Feature Pyramid Network. Previous work [11], [16], [27] simply detect hotspots on the output from the last layer of the backbone. However, we find that the feature maps of layers in

different levels are also crucial for hotspot detection, which is further verified by our experimental results in V.

To utilize the feature maps from different layers, we build the Feature Pyramid Network (FPN) [28] on top of the ResNet-50. As shown in the right part of Fig. 5, with a top-down pathway and lateral connections, multi-scale feature maps are generated. Each feature map contains different level features, and all of them can be used for detecting hotspots. Considering the potential size of hotspot regions, we generate three feature maps (P_3, P_4, P_5) with different scales, where P_k indicates that it has resolution 2^k lower than the input.

B. Classification and Localization Head

In order to detect hotspots, previous work [16] first proposes the regions of interest (ROI) that are likely to contain the hotspot with region proposal network [16]. Then these proposed regions are fed into the detection head for regression and classification to get more accurate results. However, due to the region proposal network with complex structures, the detection speed is low.

To tackle this issue, we define dense regions which will be directly passed to classification and localization head after feature extraction. The region proposal step is skipped in order to improve the detection efficiency. These predefined regions are called anchors. More details about the anchor settings are illustrated in Section IV-A.

Assume that we define M anchors for each pixel of the feature map. Given a feature map $\boldsymbol{F} \in \mathbb{R}^{H\times W\times C}$ generated by the backbone as the input, the conv layers module, which consists of four consecutive 3×3 convolution layers with C filters, outputs a new feature map $\boldsymbol{F}_c \in \mathbb{R}^{H\times W\times C}$. A feature aggregation module is then adopted to enhance the representation ability of $\boldsymbol{F}_c$, and then produces another feature map $\boldsymbol{F}_a \in \mathbb{R}^{H\times W\times C}$. The structure of the feature aggregation module will be clearly illustrated in Section III-D. The feature map F_a will be fed into the last layer, which is a 3×3 convolution layer with M filters and output the final result $\boldsymbol{F}_o \in \mathbb{R}^{H\times W\times M}$ where each element predicts the probability of each anchor containing a hotspot.

As for the regression head, it predicts the offset from each anchor to a nearby groundtruth hotspot, if one exists. The structure of the regression head is almost the same as the classification head except that the last layer has $4M$ filters. This is because the offset for each anchor is a 4-dimension vector, including the offset for a 2-d center, width, and height.

C. Corner & Center Representation Learning

Multi-task learning is a learning paradigm which aims to learn multiple related tasks jointly so that the knowledge contained in one task can be leveraged by other tasks, with the hope of improving the overall performance.

In Section III-B, the localization and classification heads are proposed to detect the potential hotspot regions with bounding boxes. Apart from the bounding box, corner and center representations are proposed in this work to further

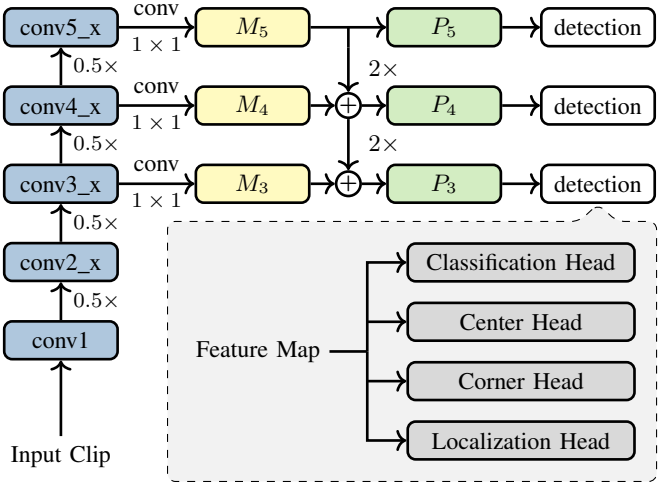


Fig. 5 The architecture of the backbone. The backbone is a combination of ResNet-50 and Feature Pyramid Network.

improve the detection performance. There are several advantages to utilize corner and center representation learning. (1) Center representation learning: hotspot region can be regarded as a small pattern in the layout design, for which center representation is proved to be friendly to detect small objects [19]. (2) Corner representation learning: corner representation is helpful for locating the target precisely [18].

