ISPD '20, September 20–23, 2020, Taipei, Taiwan
Proceedings published March 29, 2020

DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network

Rongjian Liang
Texas A&M University
College Station, Texas, United States
liangrj14@tamu.edu

Hua Xiang
IBM Research
Yorktown Heights, New York, United States
huaxiang@us.ibm.com

Diwesh Pandey
IBM Inc
Bengaluru, Karnataka, India
dpandey87@in.ibm.com

Lakshmi Reddy, Shyam Ramji and Gi-Joon Nam
IBM Research
Yorktown Heights, New York, United States
{reddyl,ramji,gnam}@us.ibm.com

Jiang Hu
Texas A&M University
College Station, Texas, United States
jianghu@tamu.edu

ABSTRACT

As the semiconductor process technology advances into sub-10nm regime, cell pin accessibility, which is a complex joint effect from the pin shape and nearby blockages, becomes a main cause for DRC violations. Therefore, a machine learning model for DRC hotspot prediction needs to consider both very high-resolution pin shape patterns and low-resolution layout information as input features. A new convolutional neural network technique, J-Net, is introduced for the prediction with mixed resolution features. This is a customized architecture that is flexible for handling various input and output resolution requirements. It can be applied at placement stage without using global routing information. This technique is evaluated on 12 industrial designs at 7nm technology node. The results show that it can improve true positive rate by 37%, 40% and 14% respectively, compared to three recent works, with similar false positive rates.

ACM Reference Format:
, Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji and Gi-Joon Nam, and Jiang Hu. 2020. DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network. In *Proceedings of the 2020 International Symposium on Physical Design (ISPD '20), March 29-April 1, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3372780.3375560

1 INTRODUCTION

Freedom from Design Rule Violation (DRV) is a fundamental requirement for chip designs. Yet, it is challenging to achieve at sub-10nm process nodes due to a compounding effect arising from multiple factors. First, design rules at advanced technology nodes are immensely complex. Second, increasingly high chip complexity results in very dense devices on silicon die and therefore makes routing resource very scarce. Last but most importantly, cell pins are very difficult to access due to compact cell layout and cause the well-known pin accessibility problem [6]. In general, DRVs in chip layout are not known until detailed routing is completed. However, it is often hard to fix all DRVs by routing alone and cell placement needs to be modified as well. As such, it is important to predict Design Rule Checking (DRC) hotspots in early layout stages, such as cell placement, and take preemptive measure to avoid excessive iterations back and forth between placement and routing.

Closely related to DRVs, routability and routing congestion predictions have been extensively studied in the past [10, 13, 15, 17]. However, routability is not necessarily equivalent to freedom of DRVs as there can be DRVs after routing completion. Also, a congested layout does not necessarily mean it is not routable, although the two issues are correlated. In general, routing congestion can be seen during or after global routing, routability can be confirmed at detailed routing and DRVs are clearly known after design rule checking. Therefore, the predictions of congestion, routability and DRVs are progressively more difficult. There are a few works that predict routability in terms of DRVs [2, 14, 16]. However, these works did not explicitly emphasize pin accessibility, which is a main cause for many DRVs in nanometer designs. The pin accessibility issue has been investigated in [6, 12, 18], however, they are mostly focused on pin accessibility alone while ignoring other DRVs. The most recent work on pin accessibility prediction is restricted to only M2 short. However, M2 short does not always dominate DRVs and there are many other types of DRVs that need to be considered as well. Our investigation on an industrial design shows that M2 short accounts for only about 20% of total DRVs. In the predictions, another important issue is whether or not global routing information is required as global routing is time consuming and makes the prediction difficult to be used frequently. Among existing works, only those for relatively easy congestion prediction [17] and specific kinds of DRV prediction [14, 18] can be performed without global routing while general DRC hotspot predictions [2, 16] still rely on global routing.

In this work, to the best of our knowledge, we introduce the first general DRC hotspot prediction technique that emphasizes pin

ISPD '20, March 29-April 1, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7091-2/20/03…$15.00
https://doi.org/10.1145/3372780.3375560

accessibility and does not rely on global routing. This is a machine learning approach based on convolutional network. A key difference from other machine-learning-based predictions [2, 14, 16–18] is that a customized convolutional network architecture, J-net, is developed as opposed to a plugin use of existing machine learning engines. This customization is used to address the mixed input resolution issue. The pin shape features for emphasizing pin accessibility has a resolution several orders of magnitude higher than other layout features such as pin density. Simply scaling low resolution features into high resolution would evidently cause inefficiency in computing. The proposed J-net architecture can be automatically tuned for various feature resolutions from different designs. We demonstrate the effectiveness of proposed approach on industrial designs at 7nm process node. In a relatively relaxed yet realistic setting, the proposed J-Net achieves 93% true positive rate (TPR) while the TPR of the previous work [16] is only 56%, and the TPRs of the extensions of works in [17] and [18] are 52% and 79% respectively, at about the same false positive rate. In a stricter setting, it can achieve 78.5% true positive rate (with false positive rate being 8.9%) when performing predictions on unseen designs. This true positive rate is 37% and 40% higher than those of [16] and [17] at about the same negative positive rate, respectively.

