Hotspot Detection via Attention-Based Deep Layout Metric Learning

Hao Geng, *Student Member, IEEE*, Haoyu Yang, Lu Zhang, Fan Yang, *Member, IEEE*, Xuan Zeng, *Senior Member, IEEE*, and Bei Yu, *Member, IEEE*

***Abstract*—With the aggressive and amazing scaling of the feature size of semiconductors, hotspot detection has become a crucial and challenging problem in the generation of optimized mask design for better printability. Machine learning techniques, especially deep learning, have attained notable success on hotspot detection tasks. However, most existing hotspot detectors suffer from suboptimal performance due to two-stage flow and less efficient representations of layout features. What is more, most works can only solve simple benchmarks with apparent hotspot patterns like ICCAD 2012 Contest benchmarks. In this article, we first develop a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are jointly performed. An attention mechanism-based deep convolutional neural network (CNN) is exploited as the backbone to learn embeddings for layout features and classify the hotspots simultaneously. The experimental results demonstrate that our framework achieves accuracy improvement over prior arts with fewer false alarms and faster inference speed on much more challenging benchmarks.**

***Index Terms*—Attention module, deep metric learning, layout hotspot detection, via layer benchmark.**

I. Introduction

As THE technology node of integrated circuits scales down to 7 nm and beyond, the techniques for lithographic processes are supposed to manage the ever-shrinking feature size. However, owing to the delayed progress of lithography techniques, lithographic processes variations emerge during the manufacturing, and thus lead to yield loss. To combat the variations, different kinds of approaches are developed. One is mask optimization through various resolution enhancement techniques (RETs) [1]–[8]. Another way to lighten variations, especially for some sensitive layout patterns (a.k.a. lithography hotspot), is so-called hotspot detection. Detecting these hotspots at the early stage of mask synthesis is imperative. Many techniques of early detection for hotspots have been proposed to ensure manufacturability. The prevalently applied lithography simulation technique could attain high accuracy, nevertheless, it is also known to be pretty time consuming. Therefore, two other types of quick detection methods are proposed as alternatives to lithography simulation. One is based on pattern matching [9], [10], and the other is machine learning driven. The pattern matching approaches take input predefined pattern library of known hotspots, but they cannot detect unknown hotspots. Fortunately, detection methods which are built upon machine learning methodologies [11]–[16], and particularly deep learning techniques [17]–[25], are able to offer fast and accurate solutions to both known and unknown hotspot patterns.

For example, motivated by the recent advances of deep learning in other domains like computer vision, Yang *et al.* [17] first proposed the hotspot detector based on a shallow convolutional neural network (CNN), where layout clips were first converted into the frequency domain via discrete cosine transform (DCT) before fed into networks. To extract DCT features in different scales, the inception modules are introduced in the work [18] to modify the structure of neural networks. In [19], to overcome the limitation imposed by the number of labeled hotspots or nonhotspots, the concept of semi-supervised learning was introduced so that unlabeled data can be harnessed as well. By considering input layout clips as binary images, Jiang *et al.* [20] employed a binarized neural network to further speed up the detection process.

These prior arts have been proven very effective, nevertheless, most of them fall into a two-stage flow, where layout feature extraction and detection process are separable. Moreover, existing techniques for layout feature extraction lack both good understandings of similarity among different layout clips and guidances from supervised information. Only a few hotspot detectors [20] ensemble the feature extraction and detection into a one-stage framework. Unfortunately, the features which are automatically learned by convolutional kernels may be ill-separated owing to the lack of discrimination-oriented similarity metrics. Without an effective weighing procedure on the features per their informative significance, those redundant or even misleading features may very well degrade the overall prediction accuracy and performance. In light of these facts, those existing works can only solve simple benchmarks

Manuscript received 26 March 2021; revised 20 July 2021; accepted 24 August 2021. Date of publication 14 September 2021; date of current version 19 July 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFA0711900 and Grant 2020YFA0711903; in part by the Research Grants Council of Hong Kong SAR under Project CUHK14208021 and Project CUHK14209420; and in part by the National Natural Science Foundation of China (NSFC) Research Projects under Grant 61822402, Grant 61774045, Grant 61929102, and Grant 62011530132. The preliminary version has been presented at the IEEE/ACM International Conference on Computer-Aided Design (ICCAD) in 2020. This article was recommended by Associate Editor P. Gupta. *(Corresponding author: Bei Yu.)*

Hao Geng, Lu Zhang, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

Haoyu Yang is with the ASIC and VLSI Research Group, NVIDIA Corporation, Austin, TX 78717 USA.

Fan Yang and Xuan Zeng are with the State Key Laboratory of ASIC and System, Microelectronics Department, Fudan University, Shanghai 201203, China.

Digital Object Identifier 10.1109/TCAD.2021.3112637

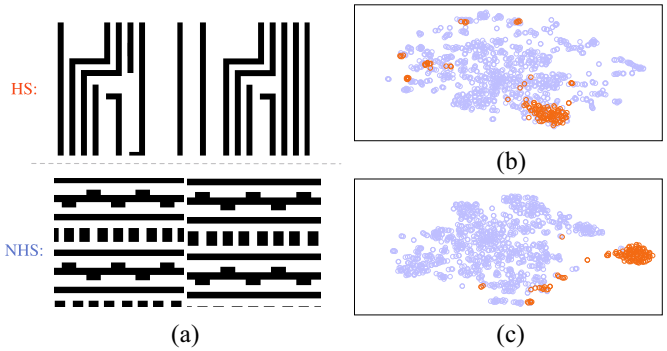


Fig. 1. Snapshots of layout clips from ICCAD12 benchmarks [26] and Barnes-Hut t-SNE [27] visualizations of feature embeddings on the same benchmarks. (a) Examples of hotspots and nonhotspots. (b) DCT feature embeddings of TCAD'19 [17]. (c) Feature embeddings of our proposed framework. "HS" is used to represent hotspot clips, while "NHS" refers to nonhotspot clips.

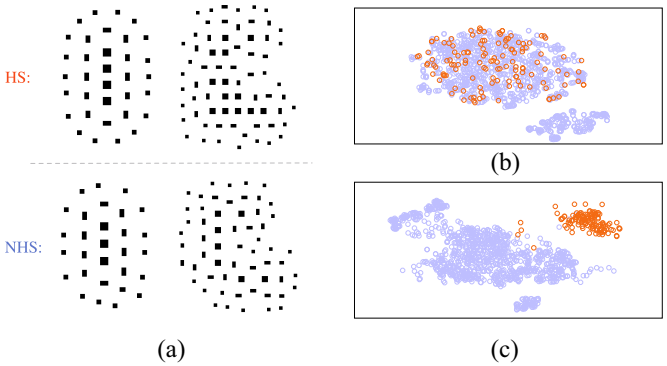


Fig. 2. Snapshots of layout clips from the via layer benchmarks and Barnes-Hut t-SNE visualizations of feature embeddings on the same benchmarks. (a) Examples of hotspots and nonhotspots. (b) DCT feature embeddings of TCAD'19. (c) Feature embeddings of our proposed framework. The via layer benchmarks consist of via patterns that contain vias and model-based SRAFs. Here, "via" refers to the via connecting multiple metal layers.

with apparent hotspot patterns like ICCAD 2012 Contest benchmarks, whilst in the case where nonhotspot patterns and hotspot patterns become increasing similar brings a big challenge to them. To visualize our concern, we sample some hotspots and nonhotspots from ICCAD12 [26] and new via layer benchmarks as exemplars in Figs. 1(a) and 2(a). Besides, the embeddings learned from TCAD'19 and ours on both benchmark suites are projected into a two-dimension space [shown in Figs. 1(a) and (c) and 2(b) and (c)]. It can be seen that, in easy benchmarks like ICCAD12, the layout clips sharing the same label are usually similar, while the layout clips from different classes have prominent differences. Opposite to ICCAD12 benchmarks, layout clips in the via benchmarks from different classes may look very similar. What's worse, the lack of diversity of via layer patterns introduces an additional challenge to existing hotspot detectors. According to the distributions of layout feature embeddings displayed in Figs. 1(b) and 2(b), we can infer that although existing works like TCAD'19 work well on ICCAD12 benchmarks, they may have poor performance on more challenging benchmarks. As demonstrated above, layout patterns in such easy benchmarks like ICCAD12 contain too much discriminative information to determine the true states of hotspot detectors.

In preliminary work, we argue that it is of great importance to learn good embeddings of layout clips, which can be assembled into a one-stage hotspot detection flow. In accordance with this argument, we have proposed an end-to-end detection framework in [23] where the multiple tasks, learning embeddings, and classification, are jointly performed and mutually benefited. Furthermore, in order to focus on important features and suppressing unnecessary ones, we have introduced an attention module and incorporate it into [23]. To the best of our knowledge, [23] is the first work for layout embedding learning seamlessly combining with hotspot detection task, and there is no prior work applying attention mechanism-based deep metric learning into hotspot detection. However, there still exist some defects in our preliminary design. For example, the vanilla convolution operator is not able to fully capture the influences caused by the propagation of diffracted light from a mask pattern. A lager receptive field of convolution operator is in demand. Additionally, the constant margin in triplet-based metric learning loss would result in over- or under-sampling problems, and consequently hinder the fitting of the model. The relationships among the layout and its own augmented versions like flipped and rotated ones are worth to be considered. To evaluate the true efficacy of existing works and the proposed detector, we adopt a much more challenging benchmark suite. Figs. 1(c) and 2(c) prove the performance of the proposed framework on both easy and more challenging benchmarks to some extent. Our main contributions are listed as follows.

