Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 16, No. 3, September 2016, pp. 208-215
http://dx.doi.org/10.5391/IJFIS.2016.16.3.208

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

CNN Based Lithography Hotspot Detection

Moojoon Shin[1,2] and Jee-Hyong Lee[1]
[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Korea
[2]Samsung Electronics, Hwasung, Korea

Ijfis

Abstract

The lithography hotspot detection process is crucial for semiconductor design development process. But, the lithography hotspot detection using optical simulation method takes much time and it slowdown the layout design development cycle. Though the geometry based approach is introduced as an alternative, it still revealed low detection performance and sophisticated framework. To solve this problem, we introduce a deep convolutional neural network based hotspot detection method. Our method made better results in ICCCAD 2012 dataset. To reach this score, we used lots of technical effort to improve the result in addition to just utilizing the nature of convolutional neural network. Inspection region reduction, data augmentation, DBSCAN clustering helped our work more stable and faster.

Keywords: Lithography hotspot detection, Convolutional neural network

1. Introduction

As the semiconductor design shrinks the physical design is more restricted by the lithography constraint because the main patterning process is still mainly relies on 193 nm lithography process. Thus, semiconductor design should be verified in the view of lithography process. The optical rule check is the golden rule of physical design verification. It calculates a projected image on silicon wafer using optical simulation, and then it finds problematic position from the image. Though it has great hotspot detection performance, it requires heavy computational cost due to its optical simulation process. It results in the slowdown of physical design development.

Since the optical simulation based verification requires heavy time consumption, the geometric verification methods have been introduced as an alternative. These methods have fast detection speed because they don't require optical simulation and infer the design layout itself. The pattern matching based method [1-3] and machine learning based method [4-7] was introduced.

The pattern matching method is useful for registered hotspot detection. It detects hotspots by evaluating geometric similarity of the pattern in the test layout with registered hotspot patterns. Though it is trustable for known hotspot, it has limit to find unknown hotspot and it result in low detection accuracy.

The machine learning method is introduced to find unknown hotspots. The machine learns the classification ability from known hotspot and non-hotspot dataset with supervised learning. Since, it learns the characteristics of the geometry of hotspot/non-hotspot during the training,

Received: Sep. 7, 2016
Revised : Sep. 13, 2016
Accepted: Sep. 14, 2016

Correspondence to: Jee-Hyong Lee
(John@skku.edu)

the trained model have ability to detect unknown hotspot.

But former machine learning method was accompanied by complex frameworks including complex layout encoding (feature extraction) and multi classifier usage due to its low-performance model usage like support vector machine (SVM). In spite of its unknown hotspot detection ability and its sophisticated framework usage, it revealed high false positive in ICCAD2012 benchmark dataset [8].

To cope with it, we introduce convolutional neural network (CNN) based hotspot method. The semiconductor layout is basically 2 dimensional image and it was proven to have great image classification performance [9, 10]. Due to its self-trainable convolution filters which play a role good feature extractor we were able to achieve better result using very simple framework: just one classifier with simple density based layout encoding. We made additional effort for improvement of our result. We first found adequate CNN topology by evaluating various network structures. And we augmented training data to prevent overfitting caused by low given samples of the dataset. Next we applied candidate region to reduce the inspection region in the test layout. Through these efforts, our method achieved 97.9% of recall and 15.6% of precision which is better than previous results [4, 5].

The remainder is organized as follows. In Section 2, we introduce the proposed method including our overall framework, hotspot model training and hotspot detection using trained CNN classifier. In Section 3, the experiment in the view point of detection performance and runtime is followed. The last section concludes the paper.

2. Proposed Method

Our approach treats the lithography hotspot detection problem because the semiconductor layout can be simply represented as an image. Thus, our hotspot detection method follows traditional image detection method as shown in Figure 1. We consider the test layout as an image and we find out the problematic points on the test layout using conventional sliding window scan method. The calibrated CNN model evaluates hotspot probability of each pixel during the sliding window step. As a result of the inspection, we get the hotspot probability map which indicates the hotspot probability of each pixel location. Finally we convert hotspot probability map to final hotspot coordinates in the test layout.

