This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

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

Ying Chen *Student Member, IEEE*, Yibo Lin *Member, IEEE*, Tianyang Gai *Student Member, IEEE*, Yajuan Su, Yayi Wei, *Member, IEEE*, David Z. Pan *Fellow, IEEE*

***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 model accuracy and generality. Experimental results demonstrate that our approach can achieve 4.6%-6.5% better accuracy at the same false alarm levels than the state-of-the-art work using 10%-50% of training data.**

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]–[12] 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 [13] of the above two techniques are proposed to combine both their advantages. Mostafa *et al.* [13] adopt a machine learning system to filter patterns, and then apply pattern matching to detect outliers. They aim to identify previously observed hotspots accurately while achieving better predictability of unseen hotspots.

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 [14], [15]. Park *et al.* [16] consider lithography imaging 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.* [17] 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 [4]. They also propose a batch active learning to improve the data efficiency by active selection of data samples for training. The active learning scheme assumes that it is possible to query the labels of data samples by lithography simulation during model construction.

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 [18]. 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.

Facing the situation that labeled data samples are difficult to obtain while unlabeled data samples are abundant and easily collected, semi-supervised learning which could make full use of vast unlabeled data has attracted attention. 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 actively explored in image recognition, neural language processing, etc [19], [20]. Co-training and graph-based method are two

The preliminary version has been presented at the Asia and South Pacific Design Automation Conference (ASP-DAC) in 2019. This work is supported by National Science and Technology Major Project of China (Grant No.2017ZX02315001 and 2017ZX02101004), US National Science Foundation (#1718570) and China Scholarship Council (No.201704910587) during Y. Chen's visit to UT Austin *(Corresponding author: Yayi Wei, David Z. Pan, Yajuan Su).*

Y. Chen, T. Gai Y. Su and Y. Wei are with Key Laboratory of Microelectronics Devices Integrated Technology, Institue of Microelectronics of Chinese Academy of Science and University of Chinese Academy of Science, Beijing, China (e-mail: weiyayi@ime.ac.cn; suyajuan@ime.ac.cn).

Y. Chen, Y. Lin, D. Z. Pan are with The Department of Electrical and Computer Engineering, The University of Texas at Austin, TX, USA (e-mail: dpan@ece.utexas.edu).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

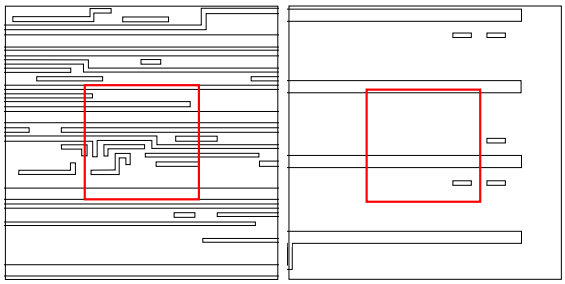
classical semi-supervised learning methods. In co-training, two learning algorithms are trained separately on two distinct views of each sample, and their predictions on new unlabeled samples are used for the enlargement of each other's training set [21]. Zhou *et al.* [22] present a co-training algorithm called tri-training which uses three classifiers instead of two. An unlabeled sample can be labeled with two classifiers' agreement on the label assignment of this sample. This could alleviate the time-consuming problem in the labeling confidence measurement of a standard co-training. Caldas *et al.* [23] propose a co-training method based on minimal learning machine, and mitigate the heavy computational cost problem with the recursive formulation in parameters learning step and a nearest neighbor procedure in the output estimation step. Label propagation algorithm is a widely used graph-based method [24]–[27], which predicts class for unlabeled samples by propagating labels of labeled samples to unlabeled samples based on the data distribution. Yu *et al.* [28] classify the labels of new observations based on label propagation results and the consensus rates computed from multiple clustering results.

For correlated tasks, designing the architecture with sharing layers could be effective. For example, Li *et al.* [29] perform saliency detection task and object class segmentation task together with a shared convolution part. Zhang *et al.* [30] utilize a multi-task deep neural network with shared hidden layers for acoustic emotion recognition. Self-paced learning could introduce samples for training from easy ones to hard ones gradually. With its benefit of alleviating the influence of ambiguous data, self-paced learning has been adopted in many semi-supervised learning related researches. Lin *et al.* [31] combine active learning and self-paced learning, the model performance is improved by selecting high confident samples with self-paced learning and querying real label of low confident samples with active learning. Zhou *et al.* [32] apply self-paced learning to alleviate the side effects of noisy samples or outliers and outperform state-of-art approaches in person re-identification.

In our preliminary work [33], we first apply multi-task network and self-paced learning on hotspot detection to overcome the limitations of conventional supervised hotspot detection. High-confidence pseudo-labeled samples are gradually incorporated for model training through self-paced learning. However, this self-paced learning approach may introduce wrongly pseudo-labeled samples for training due to the mix-up of pseudo hotspots and nonhotspots. We propose an imbalance-aware self-paced learning algorithm, which separates the pseudo hotspots and nonhotspots and selects confident samples from each dataset for training. We can obtain more confident pseudo-labeled samples for model training.

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 incorporate pseudo-labeled data samples for training gradually. It avoids the compromise of ambiguous labeling and improves the model performance.
- We develop an imbalance-aware self-paced learning algorithm to incorporate pseudo hotspots and pseudo nonhotspots separately. It further avoids labeling error and introduces more confident pseudo-labeled samples for training.
- The experimental results show that the framework can achieve 4.6%-6.5% better accuracy at the same false alarm levels than the state-of-the-art work using 10%-50% of training data on the ICCAD 2012 contest benchmarks [34].



(a) (b)

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

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. Figure 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 [35] 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

B. Problem Formulation

The performance of a hotspot detector is evaluated with following metrics [34],

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

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

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. Algorithms

In this section, we will explain how our self-paced multi-task learning model works. Figure 2 shows the overall training flow. The main stages of our framework include data preparation, classifier updating, and sample pseudo labeling & weights updating.

Data preparation: In the beginning, labeled layout clips are randomly selected for each class. The original layout clip size is $1200 \times 1200\text{nm}^2$, which is expensive for neural networks to process. To get a good feature representation, we apply DCT [4] on layout clips. In following training loops, apart from original labeled samples, weighted unlabeled samples with pseudo labels are selected for training with a self-paced learning paradigm based on their labeling confidence.

Classifier updating: At first, only a small amount of labeled samples are used for classifier training. As the framework gets mature, samples pseudo-labeled are growing, we add them into the training set along with original labeled samples and retrain the classifier.

Sample pseudo-labeling & weights updating: In our framework, we adopt a multi-task network with classification and clustering model jointly learned to alleviate the negative influence of inaccurate labeling [20]. In each iteration, the classification stream assigns pseudo-labels for unlabeled samples while the clustering stream measures the confidence of pseudo-labels with weights.

A. Convolutional Neural Networks

Convolutional neural network (CNN) is adopted as the classifier for its good performance in image related tasks [36]. CNN is mainly built with convolution layers and fully connected layers, where convolution layers extract features and fully connected layers perform classification or regression. Typically, a rectified linear unit (ReLU) layer is applied for activation following the convolution layer for its benefit to fast convergence and nonlinearity to the network. The ReLU function is defined as,

$$f(x) = \max(x, 0). \tag{2}$$

Then the max-pooling layer performs down-sampling with the benefit of feature map dimensions reducing and translation-invariance.