1) Apply attention mechanism to learn better feature embeddings.
2) Leverage deep layout metric learning model to learn the good layout feature embeddings.
3) Incorporate feature embedding and hotspot detection into a one-stage multibranch flow.
4) The deformable convolution operator is introduced to further boost the preliminary hotspot detector.
5) Redesign the loss function with adding a self-adaptive margin in the original triplet layout learning loss.
6) Consider a self-contrastive layout learning loss on different augmented views of the same layout clip image.
7) Analyze the generalized representation ability of triplet-based deep metric learning algorithm.
8) A more challenging benchmark suite that fills the void created by previous easy benchmarks is adopted to test the performance of the detectors.
9) The proposed detector improves the performance on accuracy, false alarm, and runtime of inference compared to the state-of-the-art frameworks.

The remainder of this article is organized as follows. Section II introduces some preliminaries on metrics and problem formulation in the article. Section III first gives a whole view of the proposed preliminary framework, then illustrates its backbone network with the applied inception and attention modules, as well as the multibranch design. Section IV describes the loss functions and some training strategies of the preliminary detection framework. Section V

shows the details of the enhanced hotspot detector. Section VI depicts the new via benchmarks and then provides experimental results, followed by the conclusion in Section VII.

II. Problem Formulation

Owing to the manufacturing process variation, designed layout patterns stochastically cause defects on wafers during the lithographic process. These sensitive patterns may cause reduction of manufacturing yield or even potential circuit failures. Layout patterns that are sensitive to process variations are defined as hotspots.

Accordingly, our task can be defined as an image classification problem if layout clips are translated into images. The label of a layout clip is given according to the information that whether the core region [26] of the clip contains hotspots or not.

We adopt the same metrics exploited in previous work to evaluate the performance of our proposed hotspot detector. The following show definitions of these metrics.

Definition 1 (Accuracy): The ratio between the number of correctly categorized hotspot clips and the number of real hotspot clips.

Definition 2 (False Alarm): The number of nonhotspot clips that are classified as hotspots by the classifier.

With the evaluation metrics above, our problem is formulated as follows.

Problem 1 (Hotspot Detection): Given a collection of clips containing hotspot and nonhotspot layout patterns, the objective of deep layout metric learning-based hotspot detection is training a model to learn optimal feature embeddings and classify all the clips so that the detection accuracy is maximized whilst the false alarm is minimized with a less runtime overhead.

III. Architecture of Preliminary Hotspot Detector

A. Overall Framework

The prior arts are either in a two-stage framework or lacking discriminative feature extractor. Especially, isolating the layout feature extraction and follow-up prediction may make the whole framework be susceptible to converge to suboptimal performance. The recent success of applying the end-to-end learning structure in numerous EDA applications [28]–[30] has demonstrate the argument in another angel. Considering the concern and discovery, our proposed preliminary algorithm adopts a well-designed multibranch flow which works in an end-to-end manner. During the training, the proposed multibranch flow simultaneously works on two branches: one is feature embedding and the other is classification. When the training process finishes, our hotspot detector identifies not only layout feature embeddings, but also a pretty good boundary to divide the hotspots and nonhotspots in embedding space. By the merit of the end-to-end nature, our detector is fast and flexible for both training and inference phases.

Fig. 3 shows the architecture of the proposed framework, where "⊗" stands for the element-wise multiplication. The proposed framework is composed of three main components: 1) backbone network which is based on inception structures

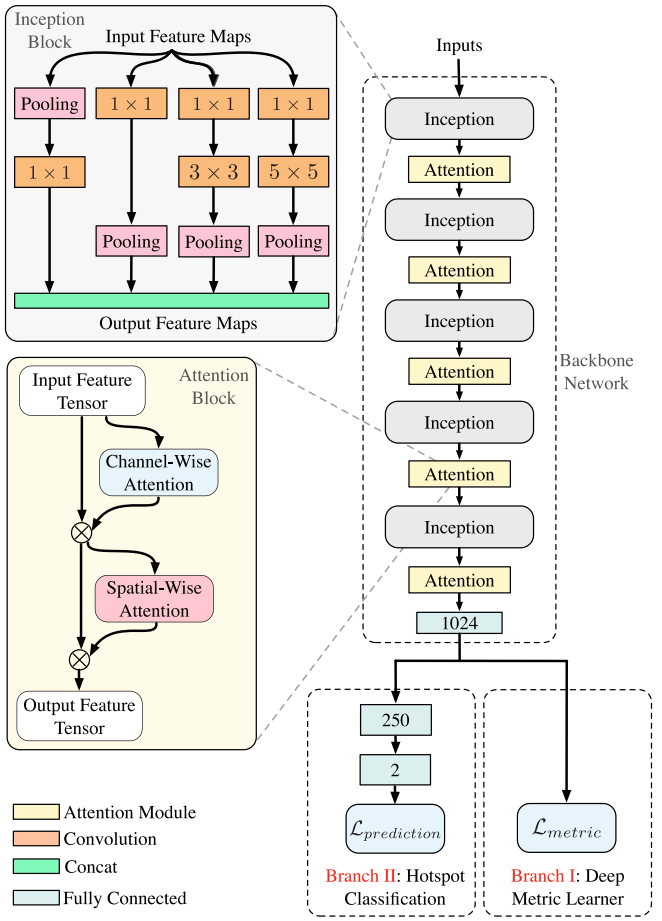


Fig. 3. Architecture of the proposed preliminary hotspot detector.

and attention modules. It is shared by two jointly learned tasks and then is split into two branches. The whole backbone includes five inception modules, five attention modules, and one fully connected layer; 2) for deep layout metric learner branch, it is guided by a triplet loss function to strive for good embeddings of layout clips; and 3) for hotspot classification branch, it behaves as an ordinary learning model-based hotspot detector as in other works.

B. Backbone: Inception Block

Recent progress of deep learning techniques in computer vision reveals that by virtue of the increasing model complexity, a deeper neural network achieves a more robust feature expression, and a higher accuracy comparing to a shallow one. However, deeper networks are susceptible to be overfitted, and gradient vanish emerges. What is worse, the turn-around-time at the inference stage and training stage are greatly affected, too.

In our context, more cases are needed to concern. For instance, a layout pattern which usually contains several rectangles is monotonous. Another example is in our via pattern, the distances between vias and surrounding subresolution assist features (SRAFs), the distances among vias have pretty large variations. A single-sized kernel cannot capture multiscale information. To tackle these issues, we employ an Inception-based structure [31] which consists of several convolutional kernels with multiple sizes. At the same level,

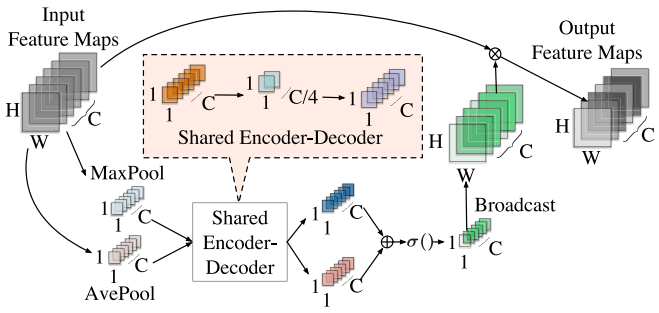


Fig. 4. Channel-wise attention module.

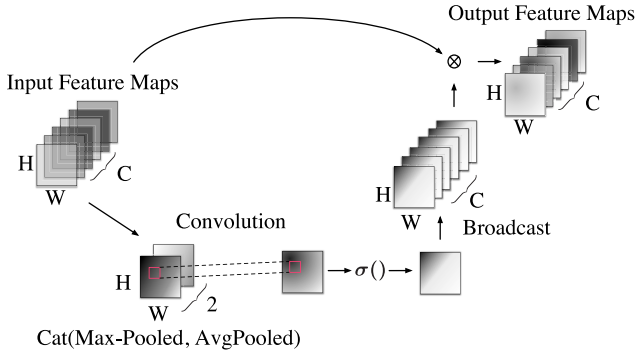


Fig. 5. Spatial-wise attention module.

these convolutional kernels perform convolution operations with different kernel scales on layout patterns, and the outputs are concatenated in the channel dimension. After going through pooling layers, the fused feature maps are scaled down in height and width dimensions. As a result, the feature maps are comprehensive and rich. The backbone network based on inception structure is illustrated in Fig. 3.

C. Backbone: Attention Block

Recently presented attention mechanism which mimics human perception has achieved much success in the computer vision domain [32]–[34]. With attention techniques, neural networks focus on the salient parts of input features and generate attention-aware feature maps. In order to capture structures of feature maps better, we exploit this mechanism and embed the corresponding modules into our backbone network.

As we know, the features include both cross-channel and spatial information on the back of convolution computations. Based on that fact, the embedded attention modules can emphasize informative parts and suppress unimportant ones along the channel and spatial axes. Channel-wise attention focuses on the informative part itself, while spatial attention concentrates on its spatial location. They are complementary to each other. To fit the motivation, the structure of one attention module consists of two subparts: one is channel-wise, and the other is spatial-wise. For a better understanding, we visualize an attention module and zoom in on its intrastructure in Fig. 3. The whole attention module sequentially infers a 1D channel attention map, and then a 2D spatial attention map. Through the broadcasting operation, each attention map will perform element-wise multiplication with input feature maps.

