# Faster Region-based Hotspot Detection

Ran Chen
Chinese University of Hong Kong

Wei Zhong
Dalian University of Technology

Haoyu Yang
Chinese University of Hong Kong

Hao Geng
Chinese University of Hong Kong

Xuan Zeng
Fudan University

Bei Yu
Chinese University of Hong Kong

## ABSTRACT

As the circuit feature size continuously shrinks down, hotspot detection has become a more challenging problem in modern DFM flows. Developed deep learning techniques have recently shown their advantages on hotspot detection tasks. However, existing hotspot detectors only accept small layout clips as input with potential defects occurring at a center region of each clip, which will be time consuming and waste lots of computational resources when dealing with large full-chip layouts. In this paper, we develop a new end-to-end framework that can detect multiple hotspots in a large region at a time and promise a better hotspot detection performance. We design a joint auto-encoder and inception module for efficient feature extraction. A two-stage classification and regression flow is proposed to efficiently locate hotspot regions roughly and conduct final prediction with better accuracy and false alarm penalty. Experimental results show that our framework enables a significant speed improvement over existing methods with higher accuracy and fewer false alarms.

## 1 INTRODUCTION

With the development of the semiconductor industry, transistor feature size shrinks rapidly, which significantly challenges manufacturing yield. To ensure the printability of layout designs, an efficient and accurate hotspot detector is indispensable. Currently, there are three main classes of methods: lithography simulation, pattern matching and machine learning. Lithography simulation is accurate but extremely time consuming. Pattern matching and machine learning based methods are proposed to speedup the hotspot detection flow while attain the detection accuracy as much as possible.

On one hand, the main idea of pattern matching is to set up a collection of hotspot layout patterns and use this collection to identify any matched patterns in a new design as hotspots [1, 2]. Although the pattern matching overcame the runtime issue, this approach cannot give a confident result on unseen hotspot patterns. On the other hand, machine learning based methods have shown the capability to solve this problem with generalized feature extractors [3–12]. A learning model is usually trained by a batch of labeled data and conducts hotspot prediction on new layout patterns efficiently.

*DAC'19, June 2–6, 2019, Las Vegas, NV, USA*
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196056

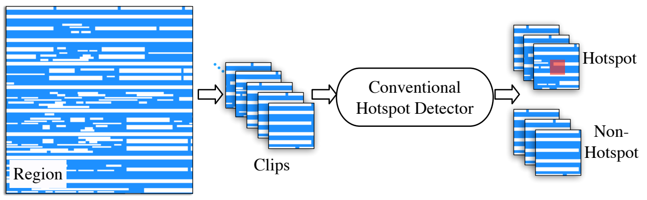

**Figure 1: Conventional hotspot detection flow.**

Convolutional neural network (CNN) is a powerful technique to improve hotspot detection performance [13–19]. It has the ability to extract the features in different levels in an automatic way. Yang *et al.* [15] proposed a deep CNN which considered the data unbalanced issue and achieved high classification accuracy. They also designed a biased learning technique for unbalanced dataset and apply discrete-cosine transformation (DCT) to give proper feature expression [16].

In all existing literatures, however, hotspot detectors only work on small clips extracted from a whole chip layout and can only detect one hotspot location at a time that occurs at a center (i.e. core in [20]) of each clip. Figure 1 illustrates a conventional hotspot detection scheme, which requires repeatedly scanning overlapping regions of a full chip design, therefore it could be a waste of computational resources and time consuming when facing with extreme large layouts. To solve this problem, we propose a new faster region-based hotspot detection framework, which can mark multiple hotspot locations within a region that is much larger than a clip applied in previous works, as shown in Figure 2. To the best of our knowledge, this is the first time a hotspot detector is designed to detect multiple process weak points within very large scale layout clips in one step feed-forward detection. The framework contains a regression and classification multi-task flow which guide to higher accuracy, higher detection speed and lower false alarm penalty. The main contributions of this paper are listed as follows:

- We construct a deep neural network specific for region-based hotspot detection task and our network framework can be efficiently trained end-to-end.
- We present a clip proposal network and a refinement stage to further improve accuracy and reduce false alarm.
- We apply a novel classification and regression strategy to reduce the detection region and make the multiple hotspot detection become realizable in large scale.
- Experimental results show that our proposed framework has great advantages over state-of-the-art and can achieve 6.14% accuracy improvement and 45× speedup on average.

