Hotspot detection using squish-net

Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, Bei Yu

Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, Bei Yu, "Hotspot detection using squish-net," Proc. SPIE 10962, Design-Process-Technology Co-optimization for Manufacturability XIII, 109620S (20 March 2019); doi: 10.1117/12.2515172

SPIE.

Event: SPIE Advanced Lithography, 2019, San Jose, California, United States

Hotspot Detection Using Squish-Net

Haoyu Yang[1,2], Piyush Pathak[2], Frank Gennari[2], Ya-Chieh Lai[2], Bei Yu[1]
[1]The Chinese University of Hong Kong, NT, Hong Kong
[2]Cadence Design Systems Inc., CA 95134, USA

ABSTRACT

Design-process weakpoints also known as hotspots cause systematic yield loss in semiconductor manufacturing. One of the main goals of DFM is to detect such hotspots. For the application of AI in hotspot detection, a variety of machine learning-based techniques have been proposed as an alternative to time expensive process simulations. Related research works range from finding efficient layout representations and features and developing reliable machine learning models. Main stream layout representations include density-based feature, pixel-based feature, frequency domain feature, concentric circle sampling (CCS) and squish pattern. However most of them are either suffering from information loss (e.g. density-based feature, and CCS), or not storage efficient (e.g. images). To address these problems, we propose a convolutional neural network called Squish-Net where the input pattern representation is in an adaptive squish form. Here, the squish pattern representation is modified to handle variations in the topological complexity across a pattern catalog, which still allows no information loss and high data compression. We show that different labeling strategies and pattern radius contribute to the trade-offs between prediction accuracy and model precision. Two imbalance-aware training strategies are also discussed with supporting experiments.

1. INTRODUCTION

Design-process weakpoints also known as hotspots cause systematic yield loss in semiconductor manufacturing.[1,2] One of the main goals of DFM is to detect such hotspots. For the application of AI in hotspot detection, a variety of machine learning-based techniques have been proposed as an alternative to time expensive process simulations.

Although machine learning has brought evolutionary benefits of computer vision tasks, such techniques cannot directly applied here due to the speciality of layout design and sign-off flows. Recent researches seek to embed machine learning techniques to complicated DFM flows. [3,4] adopt regular convolutional neural networks for layout hotspot detection tasks which take layout images as input and consider greedy solutions regarding the imbalanced training set problem. [5] formulates a machine learning model that can be used to predict edge displacements in model-based OPC, which reduces optimization runtime by a significant amount. [6] argues that pooling layers in traditional CNNs drop important information of layout context and proposes to replace the pooling layers with strided convolution layers, which achieves better hotspot predicition accuracy. [7] studies the effectiveness of generative machine learning models in mask optimization tasks, where an ILT-guided training strategy is proposed to achieve better convergence and generated mask quality. [8] proposes another hotspot detection framework, where layout clips are compressed in frequency domain with minor information loss. The compressed data packet format is naturally compatible with deep neural networks. A batch-biased learning algorithm is developed along with shallow CNNs to achieve satisfactory trade-off between prediction accuracy and false alarms. Very recently, [9] proposes an adaptive squish representation that makes the multilayer hotspot detection tasks easier.

Specifically, machine learning-based hotspot detection researches focus on finding efficient layout representations and features[3,5,8,10,11] and developing reliable machine learning models.[3,8,10,12,13] Main stream layout representations include density-based feature,[10] pixel-based feature,[3,4] frequency domain feature,[8,14] concentric circle sampling (CCS)[12] and squish pattern.[11] However, most of them are more or less suffering from drawbacks. Density-based features correspond to the ratio of geometry and spacing area which significantly drop shape relations. Pixel-based features (i.e. raw images) are not computational friendly in terms of both training and inferencing, and so are frequency domain features. Although CCS adopts mutual information for better feature correlations, it still ignores large amount of context information. Squish pattern, on the other hand, is a scan line-based layout representation, where a layout clip is cut into grids by scan lines that overlap with all shape edges. However, squish pattern extraction are likely to result in different dimensionality which is not machine

Design-Process-Technology Co-optimization for Manufacturability XIII, edited by Jason P. Cain, Proc. of SPIE Vol. 10962, 109620S · © 2019 SPIE · CCC code: 0277-786X/19/$18 · doi: 10.1117/12.2515172

learning friendly. To address this problem, [9] proposes a novel adaptive squish pattern representation which is modified to handle variations in the topological complexity across a pattern catalog.[15] This representation allows no information loss and high data compression.

