Journal of
Micro/Nanolithography, MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Lithography hotspot detection using a double inception module architecture

Jing Chen
Yibo Lin
Yufeng Guo
Maolin Zhang
Mohamed Baker Alawieh
David Z. Pan

Jing Chen, Yibo Lin, Yufeng Guo, Maolin Zhang, Mohamed Baker Alawieh, David Z. Pan, "Lithography hotspot detection using a double inception module architecture," *J. Micro/Nanolith. MEMS MOEMS* **18**(1), 013507 (2019), doi: 10.1117/1.JMM.18.1.013507.

Lithography hotspot detection using a double inception module architecture

Jing Chen,[a,b,c] Yibo Lin,[c] Yufeng Guo,[a,b,*] Maolin Zhang,[a,b] Mohamed Baker Alawieh,[c] and David Z. Pan[c,*]
[a]Nanjing University of Posts and Telecommunications, College of Electronic and Optical Engineering and College of Microelectronics, Jiangsu, China
[b]Nanjing University of Posts and Telecommunications, National and Local Joint Engineering Laboratory for RF Integration and Micro-Packaging Technologies, Jiangsu, China
[c]University of Texas at Austin, Department of Electrical and Computer Engineering, Austin, Texas, United States

Abstract. With the shrinking feature sizes of semiconductor devices, manufacturing challenges increase dramatically. Among these challenges, lithography hotspot stands out as a prominent ramification of the growing gap between design and manufacturing. Practically, a hotspot refers to the failure in printing desired patterns in lithography. As lithography hotspots have significant impacts on manufacturing yield, the detection of hotspots in the early design stage is desired to achieve fast design closure. We propose a lithography hotspot detection framework using a double inception module structure. This structure performs better in both accuracy and false alarms by widening the conventional stacked structure to benefit feature extraction and using global average pooling to keep the spatial information. Experimental results show that the proposed structure achieves better performance than existing methods. © *2019 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JMM.18.1.013507]

Keywords: lithography hotspot detection; convolution neural network; inception module; global average pooling.

Paper 19001 received Jan. 8, 2019; accepted for publication Mar. 5, 2019; published online Mar. 22, 2019.

1 Introductions

As the feature sizes keep shrinking, the gap between design and manufacturing continues to increase.[1] Lithography hotspot is one of the major issues in the lithography process from the shape distortion of printed patterns. Such distortion may not only cause performance degradation but also malfunction due to potential open/short connections. Thus, lithography hotspot is critical to the manufacturing yield of chips, and early prediction of hotspots is desired to ensure manufacturability and speed up design closure.

In literature, two approaches have been proposed to address the hotspot detection problem. One uses lithography simulation,[2–4] which leverages lithography models to simulate the contour of the circuit. This method has high accuracy; however, it requires heavy computational resources. The other refers to geometric methods, which can be further categorized into pattern matching and machine learning-based approaches.

Pattern matching[5–9] detects hotspots by comparing a pattern with known hotspot patterns in a library. Yao et al.[5] presented this by specifying process-hotspots as a library of range patterns, eventually, hotspots can be found in a matter of minutes. Yu et al.[8] expressed the topology features of the hotspot patterns by extracting their key design rules and then adopted a two-stage filtering process to locate all hotspots. Hotspots can be detected accurately and effectively with pattern matching, but the limitation is its incapability of detecting unseen hotspots.

Recently, machine learning-based detection[10–15] was proposed, motivated by the outbreak of big data and improved computer hardware performance. A machine learning model is built by learning the discriminant boundaries between hotspots and nonhotspots in the training data, then it generalizes to the testing data. Both conventional learning approaches and deep learning ones have been proposed. Yu et al.[11] proposed an accurate hotspot detection approach based on principal component analysis for feature dimension reduction and support vector machine (SVM) for classification. They reported over 80% accuracy on the testing layouts. Yu et al.[13] improved the accuracy by combining topological classification and critical feature extraction with SVM.

Moreover, fuzzy matching was proposed to integrate the advantages of machine learning and pattern matching.[16,17] Different from setting a middle decision boundary between hotspots and nonhotspots, fuzzy matching extends the decision boundary by taking advantage of pattern matching. A new fuzzy matching model was proposed by Lin et al.,[16] which could dynamically adjust the known hotspots around the fuzzy region. Later, their team[17] added a grid reduction technique to this fuzzy matching model and achieved an average accuracy of 94.5% while reducing CPU run time. However, as a matter of fact, although these methods are precise and efficient, they are composed of complex feature extraction and layout coding steps, which may be too complex in operation.

The performance of these conventional machine learning approaches relies heavily on feature engineering. Since a layout clip can be represented with an image, deep learning[18–20] was then proposed to extract features automatically. In particular, convolution neural networks (CNNs) have gained lots of attention in image classification problems.[21–26] It can automatically extract features from the training set through the convolution-pooling layers, and then build a model by

*Address all correspondence to Yufeng Guo, E-mail: yfguo@njupt.edu.cn; David Z. Pan, E-mail: dpan@ece.utexas.edu

1932-5150/2019/$25.00 © 2019 SPIE

minimizing the loss between the predicted results and labels. Shin and Lee[22] first used CNN-based lithography hotspot detection and achieved about 95.5% accuracy. Then, Yang et al.[24] also studied a deeper structure. Later, they used feature tensor generation to extract representative layout features and applied biased learning into a simple structure,[26] which achieved higher detection accuracy. However, almost all existing CNNs for lithography hotspot detection adopted stacked alternative convolution and pooling layer structures.

