Semi-Supervised Hotspot Detection with Self-Paced Multi-Task Learning

Ying Chen[1,2,3], Yibo Lin[3], Tianyang Gai[1,2], Yajuan Su[1], Yayi Wei[1,2], and David Z. Pan[3]

[1]Key Laboratory of Microelectronics Devices Integrated Technology, Institue of Microelectronics of Chinese Academy of Sciences, Beijing 100029, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]ECE Department, University of Texas at Austin, Austin, TX, USA

***Abstract*—Lithography simulation is computationally expensive for hotspot detection. Machine learning based hotspot detection is a promising technique to reduce the simulation overhead. However, most learning approaches rely on a large amount of training data to achieve good accuracy and generality. At the early stage of developing a new technology node, the amount of data with labeled hotspots or non-hotspots is very limited. In this paper, we propose a semi-supervised hotspot detection with self-paced multi-task learning paradigm, leveraging both data samples w./w.o. labels to improve the model accuracy and generality. Experimental results demonstrate promising accuracy with a limited amount of labeled training data compared to the state-of-the-art work. The source code and trained models are released on https://github.com/qwepi/SSL.**

I. Introductions

As the technology node continues to shrink, the feature sizes are getting smaller and smaller. Layout patterns are becoming more sensitive to process variations in lithography and lead to manufacturing defects. It is necessary to detect these patterns before volume production to ensure yield. These patterns are named as hotspots.

Hotspots are usually detected with lithography simulation [1]. It is able to achieve high detection accuracy but computationally expensive. Machine learning [2]–[7] and pattern matching [8]–[11] based approaches are then proposed to speedup the detection efficiency and meanwhile maintain the high accuracy. Pattern matching based approaches stores a known hotspot library and search for exact or similar matches given a new layout clip. Yu *et al.* [11] extract critical topological features of hotspots and transform them for design rule checking (DRC) to locate the hotspot positions. Although it has high confidence, it cannot handle unseen hotspots. Machine learning techniques are able to learn the correlation between layout features and hotspots/non-hotspots, develop classifiers to differentiate hotspots and non-hotspots, and thus recognize even unseen hotspots with high accuracy. In addition, hybrid methods [12] of the above two techniques are proposed to combine both their advantages.

In machine learning based hotspot detection, both conventional learning approaches and deep learning approaches are developed for hotspot detection. Models like Bayesian and bilinear techniques have been explored with various feature extraction techniques [13], [14]. Park *et al.* [15] take lithography imaging into consideration and train four SVM kernels for different types of hotspots with the aerial image intensity information to achieve high accuracy. Conventional learning approaches usually require manual feature extraction. Deep learning with conventional neural networks (CNN) has then been explored to avoid the overhead of feature engineering. Yang *et al.* [16] identify the label imbalance issue in the datasets and propose a deep CNN to achieve high classification accuracy. They then develop a biased learning technique for the unbalanced dataset with a discrete-cosine transformation (DCT) for feature tensor generation to further improve accuracy with a less deep CNN [5].

For machine learning-based hotspot detection, previous work mostly relies on supervised learning with access to a large amount of training data available. That is, there are enough data samples known to be either hotspots or non-hotspots (labeled) for model training. This condition cannot always hold in the evolution of technology nodes. At the early stage of a new technology node, the amount of labeled data samples tends to be limited, while unlabeled data samples are relatively easy to access [17]. As supervised learning can only leverage labeled data samples for training, it is likely to encounter significant performance degradation with a small amount of labeled training data.

To overcome the limitations of conventional supervised hotspot detection, we present a self-paced multi-task network for semi-supervised hotspot detection. Semi-supervised learning can leverage both labeled and unlabeled samples to help the model training, reducing the dependence to a large amount of labeled training data. It is being actively explored in image recognition, neural language processing, etc [18], [19]. The main contributions are summarized as follows.

- A multi-task neural network (MTNN) with classification and clustering streams is proposed, in which joint model training constructs inner relations and alleviates the influence of labeling error for unlabeled samples.
- A self-paced learning paradigm is developed to gradually incorporate pseudo-labeled data samples for training. It avoids the compromise of ambiguous labeling and improves the model performance.

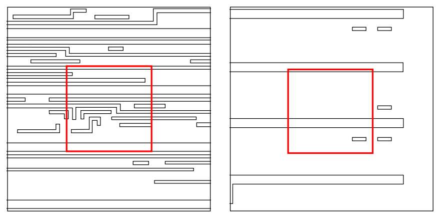

Fig. 1: (a) Hotspot and (b) non-hotspot layout clips.