The rest of the paper is organized as follows. Section 2 introduces basic concepts and gives problem formulation. Section 3 discusses the details of the proposed end-to-end neural framework. Section 4

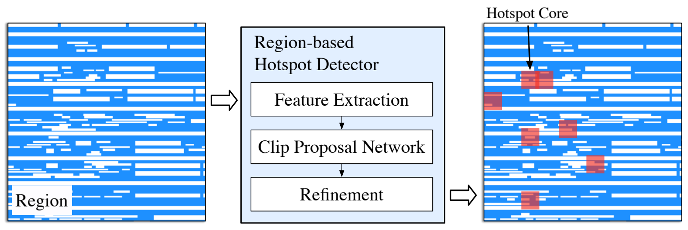

**Figure 2: The proposed region-based hotspot detection flow.**

lists experimental comparisons with state-of-the-art, followed by conclusion in Section 5.

## 2 PRELIMINARIES

The lithographic process which transfers the designed layout patterns onto the silicon wafers involves a lot of variations. Some patterns are sensitive to these variations and may cause reduction of manufacturing yield as a result of potential circuit failures. Layout patterns that are sensitive to process variations are defined as *hotspots*. We also define a *hotspot clip* as a clip that contains at least one hotspot at its core region [20]. A hotspot is correctly detected if it is located in the core region of a clip that is marked as hotspot by a hotspot detector. The core region applied in this paper is the middle third region of the clip. In this paper, the following definitions and metrics are used to evaluate performance of a hotspot detector.

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

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

It should be noted that the accuracy is also equivalent to the *true positive rate* and the false alarm corresponds to the number of *false positives*. Because a good hotspot detector aims to recognize as many real hotspots as possible and avoids incorrect predictions on non-hotspot patterns, with the evaluation metrics above, we define region-based hotspot detection (R-HSD) problem as follows.

**Problem 1** (R-HSD)**.** Given a layout region that consists of hotspot and non-hotspot patterns, the objectives of region-based hotspot detection is training a model to locate and classify all the hotspot and the non-hotspot within the region, such that the detection accuracy is maximized with minimum false alarm penalty.

## 3 R-HSD NEURAL NETWORK

Our proposed region-based hotspot detection (R-HSD) neural network, as illustrated in Figure 2, is composed of three steps: (1) feature extraction, (2) clip proposal network, and (3) refinement. In this section, we will discuss each step with details. At first glance, R-HSD problem is similar to object detection problem, which is a hot topic in computer vision domain recently. In object detection problem, objects with different shapes, types and patterns are the target to be detected. However, as we will discuss, there is a gap between hotspot detection and object detection, e.g. the hotspot pattern features are quite different from the objects in real scenes, thus typical strategies and framework utilized in object detection cannot be applied here directly.

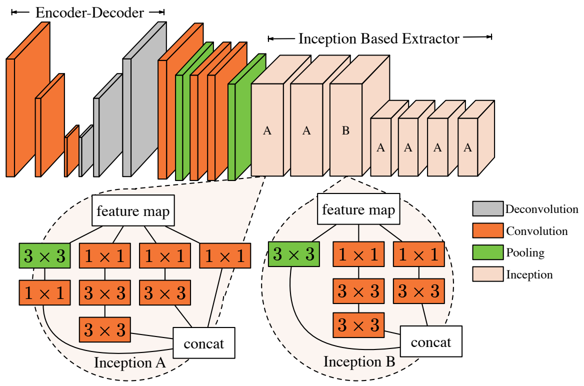

**Figure 3: Tensor structure of feature extractor.**

### 3.1 Feature Extraction

Because of wide variations between traditional objects in real scenario and VLSI layout patterns, it is extremely crucial to design an appropriate feature extractor in our neural network framework. Our feature extractor aims at transforming original layout feature non-linearly while reducing computation overhead. Besides, we also tend to enrich the feature diversity with fewer parameters. Based on these two major principles, a joint encoder-decoder structure and inception-based module for efficient feature extraction is designed, connected with three convolution layers and two max-pooling layers. This connection is applied to compress the feature map size from $224 \times 224$ to $56 \times 56$ which can bring speed up at training stage and inference stage.