Imbalanced dataset is a general problem of machine learning-based hotspot detectors which is discussed in [3]. [3] proposes an upsampling approach that duplicates the minority class (i.e. hotspot) in the training set. However such setting might significantly introduce over-fitting. In this paper, we discuss two efficient solutions to reduce the side effect of the imbalanced problem as much as possible. One solution is called balanced batch that forces the training batch to be balanced by sampling same amount instances from both class. The other solution samples non-hotspot patterns according to their CCD and complexity score distribution, such that the training set will be balanced before training stage starts. We will show two methods exhibit different trade-offs on true positive and false positive rates.

Hotspot labeling is another problem when generating the training set. Previous frameworks are usually evaluated on existing datasets with patterns being labeled as hotspots or non-hotspots.[1,3,6,16–19] However, pattern labeling is also critical in real backend design flows. Here we label a clip as hotspot or non-hotspot according to the distance between the clip center and the location where real hotspot occurs. If the distance is smaller than some threshold value, we will mark the clip as a hotspot and vice versa. It can be seen that a smaller threshold comes with less noise but more serious imbalance problem and a larger threshold makes the labels in the training set inaccurate.

The rest of the paper is organized as follows. Section 2 introduces basic terminologies related to machine learning-based hotspot detection problem. Section 3 discusses the details of our Squish-Net framework. Section 4 shows experimental results followed by conclusion in Section 5.

2. PRELIMINARIES

In this section, we will introduce basic terminologies and concepts related to this work. We adopt the same evaluation metrics as introduced in [1] which is defined based on the confusion matrix and applied in most recent hotspot detection works.

Definition 1 (True Positive (TP)) *The total number of correctly predicted hotspots is called TP. The ratio between TP and total number of hotspots is defined as accuracy or recall.*

Definition 2 (False Positive (FP)) *The number of nonhotspot locations that are reported as hotspots by the hotspot detector. False alarm rate is similarly defined by the ratio between FP and total number of nonhotspots.*

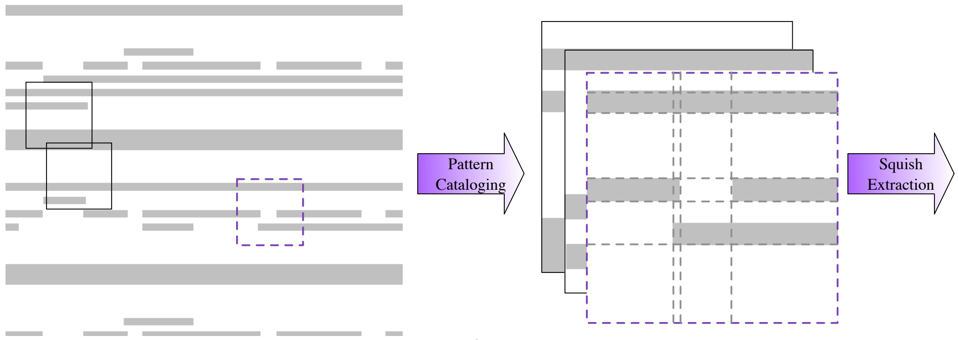
In modern DFM flows, post-tapeout inspection overhead is almost strictly related to FPs, thus we also expect most predicted positive samples are TPs. Such overhead is quantified as follows.

Definition 3 (Precision) *The ratio between TP and total number of predicted positive samples.*

Modified Squish Pattern

The classic squish pattern[11] is a lossless layout representation that consists of layout topology and geometric information.[9] As shown in Figure 1, a layout design is converted into clips for pattern analysis. Each clip is split into grids by a set of scan lines that cover all the shape edges. The topology of a given pattern can then be defined by a matrix $\boldsymbol{T}$ that has the same dimension as the pattern grids. Two vectors $\boldsymbol{\delta_x}$ and $\boldsymbol{\delta_y}$ store the grid size alone x-axis and y-axis respectively. We first show how $\boldsymbol{T}, \boldsymbol{\delta}_x$ and $\boldsymbol{\delta}_y$ are embedded into a single tensor. Here we expand $\boldsymbol{\delta}_x$ and $\boldsymbol{\delta}_y$ into the same dimensionality as $\boldsymbol{T}$ by tiling them vertically and horizontally.