The rest of the paper is organized as follows. Section 2 briefly introduces the background on semantic segmentation to help understand our method. Relation between our work and previous ones is described in Section 3 and the details of proposed customized convolutional network for DRC hotspot prediction is presented in Section 4. Section 5 shows our experimental setups and results. Section 6 concludes this paper.

2 BACKGROUND ON SEMANTIC SEGMENTATION

Our prediction method is a customization of an existing semantic segmentation technique. The background on semantic segmentation is briefly introduced here in order to help understand our method. Different from image classification, whose output tells the class of a given input image, semantic segmentation outputs the class for every pixel. As such, the output of semantic segmentation can be treated as another image with the same size and resolution as the input image.

A popular framework for semantic segmentation is Fully Convolutional Network (FCN) [9]. An FCN is composed by multiple layers of convolution, ReLU, pooling and transposed convolution operations. A convolution layer consists of a set of trainable filters (kernels) sliding over the whole image. The step of sliding is called *stride*. A pooling layer, also referred to as down-sampling layer, applies a max/mean/average filter to the input, with a stride same as the filter size. The left side of Fig. 1 shows an example of max-pooling, after which the resolution is reduced by a half. The right side of Fig. 1 is an example of transposed convolution, which realizes up-sampling. FCN organizes the order of different layers according to the so-called encoder-decoder architecture. An encoder maps the raw input to compact feature representations, while a decoder processes feature representations to produce an output. Since all its component layers do not require their inputs to be fixed sizes, FCN can take an image with variable size as input.

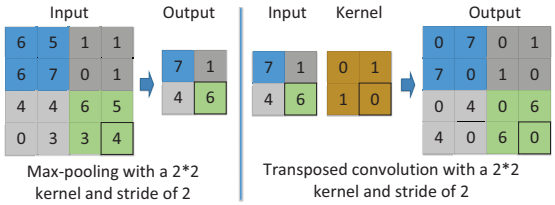

Figure 1: Examples of max-pooling and transposed-convolution operation.

U-Net [11] is a variant of FCN and designed for applications with relatively small amount of training data. It mainly consists of three parts, i.e. the encoding path, the decoding path and the bottleneck unit connecting two paths, as shown in Fig. 2. The encoding path consists of repeated downing-sampling units, denoted by DOWN in Fig. 2. The decoding path consists of repeated up-sampling units denoted by UP in Fig. 2. At each up-sampling unit, the up-sampled map is concatenated with the feature map from the short-cut. The bottleneck unit is simply the stacking of a few convolution layers. Short-cuts exist at every resolution level and help U-Net capture both local and global information. The U-Net architecture resembles

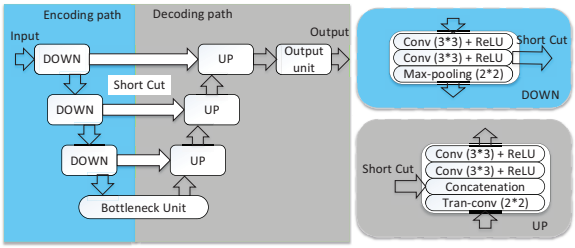

Figure 2: U-Net architecture.

the multi-grid method for solving partial differential equations [1] and multilevel optimization in EDA [4], where a system is coarsened for several iterations, the main computation is performed at the coarsest level, and then the coarse solution is iteratively refined to the original granularity level.

3 RELATION WITH PREVIOUS WORKS

Early related works were mostly focused on routing congestion prediction. There are generally two types of approaches. One is analytical estimation based on net locations, such as [10, 13]. Such approach is fast to compute, but it can be very inaccurate and is hardly adopted in industry. The other is trial routing, which is much more accurate than analytical estimations but very time consuming. The pin accessibility problem was considered in several placement works [6, 12, 15], where an accessibility is evaluated by some simple score functions. This is very helpful for cell placement to address the pin accessibility problem but does not consider routability problems caused by congestion.

Recently, machine learning techniques were investigated to obtain accurate yet fast prediction on routability and/or DRVs [2, 7,

14, 16–19]. The technique of [2] is based on support vector machine and requires global routing information as the machine learning model input. A neural network approach is proposed in [14] for predicting only short violations. In [16], a Convolutional Neural Network (CNN) model without using global routing is developed for classifying the overall number of DRVs and an FCN [9] model with global routing as input is designed for DRC hotspot prediction. A neural network ensemble approach for DRC hotspot prediction is described in [19]. However, it still requires global routing information as an input. Another neural network approach that does not rely on global routing is reported in [7]. However, this method predicts only the total number of DRVs. In [17], conditional generative adversarial network is explored for routing congestion prediction without considering DRVs and this technique is validated on FPGA designs. A CNN-based pin accessibility prediction technique is introduced in [18]. However, it focuses only on M2 short violations and not for general DRVs. This very specific application allows it to use only two adjacent standard cells as the model input. Such small receptive field can easily fail for general DRVs, which may be caused by complex joint effects in a large region. The work in [18] can be extended to take information of a larger region (patch) into account to tell whether the tile is a DRC hotspot, as in [2, 14]. However, in the work [18], mixed resolution input channels have to be mapped and concatenated to a one-dimension feature vector, which then is processed by a sequence of fully connected layers. Such feature fusion scheme would become inefficient when a large patch size is used so that the concatenated feature vector would become very long and large fully connected layers have to be employed.