In this paper, we develop a CNN-based hotspot detection framework with double inception modules. The key idea is to widen the network to extract more comprehensive features while applying global average pooling for classification; an approach that demonstrates superior results.

The main contributions of this paper are summarized as follows:

- We first propose a CNN-based architecture using double inception modules for feature extraction and global average pooling for classification.
- Experimental results show that our method achieves 97.77% average hotspot detection accuracy, with 19.52% reducing in average false alarms compared with the state-of-the-art results[26] on ICCAD 2012 contest benchmarks.[27]

The rest of the paper is organized as follows. Section 2 reviews the basic background of CNNs and introduces the lithography hotspot detection problem. Section 3 gives the whole architectures and analyzes its composition. Section 4 validates the proposed network structure with the experimental results, and Sec. 5 finally concludes the paper.

2 Preliminaries and Problem Formulation

In this section, we first introduce the detailed structure functions of CNNs. Then, the background of lithography hotspot detection is presented, followed by problem formulation.

2.1 *Convolution Neural Networks*

CNNs usually consist of two parts: the feature extractor and the classifier. The feature extractor is usually composed of convolution layers and pooling layers. Typically, robustness and uniqueness can be used to evaluate the feature extractor. Robustness can be defined as the ability of a system to resist change. The closer the similar samples are distributed in the feature space, the higher the robustness. On the other hand, uniqueness can be defined as the ability to differentiate between different samples. The more the distribution of different samples overlaps in the feature space, the poorer the uniqueness. Mathematically, the convolution operation can be expressed as

$$a_{i,j} = f\left(\sum_{d=0}^{D-1}\sum_{m=0}^{F-1}\sum_{n=0}^{F-1} w_{d,m,n}x_{d,i+m,j+n} + w_b\right),$$

where $x_{d,i,j}$ represents the i'th row and j'th column pixel at the (i, j) coordinates of d'th channel of an image, $w_{d,m,n}$ refers to the weight of the m'th row and the n'th column of d'th feature map, w_b represents the bias term of the filter, f is the activation function, and $a_{i,j}$ stands for the i'th row and j'th column element of the feature map with D channels.

Since the convolution layer is characterized by a linear convolutional operation to input followed by an activation function to obtain feature maps, it can get the features of input which will be beneficial to the uniqueness. On the other hand, the pooling layer can increase robustness by reducing the error during feature extraction. Thus, reasonably combining the convolution layer and pooling layer plays a significant role in improving the quality of extracted features.

Then, the fully connected (FC) layer acts as the classifier in CNNs by mapping the learned distributed feature representation to the sample mark space. After automatically obtaining the feature maps through the convolution-pooling layers, these feature maps are flattened into one-dimension before being fed to the FC layers. The first FC layer can be converted into a 1×1 convolution kernel with a global convolution with size equal to the output of previous convolution layers. This indeed contributes most of the parameters of the CNNs, which will increase the probability of overfitting as well. Meanwhile, the flattening operation can compromise the spatial information, which is important for vision-related tasks. In our paper, we apply global average pooling in our architecture to enhance the classifier. Further elaboration will be displayed in Sec. 3.3.

2.2 *Lithography Hotspot Detection*

The designed mask pattern is generally transferred to the silicon wafer by the lithography process. However, distortion tends to appear in the manufacturing layout due to the gap between the device size and the lithography wavelength. The layout patterns with problems after manufacturing such as bridging or breaking, which may lead to circuit failure, are called lithography hotspot. Therefore, hotspots need to be located before mask tape-out. Figure 1 shows a schematic diagram of a hotspot and a nonhotspot. The red core part of Fig. 1(a) represents a hotspot region.

2.3 *Problem Formulation*

Lithography hotspot detection can be analogous to binary classification, where the label represents hotspot or nonhotspot. During the training process, the model is built by minimizing the error between the predicted results and known labels. And during the testing process, the trained model will be employed to detect the hotspot patterns. We compare

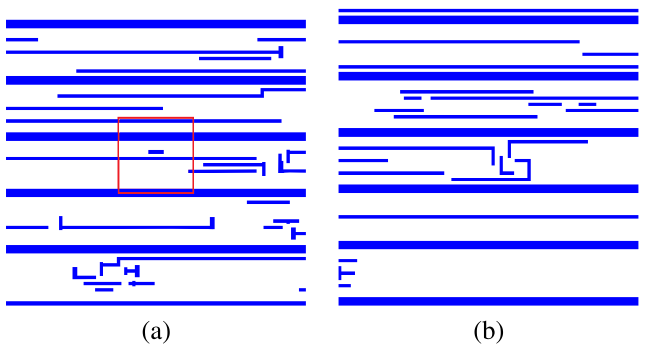

Fig. 1 (a) Hotspot and (b) nonhotspot layout clips.

the predicted results with known labels to evaluate the performance of the model.