- Experimental results show that the framework can achieve the state-of-the-art accuracy with more than 3X less labeled training data. With 10% labeled training data, it can achieve an average of 8% higher accuracy than previous work on the ICCAD 2012 contest benchmarks [20].

The rest of the paper is organized as follows. Section II introduces basic concepts and provides the problem formulation. Section III presents the detailed algorithm for the self-paced semi-supervised learning. Section IV validates the proposed framework with experimental results. Section V concludes the paper.

II. Preliminaries and Problem Formulation

In this section, we will review the background of hotspot detection and provide the problem formulation in this work.

A. Hotspot Detection

Due to small process margin in the lithography process, hotspot patterns may cause bridges or broken lines on the wafer after manufacturing. Fig. 1 gives examples of hotspot and non-hotspot. The red regions indicate known hotspot or non-hotspot. hotspots need to be detected and fixed before mask tape-out. Conventionally, lithography simulation [21] is used to do hotspot detection. To implement lithography simulation, process and optical information are needed for model calibration. The model is applied to simulate the imaging contour of layout patterns on the wafer. Problematic locations (a.k.a hotspot) could be easily recognized from the contour simulation. Lithography simulation is extremely time-consuming for full-chip verification and often slows down the design closure.

On the other hand, machine learning technique takes an input layout clip as an image. The information of whether the clip contains hotspots or not could be seen as its label. Hotspot detection based on machine learning can be formulated as an image recognition problem in which the lithography process information is stored in the correlation between input samples and their labels.

B. Problem Formulation

The performance of a hotspot detector is evaluated with following metrics [20],

$$\text{accuracy} = \frac{\text{\# of correctly predicted hotspots}}{\text{\# of hotspots}}, \tag{1a}$$

$$\text{false alarm} = \text{\# of incorrectly predicted hotspots.} \tag{1b}$$

In the terminology of statistics, accuracy is equivalent to the true-positive ratio and the false alarm is the number of false-positive predictions.

The objective of hotspot detection is maximizing accuracy with low false alarms. Recently, machine learning based approaches formulate the hotspot detection problem into a classification task, in which the labels need to be predicted given input features. Related terminologies are shown as follows.

Definition 1 (Labeled/unlabeled samples)**.** *If the class of a sample is known, the sample is called a labeled sample; otherwise, it is an unlabeled sample.*

For hotspot detection, a sample can have two classes: hotspot or non-hotspot. If we are sure whether a layout clip is a hotspot or non-hotspot, then the clip is a labeled sample; otherwise, it is unlabeled. We can obtain the label of an unlabeled sample with lithography simulation and then the sample becomes a labeled sample.

We then formulate the semi-supervised hotspot detection problem as follows:

Definition 2 (Semi-supervised hotspot detection)**.** *Given a labeled dataset containing layout clips with known labels and an unlabeled dataset containing layout clips without known labels, train a classifier to maximize the accuracy with low false alarms over the entire dataset.*

In practice, obtaining large labeled hotspot detection dataset is very expensive, as numerous lithography simulations are required to obtain the labels. However, it is relatively inexpensive to access unlabeled datasets by extracting layout clips from designs without querying for the labels. Therefore, in most of the cases, it is desired to build an accurate hotspot detector with a very limited amount of labeled samples. Whether the unlabeled dataset can be utilized to assist the model training becomes very meaningful.

III. Semi-Supervised Hotspot Detection

The major challenge in semi-supervised learning is the accuracy degradation from the insufficient labeled data and the error in assigning pseudo-labels for unlabeled data. To improve model performance, we adopt a multi-task network with classification and clustering model jointly learned to alleviate the negative influence of inaccurate labeling [19]. The overall flow is shown in Fig. 2. In MTNN, the classification stream assigns pseudo-labels for unlabeled samples while the clustering stream measures the confidence of pseudo-labels with weights. Unlabeled samples with weights for model training could reduce the influence of errors in labeling. The weighted unlabeled samples are gradually introduced for training with a self-paced learning paradigm. This self-paced learning paradigm is connected with the data preparation part. The original layout clip size is $1200 \times 1200\text{nm}^2$, which is expensive for neural networks to process. We first generate images with 1200×1200 pixels and downscale to 128×128 with the nearest-neighbor algorithm [22].