The CNN model is the key of our detection framework. This model is calibrated with given hotspot and non-hotspot images. In contrast to shallow machine learning like SVM, the CNN learns feature extraction during training step. Thus the CNN can be trained with high dimensional raw image without handcrafted feature extraction. Figure 2 illustrates the feature extraction and classification of a hotspot. An image with 76 pixel is convolved and shattered with multiple convolutional weights. Through the convolution-pool steps the input image is finally transformed to 80 ea of unit pixel image at CP4 step and it is classified as a hotspot by fully connected layers (FC1,FC2).

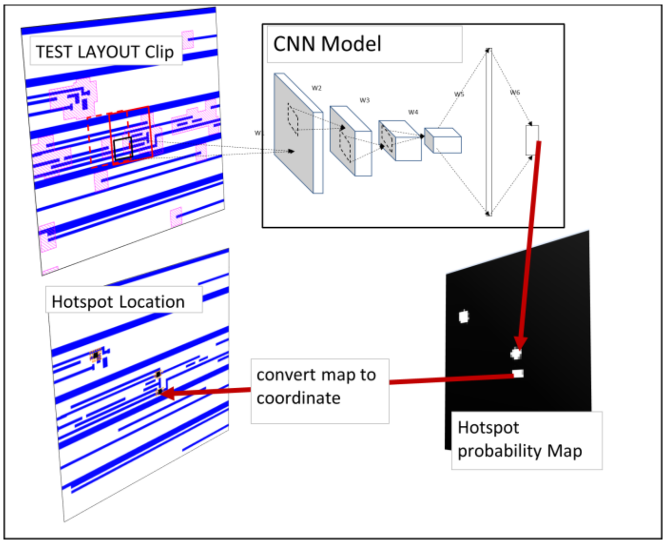

Figure 1. Our hotspot detection flow.

The whole training / test sequence is described in Figure 3. The given hotspot and non-hotspot location is transformed to two dimensional images through the density encoding step. The number of train image set is increased using data augmentation steps to prevent overfitting. The CNN is trained using stochastic gradient decent method. The calibrated CNN model is employed in hotspot detection task. Like the training step, the first step of hotspot detection task is 2D image transform. The candidate region generation which defines inspection is done concurrently. Then, large 2D image is sliding-window-scanned so that the hotspot probability map is derived. The hotspot location is finally obtained through the density-based spatial clustering (DBSCAN) on hotspot probability map.

2.1 Density Based Layout Encoding

Though the semiconductor layout represents binary image, its GDSII format is different from conventional 2D pixelated image format. Thus, various layout representation methods have been proposed for machine learning or pattern matching purpose.