For binary classification problems, there are usually four situations, namely true positive (TP), true negative (TN), false positive (FP), and false negative (FN). A TP (FN) is the case where an actual hotspot is predicted as a hotspot (nonhotspot). A TN (FP) is the case where an actual nonhotspot is predicted as a nonhotspot (hotspot). Usually, we adopted the following metrics to evaluate the performance.

Definition 1 (Accuracy): The ratio between the number of correctly predicted hotspots and the total number of actual hotspots:[24]

$$\text{accuracy} = \frac{\sum \text{TP}}{\sum(\text{TP} + \text{FN})}.$$

Definition 2 (False Alarm): The number of actual nonhotspot clips that are predicted as hotspots by the model:[24]

$$\text{false alarm} = \sum \text{FP}.$$

Meanwhile, we formulate the lithography hotspot detection problem as follows:

Problem 1 (Hotspot Detection Problem): Training a model using a training set of labeled hotspot and nonhotspot data samples, such that the accuracy can be maximized and the number of false alarms can be minimized in the testing dataset.

3 Inception Module Framework

In convolutional networks, convolutional layers for feature extraction are critical to model performance, so we introduce inception modules[28,29] with parallel convolutional kernels to extract different views to the features. In this section, we first exhibit the whole double inception module architecture. Then, we detail the benefits of the inception module for the feature extraction. Finally, the advantages of applying global average pooling[30] are analyzed.

3.1 *Double Inception Module Structure*

Figure 2 shows the proposed structure. The network structure is mainly divided into two parts: feature extraction and classification. Feature extraction is primarily based on the double inception modules. The classifier mainly consists of the global average pooling and the linear categories FC layer followed by a softmax function. Deep feature representation is attained after the feature extraction of double inception modules and gathered by 1×1 convolution layer, and then fed into global average pooling. Next, the one-dimensional output of global average pooling is inputted to the linear FC layer which has two outputs representing hotspot and nonhotspot. Meanwhile, Fig. 2 also shows the relevant configurations of its structure, such as kernel size, channels, stride, and padding type. And the blue numbers refer to the output size of the corresponding layer.

3.2 *Feature Extraction*

The feature extraction part is basically composed of inception module which can be viewed as a split-transform-merge

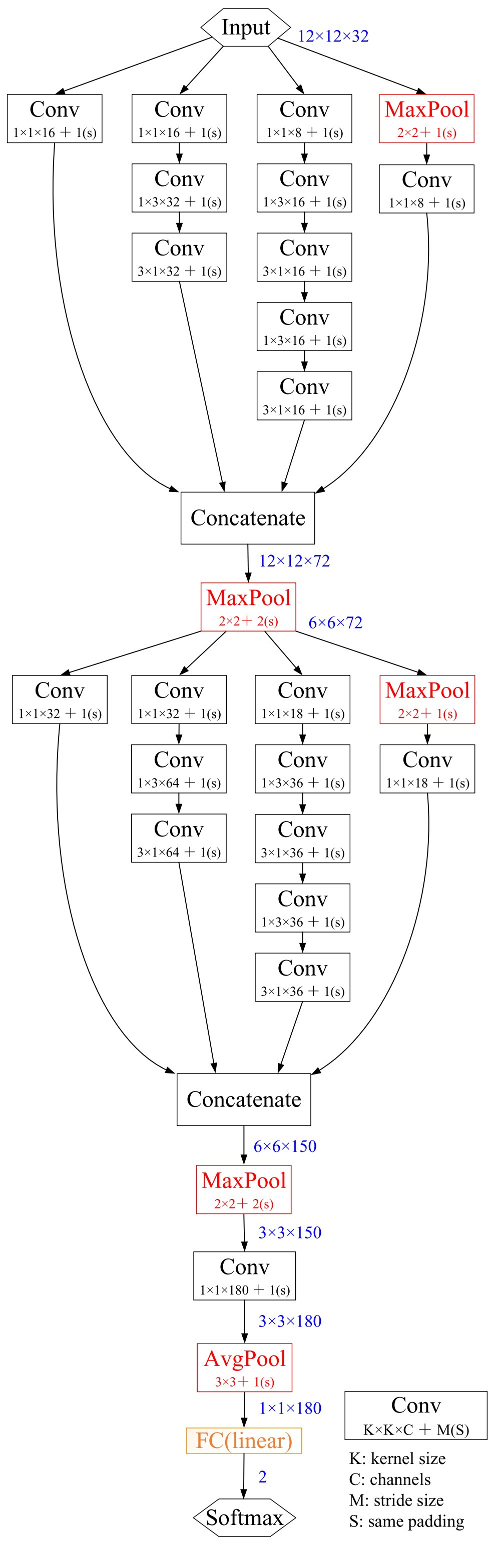

Fig. 2 Double inception module structure with all bells and whistles.

process. Figure 3 shows the concrete split-transform-merge operation principle of the inception module. First step is to apply different convolution and pooling operation to the input, respectively, and the input is split into four branches. Then, four types of feature maps encoded with different colors in Fig. 3 will be obtained after the transforming operation of convolution-pooling layers. At last, the feature maps are

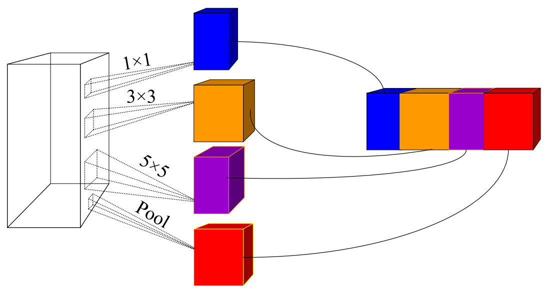

Fig. 3 The split-transform-merge operation principle of inception module.

merged into a single output vector, which represents the input of the next stage by adding the channels together.