Following the paradigm of multi-task learning, corner and center representation learning share the same backbone with the previous heads illustrated in Section III-B. To train the detector to learn these effective representations, we design corner head and center head separately. The center head receives the output $\boldsymbol{F} \in \mathbb{R}^{H\times W\times C}$ of the backbone and predicts the probability of each point being a center as well as the offset from each point to a nearby center, if one exists. The structure of the center head is composed of a single point head as shown in Fig. 6. Different from the center head, the corner head identifies both the top left corner and bottom right corner of the hotspot region. Therefore, the corner head is composed of two separate point heads.

D. Feature Aggregation Module

The Transformer Encoder applied in machine translation tasks has shown its strength in modeling all pairwise interactions between different elements in a sequence. Inspired by its mechanism, which has been introduced in Section II-B, we design a module called feature aggregation module (FAM) based on the Transformer Encoder.

With the help of FAM, we can enrich the feature map output from the conv layers module in the localization and classification heads, as shown in Fig. 4. By globally capturing the dependencies between different features with the Multi-Head Attention mechanism, the representation ability of the feature map could be effectively augmented.

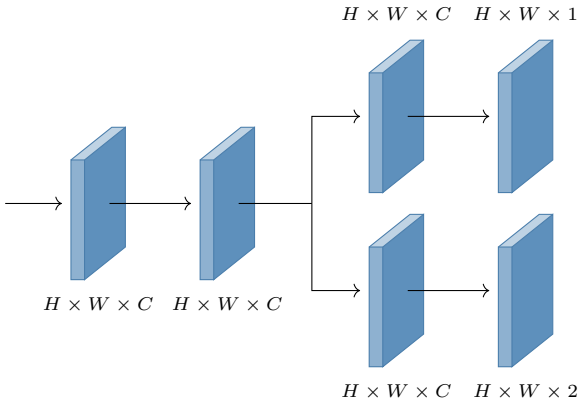


Fig. 6 Point head structure. The center head is composed of a single point head. The corner head is composed of two point heads.

To be specific, given a feature map $\boldsymbol{F}_c \in \mathbb{R}^{H\times W\times C}$, when we hope to capture the dependencies between $\boldsymbol{f}_m \in \mathbb{R}^C$ with all other features $\boldsymbol{f}_n \in \mathbb{R}^C, n \in \{1, 2, \ldots, HW\}$, $\boldsymbol{f}_m$ is regarded as *query* and $\boldsymbol{f}_n$ is regarded as *key* and *value*. Noted that $\boldsymbol{F}_c$ is the output of the conv layers module as illustrated in Section III-B. Then considering the operation via a single head of the Multi-Head Attention, which can be applied on $\boldsymbol{f}_m$ and $\boldsymbol{f}_n$ as follows:

$$\boldsymbol{h}_i = \sum_{n=1}^{HW} \frac{\exp(\boldsymbol{f}_m \boldsymbol{W}_i^Q (\boldsymbol{f}_n \boldsymbol{W}_i^K)^\top / \sqrt{C})}{\sum_{n=1}^{HW} \exp(\mathbf{f}_m \boldsymbol{W}_i^Q (\boldsymbol{f}_n \boldsymbol{W}_i^K)^\top / \sqrt{C})} \boldsymbol{f}_n \boldsymbol{W}_i^V, \quad (3)$$

where $\boldsymbol{h}_i \in \mathbb{R}^C$ is the output of head i. HW is the number of features of the feature map $\boldsymbol{F}_c \in \mathbb{R}^{H\times W\times C}$, and C is the dimension of the feature. $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{C\times C}$ are projection matrices as explained in Section II-B.

However, the computation cost is extremely expensive since the huge key set includes hundreds of candidates and the complexity for Equation (3) is $\mathcal{O}(HWC^2)$. In terms of this issue, we design a simple yet effective algorithm to reduce the amount of the selected keys. The main idea is to leverage the knowledge from the center head and corner head to guide the key selection for the feature map in the classification head and localization head separately. The key selection algorithm on the center head is demonstrated in Algorithm 1. The key set of the corner head is selected in a similar way.

As illustrated in Section III-C, the probability map output by center head describes the probability of each point being a center. We perform a 3×3 average pooling with stride 1 on the probability map and get the positions of the maximum k values (lines 2–3). Then the features in the corresponding positions of the feature map are added to the key set (lines 4–8), which will be adopted to enhance the feature map.