Machine learning-based DRC hotspot prediction can be realized with two different strategies. One is to perform semantic segmentation on entire layout like in [16] and our work. Alternatively, one can build a model for classifying a single tile (or global routing cell) using this tile and its neighboring tiles as input features [2, 14, 18]. This approach invokes many small models many times for all tiles and thus the feature data of one tile is fetched many times. By contrast, the feature data of one tile is fetched only once in semantic segmentation, and the computation among different patches can be highly amortized over the overlapping regions of those patches.

Overall, general DRC hotspot prediction without global routing information is still far from a solved problem. Our work is the first systematic study on this subject, to the best of our knowledge. Compared to most of previous machine learning-based works, which are plugin use of existing machine learning techniques with parameter tuning, our work is a customized machine learning approach to address mixed resolution issues caused by coarse-grained routing congestion and fine-grained pin accessibility.

4 CUSTOMIZED CONVOLUTIONAL NETWORK FOR DRC HOTSPOT PREDICTION

4.1 Overview and the Mixed Resolution Issue

We propose a customized convolutional network architecture, called J-Net, for DRC hotspot prediction with an explicit emphasis on pin accessibility. It takes a cell placement solution including IPs as input, and outputs a 2D array, where each entry corresponds to a tile and tells if each tile is a DRC hotspot. The tile size is similar to that of a global routing cell (gcell). The raw output of J-Net is a set of real numbers in $[-1, 1]$. By setting a threshold of 0, positive numbers indicate a hotpot, and non-positive numbers indicate a non-hotspot.

There is a key difference between our DRC hotspot prediction and conventional semantic segmentation. The output of semantic segmentation is an image of the same or a bit lower resolution as the input image. In our case, the image resolution for pin accessibility is about two orders of magnitude finer than the hotspot map. The resolution of pin image feature must be a half wire pitch or finer to ensure the recognition of a pin shape. The hotspot map is based on routing tiles and a tile typically covers 20 routing tracks or more. Therefore, a tile easily contains thousands of pin image pixels. Directly plugin use of U-Net requires us to adopt very high resolution hotspot maps, which costs a large and unnecessary computing overhead. Moreover, we need to consider tile-based features and hence features with huge resolution difference need to be simultaneously input to the model. The mixed resolution issue motivates us for a customized approach.

4.2 J-Net Convolutional Network Architecture

The proposed J-Net is an extension of the U-Net architecture [11] and can handle highly different resolutions between different input channels, and between input and output. When all input channels and the output have the same resolution, J-Net degenerates to U-Net.

A J-Net follows the typical encoder-decoder architecture, as illustrated in Fig. 3. The encoding path consists of repeated down-sampling units, which are indicated by *DOWN* in Fig. 3. Similarly, the decoding path is formed by a sequence of up-sampling units. For simplicity, we refer to the kernel size of max-pooling layer in a down-sampling unit as the kernel size of the down-sampling unit, and to the kernel size of transposed convolution layer in an up-sampling unit as the kernel size of the up-sampling unit. The $k * k$ in parenthesis in Fig. 3 represent the kernel sizes of the down-sampling and up-sampling units. Compared to U-Net, four main modifications are made in J-Net.

(1) Input channels of different resolutions are fed into different levels at the encoding path. If an input channel is fed to a middle level of the encoding path, it is concatenated with the feature representations of the same resolution produced from the previous upper level. By doing so, J-Net can easily accommodate input channels with highly different resolutions.
(2) The number of decoder levels is less than that of encoder. Therefore, the output resolution can be significantly lower than the top input channel at the encoder path. This configuration makes the encoder-decoder architecture look a flipped letter "J" and this is why this architecture is called J-Net.
(3) The number of convolution operations in each down-sampling/up-sampling unit is reduced from 2 to 1. This change greatly decreases the training parameters as well as the need for large amount of training data. This is particularly appealing in EDA applications, where training data is difficult to obtain.

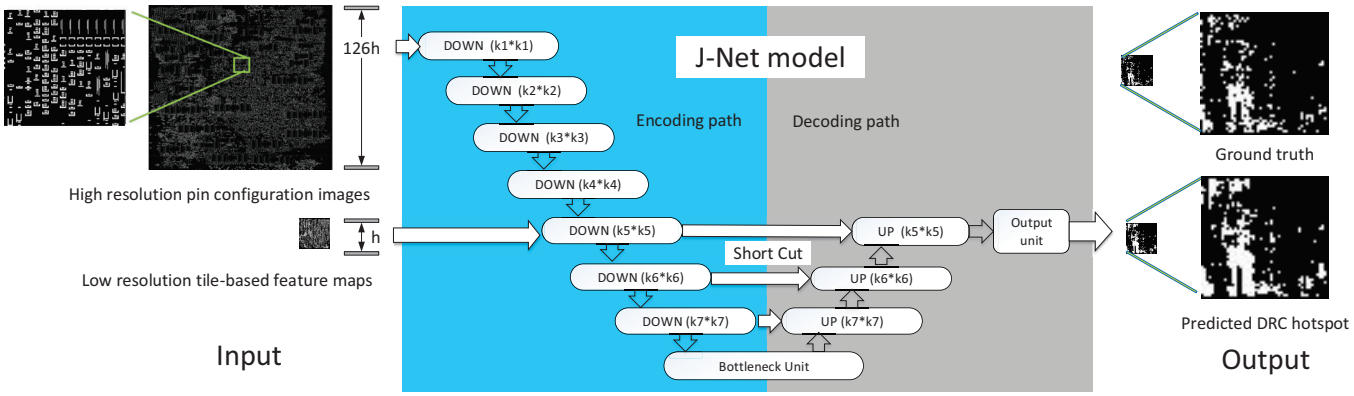

Figure 3: DRC hotspot prediction via J-Net.