The whole process of channel-wise attention is concluded in (1) and (2) visualized in Fig. 4. First, given an input feature tensor $\boldsymbol{T}$, spatial information of $\boldsymbol{T}$ is first aggregated by average-pooling and max-pooling operations, and then two different spatial context descriptors are produced. Next, two descriptors go through a shared encoder-decoder network [i.e., a single hidden layer perceptron named ED() in (2)], and output feature vectors are merged using element-wise summation. After activation operation [i.e., defined by $\sigma()$ in (2)], the elements of the channel-wise attention masks $A_c(\boldsymbol{T})$ are first broadcasted along the spatial dimension, then multiplied with corresponding elements in feature maps. Finally, the module outputs the feature tensor $\boldsymbol{T}'$. In a nutshell, the channel-wise attention module infers the channel-wise attention masks, and each mask will be multiplied element-wisely with input feature maps

$$\boldsymbol{T}' = A_c(\boldsymbol{T}) \otimes \boldsymbol{T} \tag{1}$$

$$A_c(\boldsymbol{T}) = \sigma(\mathrm{ED}(\mathrm{AvgPool}(\boldsymbol{T})) + \mathrm{ED}(\mathrm{MaxPool}(\boldsymbol{T}))). \tag{2}$$

We mathematically summarize the computation process for spatial-wise attention module in (3) and (4), and visualize it in Fig. 5. First, through pooling operations, we aggregate channel information of the input feature tensor $\boldsymbol{T}'$, which is also the output of the channel-wise attention module. Then, concatenate [denoted by Cat() in (4)] two aggregated features and perform convolution computation [i.e., Conv()] on the concatenation. After activated by sigmoid function, the spatial attention mask $A_s(\boldsymbol{T}')$ is generated. Finally, the mask will be multiplied with $\boldsymbol{T}'$ element-wisely, which is shown in the following:

$$\boldsymbol{T}'' = A_s(\boldsymbol{T}') \otimes \boldsymbol{T}' \tag{3}$$

$$A_s(\boldsymbol{T}') = \sigma\big(\mathrm{Conv}\big(\mathrm{Cat}\big(\mathrm{AvgPool}(\boldsymbol{T}'), \mathrm{MaxPool}(\boldsymbol{T}')\big)\big)\big). \tag{4}$$

D. Multibranch Design

Our end-to-end framework has two jointly performed tasks: 1) hotspot classification and 2) deep layout metric learning. The two tasks share the backbone network for feature extraction, but are guided by different loss functions. For the deep layout metric learner, the proposed triplet loss is directly calculated on the high dimensional vector which is the output of the fully connected layer in the backbone network. The learner searches for a good feature embedding which nonlinearly maps the image representations of layout patterns (i.e., gray images) into a new space. Therefore, pairs of hotspot patterns and nonhotspot patterns can be effectively measured and separated by the Euclidean distance metric. For hotspot detection, the main task is to find an appropriate boundary that well divides hotspots and nonhotspots. Via backpropagation, the guide information (i.e., gradients) from two branches update the backbone collectively.

IV. Loss Functions and Training for Preliminary Hotspot Detector

A. Metric Learning Loss in Branch I

Metric learning is typically referred to learning a distance metric or pursuing an embedding function to map images onto

a new manifold. Given a similarity metric function, similar images are projected into a neighborhood, while dissimilar images are mapped apart from each other. Note that the term "similar images" means the images share the same label. Over past decades, the machine learning community has witnessed a remarkable growth of metric learning algorithms used in a variety of tasks, such as classification [35], clustering [36], image retrieval [37], etc. However, many metric learning approaches explore only linear correlations, thus may suffer from nonlinearities among samples. Although kernel tricks have been developed, it is resource consuming to find an appropriate one. With a notable success achieved by deep learning, deep metric learning methods have been proposed to find the nonlinear embeddings. By taking advantage of deep neural networks to learn a nonlinear mapping from the original data space to the embedding space, deep metric learning methods measuring Euclidean distance in the embedding space can reflect the actual semantic distance between data points.

Contrastive loss [38] and triplet loss [39] are two conventional measures which are widely utilized in most existing deep metric learning methods. The contrastive loss is designed to separate samples of different classes with a fixed margin and pull closer samples of the same category as near as possible. The triplet loss is more effective and more complicated, which contains the triplets of anchors, positive, and negative samples. Since the triplet loss takes into consideration of higher-order relationships in embedding space and thus can achieve better performance than the contrastive loss. Therefore, in the proposed deep layout metric learning, we adopt the triplet loss.

As aforementioned, the goal of deep metric learning aims at finding a good embedding which is denoted by $f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^d$ with $\boldsymbol{w}$ as the parameter of f in our assumption. The embedding function $f_{\boldsymbol{w}}(\cdot)$ projects the layout pattern $\boldsymbol{x}$ onto a hypersphere located in a d-dimensional compact Euclidean space, where distance directly corresponds to a measure of layout similarity. In other words, the normalization constraint $||f_{\boldsymbol{w}}(\boldsymbol{x})||_2^2 = 1$ is attached to all embeddings. The mechanism behind the triplet loss is based on the following rules.

1) During training, the layout clips constitute several triplet instances, where $f_{\boldsymbol{w}}(\boldsymbol{x}_i)$, $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)$, and $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)$ denote an anchor layout clip, a layout clip sharing the same label with the anchor, and a layout clip which has the opposite label, respectively.
2) Each triplet instance indicates the triplet relationship among three layout clips as

$$S\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)\big) + M < S\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)\big) \quad \forall\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)\big) \in \mathcal{T}. \tag{5}$$

   In (5), S is the similarity measurement metric and can produce the similarity values always satisfying the triplet relationship. M refers to a margin between positive and negative pairs, and $\mathcal{T}$ collects all valid triplets in the training set with $|\mathcal{T}| = n$.
3) The Euclidean distance is employed as the metric to measure the similarity between the embedding pairs in the new feature space.

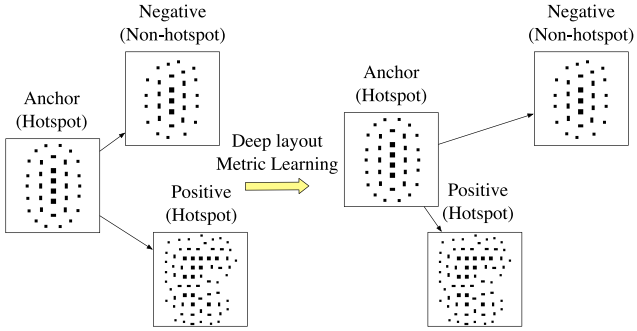


Fig. 6. Visualization of the proposed deep layout metric learning. In the worst case, the anchor is much similar to the negative than to the positive. In other words, in original space, the distance between the anchor and the negative is shorter than the one between the anchor and the positive. But after deep layout metric learning, in a new manifold, the two hotspot layout clips are kept apart from the nonhotspot clip.

To explore the relationship among triplet layout clips, the objective function of deep layout metric learning can be formulated as a loss function $\mathcal{L}_{\text{metric}}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$, which is based on the hinge loss [displayed in (6a)] with Constraint (6b). $\mathcal{L}_{\text{metric}}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$ is also called empirical loss. Minimizing the empirical loss is equivalent to minimizing the violations on the relationship defined in (5)

$$\min_{\boldsymbol{w}} \ \frac{1}{n}\sum_{i=1}^{n} \max\Big(0, M + ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big)||_2^2 - \big\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big\|_2^2\Big) \tag{6a}$$

$$\text{s.t.} \ \ \|f_{\boldsymbol{w}}(\boldsymbol{x}_i)\|_2^2 = 1 \quad \forall\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big) \in \mathcal{T}. \tag{6b}$$

The illustration for the proposed loss is shown in Fig. 6. It can be seen that after training, the layout clips of the same category will be kept apart from the one which is from the other class. The proposed layout triplet loss attempts to enforce a margin between each pair of layout clips from hotspot to nonhotspot. To prove the property, we calculate the gradients of $\mathcal{L}_{\text{metric}}$ with respect to the embedding vectors by the following:

$$\frac{\partial \mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big)} = \frac{2}{n}\big(f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big) - f_{\boldsymbol{w}}(\boldsymbol{x}_i)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big) \tag{7a}$$

$$\frac{\partial \mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)} = \frac{2}{n}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big) \tag{7b}$$

$$\frac{\partial \mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i)} = \frac{2}{n}\big(f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big) - f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{metric}}\big(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^+\big), f_{\boldsymbol{w}}\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big) \tag{7c}$$

where $\mathbf{1}$ is the indicator function which is defined as

$$\mathbf{1}(x) = \begin{cases} 1, & \text{if } x \text{ is true} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

When the distance constraint is satisfied, i.e., there are no violations on the triplet relationship, gradients become zeros. As a result, triplet loss $\mathcal{L}_{\text{metric}}$ allows the layouts from one category to live on a manifold, while still keep the distance and hence discriminative to another category.

We further analyze the effectiveness of triplet loss by offering its bias between the generalization error and the empirical error. In other words, we evaluate the upper bound of the network representation ability in a more mathematical way. In the following descriptions, for a better explanation, we use $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ expressing a valid triplet. Before showing the bias, we readily extend the pair-based definitions in [40] to the triplet scenarios, as in Lemma 1.

Lemma 1: A triplet-based metric learning algorithm has β-uniform stability ($\beta \geq 0$), if

$$\sup_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}} \left| \ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\right) \right| \leq \beta \quad \forall \mathcal{T}, \mathcal{T}_i \tag{9}$$

where ℓ indicates a loss function, $\mathcal{T}_i$ is the training set $\mathcal{T}$ with sample $\boldsymbol{x}_i$ replaced by an independent and identically distributed exemplar $\boldsymbol{x}_i'$, and $\mathcal{D}$ is some kind of distribution. $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ and $f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)$ are mapping functions learned over the training set $\mathcal{T}$ and $\mathcal{T}_i$, respectively.