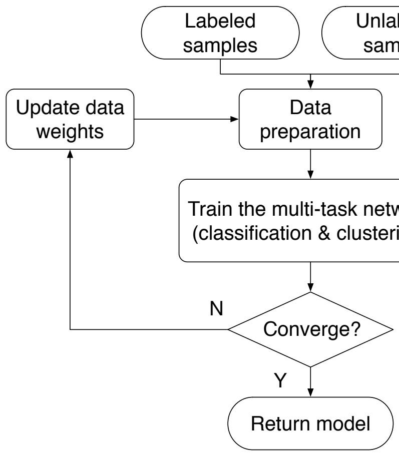

Fig. 2: Overall flow of semi-supervised learning.

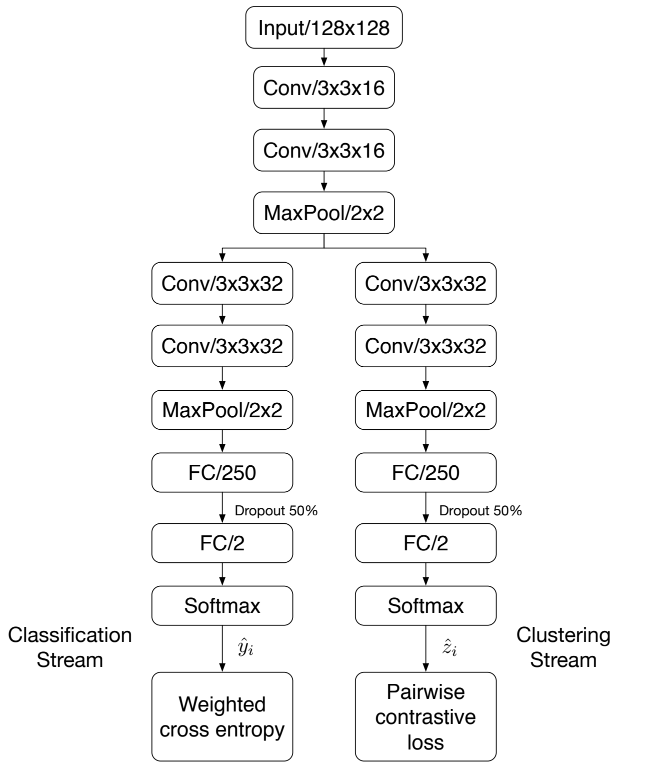

Fig. 3: The architecture of MTNN.

A. Multi-Task Neural Network Architecture

CNN is adopted as the classifier in MTNN for its good performance in image related tasks [23]. The architecture of MTNN is shown in Fig. 3. MTNN has two streams, with which models for classification and clustering are jointly learned. The two streams share layers for feature extraction at early stages and then split into two branches [24]. The shared layers include two convolutional layers, one ReLU layer, and one max pooling layer. The kernel sizes and number of kernels are annotated in the figure. The max pooling layer performs 2×2 down sampling.

The individual layers for two streams are identical except that the weights are learned separately and the loss functions are different. The classification stream behaves as an ordinary hotspot detector as in other neural network architectures. It can predict labels for unlabeled samples and its loss function for training is the weighted cross entropy where the weights come along with data samples. The loss function is defined as follows,

$$L_{class} = -\sum_{i=1}^{N} w_i (\sum_{k=1}^{K} y_i^k \log \hat{y}_i^k), \tag{2}$$

where N is the number of input samples, K is the number of classes, which is 2 in hotspot detection. Vectors y_i and $\hat{y}_i$ are the one-hot encoding of actual class labels and the softmax output for i^{th} sample, respectively. Weight w_i indicates the confidence of the i^{th} sample's label. The weights for labeled samples are set to 1, while the weights for unlabeled samples are calculated during the training with the clustering stream.

The main target of clustering stream is to determine the weights of unlabeled samples based on their distances to the predicted clusters. To further alleviate the influence of untrusted labeling, pairwise constraints [25] are introduced for the loss function of clustering. A pairwise constraint is a pair of samples with the information of whether they are similar.

Definition 3 (Similar pair)**.** *If the labels of two samples are the same, they are a similar pair, vice versa.*

For a labeled sample, its actual label is used. For an unlabeled sample, the pseudo-label predicted from the classification stream is used.