Based on the new generated key set, we further design a variant of Equation (3) as follows:

$$\boldsymbol{h}_i = \boldsymbol{f}_m + \lambda \sum_{n=1}^{k} \frac{\exp(\boldsymbol{f}_m \boldsymbol{W}_i^Q (\boldsymbol{f}_n \boldsymbol{W}_i^K)^\top / \sqrt{C})}{\sum_{n=1}^{k} \exp(\boldsymbol{f}_m \boldsymbol{W}_i^Q (\boldsymbol{f}_n \boldsymbol{W}_i^K)^\top / \sqrt{C})} \boldsymbol{f}_n \boldsymbol{W}_i^V \quad (4)$$

Algorithm 1 Key selection algorithm

Input: Center Probability Map $\boldsymbol{C} \in \mathbb{R}^{H\times W}$, Feature Map $\boldsymbol{F}_c \in \mathbb{R}^{H\times W\times C}$, Selection Number k;
Output: Key set $\boldsymbol{F}_k$;
1: $\boldsymbol{F}_k \leftarrow$ Initialized to empty set;
2: $\boldsymbol{C}' \leftarrow$ `AvgPool(`$\boldsymbol{C}$`)`;
3: $\boldsymbol{topk\_idx} \leftarrow$ the index of the maximum k values in $\boldsymbol{C}'$;
4: **for** $i \leftarrow 1, 2, \ldots k$ **do**
5: $\quad idx \leftarrow \boldsymbol{topk\_idx}[i]$;
6: $\quad \boldsymbol{f}_i \leftarrow$ the feature in the position idx of $\boldsymbol{F}_c$;
7: $\quad$ append feature f_i to key set $\boldsymbol{F}_k$;
8: **end for**
9: **return** Key set $\boldsymbol{F}_k$ with k features.

where k is the selection number of keys defined in Algorithm 1 and $\boldsymbol{f}_n$ is from the key set $\boldsymbol{F}_k$. λ is a hyperparameter to control the feature augmentation degree. Compared to Equation (3), the complexity of Equation (4) is reduced to $\mathcal{O}(kC^2)$, where k is much smaller than HW. As illustrated in Section II-B, the Multi-Head Attention is composed of multiple attention heads. In our framework, the number of the heads is set to 8.

IV. Implementation Details

A. Anchors

For each pixel of the feature map, anchors with three different aspect ratios {1:2, 1:1, 2:1} and four different sizes $\{2^0, 2^{1/4}, 2^{2/4}, 2^{3/4}\}$ are set for dense scale coverage. Since each pixel is assigned with 12 anchors and each feature map is composed of many pixels, the number of anchors is extremely large, leading to the low efficiency for training. We can benefit from getting rid of large parts of the anchors. We first define Intersection-over-Union (IoU) as follows:

$$\text{IoU} = \frac{\text{anchor} \cap \text{groundtruth}}{\text{anchor} \cup \text{groundtruth}}. \quad (5)$$

Our assignment rule is based on the IoU between anchor and groundtruth as follows:

- If the IoU between an anchor and a groundtruth is larger than 0.5, the anchor will be regarded as positive sample.
- If the IoU between an anchor and any other groundtruth is smaller than 0.4, the anchor will be regarded as negative sample.
- If the IoU between a groundtruth and any other anchor is smaller than 0.5, the anchor with the highest IoU will be regarded as positive sample.
- The rest anchors are ignored during training.

Noted that each anchor is assigned to at most one groundtruth bounding box.

B. Training Loss

The objective function presented in this work is formulated as follows:

$$L^{det} = L^{bbox} + L^{ctr} + L^{cor}, \quad (6)$$

L^{bbox}, L^{ctr} and L^{cor} are bounding box loss, center loss and corner loss correspondingly. The detail explanation on these three terms will be introduced in this section.

1) Bounding Box Loss: Based on the assignment rule in section IV-A, the ground truth classification target p_i is set to 1 if the anchor i belongs to positive sample and 0 otherwise.

Focal loss [29] is adopted to train the classification head. The classification loss function is defined as:

$$L_{cls}^{bbox}(p_i, p_i') = \begin{cases} -\alpha(1-p_i')^{\gamma}\log p_i', & p_i = 1, \\ -(1-\alpha)p_i'^{\gamma}\log(1-p_i'), & p_i = 0, \end{cases} \quad (7)$$

where α and γ are hyperparameters. p_i' is the prediction result which indicates the probability containing the hotspot.