(4) The kernel size of down-sampling/up-sampling units in U-Net is usually fixed at 2×2. By contrast, J-Net treats the kernel sizes as variable parameters and can automatically tune their values for various input channel resolutions caused by different process nodes and feature selections.

4.3 Kernel Size Tuning and Branch Path Addition

In constructing a J-Net, the number of encoding path levels and the kernel sizes of its down-sampling units are automatically tuned. The tuning is to ensure that the feature representations from different input channels have the same resolution when they are concatenated at certain level. Sometimes, such tuning is insufficient and a new branch needs to be added to the encoding path. We illustrate the tuning and branch addition of two cases through the examples as follows.

- Case 1: The resolution of one input channel is the divisor of the other channel. For example, channel1 has resolution 12600×12600 and channel2 has resolution 100×100. The relative resolution between them is $12600/100 = 126$. Prime factorization is performed on the relative resolution to obtain $126 = 2 \times 3 \times 3 \times 7$. Then, four down-sampling units are generated between channel1 and channel2 as shown in Fig. 3, with $k_1 = 2, k_2 = 3, k_3 = 3$ and $k_4 = 7$. The other down-sampling units have their kernel sizes as the default value of U-Net or other empirical values.
- Case 2: No input channel has resolution as divisor of the other channel. For example, channel1 has resolution 400×400 and channel2 has resolution 300×300. In this case, we first find the greatest common factor of the resolutions of the two channels, which is 100 in this example. Next, we compute the relative resolutions of the channels to the greatest common factor, which are 4 and 3 here. Then, factorization is performed to obtain $4 = 2 \times 2$ and $3 = 3$. As a result, there are two branches forming an encoding tree as shown in Fig. 4, as opposed to the single encoding path in Fig. 3.

Mathematically, there could be other cases. For example, two input channels with resolutions that prime to each other, such as 11×11 and 97×97. However, it is extremely rare that such strange cases can happen in practice.

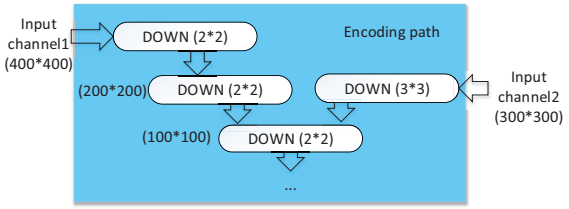

Figure 4: An example of encoding-path for two input channels of case 2.

4.4 Feature Selection

4.4.1 Pin Configuration Images. Complicated patterns formed by nearby pins and routing blockages have great impact on local routability at advanced technology nodes. In order to capture the effects of such patterns, pin configuration images are generated for each layout, which have sufficiently high resolution to accurately capture the location and shapes of pins as well as routing blockages of the entire layout. An example of pin configuration image is shown in Fig. 3. In the image, pins and blockages are assigned with different grey scales as they play different roles in pin accessibility. A relatively large pin makes it more accessible while a large blockage degrades pin accessibility. To allow machine learning models to identify complex patterns formed by pins locating at different metal layers, one pin configuration image is generated for one layer where pins reside. Four images are generated for each design in total for our benchmark designs.

4.4.2 Tile-based Features Maps. Besides the pin configuration images, some netlist and layout features are also extracted for each tile (or global routing cell). Then, the feature values of all the tiles are combined into feature maps according to their locations in the layout, and employed as inputs to J-Net. Extracted features include:

- Routing resource describing the percentage of a tile area that is occupied by IPs, which affects the number of available routing tracks in the tile.
- Connection features such as the number of local nets, whose pins are completely inside the tile, and the number of global nets, which connect to pins outside the tile.

For the technology node and test cases in our setup, each tile is $1.26\mu m \times 1.26\mu m$ large, and corresponds to 126×126 pixels in a pin configuration image. Hence, the resolution of pin configuration images are 126 times of tile-based feature maps. Both pin configuration images and tile-based feature maps are generated based on the information collected at the placement stage, via our self-developed scripts.

Note that the relative resolutions between different input channels are fixed after training the J-Net model. However, thanks to the properties inherited from the FCN architecture, the trained J-Net model can be applied to placement instances of various sizes, as long as the resolutions (unnecessary the size) of input channels keep the same as those used in the training process.

4.5 Data Augmentation

Due to the difficulty for obtaining real-world 7nm designs, the amount of available training data is quite limited. We employ the following data augmentation techniques in the training process to address this problem.