B. Multi-Task Neural Network Architecture

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 [20].

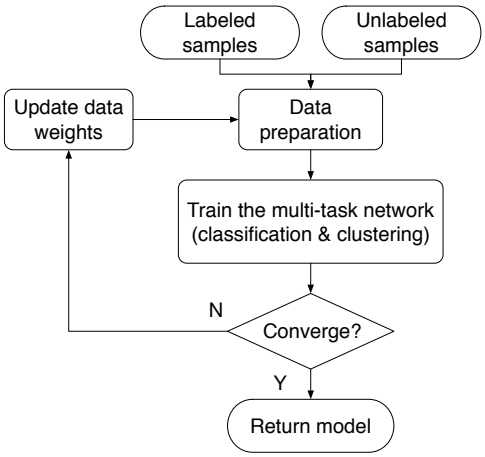


Fig. 2: Overall flow of semi-supervised learning.

The architecture of MTNN is shown in Figure 3. MTNN has two streams, with which models for classification and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

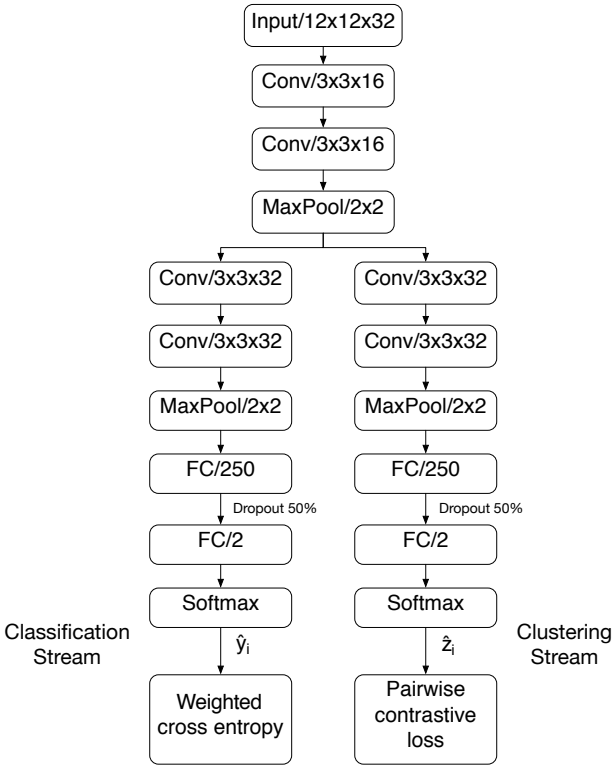


Fig. 3: The architecture of MTNN.

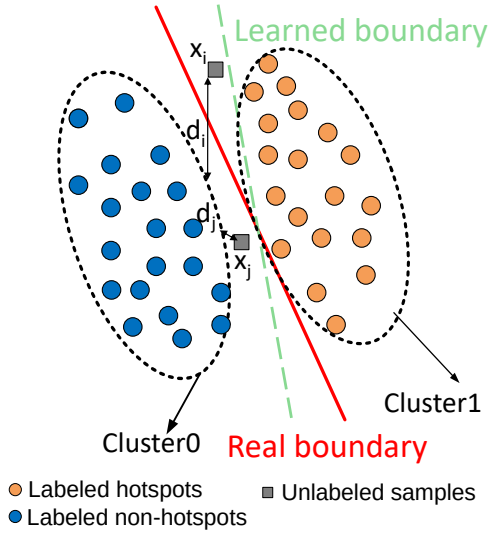


Fig. 4: Illustration of how MTNN works.

clustering are jointly learned. The two streams share layers for feature extraction at the early stages and then split into two branches [29]. The shared layers include two convolutional layers, one ReLU layer, and one max pooling layer. The kernel sizes and the number of kernels are annotated in the figure. The max pooling layer performs 2×2 downsampling.

For hotspot detection, the classification stream is used to do pseudo-labeling and the final hotspot detection, while the clustering stream is used to estimate labeling confidence with weights. Fig. 4 illustrates how the MTTNN works. The blue and orange dots are labeled hotspots and labeled nonhotspots separately, the solid red line is the real boundary between them. With jointly learned clustering and classification, we got the green dash line as the learned boundary. The grey square is the unlabeled samples. For unlabeled samples x_i and x_j, if we used the trained model to assign pseudo labels, sample x_i is pseudo-labeled wrongly while sample x_j is pseudo-labeled correctly. If we start the next training using the pseudo-labeled samples without computing weight through clustering, then wrongly-labeled x_i could compromise the final prediction performance. Clustering analyzes samples' distributions and divides samples into two clusters. Samples with the same labels tend to be in the same cluster due to their similar distributions. A sample's average distance to the predicted cluster represents its labeling confidence. The longer the distance is, the less the labeling confidence is. As shown in Fig 4, $d_i > d_j$, it means that the wrongly pseudo-labeled sample x_i has lower confidence than correctly pseudo-labeled sample x_j. x_i should have less influence on model training. Therefore, lower weight is assigned to x_i to alleviate its influence.

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,

$$L_{class} = -\sum_{i=1}^{N} w_i (\sum_{k=1}^{K} y_i^k \log \hat{y}_i^k), \tag{3}$$

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 [37] 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 the 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 output of the final softmax layer in the clustering stream could be seen as the distribution probability over different clusters.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Kullback-Leibler (KL) divergence could evaluate the similarity between two distributions by measuring their statistical distance. Therefore, the loss function for the clustering stream is built with Kullback-Leibler (KL) divergence from x_j to x_i with pairwise constraints.

$$KL(x_i||x_j) = \sum_{k=1}^{K} \hat{z}_i^k \log(\frac{\hat{z}_i^k}{\hat{z}_j^k}), \tag{4}$$

where x_i and x_j is the i^{th} and j^{th} samples, respectively, $\hat{z}_i = (\hat{z}_i^1, \hat{z}_i^2, \ldots, \hat{z}_i^k)$ and $\hat{z}_j = (\hat{z}_j^1, \hat{z}_j^2, \ldots, \hat{z}_j^k)$ are the final softmax layer outputs of the clutstering stream for samples x_i and x_j, respectively. K is the number of clusters, which equals 2 in hotspot detection problem. We use the KL-divergence in Equation 4 to evaluate the similarity between two samples, defined as *KL-distance* for brevity. The larger the KL-distance is, the less similar the two samples are. We now define pairwise cost function to convert the divergence into a cost as follows,

$$L_{pair}(x_i||x_j) = \begin{cases} KL(x_i||x_j), & \text{if } (x_i, x_j) \text{ is a similar pair,} \\ \max(0, M - KL(x_i||x_j)), & \text{otherwise,} \end{cases} \tag{5}$$

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 [37]. Therefore, the overall loss function for the clustering stream considers the pairwise costs of both sides,

$$L_{clust} = \sum_{i,j=1}^{N} \frac{1}{2}(L_{pair}(x_i||x_j) + L_{pair}(x_j||x_i)). \tag{6}$$

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,

$$d_i = \frac{\sum_{j=1}^{N} KL(x_i||x_j)\delta(x_i, x_j)}{\sum_{j=1}^{n} \delta(x_i, x_j)}, \tag{7}$$

where

$$\delta(x_i, x_j) = \begin{cases} 1, & \text{if } (x_i, x_j) \text{ is a similar pair,} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

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

$$w_i = N_u \frac{\exp(-d_i)}{\sum_{i=1}^{N_u} \exp(-d_i)}, \tag{9}$$

where N_u is the number of unlabeled samples.