$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{\delta}_{x\_tile} = \begin{bmatrix} 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \\ 75 & 13 & 78 & 83 \end{bmatrix}, \boldsymbol{\delta}_{y\_tile} = \begin{bmatrix} 25 & 25 & 25 & 25 \\ 32 & 32 & 32 & 32 \\ 96 & 96 & 96 & 96 \\ 32 & 32 & 32 & 32 \\ 32 & 32 & 32 & 32 \\ 32 & 32 & 32 & 32 \\ 45 & 45 & 45 & 45 \end{bmatrix}, \tag{1}$$



$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\delta}_x = \begin{bmatrix} 75 & 13 & 78 & 83 \end{bmatrix}$$

$$\boldsymbol{\delta}_{\boldsymbol{y}} = \begin{bmatrix} 25 & 32 & 96 & 32 & 32 & 32 & 45 \end{bmatrix}$$

Figure 1: A simple pattern example with scan lines.

which can be stacked together as a 3-channel tensor and the second and the third channel store the grid size at corresponding locations in x and y directions respectively. It can be observed that some $\boldsymbol{\delta}_{x\_tile}$ and $\boldsymbol{\delta}_{y\_tile}$ will have large variations in their entries that might lead the gradient out of control during the neural network training.

In [9], an adaptive squish pattern is proposed to effectively pad input tensors into any desired size while reducing the variations of $\boldsymbol{\delta}_{x_tile}$s and $\boldsymbol{\delta}_{y_tile}$s. The basic idea is to introduce additional scan lines that can still make the whole input tensor informative. Suppose $\boldsymbol{T} \in \mathbb{R}^{7\times 4}$ is to be extended to $\mathbb{R}^{7\times 7}$. In this case, only x direction needs to be processed. By the algorithm in [9], we are able to obtain the following adaptive squish representation.

$$\begin{aligned}
\boldsymbol{T}_a &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
\boldsymbol{\delta}_{x_a} &= \begin{bmatrix} 37.5 & 37.5 & 13 & 39 & 39 & 41.5 & 41.5 \end{bmatrix}, \\
\boldsymbol{\delta}_{y_a} &= \begin{bmatrix} 25 & 32 & 96 & 32 & 32 & 32 & 45 \end{bmatrix}.
\end{aligned}$$

3. THE METHODS

3.1 Training on the Imbalanced Dataset

3.1.1 Labeling Layout Patterns

After cataloging, a layout will be converted into a set of layout clips that contain patterns of interest. Defects might occur anywhere within some clips. Here a problem comes out that whether a clip should be labeled as hotspot as long as there are defects inside it. In most cases, lithography hotspots are not usually caused by the pattern itself but also caused by a relatively larger context region surrounding it. Intuitively, it is risky to label a clip as hotspot if defects are located near the boundary of the clip, because this clip misses lots of context information that causes the defects. Labeling such clip as hotspot will inevitably induce noises during training. Here we introduce center-to-center distance (d_{c2c}) that will be used as the criteria when labeling clips, as shown in Figure 2.

Definition 4 (Center-to-Center Distance) *The smallest distance between the center of a clip and the center of all the defects in the clip.*

The label of a clip is determined accordingly by a center-to-center distance threshold t_{c2c}. A clip is labeled as hotspot if and only if $d_{c2c} \leq t_{c2c}$.