In the inception module, as the branches get deeper, the receptive field of the feature maps increases. To elaborate on this, we consider the representation shown in Fig. 4. The blue solid line boxes refer to the input and the red dotted line boxes represent the convolution filters. As shown in Fig. 4(b), for an input region of 3×3, a 1×3 convolution can be used to convolute the 3×3 region first, then another 3×1 convolution can be used to convolute the result of the first 1×3 convolution. The result of this operation is analogous to only using a 3×3 convolution as shown in Fig. 4(a). Similarity, the 5×5 convolution is replaced by two 3×3 convolution layers as shown in Figs. 4(c) and 4(d). Therefore, the second branch in Fig. 2 counting from left to right is analogous to 3×3 convolution while the third branch is analogous to 5×5 convolution. Also, it is undeniable that the nonlinearity of the structure has increased, since the structure has developed into a deeper one with more convolution layers with each convolution layer accompanied by an activation function.

The utilization of different convolution layers and pooling layer will enhance the adaptability of the network to the input feature scale. The pixels in the input extracted region are adjacent to each other such that the correlation between them is very strong, and the parameters of the larger convolution kernel are also strongly related. If the category features of the input image are scattered widely, large convolution kernels will extract their features better. On the contrary, small convolution kernels can achieve effective feature extraction while feature distribution takes a small portion of the image. Therefore, such a wide structure can choose better features since the dense features can be learned by the shallow convolution layers and the sparse features can be obtained by the deeper convolution layers.

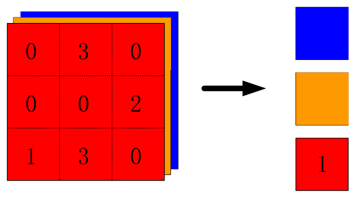

Fig. 5 Schematic diagram of global average pooling.

In summary, in most stacked structures, e.g., Refs. 22, 24, and 26, there is only one convolution kernel on the single layer. Inception module not only increases the width of the convolution layer but also uses convolution kernels of different scales on the single layer, so that convolution kernels of different sizes can extract features of different sizes, and the feature extraction ability of a single layer is enhanced.

3.3 *Classifier*

After the feature extraction of the inception module, the classifier uses the feature maps to judge upon the class label. We directly perform global average pooling[30] after the last feature maps as the classifier. Figure 5 shows the process of global average pooling by taking the average of each feature map. Three advantages come along with such operation. To begin with, the spatial information is more robust since the data put into the categories layer are directly summed out from the last feature layer. In addition, the feature maps, after global average pooling, can be claimed as categories confidence maps for it straightly reflects the correspondence between feature maps and the categories rather than training by a number of parameters of FC layer. Furthermore, the FC layer requires a large number of training parameters, which tends to result in overfitting. While global average pooling does not need any training parameters, a fact that can help avoid overfitting.

4 Experimental Results

In this section, we mainly present the experimental results that demonstrate the efficiency of our proposed approach. First, we introduce the training and testing benchmarks and the detailed configurations of our training process. Second, we compare our results with different neural network

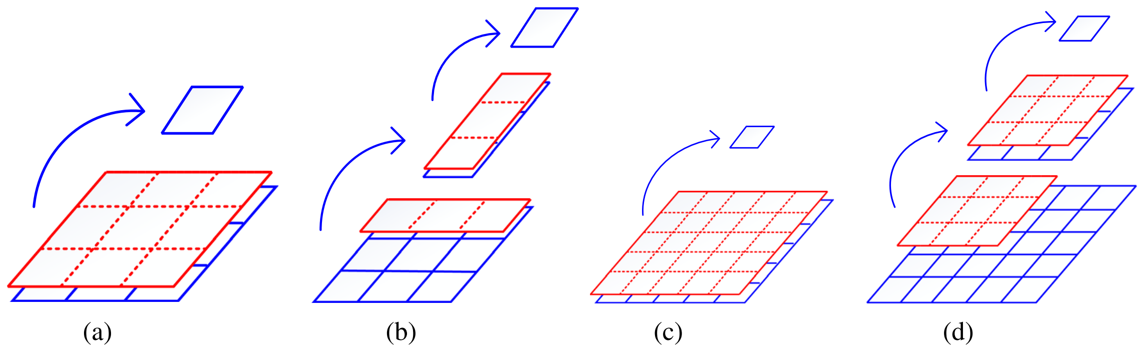

Fig. 4 Schematic diagram of feature extraction calculation. (a) 3×3 input convoluted by a 3×3 filter. (b) 3×3 input convoluted by 1×3 and 3×1 filters. (c) 5×5 input convoluted by a 5×5 filter. (d) 5×5 input convoluted by two 3×3 filters.

architectures in lithography hotspot detection. Third, we show the receiver operating characteristic (ROC) curves of our model and the baseline. Then, we show the feature visualization of various convolution layers. Next, the effects of the inception module number on the detection performance are presented. Finally, the results for the application of global average pooling on the performance are displayed as well. Our proposed method is implemented in Python 2.7 on a Linux server with 8-core 3.4 GHz CPU, Nvidia GTX 1080 GPU, and 32 GB RAM. We use tensorflow library[31] to train and test the double inception module structure.