- Cropping layouts into small ones. For large layouts, we crop their tile-based feature maps and pin configuration images into smaller ones with a sliding window. The size of the window is 50-tile-by-50-tile. On one hand, the window size is sufficiently large such that this cropping does not significantly affect the model training. On the other hand, the amount of training data is effectively increased. The stride of sliding window is 40× single-tile-length, slightly less than the length of the sliding window. As such, the cropped feature maps (so as the pin configuration images) have small overlap with each other. By allowing the overlap, more training data can be obtained. This augmentation technique can increase the number of training samples by about 10 times, since many designs in our benchmarks contain more than 150-by-150 tiles. After cropping, the size of tile-based feature maps becomes 50-by-50, while the size of pin configuration image becomes 6300-by-6300.
- Random flipping: In the training process, every time a data sample is fetched, random flipping is performed on it. The layouts after flipping should have the same routability as that before the flipping. However, the inputs to the model become different and can still enhance the training.

5 EXPERIMENT

5.1 Testcases and Data Collection

Several industrial designs at 7nm process node were used as testcases. We generated multiple placement instances for each design by performing a state-of-the-art industrial physical synthesis flow with various (but still realistic) parameter settings. After placement, pin configuration images and tile-based feature maps were generated. Then, all placement solutions were routed and design rule checking was performed to obtain the ground truth DRC hotspot maps, which were used as model training labels.

Table 1: Testcase characteristics.

Design	Chip Size (μm^2)	#IPs	#Gates	#Nets
D1	59.52×62.21	0	13569	15552
D2	59.52x×129.60	0	10796	17843
D3	211.20×391.39	0	328611	338846
D4	79.68×84.24	0	18283	25068
D5	122.88×233.28	0	68393	86640
D6	122.88×95.90	0	46001	49337
D7	138.24×80.35	0	35627	38456
D8	284.16×95.90	0	100566	108187
D9	76.80×401.76	1	52659	70338
D10	249.60×316.22	3	149456	191513
D11	280.32×489.89	16	81384	105678
D12	253.44×671.33	28	200454	247271

In total, there are 12 designs in the testcase suite, four of which have IPs (or macros). Tab. 1 summarizes the characteristics of the testcases. By varying optimization settings for each design, we obtained 166 placement instances. With the data augmentation techniques introduced in Section 4.5, the number of data samples was increased to 2268.

5.2 Comparison Setup

The effectiveness of the proposed techniques was evaluated through comparisons with recent previous works [16–18], various machine learning models and combinations of different features.

In the experiment, we compared five different machine learning models, three of which are based on recent previous works [16–18].

- **J-Net.** This is our proposed model.
- **U-Net.** Direct plugin use of the U-Net architecture [11]. As U-Net restricts that all input channels and the output have the same resolution, only tile-based features are accommodated here.
- **FCN.** The Fully Convolutional Network approach in [16]. The features used are the same as in [16].
- **cGAN.** The conditional Generative Adversarial Net approach is an extension of the work in [17]. The work was for routing congestion prediction for FPGA designs, which is different from our hotspot map prediction. We did our best to implement as close to [17] as possible. GAN is notoriously difficult to train and we adopted the techniques described in [3] to improve the training.
- **CNN.** The Convolutional Neural Network approach is an extension of the work in [18]. In contrast to considering only the information in a small region covering two nearby cells as in [18], the implemented CNN model takes the pin configuration images covering one single tile and the corresponding tile features as input to classify one tile each time.

In evaluating J-Net and U-Net, we tested several different combinations of the following features.

- **H**: high resolution pin configuration images.
- **R**: routing resource features as described in Section 4.4.
- **Cn**: connection features as described in Section 4.4.
- **Cg**: congestion map based on the global routing results.
- **D**: density features such as logic gate pin density, clock pin density, logic cell density, filler cell density, etc.

As in [16], the FCN features include R, Cg, D and additionally RUDY [13], which is a score function at each location obtained by overlapping net bounding boxes covering this location. Similar to [17], the features of cGAN contain R, D and connectivity images, which is a set of straight lines connecting pins of the same net. The input of CNN includes H, R and Cn.

5.3 Training and Testing Schemes

For the labelled data that we can obtain, two different partition schemes between training and testing sets were applied for the model evaluation.

- **Scheme1**: This partition scheme was applied on the 166 placement instances. We randomly chose 80% of the placement instances as training data, which were further increased by the data augmentation techniques described in Section 4.5. The other 20% of the placement instances were used for testing. Please note testing such placement instances may involve some same designs as those in the training set, although their placements were different. This scenario may practically exist in reality, where a few routing and DRC results of a design have already been obtained before applying machine learning on new placement solutions of the same design. Similar schemes have also been employed in previous works [2, 17].
- **Scheme2**: Twelve rounds of experiments were performed and their results were averaged. In each round, all placement instances from 11 designs were taken as training data and applied with data augmentation. Placement instances of the only remaining design were used as testing data. Each round had a distinct testing design. This is a stricter testing scheme than scheme1 as not only the testing placement instances but also the testing design are completely unseen in the training.

In our data set, only a small portion (about 4%) of the tiles belong to DRC hotspots. In other words, it is a highly imbalanced data set. Weighted loss function, i.e. giving a heavy weight to DRC hotspots during the computation of training loss, as the one used in [2], was employed to address this problem.