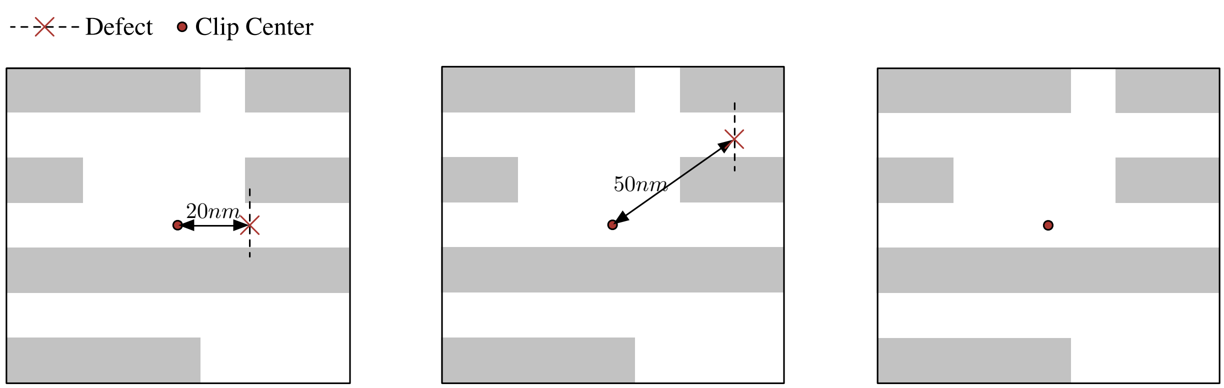
3.1.2 Imbalance-aware Training

In hotspot detection tasks, existing hotspot library is usually highly imbalanced with only a small fraction of hotspot instances.[3] We visualize the breakdown of hotspot and non-hotspot clip percentages in a $7nm$ metal layer dataset, as in Figure 3, where "rxxx"s represent clip radius in nm and the percentages are calculated when $t_{c2c} = 48nm$. As can be seen that only less than 1% of clips are hotspot. [3] addresses this problem by duplicating minority hotspot patterns in the training set and formulates a balanced pattern library. However, such strategy creates lots of repeated hotspot patterns that might cause serious overfitting.

In this paper, we adopt two solutions to alleviate this problem. In the first solution, we sample out non-hotspot clips from the training set such that instances in the training set are balanced for both categories. Previous work has shown that layout follows a normal distribution in terms of the count of critical dimension (CCD) scores.[20] In the second solution, we conduct sampling before training starts. We sample out the same amount of non-hotspot patterns as the number of hotspot patterns in the training set. The sampling procedure also makes sure that the CCD scores in the reduced training set follow a unified distribution for a better training behavior. We will show later in the experiments both solutions contribute to the imbalanced training set problem and exhibit different trade-offs between TP and FP.

3.2 The Neural Network Architecture

The detailed network configurations are listed in Table 1. Column "Layer" lists layer types and ID. Columns "Filter" and "Stride" define the size and the scan step of convolution and pooling layers. Column "Output" lists the output dimensionality of current layer. [21] has shown that shortcut links between different convolution layers allow gradients to be more easily backpropogated to early layers. In this paper, we intentionally add the output of conv$i$-1 and conv$i$-4 together before going into next level convolution stages.



(a) Hotspot (b) Non-hotspot (c) Non-hotspot

Figure 2: An example of labeling clips, assuming $t_{c2c} = 48nm$. The red dot indicates the center of a clip and the cross markers are the locations where defects occur. If a defect occurs close to the clip center, we will label it as hotspot.

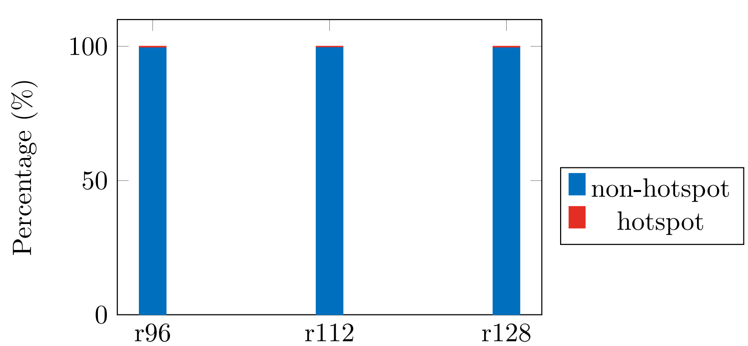


Figure 3: Breakdown of hotspot and non-hotspot clip percentages of different t_{c2c}s.

Table 1: Neural networks configuration details.

Layer	Filter	Stride	Output	Parameter
conv1-1	5×5×128	2	32×32×128	9600
conv1-2	5×5×128	1	32×32×128	409600
conv1-3	5×5×128	1	32×32×128	409600
conv1-4	5×5×128	1	32×32×128	409600
conv2-1	5×5×256	2	16×16×256	819200
conv2-2	5×5×256	1	16×16×256	1638400
conv2-3	5×5×256	1	16×16×256	1638400
conv2-4	5×5×256	1	16×16×256	1638400
conv3-1	5×5×512	2	8×8×512	3276800
conv3-2	5×5×512	1	8×8×512	6553600
conv3-3	5×5×512	1	8×8×512	6553600
conv3-4	5×5×512	1	8×8×512	6553600
conv4-1	5×5×1024	2	4×4×1024	13107200
fc1	-	-	1024	16777216
fc2	-	-	2	2048
Summary	-	-	-	59796864

Table 2: Benchmark Statistics.