Theorem 1: Assume ℓ be a loss function upper bounded by $\mathcal{B} \geq 0$, and let $\mathcal{T}$ be a training set consisting of n valid triplets drawn from distribution $\mathcal{D}$, and $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ the mapping function parameterized by $\boldsymbol{w}$ which is learned over the training set $\mathcal{T}$ by a β-uniformly stable deep metric learner. The empirical loss over $\mathcal{T}$ is $\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))$, while the expected loss of learned mapping function $f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)$ over distribution $\mathcal{D}$ is $\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)]$. Then, for $0 < \delta < 1$, with confidence 1-δ approaching to 1, the following inequality exists:

$$\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k \sim \mathcal{D}}\left[\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot), \boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k\right)\right] - \mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\right) \leq 3\beta + (2n\beta + 3\mathcal{B})\sqrt{\frac{\log \frac{1}{\delta}}{2n}}. \tag{10}$$

The proof of Theorem 1 is detailed in the Appendix. It can be observed that with the increasing of the number of training triplets, the bias converges. Our analyses for generalized representation ability of triplet-based deep metric learning demonstrate the performance of the proposed layout metric learner.

B. Classification Loss in Branch II

Except for the same backbone network, there are two fully connected layers in our classification branch. Different from those hotspot detectors in previous deep learning-based work, the proposed classifier predicts labels based on the features learned by the backbone network and deep layout metric learner. Like prior art [17], the loss function for classification, defined by $\mathcal{L}_{\text{prediction}}$, is cross-entropy loss

$$-\left(y \log\left(y^*\right) + (1-y)\log\left(1-y^*\right)\right) \tag{11}$$

where y^* is the predicted probability of a layout clip, while y is the relevant ground truth (binary indicator).

C. Training Strategies

Hotspot detection is haunted with a crucial issue that relative datasets (e.g., ICCAD12 benchmark suite [26]), no matter in academia or industry, are quite unbalanced. That is, the number of hotspots is much less than that of nonhotspots. This property results in a biased classification toward the nonhotspots. Additionally, limited by the bottleneck of hardware, it is infeasible to compute the arg min or arg max across the whole training set. Hence, sampling matters in our framework.

What we do is to generate triplets from a balanced mini-batch in an online fashion. This can be done by constructing a balanced mini-batch (i.e., the numbers of hotspots and nonhotspots are equal) in one iteration and then selecting the hard negative exemplars from within the batch. Here, we divide the negative samples based on a simple rule that easy negatives will lead the loss to become zero, whilst the hard negatives make the loss valid. Since only picking the hardest negatives leads to bad local minima early during training, we keep all anchor-positive pairs in a mini-batch while selectivity sample some hard negatives. Besides hardest negatives, we also consider some negative exemplars which are further away from the anchor than the positive samples but still hard. Those so-called "semi-hard" negatives which lie inside the margin obey the following inequality:

$$\left\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\left(\boldsymbol{x}_i^+\right)\right\|_2^2 \leq \left\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\left(\boldsymbol{x}_i^-\right)\right\|_2^2 \leq M + \left\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\left(\boldsymbol{x}_i^+\right)\right\|_2^2. \tag{12}$$

Aiming at progressively selecting false-positive samples that will benefit training, this kind of sampling strategy is widely used in deep metric learning methodologies. Many visual tasks [41]–[45] have proven its effectiveness. One reason is that it reduces the number of layout tuples that can be formed for training, and thus enhances the training efficiency.

V. Enhanced Hotspot Detector

Until now, we have designed an end-to-end, attention mechanism-based hotspot detector trained with a composite loss containing a triplet layout pattern loss and a classification loss. Nevertheless, there still exists some room to improve the performance. For instance, the predefined margin in the triplet loss function may result in under- and over-fitting problem and thus affects the discrimination of layout embeddings. Therefore, based on the preliminary design, three new concepts and techniques are introduced for further enhancement.

A. Deformable Convolution Operator in Backbone Network

As aforementioned, in hotspot detection, the hotspots may exist in the center of a layout. Due to the light propagation, diffraction, and interference of the lithography process, it also impacts the metals (polygons, vias) all around. The vanilla

convolution operator is not able to fully capture the influences caused by the propagation of diffracted light from a mask pattern. Besides, enlarging the receptive field to extract the latent information like via-via distance, metal-metal distance is also in demand. Hence, we introduce the deformable convolution operation [46] into the inception block of our backbone, which adds 2-D offsets to the regular grid sampling locations in the standard convolution. It enables free-form deformation of the sampling grid. By introducing additional convolutional layers, the offsets are learned from the preceding feature maps. The deformable convolution operator is locally, densely, and adaptively perceiving the possible hotspots with an affected, nonregular region in input layout features. With the deformable convolution operator, our detector have an effective receptive field when perceiving input layout features.

This convolution in 2-D space consists of two steps. The first is sampling using a regular grid map $\mathcal{G}$ over the input feature map $\boldsymbol{x}_{\text{input}}$, and the second is the summation of sampled values weighted $\boldsymbol{w}$. The grid $\mathcal{G}$ defines the receptive field size and dilation. For example, given a convolutional kernel of $K = 9$ sampling locations and $|\mathcal{G}| = K = 9$, $\mathcal{G} = \{(-1,-1),(-1,0),\ldots,(1,1)\}$ defines a 3×3 convolutional kernel of dilation 1. Let $\boldsymbol{l}_0$ denote a position on the output feature map $\boldsymbol{x}_{\text{output}}$. The deformable convolution can then be expressed as

$$\boldsymbol{x}_{\text{output}}(\boldsymbol{l}_0) = \sum_{\boldsymbol{l}_k \in \mathcal{G}} \boldsymbol{w}(\boldsymbol{l}_k) \cdot \boldsymbol{x}_{\text{input}}(\boldsymbol{l}_0 + \boldsymbol{l}_k) \tag{13}$$

where $\boldsymbol{l}_k$ lists all feasible locations in $\mathcal{G}$. In deformable convolution, the regular grid $\mathcal{G}$ is augmented with offsets $\{\Delta\boldsymbol{l}_k \mid k = 1,\ldots,K\}$ in which $K = |\mathcal{G}|$. Now, (13) can be written as following:

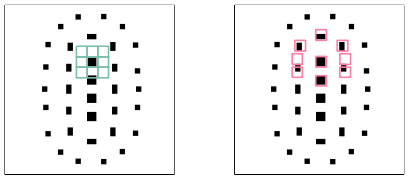
$$\boldsymbol{x}_{\text{output}}(\boldsymbol{l}_0) = \sum_{k=1}^{K} \boldsymbol{w}(\boldsymbol{l}_k) \cdot \boldsymbol{x}_{\text{input}}(\boldsymbol{l}_0 + \boldsymbol{l}_k + \Delta\boldsymbol{l}_k). \tag{14}$$

It can be observed that the sampling is on the irregular and offset locations $\boldsymbol{l}_k + \Delta\boldsymbol{l}_k$. Since the offset $\Delta\boldsymbol{l}_k$ is usually fractional, in practical implementation, bilinear interpolation is utilized to attain (14)

$$\boldsymbol{x}_{\text{input}}(\boldsymbol{l}) = \sum_{m} G(\boldsymbol{m}, \boldsymbol{l}) \cdot \boldsymbol{x}_{\text{input}}(\boldsymbol{l}). \tag{15}$$

In (15), $\boldsymbol{l}$ equalling to $\boldsymbol{l}_0 + \boldsymbol{l}_k + \Delta\boldsymbol{l}_k$ still refers to a fractional location, and $\boldsymbol{m}$ indicates all possible integral locations in the feature map $\boldsymbol{x}_{\text{input}}$. $G(\cdot,\cdot)$ is a 2-D bilinear interpolation kernel which can be separated into two 1-D kernels as: $G(\boldsymbol{m}, \boldsymbol{l}) = g(m_x, l_x) \cdot g(m_y, l_y)$ where $g(u, v) = \max(0, 1 - |u - v|)$.

The offsets Δl_k is obtained via a separate convolution layer applied over the same input feature maps $\boldsymbol{x}$. This convolutional layer is of the same spatial resolution and dilation as the current convolutional layer. The output offset fields have the same spatial resolution as the input feature map. For example, assuming a convolutional kernel of K sampling locations, the output is of $2K$ channels corresponding to the learned offsets $\{\Delta l_k\}_{k=1}^{K}$. During training, the kernel weights in this separate convolution layer are initialized to zero. Thus, the initial values of $\{\Delta l_k\}_{k=1}^{K}$ is zero. The learning rates of the added convolutional layers for offset and modulation learning are set to 0.1 times those of the existing layers. Both the convolutional kernels for generating the output features and the offsets are learned simultaneously. To learn the offsets, the gradients are back-propagated via (15). For a better understanding of the deformable convolution operation, we visualize it with a standard convolution operation in Fig. 7.



(a) (b)

Fig. 7. Visualization of standard and deformable convolutions with a 3×3 kernel size. (a) Vanilla convolution. (b) Deformable convolution. Each samples 9 pixels in the layout feature map. The deformable convolution has a larger receptive field size and thus can capture more information about neighbor vias and the via distances.

B. Self-Adaptive Margin-Based Triplet Loss

In the proposed preliminary hotspot detector, we have designed a triplet loss exploring the similarities among trilayouts and learning the good feature embedding. But the loss (6) imposes a constant margin M. The margin parameter is manually predefined based on the domain knowledge, which leads to bias and inflexibility. The intrinsic reason is twofold. In our online semi-hard negative mining, a small threshold value would produce few semi-hard samples. Consequently, the model would converge slowly and be stuck in suboptimality (usually hard and semi-hard samples contribute to the loss). In contrast, a large threshold would result in too many semi-hard training samples to make the model be overfitted. Since it impacts the performance of the loss, redesigning a dedicated, and more flexible triplet loss is crucial.