4.1 *Benchmark Information and Training Configurations*

The dataset we employed is from Ref. 26 whose dataset is processed after discrete cosine transform (DCT) from ICCAD2012 CAD contest data.[27] Each benchmark set is composed of hundreds or thousands of clips, and each input size is 12×12 with 32 channels. Table 1 lists the details of the benchmarks. It consists of four sets of 28-nm benchmarks. Columns "#HS" and "#NHS" show the total number of hotspots and nonhotspots in training and testing sets. Also, to deal with imbalanced training data, we adopt the biased learning strategy as paper[26] to adjust the training ground truth to offer trades-off between accuracy and false alarms in our structure.

Table 2 shows the related detailed configurations of our training process, such as optimizer, learning rate, bath size, and so on. Adam[32] is used as the gradient descent optimizer for training. We start with a learning rate of 0.001, then gradually decrease it by a factor of 0.5 after 3000 training sessions so that the update is small when it comes close to the optimal solution. Also, we apply a dropout ratio of 0.5 in our classifier layer after global average pooling to avoid overfitting.[33]

Table 1 ICCAD2012 28 nm benchmarks.

	Train		Test	
Dataset	#HS	#NHS	#HS	#NHS
Benchmark 2	174	5285	498	41,298
Benchmark 3	909	4643	1808	46,333
Benchmark 4	95	4452	177	31,890
Benchmark 5	26	2716	41	19,327

Table 2 Training configurations.

Training configurations	Value
Optimizer	Adam[32]
Initial learning rate	0.001
Learning rate decay	0.5
Learning rate decay step	3000
Batch size	32
Dropout ratio	0.5

4.2 *Performance Evaluation*

In recent years, there have been many studies on hotspot detection. We compare our framework with Shin[22] and Yang[24,26] whose papers proposed different neural network architectures for hotspot classification problem. Table 3 shows the comparison results under 28-nm testing benchmarks. Note that the results of Refs. 22 and 24 are collected from their paper, the results of Ref. 26 are showed in GitHub, and our results are illustrated with means and standard deviations from 10 different runs with different random seeds.

For dataset preprocessing, Shin and Lee[22] encoded the layout clips into gray-scale density images as the input and Yang et al.[24] just adopted original layout clip image as the input. We use the dataset after DCT preprocessing as Ref. 26, which is compatible with the CNNs and helps saving the forward propagation time.

For the neural network architectures, the structure of Ref. 22 consists of two components: (1) four alternating 1 convolution-1 pooling layers for feature extraction and (2) two FC layers for classification. The two major components of Ref. 24 are (1) two convolution layers, then four alternating 3 convolutions-1 pooling for feature extraction, and (2) three FC layers for classification. The Ref. 26 structure consists of two aspects: (1) two alternating 2 convolution-1 pooling layers for feature extraction and (2) two FC layers for classification. In brief, we can notice that all of them use stacked structures for feature extraction. These kinds of structures just increase the depth of the convolution layer and only apply one kind of convolution kernel on one single direction, for example, the convolution kernel size of Ref. 24 is only 3×3. Hence, the function of such feature extraction may be weak. In addition, all of those architectures choose FC layers for classification which may lose some spatial information. Our structure uses inception modules which is similar to a parallel structure, extracting features from different perspectives. Section 4.4 details the layer visualization of different branches. Also, we apply global average pooling for classification. Section 4.6 shows the experimental results of using global average pooling compared with FC layers. Among those testing results, our architecture achieves best accuracies in benchmarks 2, 4, and 5. In addition, the highest average detection accuracy is obtained by our structure. Our architecture has approximately an average of 989 false alarm value which is similar to the best in Ref. 24, but the average accuracy is 1.2% higher. Our approach has an average 104s CPU inference time advantage which is comparable to Ref. 26 and better than both Refs. 22 and 24.

4.3 *Receiver Operating Characteristic*

ROC curve can be employed to evaluate the binary classifier. It is a comparison of TP ratio and FP ratio as the criterion changes. Figure 6 shows the ROC curves under different benchmarks. Note that the blue curves are from our model and the red ones are from stacked CNN.[26] We can observe that our model can achieve better characteristics. Table 4 gives the testing accuracies of four benchmarks if we set

Table 3 Comparison on different neural network architectures.

	JM3'16[22]			JM3'17[24]			TCAD'18[26]			Ours		
Neural network architectures	Accuracy (%)	False alarm	CPU (s)	Accuracy (%)	False alarm	CPU (s)	Accuracy (%)	False alarm	CPU (s)	Accuracy (%)	False alarm	CPU (s)
Benchmark 2	98.8	1790	208	98.7	83	501	98.90	473	117	**99.21**	352	123
Benchmark 3	97.5	7077	321	98	3108	546	**98.67**	3881	136	98.22	**2735**	145
Benchmark 4	93.8	892	129	94.5	**296**	346	94.35	467	88	**96.32**	789	92
Benchmark 5	92.7	172	81	95.1	394	264	95.12	95	57	**97.32**	**80**	56
Average	95.7	2483	185	96.57	971	415	96.76	1229	100	**97.77**	989	104

Note: The bold characters show the best result under the same metric.