In addition to the classification head, the regression head predicts the offset between each anchor and its assigned groundtruth bounding box. $\boldsymbol{t}_i' = (t_x', t_y', t_w', t_h')$ and $\boldsymbol{t}_i = (t_x, t_y, t_w, t_h)$ are 4-d vector representing the regression prediction and the corresponding groundtruth target, respectively. They are defined as:

$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a, \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a), \\ t_x' &= (x' - x_a)/w_a, & t_y' &= (y' - y_a)/h_a, \\ t_w' &= \log(w'/w_a), & t_h' &= \log(h'/h_a), \end{aligned} \quad (8)$$

where (x, y, w, h) represents the center coordinates, width and height. w', w_a and w represents the width of the predicted bounding box, anchor and groundtruth bounding box separately (same as x, y and h). Smooth L_1 loss is adopted for regression loss function which is formulated as:

$$L_{reg}^{bbox}(\boldsymbol{t}_i, \boldsymbol{t}_i') = \sum_{j=1}^{4} l_{reg}^{bbox}(t_i[j], t_i'[j]), \quad (9)$$

where

$$l_{reg}^{bbox}(t_i[j], t_i'[j]) = \begin{cases} \frac{1}{2}(t_i[j] - t_i'[j])^2, & \text{if } |t_i[j] - t_i'[j]| < 1 \\ |t_i[j] - t_i'[j]| - \frac{1}{2}, & \text{otherwise.} \end{cases} \quad (10)$$

With the defined regression and classification loss function, the overall loss for the bounding box is calculated as:

$$L^{bbox} = \frac{1}{N_{anch}} \sum_i (L_{cls}^{bbox}(p_i, p_i') + p_i L_{reg}^{bbox}(\boldsymbol{t}_i, \boldsymbol{t}_i')), \quad (11)$$

where N_{anch} is the number of anchors.

2) Center & Corner Loss: Center and corner representations are adopted as the auxiliary to enhance the bounding box representation. To learn the effective feature expression, loss functions are designed for the corner head and the center head respectively. The loss for each point in the corner head is designed identically as the loss for the center head, thus we take the loss function for the center head as an example.

Similar to the anchor assignment, all the points within a feature map are divided into two categories. The points within the groundtruth bounding box are regarded as positive points and the rest are regarded as negative points. We assign each positive point with its corresponding bounding box center. For negative points, the groundtruth probability q_{xy} for (x, y) being the center is set to 0. And for positive points, we take the Gaussian kernel to describe the probability as follows:

$$q_{xy} = \exp\left(-\frac{(x - \lfloor\hat{x}\rfloor)^2 + (y - \lfloor\hat{y}\rfloor)^2}{2\sigma^2}\right), \quad (12)$$

where $(\hat{x}, \hat{y})$ is the assigned center for each positive point and σ is a hyperparameter. The classification branch of the center head predicts the probability q_{xy}' of each point being the center. A variant of focal loss [29] is adopted for the classification loss function, defined as:

$$L_{cls}^{ctr}(q_{xy}, q_{xy}') = \begin{cases} (1 - q_{xy}')^{\theta}\log(q_{xy}'), & q_{xy} = 1, \\ (1 - q_{xy}')^{\beta}(q_{xy}')^{\theta}\log(1 - q_{xy}'), & q_{xy} < 1, \end{cases} \quad (13)$$

where θ and β are hyperparameters.

Besides predicting the probability of each point being a center, the center head is trained to predict the offset $\boldsymbol{d}_{xy}'$ between each positive point (x, y) and its corresponding center $(\hat{x}, \hat{y})$. Similar to eq. (9), the regression loss function is defined as:

$$L_{reg}^{ctr}(\boldsymbol{d}_{xy}, \boldsymbol{d}_{xy}') = \sum_{j=1}^{2} l_{reg}^{ctr}(d_{xy}[j], d_{xy}'[j]), \quad (14)$$

where l_{reg}^{ctr} is taken as the same form as eq. (10).