Pairwise constraints can be generated by enumerating all pairs from the labeled or pseudo-labeled data samples. Clustering training with pairwise constraints enables better tolerance to labeling error for unlabeled data. The loss function for the clustering stream is built with Kullback-Leibler (KL) divergence from x_j to x_i with pairwise constraints.

$$KL(x_i||x_j) = \sum_{k=1}^{K} \hat{z}_i^k \log(\frac{\hat{z}_i^k}{\hat{z}_j^k}), \tag{3}$$

where x_i and x_j is the i^{th} and j^{th} samples, respectively, $\hat{z}_i$

and $\hat{z}_j$ are their corresponding clustering softmax outputs, respectively. We use this KL-divergence to measure the distance between two samples, defined as *KL-distance* for brevity. We now define pairwise cost function to convert the divergence into a cost as follows,

$$L_{pair}(x_i||x_j) = \begin{cases} KL(x_i||x_j), & \text{if } (x_i, x_j) \text{ is a similar pair,} \\ \max(0, M - KL(x_i||x_j)), & \text{otherwise,} \end{cases} \tag{4}$$

where M denotes the maximum similarity of samples belonging to two clusters. It is set to a constant value 2 for better convergence in training [25]. Therefore, the overall loss function for the clustering stream considers the pairwise costs of both sides,

$$L_{clust} = \sum_{i,j=1}^{N} \frac{1}{2}(L_{pair}(x_i||x_j) + L_{pair}(x_j||x_i)). \tag{5}$$

The pseudo-label of an unlabeled sample closer to other samples within the predicted cluster according to the KL-distance should have higher confidence due to the correlation between classification and clustering streams. Thus the confidence of the i^{th} sample is defined as its average KL-distance to other samples the same predicted cluster,

$$d_i = \frac{\sum_{j=1}^{N} KL(x_i||x_j)\delta(x_i, x_j)}{\sum_{j=1}^{n} \delta(x_i, x_j)}, \tag{6}$$

where

$$\delta(x_i, x_j) = \begin{cases} 1, & \text{if } (x_i, x_j) \text{ is a similar pair,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Then weight w_i of the i^{th} sample is defined by normalizing d_i for unlabeled samples,

$$w_i = N_u \frac{\exp(-d_i)}{\sum_{i=1}^{N_u} \exp(-d_i)}, \tag{8}$$

where N_u is the number of unlabeled samples.

B. Self-Paced Learning Paradigm

MTNN is trained only with labeled data at first. Then pseudo-labels and weights are assigned to unlabeled samples through predictions. If the unlabeled data are fed to the model training directly, data with unconfident predictions will degrade the accuracy. Therefore, a self-paced paradigm [26] is adopted to gradually introduce unlabeled data with high confidence for training. The procedure of the self-paced learning is summarized in Alg. 1. The learning can be repeated for R rounds with $R = 4$ in the experiment.

An indicator vector $v = (v_1, v_2, \ldots, v_N)$ is used to decide which samples are selected for the next training cycle. Selected samples will have v_i equal to 1; otherwise, they are 0. Labeled samples always have $v_i = 1$. For unlabeled samples, the selection criteria is based on following,

$$v_i = \begin{cases} 1, & \text{if } -w_i \sum_{k=1}^{K} y_i^k \log \hat{y}_i^k < \lambda, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where λ is the threshold to the weighted loss of each sample for selection. The underlying assumption is that small loss indicates high confidence.

The threshold λ is determined by training an auxiliary network [26] with the same structure as the classification stream. After the training of multi-task network, pseudo-labels and weights are assigned to unlabeled samples. We divide the unlabeled dataset S_u to m subsets, i.e, $\{S_u^1, S_u^2, \ldots, S_u^m\}$, equally with ascending order of the loss in the classification stream. Then we try training the auxiliary network indepdendently with S_u^1, $S_u^1 \cup S_u^2$, $\ldots$, $\{S_u^1 \cup S_u^2 \cup \cdots \cup S_u^m\}$ as the training sets, respectively, and use the initially labeled samples as the testing dataset for validation. Suppose the best validation accuracy is achieved by subsets $S_u^1 \cup \cdots \cup S_u^h$ $(1 \leq h \leq m)$. Then we choose the highest loss in the classification stream of the unlabeled samples in these subsets as the value of λ.