Radius (nm)	Train		Test	
	Hotpot	Non-Hotspot	Hotspot	Non-Hotspot
96	21688	3324144	1906	91387
112	32695	5746509	2435	136413
128	43756	7376706	2898	167887

4. EXPERIMENTAL RESULTS

4.1 Configurations and The Dataset

We implement our framework with `Tensorflow` library[22] on a Intel platform with one Tesla P100 graphic card. To verify the proposed framework, we adopt an industry $7nm$ metal layer layout. The benchmark details are listed in Table 2, where column "Radius (nm)" indicates the clip size that used for pattern cataloging and training set preparation and columns "Hotspot" and "Non-Hotspot" list the number of hotspot and non-hotspot patterns in the dataset. Because we use same anchoring policy when extracting patterns, larger radius will inevitably result in more pattern count and larger imbalanced ratio between hotspot and non-hotspot patterns. It should be noted that all layout clips are with square shapes, therefore the radius here only represents the vertical or horizontal distance from a clip center to the boundaries of the clip.

All neural network models are trained with a batch size of 64 at an initial learning rate of 0.001 that will be decayed by 0.7 every 2000 iterations. We pick 10000 as the maximum number of iterations and the best models are selected based on the cross-validation from 500 hotspot patterns that are never seen during training. All neuron weights are initialized with Xavier initializer and all biases are set to zero before the training starts. We also apply a L_2 regularizer on all neuron weights in case of overfitting.

4.2 Study of C2C Threshold

In the first experiment, we will show that larger C2C thresholds induce additional noise when generating labels of the training set. We list the hotspot prediction results in Table 3, where column "C2C (nm)" lists three

Table 3: Experimental results on different C2C thresholds.

C2C (nm)	method	TPR	FPR	precision
28	balance	0.9903	0.0649	0.2414
	sampling	0.9913	0.0979	0.1744
48	balance	0.9848	0.0734	0.2186
	sampling	0.9916	0.1368	0.1313
96	balance	0.9676	0.1597	0.1550
	sampling	0.9818	0.2157	0.1212

Table 4: Experimental results on different clip radius.

Radius (nm)	method	TPR	FPR	precision	recall
96	balance	0.9848	0.0734	0.2186	0.9848
	sampling	0.9916	0.1368	0.1313	0.9916
112	balance	0.9848	0.0603	0.2257	0.9848
	sampling	0.9979	0.1032	0.1472	0.9979
128	balance	0.9806	0.0636	0.2102	0.9806
	sampling	0.9959	0.2009	0.0788	0.9959

different C2C thresholds, column "method" includes the methods that used to handle the imbalanced training set problem and columns "TPR", "FPR" and "precision" list the model performance in terms of true positive rate, false positive rate and the precision, respectively. Rows "balance" shows the hotspot detection results with the first solution of the imbalance-aware training and "sampling" corresponds to the second solution, as introduced in previous section. It can be seen from the table that the best detection performance comes with the smallest C2C threshold with over 99% TPR and less than 10% FPR whichever imbalance-aware training solution is used. As the C2C threshold increases, we will inevitably face more contaminated labels due to the reason discussed in Section 3.1, which is also consistent with the prediction results here. It can also be seen that the "balance" and "sampling" methods exhibit trade-offs between TPR and FPR, with the "sampling" method achieving a relatively higher hotspot detection accuracy at the cost of higher FPR that also reflects lithography simulation overhead in the layout physical verification flow.

4.3 Study of Clip Radius

In the second experiment, we train the neural networks with three different training sets, which are cataloged from the same layout with same anchoring point. The only difference is the clip radius. In our experiments, we have 3 radius settings ranging from $96nm$ to $128nm$. Because the pattern clips in this work are all in square shape, here the radius only refers to the distance between the clip center to clip edges. We list the experimental results in Table 4, where column "Radius (nm)" corresponds to different clip radius when generating the training set and columns "method", "TPR", "FPR", "precision" and "recall" are defined exactly the same as in Table 3. Theoretically, larger radius attains additional context information of a layout clip that is expected to achieve better model generality and prediction accuracy, which holds when we extend the radius from $96nm$ to $112nm$. However, as can be seen in the table, when we continuously increase the radius to $128nm$, slight performance degradation can be observed. Such results can be explained by the fact that the imbalanced dataset problem is getting worse with lager pattern radius, as listed in Table 2. Two imbalance-aware training solutions also follow the same trend as discussed before, with the "balance" method attaining a higher precision.

5. CONCLUSION

In this paper, we propose a deep learning-based hotspot detection framework where the input pattern is in an adaptive squish form. We discuss and show that different layout pattern labeling strategies are associated with trade-offs caused by the training set distribution and the noise information. We also study the effect of different pattern radius and show that larger radius grants the machine learning model better context information in the training phase while, however, inducing weaker training data distribution. Two training strategies are also studied in this paper to address the imbalanced training set problem. Experiments are conducted on EUV-specific $7nm$ metal layer design that show the potential of the emerging deep learning solutions on physical verification tasks.