The essence of margin is a threshold boundary separating anchor-positive and anchor-negative pairs. Inspired by the isotonic regression that estimating a margin separately and then penalizing scores relative to the margin, we propose our self-adaptive margin-based loss. The main idea behind it is the avoidance of the over- or under-sampling problems aforementioned with the assumption that the distances (or similarities) between layout features of the positive pairs or negative pairs are samples from two distinct distributions, i.e., the positive pair distance distribution and the negative pair distance distribution. The self-adaptive margin threshold is substantially used to express the average distance of the two distributions, which should have a positive relationship with the average distance. So we exploit the average distance of two distinct distributions to self-adaptively represent our margin thresholds. The margin threshold is self-adaptive based on the two distributions of the trained model.

We first write the equation for adaptive margin M_a

$$M_a = c\left(\mu^- - \mu^+\right) = c\left(\frac{1}{N^-}\sum_{i}^{n}\left\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\left(\boldsymbol{x}_i^-\right)\right\|_2^2 - \frac{1}{N^+}\sum_{i}^{n}\left\|f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}\left(\boldsymbol{x}_i^+\right)\right\|_2^2\right) \tag{16}$$

where with N^+ and N^- the numbers of positive and negative pairs, respectively, μ^+ and μ^- are mean values of two pair distance distributions, and c is the correlation constant (1 in the experiment). Note that in the implementation, for efficient computing, we only compute the mean of two distributions in each batch. After substituting the constant M with the proposed adaptive alternative M_a, the loss function denoted as $\mathcal{L}_{\text{a-metric}}$ is showed in the following:

$$\min_{w} \ \frac{1}{n}\sum_{i=1}^{n}\max\Big(0, \mu^- - \mu^+ + \big\|f_w(\boldsymbol{x}_i) - f_w\big(\boldsymbol{x}_i{}^+\big)\big\|_2^2 - \big\|f_w(\boldsymbol{x}_i) - f_w\big(\boldsymbol{x}_i{}^-\big)\big\|_2^2\Big) \tag{17a}$$

$$\text{s.t.} \ \ \|f_w(\boldsymbol{x}_i)\|_2^2 = 1 \quad \forall\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big) \in \mathcal{T}. \tag{17b}$$

Assume that the balanced batch size is n, and N^+ and N^- are set to be $(n/2)$ and $(n/2)$, respectively. The gradients of (17) are displayed in (18). The original loss function (6) has a property that if there is no violation on the triplet relationship, associated gradients tend to be zeros. It can bee seen that in (18a) and (18c), this property still holds.

C. Self-Contrastive Layout Learning for Hotspot Detection

Learning effective layout pattern representations is a still key challenge in DFM-driven applications as it allows for efficient training on downstream tasks. In preliminary framework, we have explored the relationships among different layouts in a triplet structure. However, the correlations among the layout and its own augmented views (e.g., flipped and 180-deg-rotated versions of the layout) are not considered

$$\frac{\partial\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_w\big(\boldsymbol{x}_i{}^+\big)} = \frac{2n+4}{n^2}\big(f_w\big(\boldsymbol{x}_i{}^+\big) - f_w(\boldsymbol{x}_i)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big) \tag{18a}$$

$$\frac{\partial\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_w\big(\boldsymbol{x}_i{}^-\big)} = \frac{2n-4}{n^2}\big(f_w(\boldsymbol{x}_i) - f_w\big(\boldsymbol{x}_i{}^-\big)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big) \tag{18b}$$

$$\frac{\partial\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big)}{\partial f_w(\boldsymbol{x}_i)} = \frac{2n-4}{n^2}\big(f_w\big(\boldsymbol{x}_i{}^-\big) - f_w\big(\boldsymbol{x}_i{}^+\big)\big) \cdot \mathbf{1}\big(\mathcal{L}_{\text{a-metric}}\big(f_w(\boldsymbol{x}_i), f_w\big(\boldsymbol{x}_i{}^+\big), f_w\big(\boldsymbol{x}_i{}^-\big)\big) > 0\big). \tag{18c}$$

In the computer vision domain, many different training approaches have been proposed to learn effective representations, usually relying on visual pretext tasks. Among them, discriminative approaches [47], [48] based on contrastive learning in the latent space have recently shown great potential, achieving state-of-the-art results. State-of-the-art contrastive methods are trained by reducing the distance between representations of different augmented views of the same image. This kind of technique learns representations using objective functions similar to those used for supervised learning, but train networks to perform pretext tasks in an unsupervised manner. It is extremely useful in hotspot detection. Because the problem that the hotspot clip number is much less than the nonhotspot clip number always exists. Compared with the previous works like [17] which performs data augmentation only to enrich the diversity of the dataset, applying the aforementioned self-contrastive learning methods in hotspot detection would have a bigger promise.

Inspired by the notable success of recently proposed contrastive learning algorithms, we enhance our detector network by redesigning the original architecture. The new architecture is expected to learn representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. Three components constitute the architecture. First is the data augmentation module which transforms any given layout clip resulting in two correlated views of the same layout clip. Assume that the source of the lithography system is up-down and left-right symmetric. For hotspot detection, we can apply the following simple augmentations: 180-deg rotation, top-bottom, and left-right flipping. It is worth mentioning that in hotspot detection, we have to thoroughly select augmentation operations owing to the symmetry of the system. For example, for a dipole illumination, a hotspot layout pattern may become nonhotpot if it is rotated by 90 degrees. This is one of the substantial differences between the deep learning-based hotspot detection and typical classification in the computer vision field. These augmentations would not change the label category. The second part is the inception structure- and attention module-based CNN with its replication as its counterpart. They extract representation vectors from an original layout and its augmented layout pattern. We use this new backbone to obtain the embeddings in the latent space. The two kinds of representations are projected into a space where self-contrastive loss is applied. The last one is a contrastive loss function defined as $\mathcal{L}_{sc}$, where "sc" stands for "self-contrastive." We write the loss function in

$$\mathcal{L}_{sc} \triangleq \sum_i (1 - \boldsymbol{C}_{ii})^2 + \lambda \sum_i \sum_{j\neq j} \boldsymbol{C}_{ij}^2 \tag{19}$$

where λ is a positive constant trading off the importance of the first and second terms of the loss, and where $\boldsymbol{C}$ is defined as the cross-correlation matrix which computes the correlations between the outputs of the two identical networks along the batch dimension. Assuming the latent embeddings of two input views are e^o and e^a, the element $\boldsymbol{C}_{ij}$ in $\boldsymbol{C}$ is computed via

$$\boldsymbol{C}_{ij} = \frac{\sum_b e^o_{b,i} z^a_{b,j}}{\sqrt{\sum_b \Big(e^o_{b,i}\Big)^2}\sqrt{\sum_b \Big(e^a_{b,j}\Big)^2}}. \tag{20}$$

More specifically, the self-contrastive learning flow first produces augmented views for all layout clips of a batch sampled from a layout clip dataset. Then, the batches of augmented views and original views are fed into the proposed backbone, producing batches of latent layout embeddings of two views, respectively. Ultimately, the self-contrastive learning loss is computed and back-propagated.

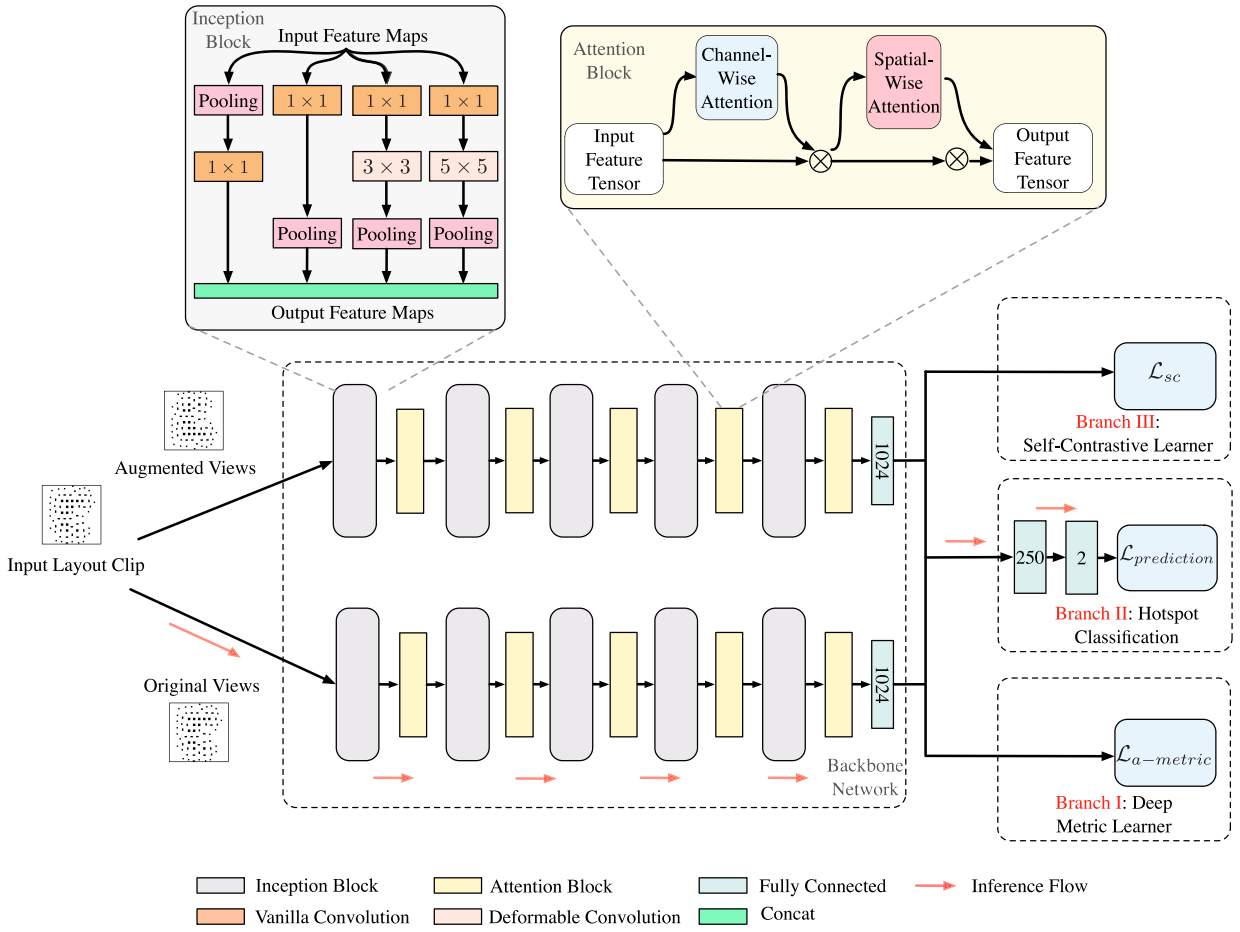


Fig. 8. Architecture of the enhanced hotspot detector.