C. 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 [38] is adopted to introduce unlabeled data with high confidence for training gradually. The learning can be repeated for R rounds with $R = 4$ in the experiment.

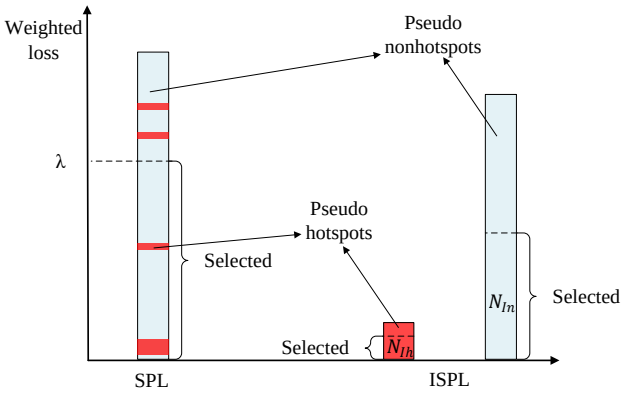


Fig. 5: Comparison of the selected pseudo-labeled samples between SPL and ISPL.

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,

$$v_i = \begin{cases} 1, & \text{if } -w_i \sum_{k=1}^{K} y_i^k \log \hat{y}_i^k < \lambda, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where λ is the threshold to the weighted loss of each sample for selection. Vectors y_i and $\hat{y}_i$ are the one-hot encoding of pseudo class labels and the softmax output of classification stream for i^{th} sample, respectively. Weight w_i indicates the confidence of the i^{th} sample's label, which is determined by Equation 9 for unlabeled samples. $-w_i \sum_{k=1}^{K} y_i^k \log \hat{y}_i^k$ is the weight loss of the classification stream for the i^{th} sample. The underlying assumption is that small loss indicates high confidence.

The threshold λ is determined by training an auxiliary network [38] with the same structure as the classification stream. After the training of the 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 independently 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. The procedure of self-paced learning (SPL) is summarized in Algorithm 1.

D. Imbalance-aware Self-Paced Learning

Definition 4 (Pseudo hotspots (PHS) and pseudo nonhotspots (PNHS)). *For an unlabeled sample, if it is predicted as a layout that contains hotspots by the classifier, we call it as a pseudo hotspot, vice versa.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Algorithm 1 Self-Paced Learning [33]

Require: Trained MTNN model M_{MTNN}, input labeled dataset S_l, unlabeled dataset S_u.
Ensure: The training dataset T for next round MTNN training;
Define λ as the threshold and acc_{max} as the max model prediction accuracy;
Define T_u as the training set for auxiliary network;
Define v_s as the indicator and l_{ws} as the weighted loss for a sample s.
$acc_{max} \leftarrow 0$;
$v_s \leftarrow 1, \forall s \in S_l$;
$v_s \leftarrow 0, \forall s \in S_u$;
Assign pseudo labels and compute weighted loss l_w for samples in S_u with M_{MTNN};
Sort S_u and l_w in ascending order of l_w, then divide S_u and l_w into m subsets, i.e, $\{S_u^1, S_u^2, \ldots, S_u^m\}$, $\{l_w^1, l_w^2, \ldots, l_w^m\}$, equally;
for $h = 1 \rightarrow m$ **do**
    $T_u \leftarrow S_u^1 \cup \cdots \cup S_u^h$;
    Train an auxiliary network with the same structure as the classification stream as MTNN with T_u;
    Compute the prediction accuracy acc_h with S_l as testing samples for the trained model;
    if $acc_h \geq acc_{max}$ **then**
        $acc_{max} \leftarrow acc_h$;
        $\lambda \leftarrow max(l_{ws}), \forall s \in S_u^h$;
    end if
end for
Update indicator $v_s, \forall s \in S_u$ according to Eq. (10);
$T \leftarrow \{s | v_s = 1, \forall s \in S_l \cup S_u\}$;
return T

Definition 5 (Pseudo false alarms)**.** *If the real label class for a pseudo hotspot is nonhotspot, then this pseudo hotspot is called a pseudo false alarm.*

In our preliminary work [33], we improve the hotspot detection accuracy with SPL but suffering the false alarms compromise. Therefore, we assume that SPL might introduce some pseudo false alarms for model training. This leads to an increasing number of false alarms during the final prediction. In Algorithm 1, pseudo hotspots and nonhotspots are mixed. Due to the different distributions between pseudo hotspots and pseudo nonhotspots, some pseudo false alarms may have a smaller weighted loss than some correctly predicted pseudo nonhotspots. In another word, comparing to other pseudo hotspots, these pseudo false alarms are less confident, but they are still more confident than most of the pseudo nonhostpots due to lower weighted loss, including ones that assigned with the right label. When choosing unlabeled samples based on the value of λ, SPL could select more pseudo false alarms before introducing high confident pseudo nonhotspots. Therefore, we further develop imbalance-aware self-paced learning (ISPL). In ISPL, we separate pseudo hotspots and pseudo nonhotspots, then consider the labeling confidence on their pseudo-labeled class. ISPL is summarized in Algorithm 2.

Algorithm 2 Imbalance-aware Self-Paced Learning

Require: M_{MTNN}, S_l, S_u.
Ensure: The training dataset T for next round MTNN training;
Define S_{Ih}, S_{In} as the imbalance-aware pseudo hotspot dataset and nonhotspot dataset for training;
Define N_{Ih}, N_{In} as the number of confident pseudo hotspots and pseudo nonhotspots;
Define s_i as the i_{th} sample in dataset;
$S_{Ih} = \varnothing, S_{In} = \varnothing$;
Assign pseudo labels and compute weighted loss l_w for samples in S_u with M_{MTNN};
Divide S_u into pseudo hotspot subset S_{uh} and pseudo nonhotspot subset S_{un};
Sort S_{uh} and S_{un} in ascending order of l_w, respectively;
Compute N_{Ih} and N_{In} according to Eq. (11);
$S_{Ih} \leftarrow s_i, s \in S_{uh}, i = 1, \ldots, N_{ih}$;
$S_{In} \leftarrow s_i, s \in S_{un}, i = 1, \ldots, N_{in}$;
$T \leftarrow S_l \cup S_{Ih} \cup S_{In}$;
return T

We divide unlabeled dataset S_u into pseudo hotspot dataset S_{uh} and pseudo nonhotspot dataset S_{un} based on assigned pseudo labels. S_{uh} and S_{un} are then sorted with ascending order of the loss in the classification stream. Samples with smaller loss tend to be more confident. In that case, we only select samples at the front from each subset for model training of next round. Since the classifier model gets mature through each iteration, the number of selected samples of each subset changes with iteration round, as shown in Equation (11);

$$N_{Ih} = N_{uh} * [1 - p + p_d * (R - 1)] \tag{11a}$$
$$N_{In} = N_{un} * [1 - p + p_d * (R - 1)] \tag{11b}$$

where N_{Ih}, N_{In} is the number of selected samples with pseudo labels as hotspot and nonhotspot, respectively, N_{uh}, N_{un} is the number of samples in S_{uh} and S_{un}, p and p_d are two manually determined hyper-parameters that control N_{Ih} and N_{In}, R is the number of interation rounds. The values of these hyperparameters are shown in Table II.