Acknowledgment

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

REFERENCES

[1] Torres, A. J., "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in [*IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*], 349–350 (2012).

[2] "ITRS." http://www.itrs.net.

[3] Yang, H., Luo, L., Su, J., Lin, C., and Yu, B., "Imbalance aware lithography hotspot detection: a deep learning approach," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)* **16**(3), 033504 (2017).

[4] Shin, M. and Lee, J.-H., "Accurate lithography hotspot detection using deep convolutional neural networks," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)* **15**(4), 043507 (2016).

[5] Matsunawa, T., Yu, B., and Pan, D. Z., "Optical proximity correction with hierarchical bayes model," in [*Proceedings of SPIE*], **9426** (2015).

[6] Yang, H., Lin, Y., Yu, B., and Young, E. F., "Lithography hotspot detection: From shallow to deep learning," in [*IEEE International System-on-Chip Conference (SOCC)*], 233–238 (2017).

[7] Yang, H., Li, S., Ma, Y., Yu, B., and Young, E. F., "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in [*ACM/IEEE Design Automation Conference (DAC)*], (2018).

[8] Yang, H., Su, J., Zou, Y., Ma, Y., Yu, B., and Young, E. F., "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2018).

[9] Yang, H., Pathak, P., Gennari, F., Lai, Y.-C., and Yu, B., "Detecting multi-layer layout hotspots with adaptive squish patterns," in [*IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*], (2019).

[10] Matsunawa, T., Gao, J.-R., Yu, B., and Pan, D. Z., "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in [*Proceedings of SPIE*], **9427** (2015).

[11] Gennari, F. E. and Lai, Y.-C., "Topology design using squish patterns," (Sept. 9 2014). US Patent 8,832,621.

[12] Zhang, H., Yu, B., and Young, E. F. Y., "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in [*IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*], 47:1–47:8 (2016).

[13] Ding, D., Torres, A. J., Pikus, F. G., and Pan, D. Z., "High performance lithographic hotspot detection using hierarchically refined machine learning," in [*IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*], 775–780 (2011).

[14] Zhang, W., Li, X., Saxena, S., Strojwas, A., and Rutenbar, R., "Automatic clustering of wafer spatial signatures," in [*ACM/IEEE Design Automation Conference (DAC)*], 71:1–71:6 (2013).

[15] Cain, J. P., Fakhry, M., Pathak, P., Sweis, J., Gennari, F. E., and Lai, Y.-C., "Pattern-based analytics to estimate and track yield risk of designs down to 7nm," in [*Proceedings of SPIE*], **10148** (2017).

[16] Wen, W.-Y., Li, J.-C., Lin, S.-Y., Chen, J.-Y., and Chang, S.-C., "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* **33**(11), 1671–1680 (2014).

[17] Matsunawa, T., Yu, B., and Pan, D. Z., "Laplacian eigenmaps-and bayesian clustering-based layout pattern sampling and its applications to hotspot detection and optical proximity correction," *Journal of Micro/-Nanolithography, MEMS, and MOEMS (JM3)* **15**(4), 043504–043504 (2016).

[18] Matsunawa, T., Yu, B., and Pan, D. Z., "Laplacian eigenmaps and bayesian clustering based layout pattern sampling and its applications to hotspot detection and OPC," in [*IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*], 679–684 (2016).

[19] Yang, F., Sinha, S., Chiang, C. C., Zeng, X., and Zhou, D., "Improved tangent space based distance metric for lithographic hotspot classification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* **36**(9), 1545–1556 (2017).

[20] Pathak, P., Krishnamoorthy, K., Wang, W.-L., Lai, Y.-C., Gennari, F. E., Somani, S., Pack, B., Schroeder, U. P., Batarseh, F., Bravo, J., et al., "Methodology to extract, data mine and score geometric constructs from physical design layouts for analysis and applications in semiconductor manufacturing," in [*Design-Process-Technology Co-optimization for Manufacturability X*], **9781**, 978109, International Society for Optics and Photonics (2016).

[21] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 770–778 (2016).

[22] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al., "TensorFlow: A system for large-scale machine learning," in [*USENIX Symposium on Operating Systems Design and Implementation (OSDI)*], 265–283 (2016).