Yang *et al.* [15] successfully applied DCT to manually extract layout pattern features. However, this manual design of the feature expression may ignore some key features thus may not give a comprehensive expression. Furthermore, the processing time of DCT is very time consuming. Compared to the manual rule based DCT, our proposed feature extractor can transform the origin layout into a network-compatible expression automatically. Embedded with the whole neural network, our feature extractor can be trained and tested end-to-end thus is fast and flexible.

The convolutional network structure designed by [15] performs well feature extraction, but its structure is too simple thus is limited to single clip hotspot detection problem. The naive replacement on DCT without redesign on further extractor is not available to region based task. There are two metrics for us to think about how to design new structure in our work. One is go deeper with more layers, the other is go wider with multiple branches.

#### 3.1.1 Encoder-Decoder Structure.

The encoder includes three convolution layers and the decoder includes three deconvolution layers. The encoder extracts the features from origin image space to high dimensions latent space by increasing the number of the convolution kernels, then the decoder downsamples the features from high dimensions to origin image space with the symmetrical kernel settings.

**Convolution Layer**. The convolution layer is the major part of the convolution neural network which has been widely used. The operation between tensor and kernel can be expressed as:

$$\mathbf{F} \otimes \mathbf{K}(j,k) = \sum_{i=1}^{c} \sum_{m_0=1}^{m} \sum_{n_0=1}^{m} \mathbf{F}(i, j-m_0, k-n_0)\mathbf{K}(m_0, n_0), \quad (1)$$

with tensor $\mathbf{F} \in \mathbb{R}^{c \times p \times p}$ and kernel $\mathbf{K} \in \mathbb{R}^{c \times m \times m}$.

**Deconvolution Layer**. In contrast to convolutional layers, deconvolution layers do the inverse operation which maps the single input feature point to multiple outputs, which can be considered as a feature generation. The expression can be written as:

$$\mathbf{T} \otimes \mathbf{K}(j,k) = \sum_{i=1}^{c} \sum_{m_0=1}^{m} \sum_{n_0=1}^{m} \mathbf{T}(i, j-m_0, k-n_0)\mathbf{K}(m-m_0, n-n_0),$$

where $\mathbf{T} \in \mathbb{R}^{c \times n \times n}$ is the tensor $\mathbf{F} \in \mathbb{R}^{c \times p \times p}$ padded with zero and $n = (m-1) \times 2 + p$, kernel $\mathbf{K} \in \mathbb{R}^{c \times m \times m}$. Here padding size is the number of pixels we fill on the border of the origin feature maps. In our experiments, padded feature map is equal to the output size. Kernel size is the size of the deconvolution kernel. We use $3 \times 3$ kernel size which is same as the encoder part. During training, the feature map of the deconvolution layer is updated with the back propagration.

#### 3.1.2 Inception-based Structure.

According to recent progress of deep learning based research in computer vision region, a deeper neural network can give a much more robust feature expression and get higher accuracy compared to shallow structure as it increases the model complexity. However it also brings sacrifice on speed at both inference stage and training stage. Another point we need to concern is that the feature expression of the layout pattern is monotonous, while feature space of layout patterns is still in low dimension after we transform it by the Encoder-Decoder structure. According to these issues, we propose an Inception-based structure. The following three points are the main rules of our design:

- Increase the number of the filters in width at each stage. For each stage, multiple filters do the convolution operation with different convolution kernel size and then concatenate them in channel direction as feature fusion.
- Prune the depth of the output channel for each stage.
- Downsample the feature map size in height and width direction.

With the above rules, the inception structure [21] can take a good balance between the accuracy and the time. The blobs shown in Figure 3 are what we apply in our framework. We construct the module A with the operation stride one and four branches. The aim of the module A is to extract multiple features without donwsampling the feature map. The operation stride of each layer in module B is two. This setting makes the out feature map half than the input, which decreases the operations in further. We only use one Module B here, because the feature map size should not be too small, while the low dimension of feature expression in final layers may bring negative affects the final result.