Figure 5 shows the difference of the selected pseudo-labeled samples between these two self-paced learning algorithms. SPL sorts unlabeled dataset S_u with ascending order of the weighted loss l_{ws}, then selects pseudo-labeled samples with weighted loss less than λ. λ is determined through the auxiliary network training. Since all pseudo-labeled samples are mixed-up, the selected dataset could be highly imbalanced and contain some wrongly pseudo-labeled samples. Meanwhile, ISPL separates pseudo hotspot dataset and pseudo nonhotspot dataset, then selects samples with a small weighted loss from each subset. The number of selected samples, N_{Ih} and N_{In}, is determined by Equation (11). Comparing to SPL, more highly confident pseudo hotspots and pseudo nonhotspots could be selected. Also, the highly imbalanced dataset could be avoided.

The procedure of the whole self-paced semi-supervised MTNN is summarized in Algorithm 3. We first initialize

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

weights and the training dataset (lines 4-6), then update them through each iteration until the neural network training is over (lines 6-17). Within each iteration, we first train the MTNN with selected training samples (line 8-9), then we compute weights for unlabeled samples and update the training dataset through SPL or ISPL(line 10-16).

Algorithm 3 Self-Paced Semi-Supervised MTNN

Require: S_l, S_u.
Ensure: MTNN with maximum accuracy and low false alarm.
1: Define 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: $w_s \leftarrow 1, \forall s \in S_l$;
5: $w_s \leftarrow 0, \forall s \in S_u$;
6: $T \leftarrow S_l$;
7: **for** $t = 1 \rightarrow R$ **do**
8: Generate pairwise constraints based on training dataset T;
9: Train MTNN with T and pairwise constraints;
10: Compute weight $w_s, \forall s \in S_u$ according to Eq. (9);
11: **if** Apply SPL **then**
12: Update dataset T, according to Algorithm 1;
13: **else if** Apply ISPL **then**
14: Updata dataset T, according to Algorithm 2;
15: **end if**
16: **end for**
17: **return** MTNN;

IV. Experimental Results

A. Experimental Setup

This self-paced MTNN is implemented in Python with Tensorflow 1.2.1 [39] 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. "#HS" and "#NHS" in "train" column denote the total number of hotspots and the total number of nonhotspots in the training set. "#HS" and "#NHS" in "test" column denote the total number of hotspots and the total number of nonhotspots in the testing set.

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 the imbalanced dataset, biased learning [4] is adopted in training to increase accuracy and reduce false alarms. The bias function is defined as follows:

$$\varepsilon = \begin{cases} \frac{1}{1+e^{\beta l}}, & \text{if } l \leq t_b \\ 0, & \text{if } l > t_b \end{cases} \quad (12)$$

where l is the training loss of the current batch with respect to unbiased ground truth, β and t_b are both manually determined parameters, β controls the influence of loss to bias, while t_b represents the threshold controling bias learning.

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

TABLE II: Training Configurations

Configurations	Value
Optimizer	Adam [40]
Initial Learning Rate	0.001
Learning Rate Decay	0.65
Bias Function Coefficient(β)	6
Bias Function Coefficient(t_b)	0, 0.15 and 0.3
LR Decay Step	3200
Batch Size	32
Loops Round (R)	4
Unlabeled Subsets Number (m)	15
ISSL Controling Constant(p)	0.2
ISSL Controling Constant(p_d)	0.058

Table II shows the details of the training configurations. Particularly, t_b varies with training steps.

B. Performance Evaluation

Training a CNN model needs about 8 minutes, general runtime for training the self-paced MTNN [33] is around 60 minutes. When applying ISPL, without multiple training of the auxiliary network, the runtime reduces to 40 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 our preliminary results in [33], as shown in Table III. "DAC" denotes to the deep biased learning approach with DCT [4]. "SSL" denotes to the original self-paced semi-supervised learning algorithm that is applied in our preliminary work [33], "ISSL" denotes to our imbalance-aware self-paced semi-supervised learning algorithm proposed in this paper. The classification stream of "ISSL" and "SSL" has the same structure as the CNN-based detector in "DAC". At ratio 0.1, both accuracy and false alarms are improved with ISSL on average of 2.78% (65.94% vs. 68.73%) and 45.3% (1830 vs. 1001). They are improved on average of 0.82% (90.63%

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

vs. 91.45%) and 8.92% (998 vs. 809) at ratio 0.5. At ratio 0.3, ISSL realizes comparable hotspot detection accuracy as SSL with much less false alarms. At a higher ratio like 0.9 and 1.0, there is no significant difference between the average accuracy among the three detectors, since enough labeled training data is available. At a lower selected ratio, comparing to SSL, ISSL reduces the numbers of false alarms greatly while keeping a competitive accuracy. The reason it happens is that ISPL could select less pseudo false alarms comparing to SPL, we will have a further discussion in Section IV-C. Particularly, the benchmark 5 is highly biased. When selecting labeled samples, we only get a small number of hotspots at first. That is the reason why the improvement of ISSL is not clear for benchmark 5, especially when the selected ratio is 0.1 (we only got 2 labeled hotspots samples).

Figure 6 plots the average testing accuracy and the standard deviations of five random seeds with different amounts of training data. When the ratio increases, the accuracy of DAC has a rising trend while the accuracy of ISSL and SSL stays high and fluctuates within a small range. With different random seeds, the accuracy of ISSL and SSL is more stable compared with that of DAC, as the deviations are smaller. Particularly, Fig. 6(b) shows an exception, there is no obvious change in accuracy when ratio changes from 0.1 to 0.9 among three detectors on benchmark 3. This may because that the original database of benchmark 3 is much larger than others, even ten percent of labeled samples contains enough hotspot and non-hotspot information for model training. The result of ISSL and SSL on benchmark 5 shows the difference from other benchmarks with a similar uptrend as DAC instead of keeping stable at a higher accuracy. This might be due to the limited hotspot samples of benchmark 5. In the case of ratio 1.0 with all labeled training data selected, ISSL and 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 the less imbalanced training set. Correspondingly, Figure 7 shows the average number of false alarms and the standard deviations of five random seeds. ISSL shows improvement on false alarms comparing to SSL at a lower selected ratio. Furthermore, for a fair comparison, we adjust the decision boundary of each model to achieve the same number of false alarms and compare the accuracy, as shown in Table IV. Comparing to SSL, the accuracy of ISSL for training data ratio 0.1, 0.3, and 0.5 are much better, with the improvements on average of 2.54%, 1.99% and 1.66%.

We further explore the efficacy of the imbalance-aware self-paced learning paradigm for different ratios of training data in Figure 8. As unlabeled samples are gradually introduced into training, the model is essentially training from “easy” to “mature” through each round [41]. 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.