5.4 Performance Metrics

Due to the large imbalance between the number of positive samples (hotspots) and negative samples (non-hotspots), simple tile-wise accuracy is no longer an effective measure of prediction performance. For example, consider a case where 90% samples are positive and a model simply predicts all the samples are positive. In this case, accuracy of 90% is obtained although the model almost does nothing useful. In our experiment, the following metrics were employed to evaluate the effectiveness of different approaches.

1) *True Positive Rate (TPR), False Positive Rate (FPR), Precision and F1-score:* Table 2 summarizes the calculation of TP, TN, FP and PN, based on which TPR (also known as recall), FPR, Precision and F1-score can be estimated.

Table 2: Calculation of prediction accuracy.

		Prediction Result	
		Positive	Negative
Ground truth	Positive	TP	FN
	Negative	FP	TN

$$TPR = \frac{TP}{TP+FN}$$
$$FPR = \frac{FP}{FP+TN}$$
$$Precision = \frac{TP}{FP+TP}$$
$$F1 = 2 \times \frac{TPR \times Precision}{TPR+Precision}$$

2) *Area Under Curve (AUC):* Receiver Operating Characteristic (ROC) curve is the TPR vs. FPR curve, which indicates the trade-off between TPR and FPR by varying the classification thresholds. This is similar to power-delay trade-off curves in chip designs. A large AUC of ROC-curve [8] implies better prediction performance. An AUC of ROC-curve of 1 means perfect prediction, and random guesses result in 0.5 AUC. AUC of Precision-Recall (PR) curve [5] is also a commonly used metric for evaluating classification performance. A larger AUC of PR-curve implies better prediction performance too.

5.5 Experiment Results

5.5.1 *Results from Training & Testing Scheme1.*

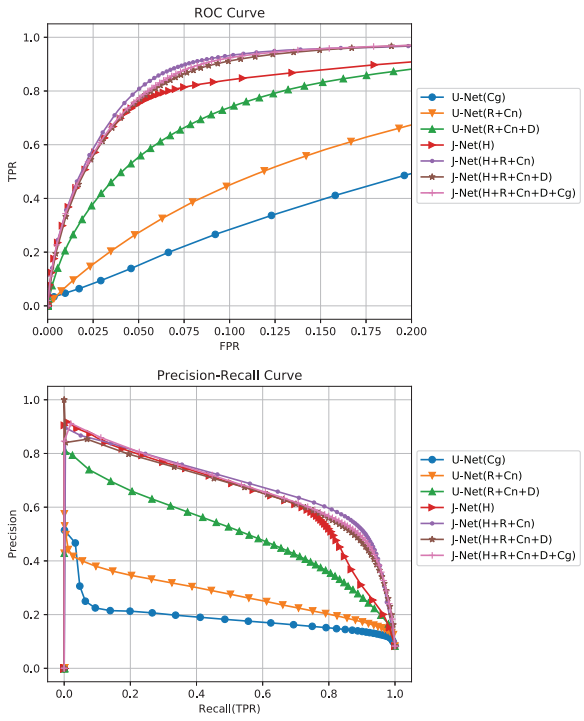

Figure 5: ROC and PR curves for different features (scheme1).

Effects of Different Features on J-Net and U-Net. The ROC and PR curves are shown in Fig. 5 while the results on different metrics

are listed in Tab. 3. One can see that pin configuration images have a great impact on prediction performance. Even when it is used as the only feature (J-Net(H)), the prediction performance is much better than employing numerous tile-based feature maps (U-Net(R+Cn+D)). Although the role of pin configuration maps is critical, the effect of tile-based features is also significant. For FPR around 10%, tile-based features can improve TPR from 83% to 93%. Overall, J-Net can achieve around 93% TPR when FPR is less than 10%. Another observation is that there is a large gap between global routing congestion map (Cg) and DRC hotspot in our testcases. When congestion map is used as the only feature (U-Net(Cg)), the TPR is as less than 30% when the FPR is 9.21%.

Note that using more features as input (compare J-Net (H+R+Cn) with J-Net(H+R+Cn+D+Cg)) does not necessarily enhance the prediction performance, since it might induce the over-fitting problem, given the limited amount of training data.

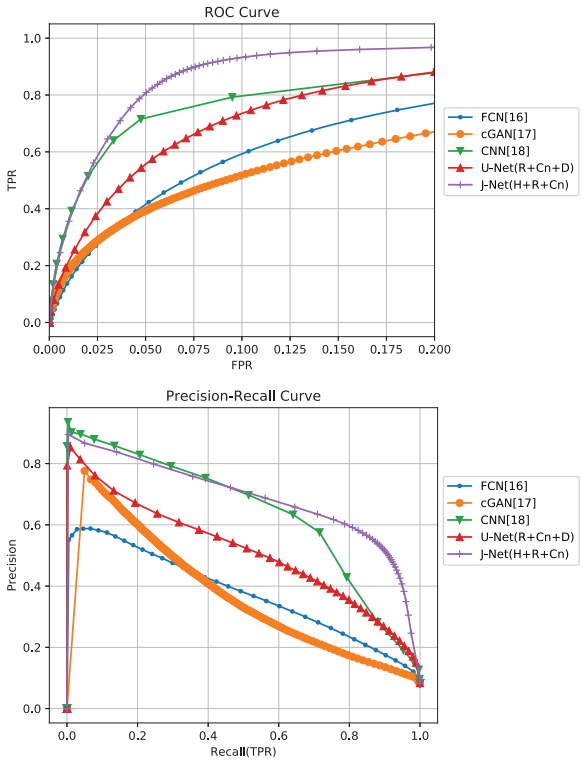

Figure 6: ROC and PR curves of different models and previous works (scheme1).