The $1 \times 1$ convolution kernel with low channel numbers brings the dimension reduction which controls the number of the parameters and operations. The multiple branches bring more abundant feature expressions, which give the network ability to do the kernel selection with no operation penalty.

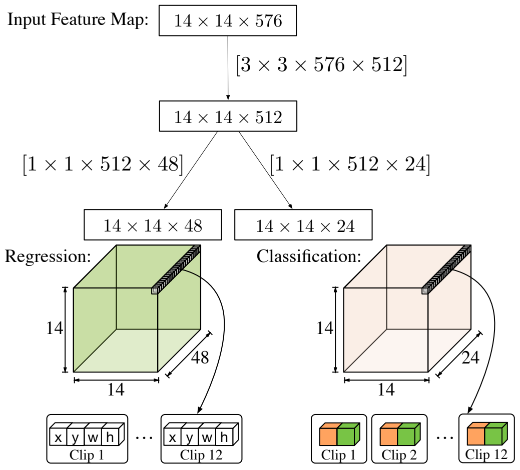

**Figure 4: Kernel Work flow of Clip Proposal Network.**

## 3.2 Clip Proposal Network

Given the extracted features, a clip proposal network is developed here to detect potential hotspot clips. For both feature maps and convolutional filters, the tensor structures of the clip proposal network are illustrated in Figure 4. Per preliminary experiments, clips with single aspect ratio and scale (e.g. square equal to the ground truth) may lead to bad performance. Therefore, for each pixel in feature map, a group of 12 clips with different aspect ratios are generated. The network is split into two branches: one is for classification and the other is for regression. In classification branch, for each clip, a probability of hotspot and a probability of non-hotspot are calculated through softmax function. In regression branch, the location and the shape of each clip is determined by a vector $[x, y, w, h]$.

#### 3.2.1 Clip Pruning.

While the number of clips is extremely large during training. If we reserve all the clips generated, it takes a long time to converge to low classification and regression loss at training stage but still lead to bad performance at inference stage. Because these redundant clips have medium intersection area to the groundtruth which are the noises to the classifier. For the clip regression task, it is not reasonable to consider linear regression on these clips with large offset to the groundtruth clips. To overcome this problem, we consider automatic clip pruning in our neural network.

We first define Intersection of Union (IoU) as follows:

$$\text{IoU} = \frac{clip_{groundtruth} \bigcap clip_{generated}}{clip_{groundtruth} \bigcup clip_{generated}}. \quad (2)$$

We then apply the following clip pruning rules:

- A clip's IoU with ground truth clip higher than 0.7 should be reserved as positive sample;
- The clip's IoU with any ground truth highest score should be reserved as positive sample;
- A clip's IoU with ground truth clip lower than 0.3 should be reserved as negative sample;
- Rest of clips do no contribution to the network training.

#### 3.2.2 Hotspot Non-Maximum Suppression.

After the classification and regression, the distance between some neighbor hotspots is quite close to each other, there exists set of overlapped

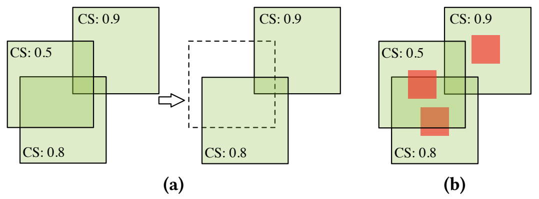

Figure 5: Examples of (a) conventional non-maximum suppression, and (b) the proposed hotspot non-maximum suppression.

clips which have the same regression target. To avoid these redundant calculation at training and inference stages, we develop a hotspot non-maximum suppression (h-nms) strategy to remove these clips. The h-nms strategy is shown in Algorithm 1.

**Algorithm 1** hotspot non-maximum suppression

```
1:  sorted_ws ← sorted clip set;
2:  $k$ ← size of clip set;
3:  for $i \leftarrow 1, 2, \ldots, k$ do
4:      current_w ← sorted_ws[$i$];
5:      for $j \leftarrow i, i+1, \ldots, k$ do
6:          compared_w ← sorted_ws[j];
7:          $Overlap$ ← Centre_IoU(current_w, compared_w);
8:          if $Overlap > threshold$ then
9:              Remove compared_w; $k \leftarrow k - 1$;
10:         end if
11:     end for
12: end for
13: return sorted_ws;
```

The elements of *sorted_ws* are arranged in descending order according to the classification score (line 1). `Centre_IoU` is a function returning the IoU score which focus on overlap of cores (line 7). If the IoU is larger than the threshold, we remove the clip with the lower score from the list (lines 8–10). The removed clips will not contribute to further operation. Note that in our experiment, the IoU threshold value is set to 0.7.