C. ISPL vs. SPL

We develop ISPL under the assumption that pseudo hotspots and pseudo nonhotspots have a different standard of labeling confidence measurement. Due to that, SPL could introduce some less confident pseudo hotspots before selecting high confident pseudo nonhotspots. By considering the labeling confidence separately, ISPL could select more confident samples and improve the trained model performance. The experimental results shown in Table III proves the effectiveness of ISPL.

To demonstrate our assumption, we extract both the pseudo labels and the true labels for all unlabeled samples, then compare the number of samples in different pseudo-labeled situations. For a clear illustration, we set the selected ratio as 0.1 and random seed as 150, the pseudo-labeled situation for all four benchmarks are as shown in Fig. 9. We also compute the exact numbers of samples in different pseudo-labeled status, as shown in Table V. ”Total #PHS” and ”Total #PNHS” denote the total number of PHS and the total number of PNHS in the unlabeled dataset. ”Selected #PHS” and ”Selected #PNHS” denote the total number of PHS and the total number of PNHS in the selected dataset for next training. ”Selected #CPHS” and ”Selected #CPNHS” denote the total number of correctly predicted pseudo hotspots (CPHS) and the total number of correctly predicted pseudo nonhotspots (CPNHS) in the selected dataset for next training. ”Selected #WPHS” and ”Selected #WPNHS” denote the total number of wrongly predicted pseudo hotspots (WPHS) and the total number of wrongly predicted pseudo nonhotspots (WPNHS) in the selected dataset for next training.

In Fig. 9, all samples are sorted with the ascending order of their weighted loss, the dots with different colors represent the pseudo-labeling status of the unlabeled samples. Take benchmark 2 as an example, as shown in Fig. 9 (a), PNHS distribute evenly among the range of the weighted loss, while PHS tends to have the larger or smaller weighted loss. Fig. 9 (b) and Fig. 9 (c) shows the weighted loss distribution of samples with correctly predicted and wrongly predicted pseudo labels. Most PNHS are predicted correctly, while the CPHS tends to have the small weighted loss. In Fig. 9 (d), we compare the weighted loss of the CPNHS and the WPHS. Many WPHS have a smaller weighted loss than CPNHS. N_{Ih}, N_{In} for ISPL and λ for SPL are shown with three dash lines. Clearly, in the selected unlabeled samples for the next training, ISPL introduces less wrongly predicted pseudo samples, especially WPHS. Table V also proves it. SPL could select about 30% less wrongly predicted pseudo samples comparing to SPL. Therefore, comparing to SPL, ISPL could select less wrongly predicted pseudo samples, especially pseudo false alarms. With less pseudo false alarms forming the new training dataset, the numbers of false alarms predicted by the newly trained model could be reduced greatly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

D. Receiver Operating Characteristic Curves

As a binary classification problem, hotspot detection could have four cases of prediction results: (1) true positive (TP): correctly predicted hotspots; (2) false positive (FP): incorrectly predicted hotspots (a.k.a false alarm); (3) true negative (TN): correctly predicted nonhotspots; (4) false negative (FN): incorrectly predicted nonhotspots. True positive rate (TPR) and false positive rate (FPR) are defined as follows:

$$\text{TPR} = \frac{TP}{TP + FN}, \tag{13a}$$

$$\text{FPR} = \frac{FP}{FP + TN}. \tag{13b}$$

TPR corresponds to hotspot detection accuracy, while FPR denotes false alarm rate. By shifting the decision boundary of each model, we plot the receiver operating characteristic (ROC) curves which depict the trade-off between TPR and FPR to evaluate the performance of the three detectors. The ROC curves of each benchmark with different ratios of selected labeled samples are as shown in Figure 10, the vertical line refers to the same FPR values reported by the DAC work [4]. ISSL achieves the best performance at a lower selected ratio like 0.1 and 0.3. At a higher selected ratio like 0.9 and 1.0, the ROC curves of ISSL, SSL and DAC overlap with each other since there is no significant performance improvement among these three detectors. Particularly, for benchmark 5, we found that the TPR fluctuates heavily across random seeds when the selected ratio is 0.1 and 0.3. As mentioned before, this might be due to the limited hotspot samples of benchmark 5.

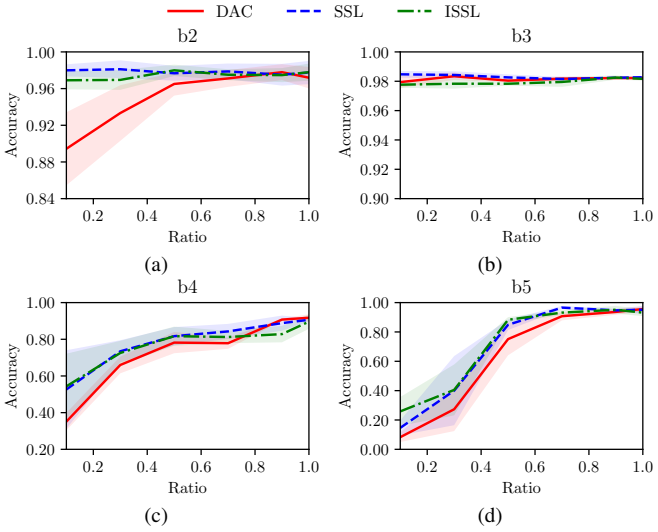


Fig. 6: Comparison of testing accuracy versus ratio of training dataset. Both average and standard deviation values are drawn for different runs.

V. Conclusion

A semi-supervised hotspot detection framework with self-paced multi-task learning is presented for lithography hotspot detection. With the joint learning of a classification model and a clustering model, MTNN is able to leverage unlabeled data samples for training. The classification stream of MTNN assigns pseudo labels for unlabeled samples while the clustering stream measures the confidence of the pseudo-labeling with weights. This could help alleviate the negative influence of samples with unconfident pseudo labels. Additionally, we propose an imbalance-aware self-paced learning paradigm to incorporate confident pseudo hotspots and nonhotspots separately, which can reduce the number of wrongly labeled samples introduced for training. The experimental results demonstrate the efficiency of imbalance-aware self-paced learning. Also, our framework can achieve 4.6%-6.5% better accuracy at the same false alarm levels than the state-of-the-art work using 10%-50% of training data. This framework

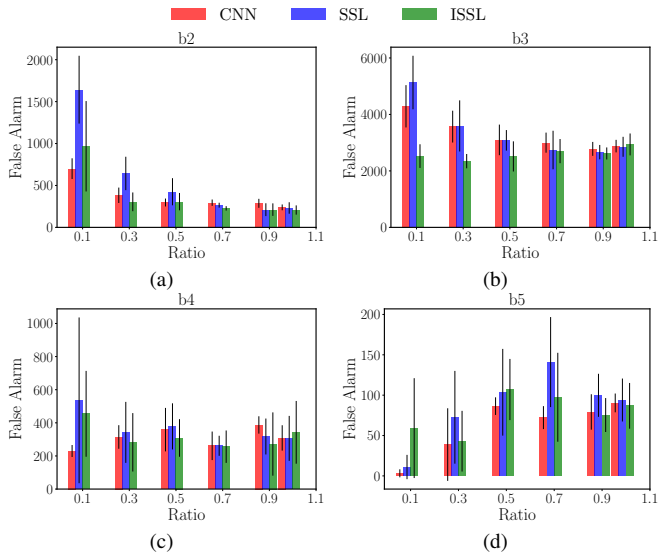


Fig. 7: Comparison of testing false alarms versus ratio of training dataset. Both average and standard deviation values are drawn for different runs.