Algorithm 1 Self-Paced Semi-Supervised MTNN

```
Require: Input labeled dataset $S_l$ and unlabeled dataset $S_u$.
Ensure: MTNN with maximum accuracy and low false alarm.
 1: Define $v_s$ as the indicator and $w_s$ as the weight for a sample $s$;
 2: Define $T$ as the training set;
 3: Define $R$ as the maximum rounds for self-paced learning;
 4: $v_s \leftarrow 1, w_s \leftarrow 1, \forall s \in S_l$;
 5: $v_s \leftarrow 0, w_s \leftarrow 0, \forall s \in S_u$;
 6: for $t = 1 \rightarrow R$ do
 7:     $T \leftarrow \{s | v_s = 1, \forall s \in S_l \cup S_u\}$;
 8:     Generate pairwise constraints based on training dataset $T$;
 9:     Train MTNN with $T$ and pairwise constraints;
10:     Assign pseudo-labels to $s \in S_u$ using the classification stream;
11:     Compute weight $w_s, \forall s \in S_u$ according to Eq. (8);
12:     Update indicator $v_s, \forall s \in S_u$ according to Eq. (9);
13: return MTNN;
```

IV. Experimental Results

This self-paced MTNN is implemented in Python with Tensorflow 1.2.1 [27] on a Linux server with an 8-core 3.4GHz CPU, a Nvidia GTX 1080 GPU, and 32GB memory. The framework is validated on 28-nm industrial benchmarks from ICCAD2012 CAD contest as described in Table I. The benchmark b1 is omitted as it is too small. To verify modeling performance with different amount of labeled data, the network is trained using different ratios of labeled samples, i.e., {0.1,0.3,0.5,0.7,0.9,1.0}. When randomly generating labeled samples from the training datasets, we keep the ratio between hotspots and non-hotspots the same as that in the original dataset. The unselected samples are regarded as unlabeled samples. Moreover, to avoid statistical instability in randomness, each experiment is repeated for five times with different random seeds and the average results are reported. To handle

TABLE I: Statistics on ICCAD2012 28nm Benchmarks

Dataset	Train		Test	
	#HS	#NHS	#HS	#NHS
b2	174	5285	498	41298
b3	909	4643	1808	46333
b4	95	4452	177	31890
b5	26	2716	41	19327

the imbalanced dataset, biased learning [5] is adopted in training to increase accuracy and reduce false alarms. Initial learning rate for training is 0.001 and batch size is 32. The unlabeled samples are divided into 15 subsets for auxiliary network training. The self-paced learning paradigm loops for 4 rounds ($R = 4$).

General runtime for training the self-paced MTNN is around 60 minutes. The prediction time for each test case with several thousands of clips in Table I is around 2 minutes, which could enable huge time savings compared with lithography simulation. Thus we will not separately report runtime in the following discussion.

We compare accuracy and false alarm on the testing dataset with two machine learning based detectors, as shown in Table II. “DAC” denotes the deep biased learning approach with DCT [5], and “SSL” denotes our self-paced semi-supervised learning algorithm. The classification stream of ”SSL” has the same structure as the CNN-based detector in ”DAC”. At a lower selected ratio like 0.1, 0.3, 0.5, 0.7, our framework achieves better hotspot detection accuracy on average of 65.94%, 77.50%, 90.63% and 94.23% with slight false alarm compromise, respectively. At a higher ratio like 0.9 and 1.0, there is no significant difference between the average accuracy between the two detectors, since enough labeled training data is available.

Fig. 4 plots the average testing accuracy and the standard deviations of five random seeds with a different amount of training data. When the ratio increases, the accuracy of DAC has a rising trend while the accuracy of SSL stays high and fluctuates within a small range. With different random seeds, the accuracy of SSL is more stable compared with that of DAC, as the deviations are smaller. Result of SSL on benchmark5 shows difference from other benchmarks with a similar uptrend as DAC instead of keeping stable at a higher accuracy. This may due to the limitated hotspot samples of benchmark5. In the case of ratio 1.0 with all labeled training data selected, SSL trains MTNN once without the self-paced learning paradigm and realizes better performance on benchmark 2 and 3. It indicates that forcing similar pairs to become closer in the clustering stream of MTNN can help the generalization of the discriminative model, especially for less imbalanced training set.