Compared to conventional non-maximum suppression method, our proposed method takes advantage of the structural relation between core region and clips which avoid the error dropout during the training. An example is shown in Figure 5, the clip with 0.5 classification score (CS) is removed in conventional method, while saved in our method if we consider the core within each clip.

## 3.3 Refinement

After the prediction of the first classification and regression in clip proposal network stage, we get a rough prediction on hotspot localization and filtered region which is classified as non-hotspot. While the greedy method of clip filtering cannot guarantee all the reserved clips are classified correctly, the false alarm may be too high. To bring a robust detection with lower false alarm, we further construct refinement stage in the whole neural network, which includes a Region of Interests (RoI) pooling layer, three inception modules, as well as another classification and regression. The structure of Refinement is shown in Figure 6.

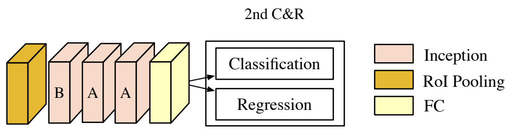

Figure 6: Tensor structure of Refinement.

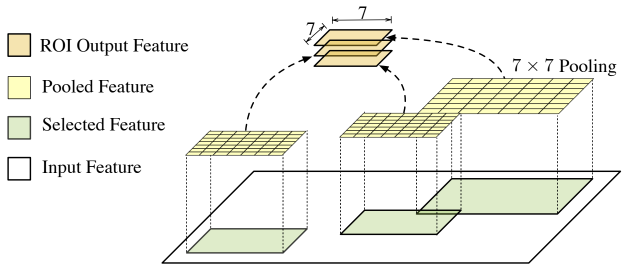

Figure 7: Visualized $7 \times 7$ RoI pooling.

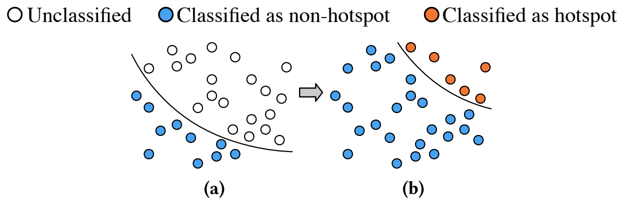

Figure 8: (a) 1st hotspot classification in clip proposal network; (b) The labelled hotspots are fed into 2nd hotspot classification in refinement stage to reduce false alarm.

**RoI Pooling**. The coordinates of each clip are the actual location from the origin input image. We scale down the coordinates to conform with the spatial extent of the last feature map before the refinement. In traditional image processing, the most common ways to resize the image are cropping and warping. The crop method cuts the pattern boundary to fix the target size which leads to information loss. The warping will reshape the size which change the shape of origin features. Here we apply Region of Interests (RoI) pooling to transform the selected feature region $h \times w$ to a fixed spatial size of $H \times W$ ($H$ and $W$ are the hyperparameters, and we use $7 \times 7$ in this work). For each pooled feature region $\lfloor h/H \times w/W \rfloor$, the max-pooling is applied independently. The RoI pooling transforms clips with different sizes into a fixed size which reserves the whole feature information and makes further hotspot classification and regression feasible. Figure 7 gives an example of RoI pooling operations.

Besides classification and regression in clip proposal network, here additional classification and regression are designed to fine-tune the clip location and give a more reliable classification result. At this stage, most non-hotspot clips have been removed, thus two stage of hotspot classification can efficiently reduce false alarm. Figure 8 illustrates the flow of the two stage hotspot classification.