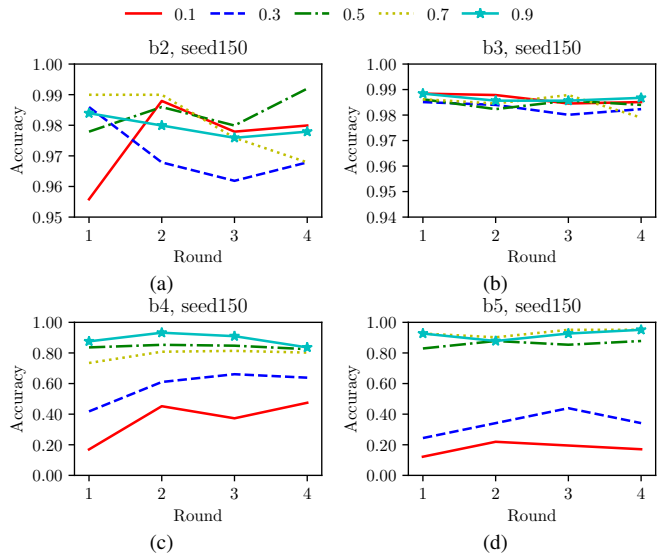


Fig. 8: Accuracy in training rounds of ISSL for one random seed. Curves for different ratios of training dataset from benchmarks b2-b5 are plotted.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

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

Ratio		b2			b3			b4			b5			Average		
		DAC	SSL	ISSL	DAC	SSL	ISSL	DAC	SSL	ISSL	DAC	SSL	ISSL	DAC	SSL	ISSL
0.1	Accuracy(%)	89.44	**97.99**	96.91	97.94	**98.47**	97.77	35.14	52.66	**54.35**	8.29	14.63	**25.85**	57.70	65.94	**68.72**
	#FA	**700**	1643	968	4288	5130	**2524**	**230**	536	454	**3**	11	59	1305	1830	**1001**
0.3	Accuracy(%)	93.33	**98.11**	96.95	98.34	**98.43**	97.83	65.99	**73.45**	72.54	27.32	40.00	**40.49**	71.24	**77.50**	76.95
	#FA	383	643	**305**	3569	3593	**2345**	315	342	**282**	**39**	73	43	1076	1163	**744**
0.5	Accuracy(%)	96.51	97.67	**97.99**	98.04	**98.26**	97.83	78.19	**81.69**	81.69	75.12	84.88	**88.29**	86.97	90.63	**91.45**
	#FA	**297**	425	307	3098	3083	**2513**	359	379	**309**	**86**	104	107	960	998	**809**
0.7	Accuracy(%)	97.11	**97.87**	97.51	**98.17**	98.15	97.95	77.85	**84.29**	81.24	90.73	**96.59**	93.17	90.97	**94.23**	92.47
	#FA	294	265	**227**	3001	2740	**2701**	261	261	**256**	**72**	141	97	907	852	**820**
0.9	Accuracy(%)	**97.79**	97.51	97.47	98.22	98.24	**98.25**	**90.73**	88.81	82.82	93.66	94.71	**95.12**	**95.10**	94.82	93.42
	#FA	287	**211**	212	2780	2665	**2618**	387	317	**272**	79	100	**75**	883	823	**794**
1.0	Accuracy(%)	97.19	97.75	**97.79**	98.22	**98.27**	98.16	**91.75**	90.62	89.60	**95.61**	95.12	93.17	**95.69**	95.44	94.68
	#FA	239	231	**207**	2878	**2854**	2938	309	**306**	343	90	94	**87**	879	**871**	894

TABLE IV: Accuracy Comparison at the Same Numbers of False Alarm as the DAC work [4]

Ratio	b2				b3				b4				b5				Average		
	#FA	Accuracy(%)			#FA	Accuracy(%)			#FA	Accuracy(%)			#FA	Accuracy(%)			Accuracy(%)		
	DAC	DAC	SSL	ISSL	DAC	DAC	SSL	ISSL	DAC	DAC	SSL	ISSL	DAC	DAC	SSL	ISSL	DAC	SSL	ISSL
0.1	700	89.44	93.11	**95.24**	4288	97.94	**98.30**	98.29	230	35.14	42.71	**44.16**	3	8.29	10.67	**17.29**	57.70	61.20	**63.74**
0.3	383	93.33	97.03	**97.28**	3569	98.34	**98.52**	98.10	315	65.99	73.36	**75.59**	39	27.32	34.21	**40.12**	71.24	75.78	**77.77**
0.5	297	96.51	97.19	**97.90**	3098	98.04	**98.25**	97.96	359	78.19	80.82	**82.95**	86	75.12	83.24	**87.32**	86.97	89.87	**91.53**
0.7	294	97.11	**98.00**	97.93	3001	98.17	**98.24**	98.12	261	77.85	**84.43**	82.17	72	90.73	93.17	**94.00**	90.97	**93.46**	93.05
0.9	287	97.79	**98.38**	97.83	2780	98.22	98.27	**98.30**	387	**90.73**	89.87	86.94	79	93.66	94.15	**95.12**	95.10	**95.17**	94.55
1.0	239	97.19	98.20	**98.28**	2878	98.22	**98.26**	98.17	309	**91.75**	90.96	90.28	90	**95.61**	95.61	92.68	**95.69**	95.76	94.85

TABLE V: The Number of Selected Samples in Different Pseudo Label Status (Ratio=0.1, seed=150)

	b2		b3		b4		b5	
	SPL	ISPL	SPL	ISPL	SPL	ISPL	SPL	ISPL
Total #PHS	412		1173		81		6	
Total #PNHS	4502		3824		4012		2462	
Selected #PHS	217	186	959	804	74	35	6	5
Selected #PNHS	3638	3744	3107	3192	3993	3238	2462	1969
Selected #CPHS	126	122	774	735	28	22	4	3
Selected #CPNHS	3632	3737	3098	3183	3944	3211	2442	1949
Selected #WPHS	91	64	185	69	46	13	2	2
Selected #WPNHS	6	7	9	9	49	27	20	20

has the potential to provide satisfactory hotspots detection at the early stage of new technology node development when hotspot data is limited.

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 *IEEE/ACM Asia and South Pacific Design Automation Conference (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] 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.

[4] 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 Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[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 *ACM/IEEE Design Automation Conference (DAC)*. ACM, 2013, p. 67.