We further explore the efficacy of the self-paced learning paradigm for different ratios of training data in Fig. 5. As unlabeled samples are gradually introduced into training, the model is essentially training from “easy” to “mature” through each round [28]. From the showing results, we can see that through each iteration, the accuracy is gradually increased. This uptrend is sharper especially when the selected ratio is low. We can see that for low ratios like 0.1 and 0.3, there is an obvious trend of gradually increasing accuracy with different rounds. For high ratios like 0.7 and 0.9, more fluctuation at high accuracy is observed.

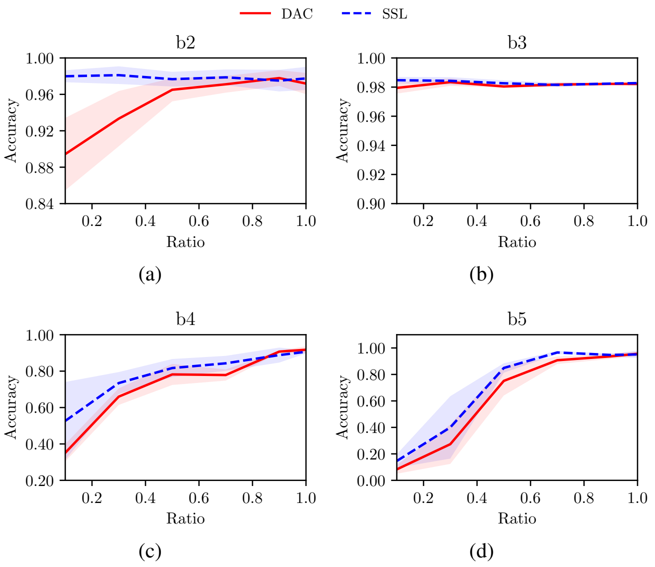

Fig. 4: Comparison of testing accuracy versus ratio of training dataset. Both average and standard deviation values are drawn for different runs.

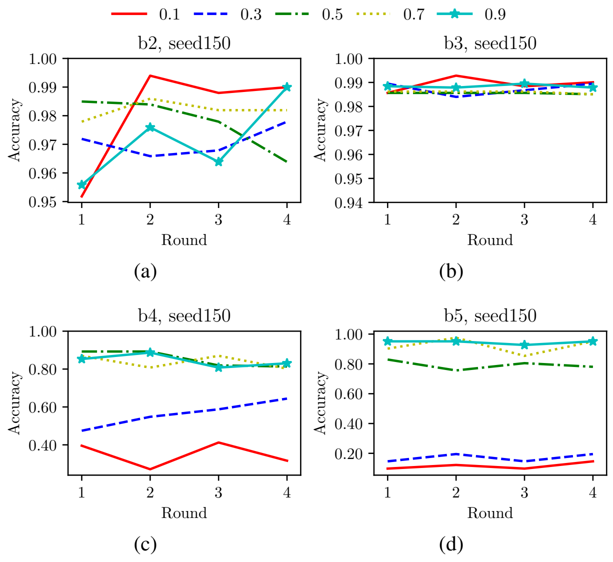

Fig. 5: Accuracy in training rounds of SSL for one random seed. Curves for different ratios of training dataset o benchmarks b2-b5 are plotted.

V. Conclusion

A semi-supervised hotspot detection framework with self-paced multi-task learing is presented. With the joint learning

TABLE II: Accuracy and False Alarm Comparison for Different Amount of Labeled Training Data.

Ratio		benchmark2		benchmark3		benchmark4		benchmark5		Average	
		DAC	SSL	DAC	SSL	DAC	SSL	DAC	SSL	DAC	SSL
0.1	accuracy	89.44%	97.99%	97.94%	98.47%	35.14%	52.66%	8.29%	14.63%	57.70%	**65.94%**
	false alarm	700	1643	4288	5130	230	536	3	11	1305	1830
0.3	accuracy	93.33%	98.11%	98.34%	98.43%	65.99%	73.45%	27.32%	40.00%	71.24%	**77.50%**
	false alarm	383	643	3569	3593	315	342	39	73	1076	1163
0.5	accuracy	96.51%	97.67%	98.04%	98.26%	78.19%	81.69%	75.12%	84.88%	86.97%	**90.63%**
	false alarm	297	425	3098	3083	359	379	86	104	960	998
0.7	accuracy	97.11%	97.87%	98.17%	98.15%	77.85%	84.29%	90.73%	96.59%	90.97%	**94.23%**
	false alarm	294	265	3001	2740	261	261	72	141	907	**852**
0.9	accuracy	97.79%	97.51%	98.22%	98.24%	90.73%	88.81%	93.66%	94.71%	95.10%	94.82%
	false alarm	287	211	2780	2665	387	317	79	100	883	**823**
1.0	accuracy	97.19%	97.75%	98.22%	98.27%	91.75%	90.62%	95.61%	95.12%	95.69%	95.44%
	false alarm	239	231	2878	2854	309	306	90	94	879	**871**