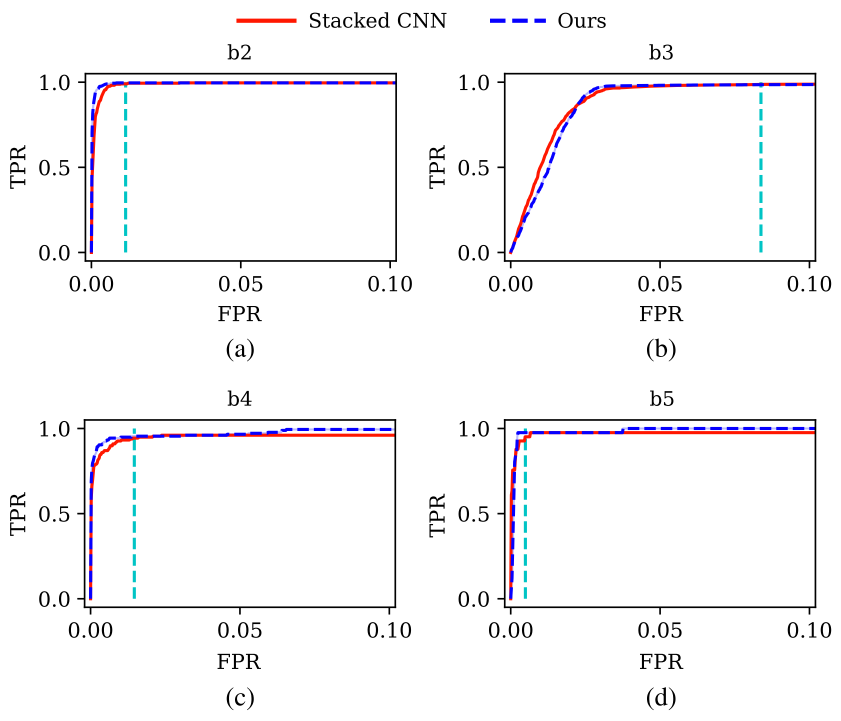

Fig. 6 ROCs of different benchmarks. (a) Benchmark 2. (b) Benchmark 3. (c) Benchmark 4. (d) Benchmark 5.

Table 4 Comparison on testing accuracy under same false alarms.

	Benchmark 2	Benchmark 3	Benchmark 4	Benchmark 5
Structures	Accuracy (%)	Accuracy (%)	Accuracy (%)	Accuracy (%)
Stacked CNN[26]	98.90	98.67	94.35	95.12
Ours	99.59	98.61	95.48	97.56

the same criterion that the false alarm values are same as baseline. We can notice that the testing accuracy is increased by 0.69% for benchmark 2, 1.13% for benchmark 4, and 2.44% for benchmark 5, while it is decreased by 0.06% for benchmark 3.

4.4 *Feature Maps from Different Inception Branches*

Figure 7 shows the output feature maps from different branches of the first inception module given the same input. Figure 7(a) shows the original layout clip. The feature maps from different branches are very diverse with different perspectives of emphasis on the input layout clip. For example, Fig. 7(b) mostly captures the horizontal lines. Figure 7(c) shows more dots, while the horizontal lines are blurry. With all these feature maps concatenated, the input can be described in a holistic way, which turns out to be beneficial for the classification task.

4.5 *Best Number of Inception Modules*

In this section, lithography hotspot detection performance is evaluated when the number of inception modules varies. We evaluate the performance of networks with one, two, and