[6] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE/ACM International Conference on Computer-Aided Design (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 *IEEE/ACM International Conference on Computer-Aided Design (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 *ACM/IEEE Design Automation Conference (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 Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 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 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2006,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

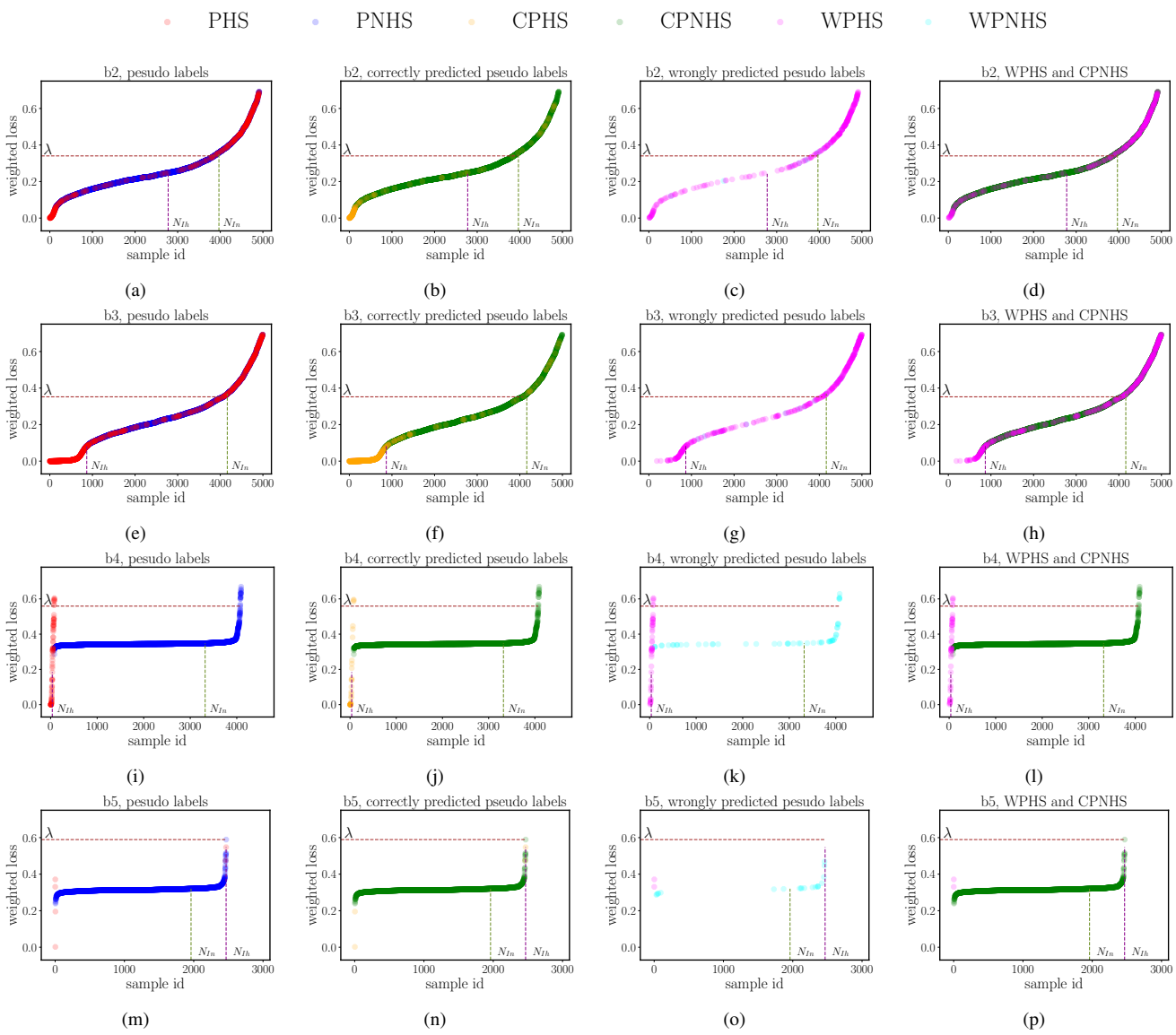


Fig. 9: Pseudo label status illustration (ratio=0.1, seed=150)

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 *ACM/IEEE Design Automation Conference (DAC)*. ACM, 2012, pp. 1167–1172.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] J. W. Park, A. Torres, and X. Song, "Litho-aware machine learning for hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 7, pp. 1510–1514, 2018.

[17] 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.

[18] 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.

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

[20] 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.

[21] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, 1998, pp. 92–100.

[22] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[23] W. L. Caldas, J. P. Gomes, and D. P. Mesquita, "Fast co-mlm: An efficient semi-supervised co-training method based on the minimal learning machine," *New Generation Computing*, vol. 36, no. 1, pp. 41–58, 2018.

[24] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, language

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

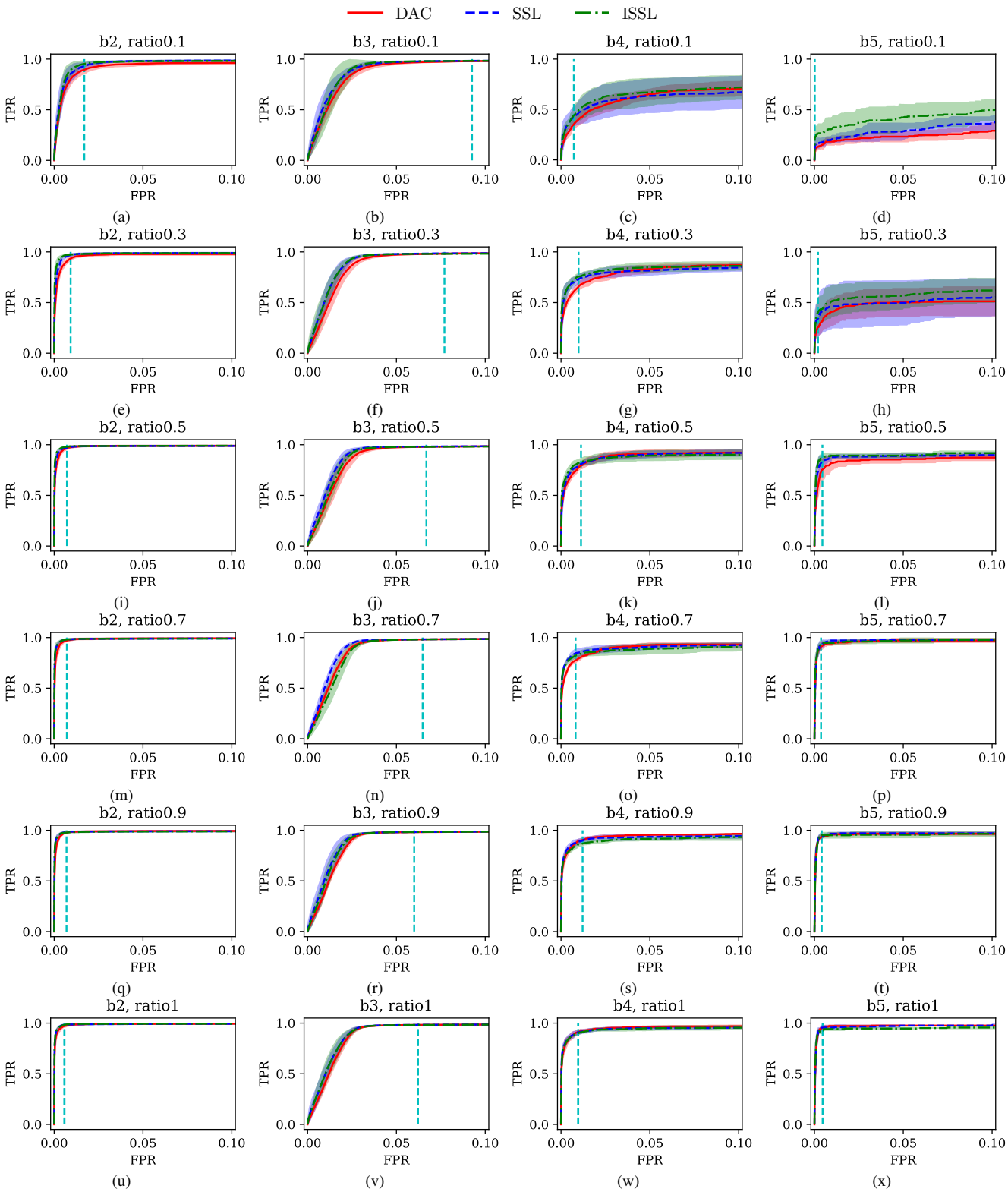


Fig. 10: The Comparison of ROC curves. Both average and standard deviation values are drawn for different runs.

technologies institute, school of computer science Pittsburgh, PA, 2005.