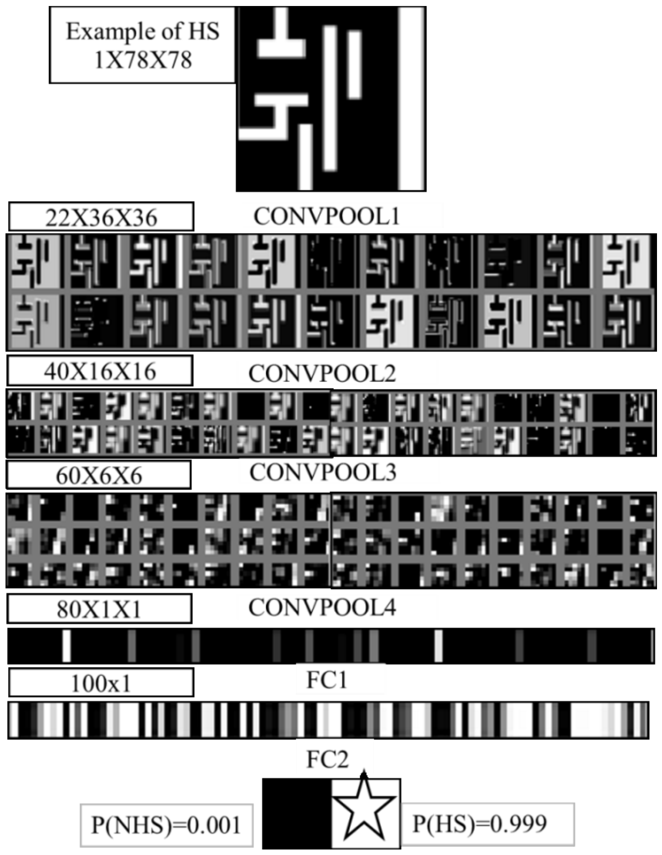

Figure 2. Feature extraction and classification inside our CNN model.

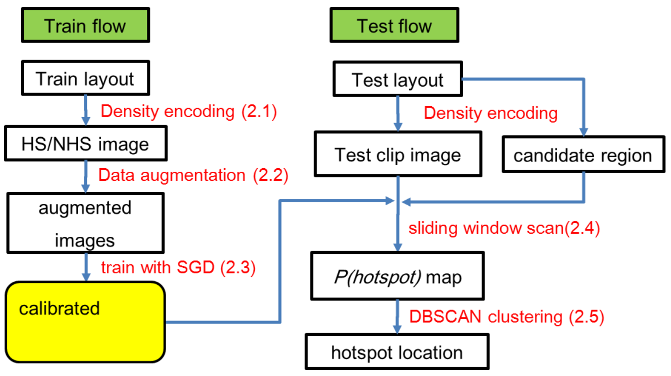

Figure 3. Overall flow of train / test flow.

The most classic way is density based encoding. This method converts the layout to 2D grayscale image convert it to 1D array [6]. The novel methods with string based encoding [4] and edge based encoding [5] were introduced too.

Our method is basically equal to density based encoding. But there are two differences: The first difference is that we use small image grid as 10nm to minimize the information loss by aliasing. Though other works use large image grid due to dimensionality issue [6, 7], The CNN can efficiently handle high dimensional image due to its convolution filter and pooling operator. The other difference is that CNN can process 2D image directly while previous density based layout encoding flattened 2D image into 1D array to fit the data into shallow models. The example of our layout encoding is illustrated in Figure 4.

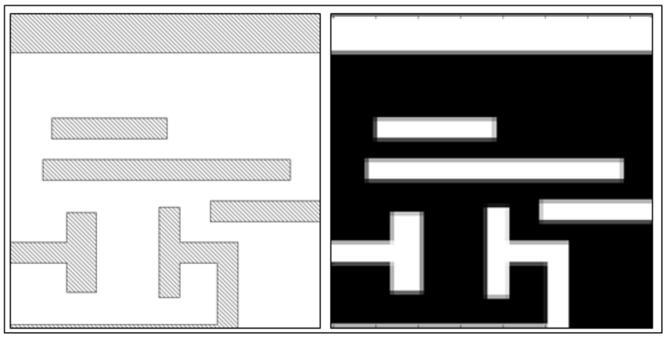

Figure 4. The original layout and density encoded layout with 10 nm image grid.

2.2 Training Data Augmentation

The number of given training sample in ICCAD2012 dataset is very small. As shown in Table 1, it provides just 500-5000 samples. Thus, we applied data augmentation to increase the number of training samples. The semiconductor layout pattern cannot maintain similar characteristics through scaling, color variation and rotation in same design level. Thus, the available options are image shift and image flip transform.

We set different maximum shift value for each hotspot and non-hotspot samples. We set 70 nm shift range for hotspot samples for accurate localization and 600 nm shift range for non-hotspot samples to learn plenty non-hotspot patterns to reduce the false positive. Note that semiconductor layout has much more various non-hotspot patterns than hotspot patterns.