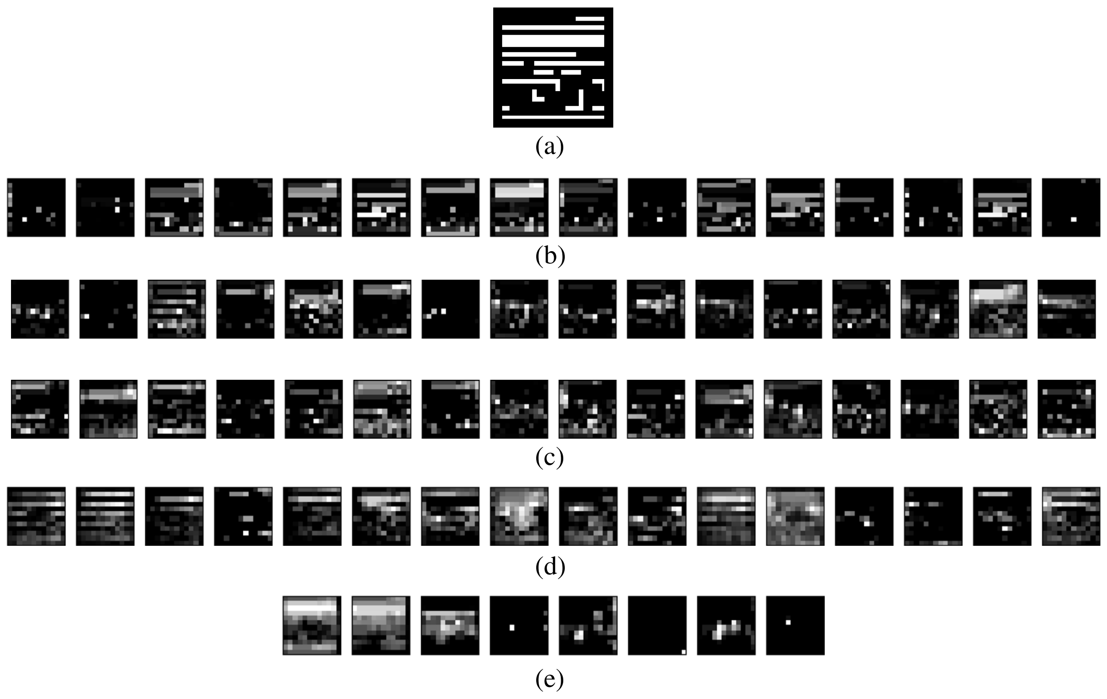

Fig. 7 Feature maps from different branches of the first inception module. (a) Input layout clip. (b) Feature maps output for branch 1. (c) Feature maps output for branch 2. (d) Feature maps output for branch 3. (e) Feature maps output for branch 4. Branches are counted from left to right as that in Fig. 2.

Table 5 Experimental results using different number of inception modules.

	Benchmark 2		Benchmark 3		Benchmark 4		Benchmark 5		Average	
Number of inception modules	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm
One	99.21	414	98.28	3833	96.78	2198	97.56	117	97.96	1641
Two	99.21	352	98.22	2735	96.32	789	97.32	80	97.77	989
Three	98.59	186	98.10	2396	94.86	379	96.58	72	97.03	771

three inception modules, while the other configurations remain the same as that in Fig. 2. Table 5 shows the results under different number of inception modules. We can observe that using one inception module can achieve the highest accuracy at the cost of a higher false alarm. The accuracy drops by 0.19% with two inception modules, while the number of false alarms is reduced by around 39.73%. Meanwhile, using three inception modules further reduces the false alarm value about 22.04%, but the accuracy is 0.74% lower compared with that using two inception modules. Actually, this is trade off between accuracy and false alarm. The false alarm can be reduced with reduced accuracy by adjusting the threshold of the bias. Consider that fewer inception modules also decrease the depth and parameters of

Table 6 Experimental results of using global average pooling compared with FC layer.

	Benchmark 2		Benchmark 3		Benchmark 4		Benchmark 5		Average	
Classifier	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm	Accuracy (%)	False alarm
FC layer	98.99	389	98.27	2907	95.14	555	95.36	106	96.94	990
Global average pooling	99.21	352	98.22	2735	96.32	789	97.32	80	**97.77**	**989**

the network, which usually enables faster and easier training. Therefore, we apply two inception modules in our architecture and call it "double inception module architecture."

4.6 *Impacts of Global Average Pooling*

As stated in Sec. 3.3, global average pooling does a better job in classification compared to the FC layer for it contains more spatial information and does not require any training parameters. Table 6 compares the experimental results between applying global average pooling and FC layer after feature extraction. One can notice that the results are almost similar in terms of false alarms, but the accuracy of using global average pooling is 0.83% higher than the FC layer, which indicates that the application of global average pooling indeed outperforms the FC layer in our structure. Hence, we adopt global average pooling in our architecture.

5 Conclusion

In this paper, we propose a lithography hotspot detection framework with double inception modules. We construct the feature extractor by two inception modules which can get features from different kernel sizes. Meanwhile, the global average pooling is applied to obtain more spatial information and avoid overfitting. The experimental results show that our method outperforms the state-of-the-art detection methods and provides another framework for lithography hotspot detection.

Acknowledgments

This work is supported in part by the US National Science Foundation (Grant No. 1718570) and the National Natural Science Foundation of China (Grant Nos. 61574081 and 61874059).