[25] B. Wang and J. Tsotsos, "Dynamic label propagation for semi-supervised multi-class multi-label classification," *Pattern Recognition*, vol. 52, pp. 75–84, 2016.

[26] C. Xiong and T.-K. Kim, "Set-based label propagation of face images," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*. Citeseer, 2012, pp. 1433–1436.

[27] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," 2001.

[28] J. Yu and S. B. Kim, "Consensus rate-based label propagation for semi-supervised classification," *Information Sciences*, vol. 465, pp. 265–284, 2018.

[29] 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2912948, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

for salient object detection," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3919–3930, 2016.

[30] Y. Zhang, Y. Liu, F. Weninger, and B. Schuller, "Multi-task deep neural network with shared hidden layers: Breaking down the wall between emotion representations," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 4990–4994.

[31] L. Lin, K. Wang, D. Meng, W. Zuo, and L. Zhang, "Active self-paced learning for cost-effective and progressive face identification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 1, pp. 7–19, 2018.

[32] S. Zhou, J. Wang, D. Meng, X. Xin, Y. Li, Y. Gong, and N. Zheng, "Deep self-paced learning for person re-identification," *Pattern Recognition*, vol. 76, pp. 739–751, 2018.

[33] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan, "Semi-supervised hotspot detection with self-paced multi-task learning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2019.

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

[35] 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.

[36] 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.

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

[38] 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.

[39] 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

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] 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.

Ying Chen received the B.S. degree in microelectronics and finance from Xi'an Jiaotong University, Xi'an, China, in 2013. She is currently pursuing the Ph.D. degree at the Department of Microelectronics and Solid State Electronics, Institute of Microelectronics of the Chinese Academy of Sciences. Her research interests include physical design and design for manufacturability.

Yibo Lin (S'16-M'18) received the B.S. degree in microelectronics from Shanghai Jiaotong University in 2013, and his Ph.D. degree from the Electrical and Computer Engineering Department of the University of Texas at Austin in 2018. He is currently a postdoctoral researcher at the same university. His research interests include physical design, machine learning applications, emerging technology in VLSI CAD, and hardware security.

He is a recipient of the Best Paper Award at Integration, the VLSI Journal 2018, Franco Cerrina Memorial Best Student Paper Award at SPIE Advanced Lithography Conference 2016, and the University Continuing Fellowship at the University of Texas at Austin in 2017. He has interned at Toshiba, IMEC, Cadence, and Oracle.

Tianyang Gai received the B.S. degree in Physics and Microelectronics from Shandong University, Qingdao, China, in 2016. He is currently pursuing the M.S. degree at the Department of Microelectronics and Solid State Electronics, Institute of Microelectronics of the Chinese Academy of Sciences. His research interests include physical design and design for manufacturability.

Yajuan Su received her B.S. And M.S. degree in microelectronics from University of Electronic Science and Technology of China in 1995 and 1998 respectively, and the Ph.D degree in microelectronics from Tsinghua University in 2005. She is currently an associate professor in Institute of Microelectronics of the Chinese Academy of Sciences. Her research interests include graphene devices and MEMS/NEMS switching devices.

Yayi Wei received his M.S. degrees in Electrics from Institute of Electrics, Chinese Academy of Sciences in 1992, and the Ph.D. degree from Max Planck Institute for Solid State Research/ University Stuttgart in 1998. He was a postdoctoral researcher in Oak Ridge National Laboratory from 1998 to 2001, a senior/staff engineer in Infineon Technologies from 2001 to 2007, a senior staff scientist in AZ Electronic Materials from 2007 to 2008, a principal member of technical staff in Global Foundries from 2009 to 2013. Currently, he is a professor in Institute of Microelectronics of the Chinese Academy of Sciences. His research interests include immersion lithography process and computational lithography, lithography materials and equipment.

David Z. Pan (S'97-M'00-SM'06-F'14) received his B.S. degree from Peking University, and his M.S. and Ph.D. degrees from UCLA. From 2000 to 2003, he was a Research Staff Member with IBM T. J. Watson Research Center. He is currently the Engineering Foundation Professor at the Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, physical design, analog design automation, and CAD for emerging technologies such as 3D-IC and nanophotonics. He has published over 340 technical papers, and is the holder of 8 U.S. patents. He has graduated 25 PhD students who are now holding key academic and industry positions.

He has served as a Senior Associate Editor for ACM Transactions on Design Automation of Electronic Systems (TODAES), an Associate Editor for IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on Very Large Scale Integration Systems (TVLSI), IEEE Transactions on Circuits and Systems PART I (TCAS-I), IEEE Transactions on Circuits and Systems PART II (TCAS-II), IEEE Design & Test, Science China Information Sciences, Journal of Computer Science and Technology, IEEE CAS Society Newsletter, etc. He has served in the Executive and Program Committees of many major conferences, including DAC, ICCAD, ASPDAC, and ISPD. He is the ASPDAC 2017 Program Chair, ICCAD 2018 Program Chair, DAC 2014 Tutorial Chair, and ISPD 2008 General Chair.

He has received a number of awards for his research contributions, including the SRC 2013 Technical Excellence Award, DAC Top 10 Author in Fifth Decade, DAC Prolific Author Award, ASP-DAC Frequently Cited Author Award, 16 Best Paper Awards at premier venues (GLSVLSI 2018, VLSI Integration 2018, HOST 2017, SPIE 2016, ISPD 2014, ICCAD 2013, ASPDAC 2012, ISPD 2011, IBM Research 2010 Pat Goldberg Memorial Best Paper Award, ASPDAC 2010, DATE 2009, ICICDT 2009, SRC Techcon in 1998, 2007, 2012 and 2015) plus 14 additional Best Paper Award nominations at DAC/ICCAD/ASPDAC/ISPD, Communications of the ACM Research Highlights (2014), ACM/SIGDA Outstanding New Faculty Award (2005), NSF CAREER Award (2007), SRC Inventor Recognition Award three times, IBM Faculty Award four times, UCLA Engineering Distinguished Young Alumnus Award (2009), UT Austin RAISE Faculty Excellence Award (2014), and many international CAD contest awards, among others. He is a Fellow of IEEE and SPIE.