Figure 5 describes the data augmentation process. First we select the multiple random positions inside shift range and we get the input images (780×780 nm^2) and apply flip transform. In addition, we adjust the number of sample selected inside shift range to balance the HS (hotspot) / NHS (non-hotspot) ratio. The result of data augmentation has more than 100,000 samples and the NHS/HS ratio is less than 5.

2.3 CNN Model Calibration

The network structure of our network is described in Table 2. Our CNN model is composed of 4 conv-pool layers (CP) and

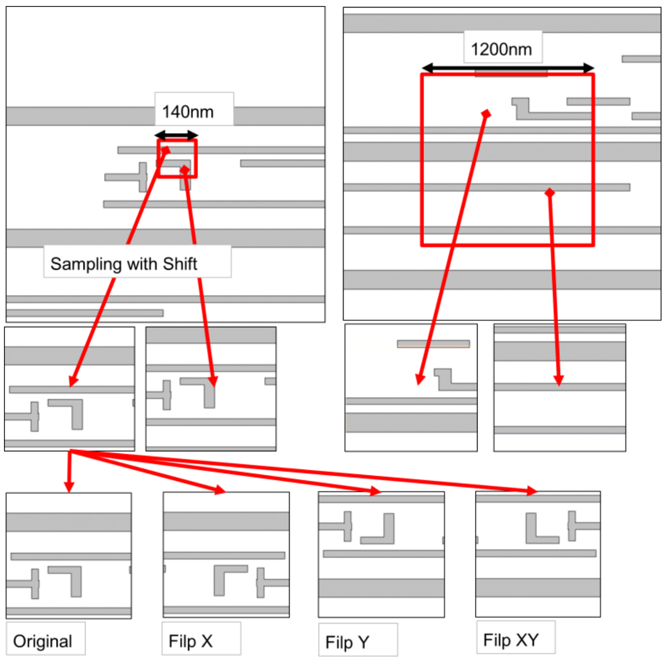

Figure 5. Our data augmentation sequence composed of shift / filp transform.

Table 1. ICCAD2012 Benchmark composition

Dataset	Train		Test	
	#HS	#NHS	#HS	#NHS
Benchmark 1 (3 nm)	99	340	226	319
Benchmark 2 (28 nm)	174	5285	499	4146
Benchmark 3 (28 nm)	909	4643	1847	3541
Benchmark 4 (28 nm)	95	4452	192	3386
Benchmark 5 (28 nm)	26	2716	42	2111

2 fully connected layers (FC). The each conv-pool layers have Max-Pooling with ratio of 2 and rectified linear unit (reLU) as activation function. Also each fully connected layers use hyperbolic tangent (Tanh) as activation function. The output of FC2 is converted to probability of hotspot by soft-max function and its probability is converted to the cost using cross entropy function. It is trained with stochastic gradient decent (SGD) to reduce the cross entropy cost. The learning rate of SGD is 0.02 and mini batch size is 100. The stopping condition is when the cost (cross entropy) is less than 0.005.

2.4 Hotspot Detection by Scanning Calibrated CNN Model

The Figure 1 illustrates hotspot detection using calibrated model. The trained model scans the entire area to get a hotspot probability map. This scan inspection step is time-challenging because the structure of CNN is much heavier than SVM or ANN. To reduce runtime, we reduce the scan inspection area defining candidate regions and we scan not every pixel but every 5 pixels for x and y directions considering the smoothness of probability map.