References

1. T. Jhaveri et al., "Maximization of layout printability/manufacturability by extreme layout regularity," *Proc. SPIE* **6156**, 615609 (2006).
2. E. Roseboom et al., "Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs," *Proc. SPIE* **6521**, 65210C (2007).
3. T. Mitsuhashi, "Method and system for lithography simulation and measurement of critical dimensions with improved cd marker generation and placement," US Patent 8,364,452 (2013).
4. M. C. Simmons et al., "A state-of-the-art hotspot recognition system for full chip verification with lithographic simulation," *Proc. SPIE* **7974**, 79740M (2011).
5. H. Yao et al., "Efficient process-hotspot detection using range pattern matching," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, ACM, pp. 625–632 (2006).
6. B. Yu et al., "Machine learning and pattern matching in physical design," in *20th Asia and South Pac. Des. Autom. Conf. (ASP-DAC)*, IEEE, pp. 286–293 (2015).
7. W. Fan, X. Wang, and Y. Wu, "Incremental graph pattern matching," *ACM Trans. Database Syst.* **38**(3), 1–47 (2013).
8. Y.-T. Yu et al., "Accurate process-hotspot detection using critical design rule extraction," in *Proc. 49th Annu. Des. Autom. Conf.*, ACM, pp. 1167–1172 (2012).
9. J.-Y. Wuu et al., "Rapid layout pattern classification," in *Proc. 16th Asia and South Pac. Des. Autom. Conf.*, IEEE Press, pp. 781–786 (2011).
10. T. Matsunawa et al., "A new lithography hotspot detection framework based on adaboost classifier and simplified feature extraction," *Proc. SPIE* **9427**, 94270S (2015).
11. B. Yu et al., "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *J. Micro/Nanolithogr. MEMS MOEMS* **14**(1), 011003 (2014).
12. J.-R. Gao, B. Yu, and D. Z. Pan, "Accurate lithography hotspot detection based on PCA-SVM classifier with hierarchical data clustering," *Proc. SPIE* **9053**, 90530E (2014).
13. Y.-T. Yu et al., "Machine-learning-based hotspot detection using topological classification and critical feature extraction," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **34**(3), 460–470 (2015).
14. D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. 46th Annu. Des. Autom. Conf.*, ACM, pp. 545–550 (2009).
15. J. A. T. Robles et al., "Hotspot detection based on machine learning," US Patent 8, 402, 397 (2013).
16. S.-Y. Lin et al., "A novel fuzzy matching model for lithography hotspot detection," in *50th ACM/EDAC/IEEE Des. Autom. Conf. (DAC)*, IEEE, pp. 1–6 (2013).
17. W.-Y. Wen et al., "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **33**(11), 1671–1680 (2014).
18. G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.* **18**(7), 1527–1554 (2006).
19. V. Borisov and J. Scheible, "Lithography hotspots detection using deep learning," in *15th Int. Conf. Synth., Modeling, Anal. and Simul. Methods and Appl. Circuit Des. (SMACD)*, IEEE, pp. 145–148 (2018).
20. T. Matsunawa, S. Nojima, and T. Kotani, "Automatic layout feature extraction for lithography hotspot detection based on deep neural network," *Proc. SPIE* **9781**, 97810H (2016).
21. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105 (2012).
22. M. Shin and J.-H. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *J. Micro/Nanolithogr. MEMS MOEMS* **15**(4), 043507 (2016).
23. Y. Watanabe et al., "Accurate lithography simulation model based on convolutional neural networks," *Proc. SPIE* **10147**, 101470K (2017).
24. H. Yang et al., "Imbalance aware lithography hotspot detection: a deep learning approach," *J. Micro/Nanolithogr. MEMS MOEMS* **16**(3), 033504 (2017).
25. H. Yang et al., "Lithography hotspot detection: from shallow to deep learning," in *30th IEEE Int. Syst.-on-Chip Conf. (SOCC)*, IEEE, pp. 233–238 (2017).
26. H. Yang et al., "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2018).
27. J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, IEEE, pp. 349–350 (2012).
28. C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 1–9 (2015).
29. C. Szegedy et al., "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 2818–2826 (2016).
30. M. Lin, Q. Chen, and S. Yan, "Network in network," in *Int. Conf. on Learning Representations (ICLR)*, pp. 1–10 (2013).
31. M. Abadi et al., "Tensorflow: a system for large-scale machine learning," in *Proc. 12th USENIX Conf. Oper. Syst. Des. and Implementation*, pp. 265–283 (2016).
32. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations. (ICLR)*, pp. 1–15 (2015).
33. N. Srivastava et al., "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014).

Jing Chen is currently pursuing her PhD at Nanjing University of Posts and Telecommunications, Nanjing, China. Now, she is a visiting scholar at the University of Texas at Austin. She received her BS degree in microelectronics from Nanjing University of Posts and Telecommunications in 2016. Her research interests include machine learning, design for manufacturability, and power devices.

Yibo Lin received his PhD at the Department of Electrical and Computer Engineering, University of Texas at Austin, in 2018. He received his BS degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. His research interests include machine learning and design for manufacturability.

Yufeng Guo is currently the dean of the College of Electronic and Optical Engineering and the vice-director of the National and Local Joint Engineering Laboratory of RF Integration and Micro-Assembly Technology, Nanjing University of Posts and Telecommunications, Nanjing, China. He received his PhD in microelectronics and solid-state electronics from the University of Electronic Science and Technology of China, Chengdu, in 2005. His research interests include power devices, and nanoscale device design and modeling.

Maolin Zhang is currently pursuing his PhD at Nanjing University of Posts and Telecommunications, Nanjing, China. He received his BS degree in microelectronics from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016. His research interests include nanoscale device design and modeling.

Mohamed Baker Alawieh is currently pursuing his PhD in electrical and computer engineering with the University of Texas at Austin, Austin, Texas, USA. He received his MS degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, in 2016 and his BS degree in electrical and computer engineering from the American University of Beirut, Beirut, Lebanon, in 2014. His current research interests include machine learning and computer-aided design for VLSI.

David Z. Pan is Engineering Foundation Professor at ECE Department, UT Austin. His research interests include cross-layer design for manufacturability, reliability, security, and CAD for emerging technologies. He has published over 320 technical papers and graduated 25 PhDs. He has received numerous awards, including SRC Technical Excellence Award and many best paper awards at premier venues. He is a fellow of IEEE and SPIE.