By combining the regression and classification loss functions, the overall loss for the corner head is calculated as:

$$L^{ctr} = \frac{1}{N_{ctr}} \sum_x \sum_y (L_{cls}^{ctr}(q_{xy}, q_{xy}') + \mathbb{I}_{q_{xy}>0} L_{reg}^{ctr}(\boldsymbol{d}_{xy}, \boldsymbol{d}_{xy}')), \quad (15)$$

where N_{ctr} is the number of centers for a given input. The indicator function $\mathbb{I}_{q_{xy}>0}$ points out that the negative points are omitted for regression part.

V. Experimental Results

We implement our proposed hotspot detector on a platform with the Xeon Silver 4114 CPU processor and NVIDIA TITAN Xp Graphic card. We evaluate the performance of our framework on the ICCAD CAD Contest 2016 Benchmarks [30] which contains four designs that are shrunk to match EUV metal layer design rules. According to the results detected with the industrial $7nm$ metal layer EUV lithography simulation technique, the hotspot is precisely located. Since the first benchmark design contains limited defects checked by lithography simulation, we only conduct the experiments on the rest three ones. For each layout, we split it into two parts with equal area size, among which one part is used for training, and the other part is used for testing. Due to the extremely large size of the whole layout, small clips are cropped from the layout and are then fed into the hotspot detector. More details are illustrated in TABLE II.

Noted that `case2`, `case3` and `case4` are the names of the three benchmarks. Column "Train #HS" and "Test #HS" indicate the total number of hotspots in the training and test set, while column "Train #Clips" and "Test #Clips" refer to the total number of clips in the training and test set. "Training

TABLE II Benchmark Information

Bench	Train #HS	Test #HS	Train #Clips	Test #Clips	Training Set Size ($\mu m \times \mu m$)	Test Set Size ($\mu m \times \mu m$)
case2	40	39	1000	8	6.95 × 3.75	6.95 × 3.75
case3	1388	1433	1000	33	12.91 × 10.07	12.91 × 10.07
case4	90	72	1000	55	79.95 × 42.13	79.95 × 42.13

TABLE III Comparison with State-of-the-art

Bench	TCAD'19 [27]			DAC'19 [16]			Ours		
	Accu(%)	FA	Time(s)	Accu(%)	FA	Time(s)	Accu(%)	FA	Time(s)
case2	77.78	48	60.0	93.02	17	2.0	94.87	6	1.0
case3	91.20	263	265.0	94.5	34	10.0	97.2	26	4.0
case4	100.00	511	428.0	100.0	201	6.0	100	70	6.0
Average	89.66	274.00	251.00	95.84	84.00	6.00	**97.31**	**34.00**	**3.67**
Ratio	0.92	8.06	67.84	0.98	2.47	1.62	**1.00**	**1.00**	**1.00**

Set Size" and "Test Set Size" denote the resolution of the training and test set for each benchmark respectively. Clips in the training set are obtained by randomly cropping the layouts of the training part for 1000 times to get sufficient data for training. Different from the training set, we uniformly crop the layout in the test set. The size of each clip is 256×256 (corresponding to $2.56\mu m \times 2.56\mu m$), on which hotspots may appear or not.

TABLE III shows the results of our proposed framework and several other state-of-the-art hotspot detectors. "TCAD'19" lists the result of a deep learning-based hotspot detector proposed in [27] that adopts a high-dimensional feature extraction method and biased learning algorithm which can reduce the size of training instances. "DAC'19" shows the result of a faster region-based hotspot detector in [16] which first proposes a detector capable of detecting multiple hotspots in a large area for each inference. The comparison results illustrate that our model has satisfactory detection accuracy on each case. Especially, the average accuracy of our framework achieves 97.31 compared to 95.88 and 89.66 for DAC'19 [16] and TCAD'19 [27], respectively. Besides, the efficiency superiority of our proposed hotspot detector can also be noticed that it achieves 67.84× speedup compared to TCAD'19 [27], which can only detect whether the center of an input clip has a hotspot. Also, compared to DAC'19 [16], the inference speed of our model is faster because of the simpler architecture of the whole framework. Moreover, the advantage of our framework can also be noticed that it suppresses the false alarm effectively, which decreases 87.6% and 59.5% of the FA reported by TCAD'19 and DAC'19.