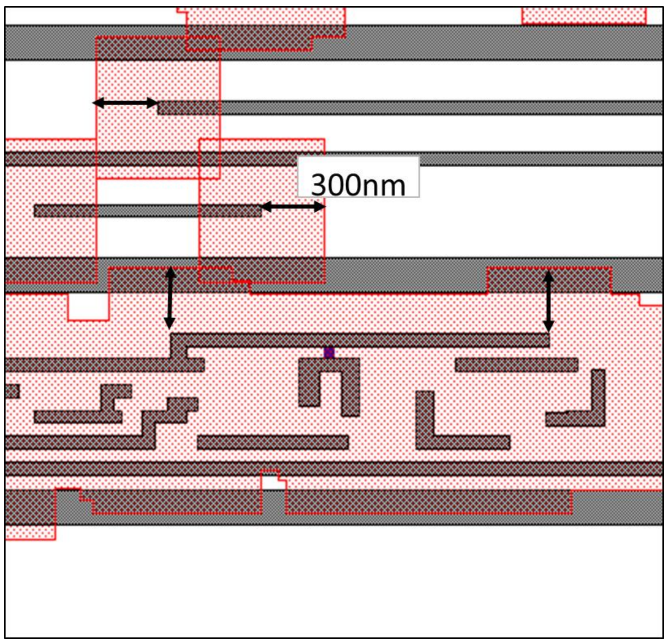

Figure 6. The example of candidate region (region in red). The location of hotspot (black marks).

Candidate region is the area which includes all possible hotspots. This region is formed by expanding all layout vertices to certain length. Figure 6 is the example of candidate region. By identifying candidate regions, we can reduce the inspection area to about 50% of the whole layout. Moreover, we can eliminate false positives outside of the candidate region. We setup the candidate region expanded 300 nm from the original layout vertices. Note that the extension value is determined to cover all given hotspot in the training layout.

2.5 Hotspot Coordinate Extraction with DBSCAN Clustering

After we extract the hotspot probability map though the scan step, the exact location of hotspot should be extracted. To extract the location of hotspot, we find potential hotspot pixels whose probability of hotspot is more than 0.5. Generally these potential hotspot are clustered in certain location and we can say this clustered points can be identified as one hotspot.

Thus we clarify the centers of potential hotspot clusters using density-based spatial clustering (DBSCAN) [11] clustering.

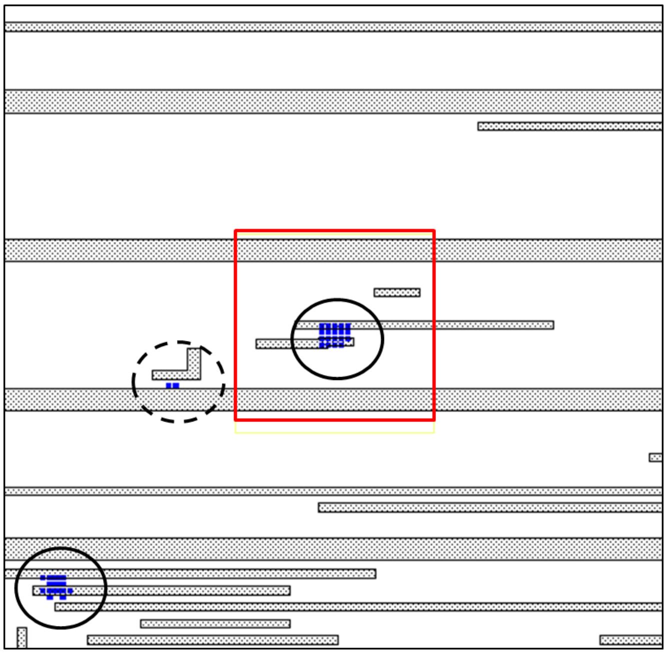

Figure 7. The example of hotspot detection in a layout clip. Potential hotspot (dots), accepted clusters (circle) and not accepted cluster (dashed circle).

DBSCAN clustering is useful for our requirement because it can make arbitrary number of spatial clusters, and we can discard insignificant potential hotspot clusters so that false positive is reduced.