Comparison Among Different Models and Previous Works. The ROC and PR curves of different models are displayed in Fig. 6 and the numerical results are listed in Tab. 4. Among the models, FCN is based on the previous work [16], and cGAN and CNN are extensions of the works in [17] and [18], respectively. The prediction performance of J-Net is evidently superior to other models. At FPR of around 10%, J-Net achieves TPR of 93%, while the TPRs of FCN, cGAN and CNN are only 56%, 52% and 79%, respectively. One important reason for the inferior performance of the FCN [16] and the cGAN [17] is that they do not consider the pin-accessibility issue, which is a significant cause of DRVs at 7nm process nodes. In contrast to considering only the information in two nearby cells as in [18], we extended the CNN model to take into account the information in one tile each time. But the prediction accuracy is still not as high as that reported in [18], for which one important reason might be that general DRVs rather than only M2 short are considered in our experiment. Using a larger patch size might help improve the prediction accuracy of the CNN approach, but significant overhead in run-time and memory-usage would be induced, as elaborated in Section 3.

5.5.2 Results from Training & Testing Scheme2. For scheme2, the comparisons for different features and different models are summarized in Tab. 5 and Tab. 6, respectively. Each data entry in the tables is the average result of 12 rounds. As scheme2 is a stricter scenario than scheme1, all results degrade compared those of scheme1. However, key observations are not changed here. First, pin configuration map is perhaps the most important feature. Second, J-Net significantly outperforms the other models including the previous works [16, 17]. Compared with FCN[16], cGAN[17] and U-Net(R+Cn+D), our J-Net(H+R+Cn) achieves 37%, 40% and 22% higher TPR, respectively, at similar FPRs.

5.5.3 Training, Feature Extraction and Inference Time. For each round of scheme1 training, the training time of FCN or U-Net on a GeForce RTX 2080-Ti GPU is about 2 hours. cGAN takes longer time to train while training time of J-Net or CNN is around 30 hours. Since the model can be reused repeatedly for different and unseen designs of the same technology node, the amortized training cost is limited. For feature extraction and inference time, the dominating factor is whether or not global routing information is required. Consider circuit D8 as an example, which has about 101K gates and 108K nets. Without global routing, the total run-time on feature extraction, data transformation and inference by each of cGAN, CNN, U-Net and our J-Net is less than 1 minute for one layout design. However, the global routing of D8 takes several hours and it is required by the FCN method [16].

6 CONCLUSION AND FUTURE RESEARCH

A machine learning approach is proposed for predicting DRC hotspot at cell placement stage for sub-10nm designs. It is a customized machine learning architecture that can simultaneously take high resolution pin images and low resolution placement information as input features. This approach is evaluated on twelve industrial designs at 7nm process node and compared with three recent works. The results show that its prediction performance considerably outperforms the previous works and has fast feature extraction and inference time as no global routing information is required. In future research, we will evaluate its impact to overall design flows. Further, we will study how to handle non-deterministic behaviors of parallel detailed routers.

7 ACKNOWLEDGMENTS

This work is partially supported by Semiconductor Research Corporation Tasks 2810.021 and 2810.022 through UT Dallas' Texas Analog Center of Excellence (TxACE).

Table 3: Results from different features for U-Net and J-Net (scheme1).

			FPR≤ 5%				FPR≤ 10%			
	AUC (ROC)	AUC (PR)	FPR	TPR	precision	F1-score	FPR	TPR	precision	F1-score
U-Net(Cg)	0.738	0.192	4.58%	13.91%	21.48%	16.89%	9.21%	26.61%	20.66%	23.26%
U-Net(R+Cn)	0.835	0.277	4.79%	26.40%	33.21%	29.42%	9.83%	44.48%	28.98%	35.09%
U-Net(R+Cn+D)	0.913	0.509	4.58%	52.94%	51.04%	51.97%	9.60%	72.89%	40.63%	52.18%
J-Net(H)	0.937	0.647	4.84%	74.67%	58.19%	65.41%	9.12%	83.34%	45.16%	58.58%
J-Net(H+R+Cn)	0.958	0.686	4.67%	78.64%	60.30%	68.26%	9.75%	92.98%	46.24%	61.77%
J-Net(H+R+Cn+D)	0.958	0.666	4.76%	75.19%	58.77%	65.97%	9.93%	91.12%	45.26%	60.48%
J-Net(H+R+Cn+D+Cg)	0.959	0.680	4.74%	75.77%	59.03%	66.36%	9.49%	91.72%	46.56%	61.77%

Table 4: Comparison among different models and with previous works (scheme1).