## 3.4 Loss Function Design

We design a multi-task loss function called Classification and Regression (C&R) to calibrate our model. As shown in Figure 4 and Figure 6, C&R is applied both in clip proposal network stage and refinement stage. The input tensors of 1st C&R are boxes in Figure 4. $W$, $H$ and $C$ are width, height and channel respectively. The probability score of the hotspot, non-hotspot and prediction of clip coordinates are grouped in channel direction. Here $x$, $y$ are the coordinates of the hotspot, which means the centre of the clip area. $w$, $h$ are the width and height of the clip. In 2nd C&R, the tensor flow of the classification and regression are same as [22] using fully-connected layers.

In the task of region-based hotspot detection, $h_i$ is the predicted probability of clip $i$ being a hotspot, $h'_i$ is the groundtruth of clip $i$. $\boldsymbol{l}_i = \{l_x, l_y, l_w, l_h\} \in \mathbb{R}^4$ and $\boldsymbol{l}'_i = \{l'_x, l'_y, l'_w, l'_h\} \in \mathbb{R}^4$ are assigned as coordinates of clips with index $i$ representing the encoded coordinates of the prediction and groundtruth respectively. The encoded coordinates can be expressed as:

$$\begin{aligned} l_x &= (x - x_g)/w_g, & l_y &= (y - y_g)/h_g, \\ l'_x &= (x' - x_g)/w_g, & l'_y &= (y' - y_g)/w_g, \\ l_w &= \log(w/w_g), & l_h &= \log(h/h_g), \\ l'_w &= \log(w'/w_g), & l'_h &= \log(h'/h_g), \end{aligned} \tag{3}$$

Variables $x, x_g$ and $x'$ are for the prediction of clip, g-clip and groundtruth clip respectively (same as the y,w and h).

The classification and regression loss function for clips can be expressed as:

$$\begin{aligned} L_{C\&R}(h_i, \boldsymbol{l}_i) =& \alpha_{loc} \sum_i h'_i l_{loc}(\boldsymbol{l}_i, \boldsymbol{l}'_i) + \sum_i l_{hotspot}(h_i, h'_i) \\ &+ \frac{1}{2}\beta(\|\boldsymbol{T}_{loc}\|_2^2 + \left\|\boldsymbol{T}_{hotspot}\right\|_2^2), \end{aligned} \tag{4}$$

where $\beta$ is a hyperparameter which control the regularization strength. $\alpha_{loc}$ is the hyperparameter which control the balance between two tasks. $\boldsymbol{T}_{loc}$ and $\boldsymbol{T}_{hotspot}$ are the weights of the neural network. For elements $l_i[j]$ and $l'_i[j]$ $(j \in [1, 4])$ in $\boldsymbol{l}_i, \boldsymbol{l}'_i$ respectively, $l_{loc}$ can be expressed as

$$l_{loc}(l_i[j], l'_i[j]) = \begin{cases} \frac{1}{2}(l_i[j] - l'_i[j])^2, & \text{if } |l_i[j] - l'_i[j]| < 1, \\ |l_i[j] - l'_i[j]| - 0.5, & \text{otherwise,} \end{cases} \tag{5}$$

which is a robust $L_1$ loss used to avoid the exploding gradients problem at training stage. $l_{hotspot}$ is the cross-entropy loss which is calculated as:

$$l_{hotspot}(h_i, h'_i) = -(h_i \log h'_i + h'_i \log h_i). \tag{6}$$

In Equation (4), we apply the $L_2$ regularization to the loss function, which is the sum of the squares of all the weights in the network. The L2 regularization penalizes peaky weight vectors and prefers diffuse weight vectors. Due to multiplicative interactions between weights and features, the L2 regularization term has appealing property of encouraging the network to use all of its inputs rather than skewed on partial of its inputs.