DBSCAN algorithm has two parameters to control. One is *minPts* (minimum points) and the other is reachable range. DBSCAN clustering accepts clusters if there are more than *minPts* points within reachable range. Especially, for our work, the *minPts* parameter plays a role of the threshold setting.

Figure 7 describes the example of a DBSCAN clustering result. Two clusters are reported as hotspots (circle), while one non-significant cluster is ignored (dashed circle).

3. Experiment Results

Our proposed method is implemented with an i7 CPU, a 64 GB main memory and a GeForce TITAN X GPU. We use Theano library [12] to train and test the CNN model. The proposed method was evaluated using industrial benchmarks titled ICCAD2012 CAD dataset [8] described in Table 1. These 32 nm, 28 nm datasets has around 500-5000 clips whose size is around 5 μm square. Each clip in the train and test layouts includes either HS (hotspot) or NHS (non-hotspot) core region which indicates the location of hotspot or non-hotspot. In the training step, the HS and NHS core regions indicates the location where the hotspot and non-hotspot training samples are. Also, the HS core region indicates the true hotspot locations for the hotspot detection step.

Table 2. The network topology of our CNN model

Case.1	CP1	CP2	CP3	CP4	FC1	FC2
# maps	22	40	60	80	100	2
Input size	76	36	16	6		
Filter size	5	5	5	5		
Max Pool	2	2	2	2		
Output size	36	16	6	1		
Activation	reLU	reLU	reLU	reLU	Tanh	Tanh

The contest dataset also includes the reference verifier which indicates the performance of the result. The verifier checks whether the detected spots in the test layout are located near the given HS core or not. If the spots are near given hotspot core region with the distance of 1.2 μm, they become *Hit*, else they will be *Extra*. For example, the rectangle in the center of Figure 7 is a hotspot core region. Since there is one detected hotspot near the HS core region and the other far from the core region, we have one *Hit* and one *Extra* from the Figure 7.

We convert the number of *Hit* and *Extra* (#Hit / #Extra) from the result of the reference verifier to precision Eq. (1) and recall Eq. (2) which is commonly used in evaluation criteria for image detection task. The #*HS* means the number of given hotspot. If more given hotspots are identified by hotspot detector, higher recall will be reported. Likewise, if the detected points are more likely to be Hit, the precision will be increased. Thus higher precision and higher recall can be represented as superiority of the detection method.

$$precision = \frac{\#Hit}{\#Hit + \#Extra}, \tag{1}$$

$$recall = \frac{\#Hit}{\#HS}. \tag{2}$$

3.1 Network Topology Selection

We choose the topology of CNN through the evaluation of detection performance. The precision-recall curve regarding the stacks of convolution layers are illustrated in Figure 8. The curve of our results is derived by sweeping the threshold value (*minPts* of DBSCAN). We chose 4CONV-pool structure which

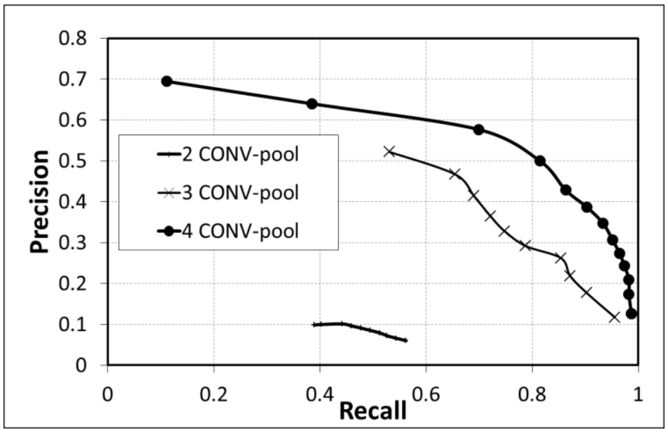

Figure 8. Detection performance with respect to network structure.

made best performance in Benchamrk1 dataset. The detailed network topology is described in Table 2.