To investigate the behavior of our designed components, we carry out ablation studies to examine how different configurations affect performance as shown in Fig. 7. The histogram shows that with the FPN, the detection accuracy improves significantly, which reveals the importance of merging feature maps from different layers. By detecting on the feature maps from different layers, multi-level features from the input clip could be utilized for detection, which contribute to the robustness of the framework. Besides, with the center head and corner head, we obtain 2.21% improvement on accuracy

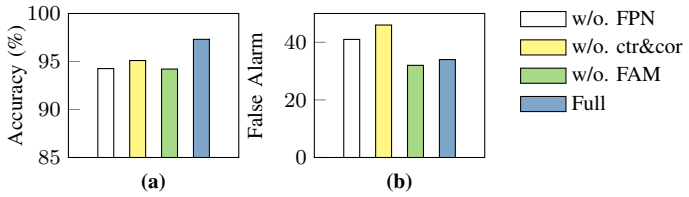


Fig. 7 Comparison among different configurations on (a) average accuracy and (b) average false alarm.

and a reduction on average False Alarm, indicating that the knowledge learned in the corner head and center head can be positively leveraged for the regression and classification for bounding boxes. In addition, with feature aggregation module, we further achieve 3.1% improvement on accuracy, demonstrating that by adopting the self-attention module, the detector learns more informative representations and further achieves better detection performance.

VI. Conclusion

In this paper, we proposed an end-to-end one-stage hotspot detection framework. We take advantage of the corner and center representation to improve both classification and localization accuracy. Our feature aggregation module provides a new way to aggregate different features and further generate the enhanced features. We further exploit a sampling strategy for FAM to reduce the computation cost effectively. The experimental results demonstrate the superiority of our framework over current deep learning-based detectors on both accuracy and efficiency. With the development of manufacturing techniques for semiconductors, layouts are becoming more and more complex. We hope the framework proposed in this work can provide a more powerful solution to advanced design for manufacturability research.

Acknowledgment

This work is partially supported by National Key R&D Program of China 2020YFA0711900, 2020YFA0711903, and The Research Grants Council of Hong Kong SAR (No. CUHK14209420).

REFERENCES

[1] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *Proc. DAC*, 2013, pp. 68:1–68:6.

[2] 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 TCAD*, vol. 33, no. 11, pp. 1671–1680, 2014.

[3] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," in *Proc. DAC*, 2012, pp. 1167–1172.

[4] S. S.-E. Tseng, W.-C. Chang, I. H.-R. Jiang, J. Zhu, and J. P. Shiely, "Efficient search of layout hotspot patterns for matching SEM images using multilevel pixelation," in *Proc. SPIE*, vol. 10961, 2019.

[5] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. DAC*, 2009, pp. 545–550.

[6] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE TCAD*, vol. 30, no. 11, pp. 1621–1634, 2011.

[7] 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. DAC*, 2013, pp. 671–676.

[8] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng, "Sparse VLSI layout feature extraction: A dictionary learning approach," in *Proc. ISVLSI*, 2018, pp. 488–493.

[9] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan, "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *JM3*, vol. 14, no. 1, p. 011003, 2015.

[10] H. Zhang, B. Yu, and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *Proc. ICCAD*, 2016, pp. 47:1–47:8.

[11] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: a deep learning approach," *JM3*, vol. 16, no. 3, p. 033504, 2017.

[12] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proc. DAC*, 2017, pp. 62:1–62:6.

[13] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, pp. 1–1, 08 2020.

[14] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," in *Proc. ICCAD*, 2020.

[15] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. ASPDAC*, 2019, pp. 299–304.

[16] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu, "Faster region-based hotspot detection," in *Proc. DAC*, 2019, pp. 146:1–146:6.

[17] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network," in *Proc. DAC*, 2019, pp. 147:1–147:6.

[18] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proc. ECCV*, 2018, pp. 734–750.

[19] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018.

[23] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE TPAMI*, vol. 39, no. 6, pp. 1137–1149, 2016.

[26] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *arXiv preprint arXiv:1605.06409*, 2016.

[27] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, vol. 38, no. 6, pp. 1175–1187, 2019.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, 2017, pp. 2117–2125.

[29] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *Proc. ICCV*, pp. 2999–3007, 2017.

[30] R. O. Topaloglu, "ICCAD-2016 CAD contest in pattern classification for integrated circuit design space analysis and benchmark suite," in *Proc. ICCAD*, 2016, pp. 41:1–41:4.