## 4 EXPERIMENTAL RESULTS

Our region-based hotspot detection flow is evaluated on ICCAD CAD Contest 2016 Benchmarks [25], which contains four designs are shrunk to match EUV metal layer design rules. Ground truth hotspot locations are label according to the results of industrial $7nm$ metal layer EUV lithography simulation under a given process window. Because there are no defects found with lithography simulation on the first benchmark, all our experiments are conducted only on rest three designs, each of which is split in half with one part used for training and the other one used for testing. We implement our large scale hotspot detection framework with Tensorflow [26] in Python, and test it on a platform with a Xeon Silver 4114 processor and an Nvidia GTX Titan graphic card. *Note that three training layouts are merged together to train one model that will be used in the inference stage.* In the following experiments, the neural network is trained with following parameter settings: *input size* = 256×256 ( corresponding to $2.56nm \times 2.56nm$ ), *batch size* = 12, *initial learning rate* = 0.002 (decay ten times for each 30000 steps), *aspect ratio* = [0.5, 1.0, 2.0], *scales* = [0.25, 0.5, 1.0, 2.0]. The parameters of the loss function are not carefully chosen, what we use are $\beta = 0.2$, $\alpha_{loc} = 2.0$. At inference stage, we follow the same data generation rule applied in [16].

We list the detailed result comparison in Table 1. Column "Bench" lists three benchmarks used in our experiments. Columns "Accu", "FA", "Time" denotes hotspot detection accuracy, false alarm count and detection runtime, respectively. Column "TCAD'18" lists the result of a deep learning-based hotspot detector proposed in [16] that adopts frequency domain feature extraction and biased learning strategy. We also implement two baseline frameworks that employ Faster R-CNN [23] and SSD [24], respectively, which are two classic techniques match our region-based hotspot detection objectives well. The corresponding results are listed in columns "Faster R-CNN [23]" and "SSD [24]". The results show that our framework get better hotspot detection accuracy on average with 6.14% improvement with ∼ 200 less false alarm penalty compared to [16]. Especially, our framework behaves much better on `Case2` with 93.02% detection accuracy compared to 77.78%, 1.8% and 71.9% for [16], Faster R-CNN and SSD respectively. The advantage of the proposed two-stage classification and regression flow can also be seen here that [16] achieves similar hotspot detection accuracy compared to our framework but has extremely large false alarms that will introduce additional. It should be noted that the detection runtime is much faster than [16] thanks to the region-based detection scheme. We can also observe that although Faster R-CNN and SSD are originally designed for large region object detection, they perform very poor on hotspot detection tasks which reflects the effectiveness and efficiency of our customized framework.

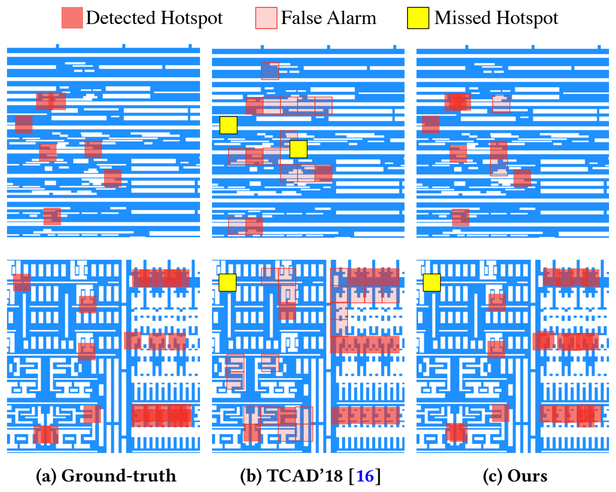

**Figure 9: Visualization of different hotspot detection results.**

Table 1: Comparison with State-of-the-art

| Bench | TCAD'18 [16] | | | Faster R-CNN [23] | | | SSD [24] | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Accu (%) | FA | Time (s) | Accu (%) | FA | Time (s) | Accu (%) | FA | Time (s) | Accu (%) | FA | Time (s) |
| Case2 | 77.78 | 48 | 60.0 | 1.8 | 3 | 1.0 | 71.9 | 519 | 1.0 | 93.02 | 17 | 2.0 |
| Case3 | 91.20 | 263 | 265.0 | 57.1 | 74 | 11.0 | 57.4 | 1730 | 3.0 | 94.5 | 34 | 10.0 |
| Case4 | 100.00 | 511 | 428.0 | 6.9 | 69 | 8.0 | 77.8 | 275 | 2.0 | 100.00 | 201 | 6.0 |
| Average | 89.66 | 274.0 | 251.0 | 21.9 | 48.7 | 6.67 | 69.0 | 841.3 | 2.0 | 95.8 | 84 | 6.0 |
| Ratio | 1.00 | 1.00 | 1.00 | 0.24 | 0.18 | 0.03 | 0.87 | 3.07 | **0.01** | **1.07** | **0.31** | 0.02 |