D. Overall Architecture of the Enhanced Hotspot Detection Flow

The illustration of the learning diagram of the proposed enhanced hotspot detector is in Fig. 8. The learning flow of the enhanced detector still works in an end-to-end, multi-branch manner. We replace some vanilla convolution operators with the deformable convolutions in inception blocks, while the attention modules keep the same as those in the preliminary hotspot detector. It can be seen that the backbone looks like the siamese network structure since the learning flow of our detector includes the diagram of self-contrastive learning on different augmented views of the same layout inputs. The loss $\mathcal{L}_{sc}$ (19) maximizes the agreement between differently augmented views of the same layout input. Besides, the self-adaptive margin-based triplet loss $\mathcal{L}_{\text{a-metric}}$ [namely, (17)] substitutes for the original $\mathcal{L}_{\text{metric}}$ [i.e., (6)] with a constant margin. On the other hand, we still keep the loss function defined by $\mathcal{L}_{\text{prediction}}$ in (11) for classification. During training, the total loss which is equivalent to the summation of $\mathcal{L}_{sc}$, $\mathcal{L}_{\text{a-metric}}$, and $\mathcal{L}_{\text{prediction}}$ back-propagates the gradients to update the whole network. To avoid confusion, we use the red arrows marking the inference flow of the framework, in other words, the route to detect the layout hotspots.

TABLE I
BENCHMARK STATISTICS

Benchmarks	Training Set		Testing Set		Size/Clip
	HS#	NHS#	HS#	NHS#	(μm^2)
ICCAD12	1204	17096	2524	13503	3.6 × 3.6
Via-1	3430	10290	2267	6878	2.0 × 2.0
Via-2	1029	11319	724	7489	2.0 × 2.0
Via-3	614	19034	432	12614	2.0 × 2.0
Via-4	39	23010	26	15313	2.0 × 2.0
Via-Merge	5112	63653	3449	42294	2.0 × 2.0

VI. EXPERIMENTAL RESULTS

The implementation of our framework is in `Python` with the `TensorFlow` library [49], and we test it on a platform with the Xeon Silver 4114 CPU processor and Nvidia TITAN Xp Graphic card. To verify the effectiveness and efficiency of our detector, two benchmarks are employed. One is ICCAD12 benchmarks [26], and the other is a more challenging via layer benchmark suite which is under 45 nm technology node. For fair comparisons against previous works, following these arts, all 28 nm designs in ICCAD12 benchmarks [26] are merged into a unified case named `ICCAD12`. The details for `ICCAD12` and the via benchmarks are listed in Table I.

TABLE II
COMPARISON WITH STATE OF THE ARTS

Bench	TCAD'19 [17]			DAC'19 [20]			ASPDAC'19 [19]			JM3'19 [18]			ADM-HSD [23]			eADM-HSD		
	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)
ICCAD12	98.40	3535	502.70	98.54	3260	561.28	97.70	2883	433.27	97.82	2651	505.67	98.42	2481	143.79	98.77	2510	204.16
Via-1	71.50	773	43.36	89.85	1886	57.76	74.98	1805	47.17	89.19	2624	47.87	93.42	1589	19.83	94.35	1492	29.80
Via-2	65.06	1290	40.02	73.00	1222	21.66	54.28	1797	40.89	38.81	454	43.06	86.32	1100	13.22	87.70	1045	20.11
Via-3	48.15	760	60.23	73.38	3406	43.15	35.88	819	70.65	21.06	42	67.13	88.20	2105	20.69	90.28	2072	30.64
Via-4	76.92	155	67.44	73.08	15288	51.98	65.38	100	80.81	46.15	21	77.15	80.77	152	20.70	84.61	157	26.71
Via-Merge	88.01	7633	165.85	90.42	9295	105.30	76.66	7268	220.15	84.34	6759	170.91	92.20	6453	59.74	92.81	5526	83.50
Average	74.67	2357.67	146.60	83.06	5726.17	140.19	67.48	2445.33	148.82	62.90	2091.83	151.97	89.89	2313.33	46.33	91.42	2133.68	65.59
Ratio	0.83	1.02	3.16	0.92	2.48	3.03	0.75	1.06	3.21	0.70	**0.90**	3.28	1.00	1.00	**1.00**	**1.02**	0.92	1.35

Columns "Train HS#" and "Train NHS#" list the total number of hotspots and the total number of nonhotspots in the training set, whilst columns "Test HS#" and "Test NHS#" refer to the total number of hotspots and the total number of nonhotspots in the testing set. "Size/Clip (μm^2)" shows the resolution for each benchmark. It is manifest that the via benchmark suite has four individual cases, which are arranged in order of the density of target designs. For example, Via-1 has the highest density of target designs in its layout clips among other cases. As a result, it contains the most hotspot patterns. In a low-density case, like Via-4, the number of hotspot clips are reduced accordingly. We also merge all four small cases into a big one named "Via-Merge." Note that, as listed in Table I, images in the testing set of ICCAD12 have a resolution of 3600×3600 which is larger than the images in the via benchmarks of 2048×2048, and in all benchmarks, the pixel size is 1 nm.

Mask images in our employed benchmarks are from distinct stages of optical proximity correction (OPC), and have different targets. For the ICCAD12 benchmark, layout patterns are before OPC, while regarding the rest via layer benchmarks, they are from intermediate OPC results (i.e., inserting SRAFs without OPC). We expect to mimic two scenarios. One is to target at finding and revising problematic designs at an early stage of the whole layout verification flow which is associated with ICCAD12 benchmark. This benchmark is labeled according to the results of the entire layout verification flow containing OPC and lithography simulation. The other is to demonstrate the potential of incorporating the hotspot detectors into OPC engines and facilitate the procedure, which corresponds to the via layer benchmarks (each layout has model-based SRAFs and un-OPCed vias). The labeling on hotspot or nonhotspot patterns in via layer benchmarks is based on the lithography simulation results of the current OPC step.

Table II summarizes the comparing results between the proposed frameworks and several state-of-the-art hotspot detectors. Column "Bench" lists six benchmarks used in our experiments. Columns "Accu," "FA," and "Time" are hotspot detection accuracy, false alarm count, and detection runtime, respectively. Columns "TCAD'19 [17]," "DAC'19 [20]," "ASPDAC'19 [19]," and "JM3'19 [18]" denote the results of selected baseline frameworks, respectively. The rest two columns, "ADM-HSD" and "eADM-HSD," stand for preliminary hotspot detection framework proposed in [23] and the enhanced detection flow presented in this work. We can see that ADM-HSD outperforms TCAD'19 averagely with 15.22% improvement on detection accuracy and 2% less false alarm penalty, while the enhanced flow, eADM-HSD, surpass ADM-HSD with an average accuracy of 91.42% and further decrease on the false alarm. Besides, ADM-HSD and eADM-HSD behaves much better on average detection accuracy compared to 74.67%, 83.06%, 67.48%, and 62.9% for TCAD'19, DAC'19, ASPDAC'19, and JM3'19, respectively. Moreover, the advantage of the proposed two one-stage multi-branch flows can also be noticed that they achieve almost $3\times$ speedup compared to previous two-stage flows.

Note that DAC'19 exhibits a slightly better accuracy on ICCAD12 case, it suffers from high false alarm penalties over almost all cases due to the nature of the binarized neural network. It is worth mentioning that to compare the detection performance in an easier fashion (i.e., evaluating the accuracy with a comparatively close false alarm to the one of [23]), we adjust the hotspot/nonhotspot threshold in the biased learning strategy exploited in ASPDAC'19. As a result, both of accuracy and false alarm of ASPDAC'19 move upward comparing with the results presented in our previous work [23]. Besides, it is crystal lucid that in Table II, the accuracy of most works like TCAD'19, DAC'19, and ASPDAC'19 on Via-Merge is higher than that on other via benchmarks. What's more, excluding Via-1 case, our proposed works also behave similarly. One main factor for this phenomenon is that the number of the hotspot and nonhotspot layout patterns in Via-Merge are much larger than the others. Meanwhile, the diversity of Via-Merge is also far more abundant than others. To put it from another angle, the models have more chance to be calibrated better with the training dataset of the Via-Merge benchmark. When it turns to our framework, as shown by Theorem 1 with the proof in the Appendix, with the increasing of the number of training triplets, the generalized representation ability of triplet-based deep metric learning have high probability to get improved.