3.2 Performance Evaluation

We discuss our results of all test cases and compare it with other methods. The methods proposed by B. Yu et al. [4] and Y.-T. Yu et al. [5] employed multiple SVM classifiers assisted their novel layout encoding methods. Though these SVM-based methods achieved fairly high recall value, it is suffered from a relatively low precision which is caused by high *Extra* (false positives).

Though our approach uses very simple framework with just one classifier, our result outperforms the other results in all category in Table 3. Our high detection performance is mainly due to the high performance of CNN classifiers and extra reduction skills which includes DBSCAN clustering in Section 2.5 and candidate region method in Section 2.4. The detection threshold (*minPts* of DBSCAN clustering) is set to 7 for all Benchmarks.

Figure 9 shows the example of *Hit, Missing*, *Extra* patterns. The dashed lines placed on the centers of images are the location of true hotspot, and the red rectangles are the detected hotspot locations by our method. The *Hit* in Figure 9(a) means the detected points are located near hotspot location. The *Extra* in Figure 9(b) indicate that the detected points are not near the true hotspot position. The *Missing* in Figure 9(c) show that true hotspots are failed to be detected.

3.3 Detection Time

Though we increased the detection performance using CNN model, its heavy model structure is still remained to disadvantage. Though GPU computation and other runtime reduction methods are applied, it still takes heavy time to inspect the layout. As shown in Table 4, it has normalized time is around 25-60 GPU h/mm^2. Though our method is faster than optical verification method whose runtime is over 100 CPU h/mm^2, it does not meet the of ICCAD2012 contest requirement (1-10 CPU h/mm^2) [8]. But, we can see the normalized runtime is proportional to the candidate region ratio (CR ratio). Thus we can see the candidate region is efficient for detection time reduction. We will further conduct the research for detection runtime

Table 3. Detailed performance comparison

Test layout	Methods	Recall	Precision
Benchmark1	B. Yu [4]	0.810	0.202
	Y.-T. Yu [5]	0.947	0.125
	Ours	**0.951**	**0.306**
Benchmark2	B. Yu [4]	0.811	0.039
	Y.-T. Yu [5]	0.982	0.040
	Ours	**0.995**	**0.190**
Benchmark3	B. Yu [4]	0.909	0.089
	Y.-T. Yu [5]	0.919	0.109
	Ours	**0.985**	**0.138**
Benchmark4	B. Yu [4]	0.870	0.054
	Y.-T. Yu [5]	0.859	0.043
	Ours	**0.989**	**0.078**
Benchmark5	B. Yu [4]	0.805	0.047
	Y.-T. Yu [5]	0.929	0.031
	Ours	**0.976**	**0.068**
Average	B. Yu [4]	0.841	0.086
	Y.-T. Yu [5]	0.927	0.070
	Ours	**0.979**	**0.156**

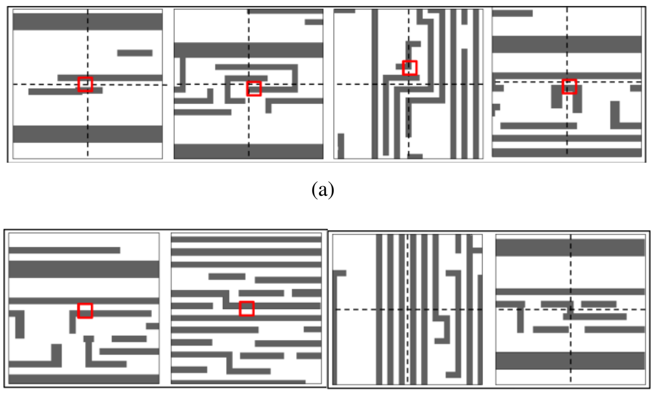

Figure 9. Example of Hit (a), Extra (b), and Missing (c).