of a classification model and a clustering model, MTNN is able to leverage unlabeled data samples for training. By gradually incorporating the unlabeled samples through a self-paced learning paradigm, the model can achieve the state-of-the-art accuracy with a much smaller amount of labeled training data. Future work includes the exploration of various feature representations such as the discrete cosine transformation [5] to improve the generalization of models.

References

[1] C.-W. Lin, M.-C. Tsai, K.-Y. Lee, T.-C. Chen, T.-C. Wang, and Y.-W. Chang, "Recent research and emerging challenges in physical design for manufacturability/reliability," in *ASPDAC*. IEEE, 2007, pp. 238–243.

[2] M. Shin and J.-H. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 15, no. 4, p. 043507, 2016.

[3] 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 *DAC*. ACM, 2013, p. 67.

[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 (JM3)*, vol. 14, no. 1, p. 011003, 2014.

[5] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on CAD*, 2018.

[6] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *ICCAD*, vol. 30, no. 11, pp. 1621–1634, 2011.

[7] H. Zhang, B. Yu, and E. F. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *ICCAD*. ACM, 2016, p. 47.

[8] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-y. Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *DAC*. IEEE, 2013, pp. 1–6.

[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 Transactions on CAD*, vol. 33, no. 11, pp. 1671–1680, 2014.

[10] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *ICCAD*. ACM, 2006, pp. 625–632.

[11] 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 *DAC*. ACM, 2012, pp. 1167–1172.

[12] S. Mostafa, J. A. Torres, P. Rezk, and K. Madkour, "Multi-selection method for physical design verification applications," in *Design for Manufacturability through Design-Process Integration V*, vol. 7974. International Society for Optics and Photonics, 2011, p. 797407.

[13] T. Matsunawa, B. Yu, and D. Z. Pan, "Optical proximity correction with hierarchical bayes model," in *Optical Microlithography XXVIII*, vol. 9426. International Society for Optics and Photonics, 2015, p. 94260X.

[14] H. Zhang, F. Zhu, H. Li, E. F. Young, and B. Yu, "Bilinear lithography hotspot detection," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*. ACM, 2017, pp. 7–14.

[15] J. W. Park, A. Torres, and X. Song, "Litho-aware machine learning for hotspot detection," *IEEE Transactions on CAD*, vol. 37, no. 7, pp. 1510–1514, 2018.

[16] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: a deep learning approach," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 16, no. 3, p. 033504, 2017.

[17] Y. Lin, Y. Watanabe, T. Kimura, T. Matsunawa, S. Nojima, M. Li, and D. Z. Pan, "Data efficient lithography modeling with residual neural networks and transfer learning," in *Proceedings of the 2018 International Symposium on Physical Design*. ACM, 2018, pp. 82–89.

[18] X. Zhu, "Semi-supervised learning literature survey," *Computer Science, University of Wisconsin-Madison*, vol. 2, no. 3, p. 4, 2006.

[19] S. Wu, Q. Ji, S. Wang, H.-S. Wong, Z. Yu, and Y. Xu, "Semi-supervised image classification with self-paced cross-task networks," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 851–865, 2018.

[20] J. A. Torres, "Iccad-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite," in *ICCAD*. IEEE, 2012, pp. 349–350.

[21] J. Kim and M. Fan, "Hotspot detection on post-opc layout using full-chip simulation-based verification tool: a case study with aerial image simulation," in *23rd Annual BACUS Symposium on Photomask Technology*, vol. 5256. International Society for Optics and Photonics, 2003, pp. 919–926.

[22] "Pillow," https://pillow.readthedocs.io/.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[24] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang, "Deepsaliency: Multi-task deep neural network model for salient object detection," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3919–3930, 2016.

[25] Y.-C. Hsu and Z. Kira, "Neural network-based clustering using pairwise constraints," *arXiv preprint arXiv:1511.06321*, 2015.

[26] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-paced learning with diversity," in *Advances in Neural Information Processing Systems*, 2014, pp. 2078–2086.

[27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.