An ablation study is performed on the preliminary flow [23] to investigate how different configurations affect the performance. Fig. 9 illustrates the contribution of attention module, inception block, and layout metric learning loss to our flow. "w/o. Atten" refers to the detector without attention modules, "w/o. Incept" stands for the detector with inception blocks replaced by vanilla convolutional layers, "w/o. Metric" denotes the detector trained without layout metric learning loss, whilst "Full" is our detector with entire techniques. The histogram shows that with attention modules, 1.29% accuracy

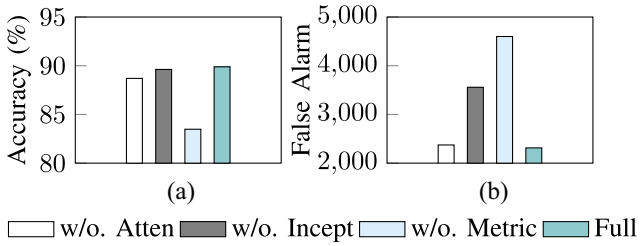


Fig. 9. Comparisons among different configurations on (a) average accuracy and (b) average false alarm of the preliminary hotspot detector (ADM-HSD) [23].

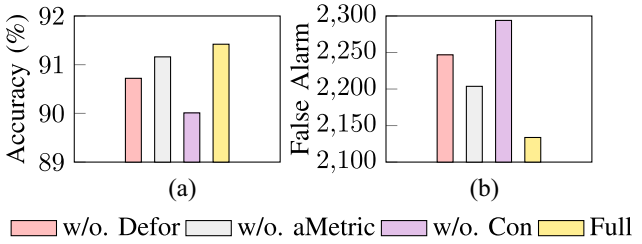


Fig. 10. Comparisons among new different settings on (a) average accuracy and (b) average false alarm of the enhanced hotspot detector.

improvement without additional false alarms on average is achieved, which confirms that the attention modules help the backbone network extract feature more efficiently. With inception blocks, we get a notable reduction on false alarm penalties, i.e., 1245 less on average, which means under the same experiment settings, the inception blocks capture richer information on layout patterns than vanilla convolutional layers. Comparing the whole framework with the model trained without layout metric learning, the model trained with the layout metric learning loss reduces 49.72% of the false alarm and get 6.41% further improvement on accuracy.

As the above ablation study on the preliminary flow [23], we explore the impact of each proposed technique on the enhanced detection flow as well. Fig. 10 records the contributions of deformable convolution operator in inception block, the self-adaptive margin in triplet layout metric learning loss, and the self-contrastive layout learning loss to our flow. "w/o. Defor" means the detector without deformable convolution, and "w/o. aMetric" represents the detector trained with self-adaptive margin degrading to a constant margin (e.g., 0.5), and "w/o. Con" corresponds to the detector trained without self-contrastive layout learning loss, while "Full" implies the enhanced hotspot detector with all proposed techniques. It can be seen that without applying deformable convolution operation, the average accuracy drops to 90.72% with about 5% additional false alarms on average. We can infer that the deformable convolution operation benefit to the detection flow by extracting more layout information in a larger receptive filed. The bar graphs in Fig. 10 also illustrate that with a self-adaptive margin, the detection flow performs better on both two metrics. This observation demonstrates that the self-adaptive margin can avoid over- and under-fitting to some extents. Furthermore, comparing with the detector trained without self-contrastive layout learning, the detector

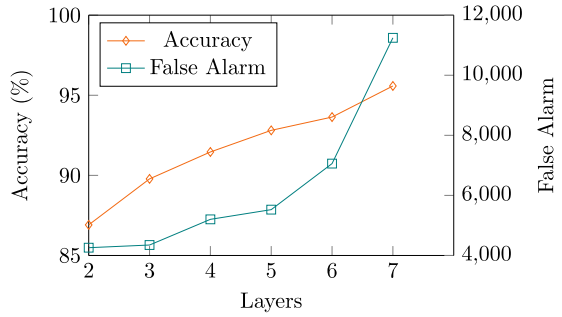


Fig. 11. Experiment to determine the number of layers.

trained with self-contrastive layout learning loss achieves a 1.41% increase on accuracy and approximately a 7% reduction on false alarm.

We also conduct an experiment to explore the impact on detection performance brought about by the number of layers. Notice that for simplicity, we treat an inception module with one attention module as a "layer," and the `Via-Merge` benchmark is exploited in the experiment. The curves in Fig. 11 suggest that as an overall trend, accuracy and false alarm rise simultaneously regarding the increase of the number of layers. The accuracy uplifts gradually. By contrast, the upward trend of false alarm becomes more rapid after the number of layers going up to 5. Namely, the classification performance degrades. A similar performance degradation problem has been exposed and verified in [50]. The results of our experiment demonstrate that configuring the number of layers to be 5 is an optimal solution to tradeoff between the accuracy and the false alarm.

In spite of the above experimental results and analyses, we would like to extend the discussion to the scalability concern. The formulation of our method is the classification on layout clips. The main factor is that directly inserting the larger full-chip design into a deep learning-based detector may cause memory issue. Besides, the inappropriate downscaling of the layout image may change the ratio of width of wire-length and spacing, which impacts the classification performance. Instead, we exploit layout clips which are small layout "snippets" contained within a square-shaped region. Generally, they are clipped from the original layout design via a sliding window. In other words, the positions of the layout clips in the full-chip design are known. In addition, the total area of the clips is in linear proportion to the area of the original full-chip design. As a consequence, our inference time will linearly rise respecting to the increase of the area of the full-chip design. In some extent, classifying the layout clip is equivalent to the detecting and locating the hotspots in the full-chip design.

VII. Conclusion

In this article, for the first time, we have proposed a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are simultaneously performed. Inception block with deformable convolution operation is exploited in the proposed enhanced hotspot detector to enlarge the receptive filed. We further exploit attention techniques to

make backbone network self-adaptively emphasize on more informative patterns in layout clips. The deep layout metric learning mechanism with a self-adaptive margin offers a new solution to extract features from via layer patterns that contain much less discriminative information than metal layer patterns. Apart from considering the triplet relationship among different layouts, we design a self-contrastive layout learning paradigm to explore the correlations among the layout and its own augmented views. Additionally, to test the true performance of hotspot detectors, a new via layer benchmark suite has been used for comprehensive verification. The experimental results demonstrate the superiority of our framework over current deep learning-based detectors. The corresponding mathematical analyses are provided as the theoretical pillars. With the transistor size shrinking rapidly and the layouts becoming increasingly complicated, we hope to apply our ideas into more general VLSI layout feature learning.

APPENDIX

Proof: The proof follows from [51] and [52]. $F_{\mathcal{T}}$ is defined as a replacement to $\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]-\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))$. In (21a) and (21b), we exploit the triangle inequality. Then, the upper bound of (21b) is attained by using Jensen's inequality and the definition of $\mathcal{L}$. With the combination of the triangle inequality, β-uniform stability, and $\mathcal{B}$-boundedness, the further bound is found in (21d)

$$\begin{aligned}\left|F_{\mathcal{T}}-F_{\mathcal{T}_i}\right| &= \Big|\mathcal{L}\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)\Big)-\mathcal{L}\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\big) \\ &\quad+\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\big[\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\big)\big] \\ &\quad-\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\Big[\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big)\Big]\Big| \quad (21a)\\ &\le \Big|\mathcal{L}\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\big)-\mathcal{L}\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)\Big)\Big| \\ &\quad+\big|\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\big[\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\big)\big] \\ &\quad-\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\Big[\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big)\Big| \quad (21b)\\ &\le \frac{1}{|\mathcal{T}|}\Bigg|\sum_{j\neq i,k\neq i,l\neq i}\Big(\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l\big)\,x \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l\Big)\Big) \\ &\quad+\sum_{j\neq i,k\neq i}\Big(\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i\big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i'\Big)\Big) \\ &\quad+\sum_{j\neq i,l\neq i}\Big(\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i,\boldsymbol{x}_l\big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i',\boldsymbol{x}_l\Big)\Big) \\ &\quad+\sum_{k\neq i,l\neq i}\Big(\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_k,\boldsymbol{x}_l\big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i',\boldsymbol{x}_k,\boldsymbol{x}_l\Big)\Big)\Bigg|+\beta \quad (21c)\\ &\le \beta+\frac{3\mathcal{B}}{n}+\beta\le 2\beta+\frac{3\mathcal{B}}{n}. \quad (21d)\end{aligned}$$

Based on the upper-bound obtained in (21), we utilize McDiarmid's inequality [53] to obtain the following:

$$\Pr[F_{\mathcal{T}}\ge\epsilon+\mathbb{E}[F_{\mathcal{T}}]]\le\exp\left(\frac{-2n\epsilon^2}{(2n\beta+3B)^2}\right). \quad (22)$$

With δ set to be $\exp([-2n\epsilon^2]/[(2n\beta+3\mathcal{B})^2])$, ϵ equals to $(2n\beta+3\mathcal{B})\sqrt{([\log(1/\delta)]/2n)}$. Hence, with confidence 1-δ, (23) exists

$$F_{\mathcal{T}}\le\mathbb{E}[F_{\mathcal{T}}]+(2n\beta+3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \quad (23)$$

For an effective bound, $\beta=o(1/\sqrt{n})$. Assume $\beta=\mathcal{O}(n^p)$. Since $\lim_{n\to+\infty}(-n/[(2n\beta+3\mathcal{B})^2])=-\infty$, $1>2*(1+p)$ holds and $\beta=o(1/\sqrt{n})$.