Table 4. Hotspot detection runtime

Test layout	# of Blocks	CRatic (%)	Area (mm^2)	Test time (min)	Normalized time (h)/mm^2
Benchmark1	541	16.6	0.0086	13	25.19
Benchmark2	4645	44.1	0.0743	224	50.25
Benchmark3	5282	63.4	0.0845	302	59.57
Benchmark4	3559	30.7	0.0569	121	35.44
Benchmark5	2152	33.8	0.0344	76	36.82

reduction.

4. Conclusion

We introduced an accurate hotspot detection framework using convolutional neural network. It was combined by the powerful classification performance of CNN, training data augmentation, candidate region usage, and DBSCAN clustering, to make our work better. Through these technical efforts, we were able to outperform previous SVM based hotspot detection framework.

References

[1] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, “Efficient process-hotspot detection using range pattern matching,” in *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, 2006, pp. 625-632. http://dx.doi.org/10.1145/1233501.1233630

[2] Y. T. Yu, Y. C. Chan, S. Sinha, I. H. R. Jiang, and C. Chiang, “Accurate process-hotspot detection using critical design rule extraction,” in *Proceedings of the 49th Annual Design Automation Conference*, San Francisco, CA, 2012, pp. 1167-1172. http://dx.doi.org/10.1145/2228360.2228576

[3] A. B. Kahng, C. H. Park, and X. Xu, “Fast dual-graph-based hotspot filtering,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1635-1642, 2008. http://dx.doi.org/10.1109/TCAD.2008.927765

[4] 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,” *Journal of Micro/Nanolithography, MEMS, and MOEMS*, vol. 14, no. 1, pp. 1-12, 2014. http://dx.doi.org/10.1117/1.JMM.14.1.011003

[5] Y. T. Yu, G. H. Lin, I. H. R. Jiang, and C. Chiang, “Machine-learning-based hotspot detection using topological classification and critical feature extraction,” in *Proceedings of the 50th Annual Design Automation Conference*, Austin, TX, 2013, pp. 671-676. http://dx.doi.org/10.1145/2463209.2488816

[6] J. Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, “Rapid layout pattern classification,” in *Proceedings of the 2011 16th Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2011, pp. 781-786. http://dx.doi.org/10.1109/ASPDAC.2011.5722295

[7] 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 *Proceedings of Society of Photo-Optical Instrumentation Engineers (vol. 9427)*, San Jose, CA, 2015, pp. 1-11. http://dx.doi.org/10.1117/12.2085790

[8] J. A. Torres, “ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite,” in *Proceedings of 2012 International Conference on Computer-Aided Design*, San Jose, CA, 2012, pp. 349-350. http://dx.doi.org/10.1145/2429384.2429457

[9] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Mitosis detection in breast cancer histology images with deep neural networks,” in *Proceedings of 2013 Medical Image Computing and Computer-Assisted Intervention*, Nagoya, Japan, 2013, pp. 411-418. http://dx.doi.org/10.1007/978-3-642-40763-5_51

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of 2012 Neural Information Processing Systems*, Lake Tahoe, Canada, 2012, pp. 1-9.

[11] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996, pp. 226-231.

[12] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: a CPU and GPU math compiler in Python,” in *Proceedings of the 9th Python in Science Conference*, Austin, TX, 2010, pp. 1-7.

Moojoon Shin received B.S. degrees Kyungpook National University in 2009. He has worked in the Semiconductor Foundry Center of Samsung Electronics as an engineer in the computational lithography field since 2009. Since 2015, he has been on leave from the company to pursue M.S. degree in the department of Semiconductor Engineering of Sungkyunkwan University. His interest is deep learning and machine-learning application to computational lithography.

Jee-Hyong Lee received his B.S., M.S. and Ph.D. in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), in 1993, 1995 and 1999, respectively. He joined Sungkyunkwan University, Korea in 2002 and is currently a Professor in the Department of Computer Science and Engineering. His research interests include machine learning, data mining and pattern analysis.