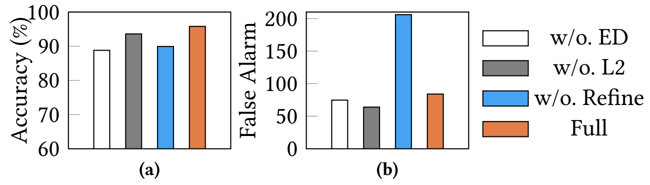

**Figure 10: Comparison among different settings on (a) average accuracy and (b) average false alarm.**

We also study the how different configurations of our framework affect the performance. Figure 10 illustrates the contribution of encoder-decoder structure, L2 regularization and refinement stage to our backbone neural network. "w/o. ED" denotes the framework without the encoder-decoder structure, "w/o. L2" stands for the framework without the L2 regularization, "w/o. Refine" denotes the framework without the refinement classification and regression, and "all" is our framework with entire techniques. The histogram shows that with the encoder-decoder structure, we get 7% accuracy improvement on average, which indicate that the encoder-decoder structure gives a more efficient feature expression than the original input. With the L2 regularization, we get 2.2% improvement in all cases, which means under the same experiment settings, the L2 regulariation resolves the overfitting problem effectively. Comparing the whole framework with the model without Refinement, the model with Refinement reduces 59.2% of the false alarm and get 5.88% further improvement on accuracy.

## 5 CONCLUSION

In this paper, we have proposed an innovative end-to-end region-based hotspot detection framework. Our feature extractor provides a self-adaptive way perform feature transformation that is very compatible with convolution neural networks and time saving. The clip proposal network locates the potential hotspot in a regression way, which is more efficient. We take advantage of L2 regularization's property to prevent over-fitting and get higher performance. Additionally, our classification and regression strategy with refinement reduces false alarms and increases the accuracy with a remarkable speed. The experimental results show that our framework outperforms the current deep learning based models. We hope this paper can give a new perspective on deep learning based hotspot detection and provide a more powerful solution for advanced design for manufactorability (DFM) research.

## ACKNOWLEDGMENTS

This work is supported in part by The Research Grants Council of Hong Kong SAR (Project No. CUHK24209017).

## REFERENCES

[1] 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, 2014.
[2] 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.
[3] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. DAC*, 2009.
[4] 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, 2011.
[5] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Proc. ASPDAC*, 2012.
[6] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in *Proc. SPIE*, vol. 9427, 2015.
[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," *IEEE TCAD*, vol. 34, no. 3, 2015.
[8] 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, 2015.
[9] T. Matsunawa, B. Yu, and D. Z. Pan, "Laplacian eigenmaps-and bayesian clustering-based layout pattern sampling and its applications to hotspot detection and optical proximity correction," *JM3*, vol. 15, no. 4, 2016.
[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.
[11] Y. Tomioka, T. Matsunawa, C. Kodama, and S. Nojima, "Lithography hotspot detection by two-stage cascade classifier using histogram of oriented light propagation," in *Proc. ASPDAC*, 2017.
[12] H. Zhang, F. Zhu, H. Li, E. F. Y. Young, and B. Yu, "Bilinear lithography hotspot detection," in *Proc. ISPD*, 2017.
[13] T. Matsunawa, S. Nojima, and T. Kotani, "Automatic layout feature extraction for lithography hotspot detection based on deep neural network," in *SPIE Advanced Lithography*, vol. 9781, 2016.
[14] M. Shin and J.-H. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *JM3*, vol. 15, no. 4, 2016.
[15] 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, 2017.
[16] 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*, 2018.
[17] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. DAC*, 2018.
[18] W. Ye, Y. Lin, M. Li, Q. Liu, and D. Z. Pan, "LithoROC: lithography hotspot detection with explicit ROC optimization," in *Proc. ASPDAC*, 2019.
[19] 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.
[20] A. J. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, 2012.
[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.
[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015.
[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. ECCV*, 2016.
[25] R. O. Topaloglu, "ICCAD-2016 CAD contest in pattern classification for integrated circuit design space analysis and benchmark suite," in *Proc. ICCAD*, 2016.
[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016.