Equation (24) shows the searching computing of the upper bound of $\mathbb{E}[F_{\mathcal{T}}]$. Note that from (24a) and (24b), we harness a fact that replacing the examples with i.i.d exemplars does not change the expected computation. More specifically, $\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))]=\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot))]$

$$\mathbb{E}[F_{\mathcal{T}}]=\mathbb{E}\big[\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\big[\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\big)\big]-\mathcal{L}\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\big)\big] \quad (24a)$$

$$\begin{aligned}&\le\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}} \\ &\quad\times\Bigg[\Bigg|\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\big) \\ &\qquad-\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big)\Bigg|\Bigg] \quad (24b)\\ &=\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}} \\ &\quad\times\Bigg[\Bigg|\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad+\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad+\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad-\ell\big(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\big)\Bigg|\Bigg] \quad (24c)\\ &\le\mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}} \\ &\quad\times\Bigg[\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\Big|\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big)\Big| \\ &\qquad+\Big|\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big) \\ &\qquad-\ell\Big(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\Big)\Big|\end{aligned}$$

$$+\left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)-\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\Bigg]\leq 3\beta. \tag{24d}$$

In (24), $f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot)$ is the mapping function learned over the training set $\mathcal{T}$ with $\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k$ replaced by $\boldsymbol{x}'_i,\boldsymbol{x}'_j,\boldsymbol{x}'_k$. Combing the results of (23) and (24), Inequality (10) holds. Therefore, with $\beta = o(1/\sqrt{n})$, the generalization gap will converge in the order of $\mathcal{O}(1/\sqrt{n})$ with high confidence 1-δ. The proof completes. ■

REFERENCES

[1] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," in *Proc. ICCAD*, Irvine, CA, USA, 2017, pp. 81–88.

[2] 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*, San Francisco, CA, USA, 2018, pp. 1–6.

[3] H. Geng, W. Zhong, H. Yang, Y. Ma, J. Mitra, and B. Yu, "SRAF insertion via supervised dictionary learning," in *Proc. ASPDAC*, 2019, pp. 406–411.

[4] H. Geng, W. Zhong, H. Yang, Y. Ma, J. Mitra, and B. Yu, "SRAF insertion via supervised dictionary learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2849–2859, Oct. 2020.

[5] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *Proc. ICCAD*, 2020, pp. 1–9.

[6] H. Yang *et al.,* "VLSI mask optimization: From shallow to deep learning," *Integr. VLSI J.*, vol. 77, pp. 96–103, Mar. 2021.

[7] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A GPU-enabled level set method for mask optimization," in *Proc. DATE*, Grenoble, France, 2021, pp. 1835–1838.

[8] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "DevelSet: Deep neural level set for instant mask optimization," in *Proc. ICCAD*, 2021, pp. 1–9.

[9] 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.

[10] 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, Art. no. 109610B.

[11] 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*, Sydney, NSW, Australia, 2012, pp. 263–270.

[12] 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," *J. Micro Nanolithogr. MEMS MOEMS*, vol. 14, no. 1, 2015, Art. no. 011003.

[13] 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, Art. no. 94270S.

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

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

[16] 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.

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

[18] J. Chen, Y. Lin, Y. Guo, M. Zhang, M. B. Alawieh, and D. Z. Pan, "Lithography hotspot detection using a double inception module architecture," *J. Micro Nanolithogr. MEMS MOEMS*, vol. 18, no. 1, 2019, Art. no. 013507.

[19] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan, "Semi-supervised hotspot detection with self-paced multi-task learning," in *Proc. ASPDAC*, 2019, pp. 420–425.

[20] 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. 1–6.

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

[22] R. Chen *et al.*, "Faster region-based hotspot detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Sep. 3, 2020, doi: 10.1109/TCAD.2020.3021663.

[23] H. Geng *et al.*, "Hotspot detection via attention-based deep layout metric learning," in *Proc. ICCAD*, San Diego, CA, USA, 2020, pp. 1–8.

[24] B. Zhu *et al.*, "Hotspot detection via multi-task learning and transformer encoder," in *Proc. ICCAD*, 2021, pp. 1–8.

[25] R. Chen *et al.*, "A unified framework for layout pattern analysis with deep causal estimation," in *Proc. ICCAD*, 2021, pp. 1–9.

[26] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, San Jose, CA, USA, 2012, pp. 349–350.

[27] L. Van Der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, 2014.

[28] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, "Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks," in *Proc. ASPDAC*, 2021, pp. 898–903.

[29] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, "Deep H-GCN: Fast analog IC aging-induced degradation estimation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Aug. 24, 2021, doi: 10.1109/TCAD.2021.3107250.

[30] H. Geng, F. Yang, X. Zeng, and B. Yu, "When wafer failure pattern classification meets few-shot learning and self-supervised learning," in *Proc. ICCAD*, 2021, pp. 1–8.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, Las Vegas, NV, USA, 2016, pp. 2818–2826.

[32] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. CVPR*, Honolulu, HI, USA, 2017, pp. 3156–3164.

[33] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proc. ECCV*, 2018, pp. 3–19.

[34] J. Fu *et al.*, "Dual attention network for scene segmentation," in *Proc. CVPR*, Long Beach, CA, USA, 2019, pp. 3146–3154.

[35] Y. Cui, F. Zhou, Y. Lin, and S. Belongie, "Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop," in *Proc. CVPR*, Las Vegas, NV, USA, 2016, pp. 1153–1162.

[36] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. NIPS*, 2003, pp. 521–528.

[37] X. Gao, S. C. Hoi, Y. Zhang, J. Wan, and J. Li, "SOML: Sparse online metric learning with application to image retrieval," in *Proc. AAAI*, 2014, pp. 1206–1212.

[38] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proc. CVPR*, Columbus, OH, USA, 2014, pp. 1875–1882.

[39] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based CNN with improved triplet loss function," in *Proc. CVPR*, Las Vegas, NV, USA, 2016, pp. 1335–1344.

[40] R. Jin, S. Wang, and Y. Zhou, "Regularized distance metric learning: Theory and algorithm," in *Proc. NIPS*, 2009, pp. 862–870.

[41] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. CVPR*, Boston, MA, USA, 2015, pp. 815–823.

[42] C. Huang, C. C. Loy, and X. Tang, "Local similarity-aware deep feature embedding," in *Proc. NIPS*, 2016, pp. 1262–1270.

[43] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," in *Proc. ICCV*, Venice, Italy, 2017, pp. 814–823.

[44] B. Harwood, B. G. V. Kumar, G. Carneiro, I. Reid, and T. Drummond, "Smart mining for deep metric learning," in *Proc. ICCV*, Venice, Italy, 2017, pp. 2821–2829.

[45] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai, "Hard-aware point-to-set deep metric for person re-identification," in *Proc. ECCV*, 2018, pp. 188–204.

[46] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. ICCV*, Venice, Italy, 2017, pp. 764–773.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020, pp. 1597–1607.
[48] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021. [Online]. Available: https://arxiv.org/abs/2103.03230.
[49] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.
[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Las Vegas, NV, USA, 2016, pp. 770–778.
[51] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.
[52] B. Wang, H. Zhang, P. Liu, Z. Shen, and J. Pineau, "Multitask metric learning: Theory and algorithm," in *Proc. AISTATS*, 2019, pp. 3362–3371.
[53] C. McDiarmid, "On the method of bounded differences," *Surveys Comb.*, vol. 141, no. 1, pp. 148–188, 1989.

Hao Geng (Student Member, IEEE) received the M.E. degree from the Department of Electronic Engineering and Information Sciences, University of Science and Technology of China, Hefei, China, in 2015, and the M.Sc. degree in machine learning from the Department of Computing, Imperial College London, London, U.K., in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include design space exploration, machine learning, deep learning, and optimization methods with applications in VLSI CAD.

Mr. Geng received the Best Paper Award Nomination from ASPDAC'2019.

Haoyu Yang received the B.E. degree from Qiushi Honors College, Tianjin University, Tianjin, China, in 2015, 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 a Research Scientist with NVIDIA ASIC and VLSI Research Group, Austin, TX, USA. His research interests include machine learning in VLSI design for manufacturability, high performance VLSI physical design with parallel computing, and machine learning security.

Dr. Yang received the Nick Cobb Scholarship from SPIE'2019 Advanced Lithography, the Best Paper Award Nomination from ASPDAC'2019, and the Best Poster Presentation from the student research forum of ASPDAC'2019.

Lu Zhang received the B.Sc. degree in microelectronics from Fudan University, Shanghai, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Her current research interest is machine learning for EDA.

Fan Yang (Member, IEEE) received the B.S. degree from Xi'an Jiaotong University, Xi'an, China, in 2003, and the Ph.D. degree from Fudan University, Shanghai, China, in 2008.

From 2008 to 2011, he was an Assistant Professor with Fudan University. He is currently an Associate Professor with the Microelectronics Department, Fudan University. His research interests include model order reduction, circuit simulation, high-level synthesis, yield analysis, and design for manufacturability.

Xuan Zeng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Fudan University, Shanghai, China, in 1991 and 1997, respectively.

She is a Full Professor with the Department of Microelectronics, Fudan University. She was the Director of the State Key Laboratory of ASIC and System from 2008 to 2012. She was a Visiting Professor with the Department of Electrical Engineering, Texas A&M University, College Station, TX, USA, and the Department of Microelectronics, Technische Universiteit Delft, Delft, The Netherlands, in 2002 and 2003, respectively. Her current research interests include analog circuit modeling and synthesis, design for manufacturability, high-speed interconnect analysis and optimization, and circuit simulation.

Prof. Zeng received the Best Paper Award from the 8th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference 2017. She received Changjiang Distinguished Professor with the Ministry of Education Department of China in 2014, the Chinese National Science Funds for Distinguished Young Scientists in 2011, the First-Class of Natural Science Prize of Shanghai in 2012, 10th For Women in Science Award in China in 2013, and Shanghai Municipal Natural Science Peony award in 2014. He is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: PART—II, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and *ACM Transactions on Design Automation of Electronic Systems*.

Bei Yu (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received seven Best Paper Awards from ASPDAC 2021, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, ICCAD 2013, ASPDAC 2012, and six ICCAD/ISPD contest awards. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of IEEE TCCPS Newsletter.