Metric	FCN[16]	cGAN[17]	CNN[18]	U-Net	J-Net
AUC (ROC)	0.867	0.818	0.927	0.913	0.958
AUC (PR)	0.375	0.360	0.639	0.509	0.686
FPR	9.01%	9.93%	9.50%	9.60%	9.75%
TPR	56.48%	51.67%	79.2%	72.89%	92.98%
Precision	35.12%	31.94%	42.9%	40.63%	46.24%
F1-score	43.31%	39.48%	55.7%	52.18%	61.77%
Need GR?	Y	N	N	N	N

Table 5: Results from different features(scheme2).

	AUC(ROC)	AUC(PR)
U-Net(R+Cn)	0.731	0.250
U-Net(R+Cn+D)	0.854	0.415
J-Net(H)	0.883	0.582
J-Net(H+R+Cn)	0.913	0.604
J-Net(H+R+Cn+D)	0.900	0.586
J-Net(H+R+Cn+D+Cg)	0.911	0.599

Table 6: Comparison among different models and with previous works (scheme2).

Metric	FCN[16]	cGAN[17]	U-Net	J-Net
AUC (ROC)	0.788	0.714	0.854	0.913
AUC (PR)	0.279	0.287	0.415	0.604
FPR	9.10%	9.69%	9.47%	8.90%
TPR	41.04%	38.1%	56.08%	78.47%
Precision	31.28%	29.9%	35.80%	46.16%
F1-score	32.25%	29.9%	39.32%	53.95%

REFERENCES

[1] William L Briggs, Steve F McCormick, et al. 2000. *A multigrid tutorial*. Vol. 72. Siam.
[2] Wei-Ting J Chan, Pei-Hsin Ho, Andrew B Kahng, and Prashant Saxena. 2017. Routability optimization for industrial designs at sub-14nm process nodes using machine learning. In *Proceedings of ACM International Symposium on Physical Design (ISPD)*. 15–21.
[3] Soumith Chintala. 2016. Tips and tricks to make GANs work. https://github.com/soumith/ganhacks
[4] Jason Cong and Joseph R Shinnerl. 2013. *Multilevel optimization in VLSICAD*. Vol. 14. Springer Science & Business Media.
[5] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of International Conference on Machine Learning (ICML)*. 233–240.
[6] Yixiao Ding, Chris Chu, and Wai-Kei Mak. 2017. Pin accessibility-driven detailed placement refinement. In *Proceedings of ACM International Symposium on Physical Design (ISPD)*. 133–140.
[7] Yu-Hung Huang, Zhiyao Xie, Guan-Qi Fang, Tao-Chun Yu, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2019. Routability-Driven Macro Placement with Embedded CNN-Based Prediction Model. In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 180–185.
[8] Mingrui Liu, Xiaoxuan Zhang, Zaiyi Chen, Xiaoyu Wang, and Tianbao Yang. 2018. Fast stochastic AUC maximization with O (1/n)-convergence rate. In *Proceedings of International Conference on Machine Learning (ICML)*. 3195–3203.
[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3431–3440.
[10] Jinan Lou, Shashidhar Thakur, Shankar Krishnamoorthy, and Henry S Sheng. 2002. Estimating routing congestion using probabilistic analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21, 1 (2002), 32–41.
[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI)*. 234–241.
[12] Jaewoo Seo, Jinwook Jung, Sangmin Kim, and Youngsoo Shin. 2017. Pin accessibility-driven cell layout redesign and placement optimization. In *Proceedings of IEEE/ACM Design Automation Conference (DAC)*. 1–6.
[13] Peter Spindler and Frank M. Johannes. 2007. Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement. In *Proceedings of Conference on Design, Automation and Test in Europe (DATE)*. 1226–1231.
[14] Aysa Fakheri Tabrizi, Logan Rakai, Nima Karimpour Darav, Ismail Bustany, Laleh Behjat, Shuchang Xu, and Andrew Kennings. 2018. A machine learning framework to identify detailed routing short violations from a placed netlist. In *Proceedings of IEEE/ACM Design Automation Conference (DAC)*. 1–6.
[15] Taraneh Taghavi, Zhuo Li, Charles Alpert, Gi-Joon Nam, Andrew Huber, and Shyam Ramji. 2010. New placement prediction and mitigation techniques for local routing congestion. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 621–624.
[16] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2018. RouteNet: routability prediction for mixed-size designs using convolutional neural network. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.
[17] Cunxi Yu and Zhiru Zhang. 2019. Painting on Placement: Forecasting Routing Congestion using Conditional Generative Adversarial Nets. In *Proceedings of IEEE/ACM Design Automation Conference (DAC)*. 1–6.
[18] Tao-Chun Yu, Shao-Yun Fang, Hsien-Shih Chiu, Kai-Shun Hu, Philip Hui-Yuh Tai, Cindy Chin-Fang Shen, and Henry Sheng. 2019. Pin Accessibility Prediction and Optimization with Deep Learning-based Pin Pattern Recognition. In *Proceedings of IEEE/ACM Design Automation Conference (DAC)*. 1–6.
[19] Wei Zeng, Azadeh Davoodi, and Yu Hen Hu. 2018. Design Rule Violation Hotspot Prediction Based on Neural Network Ensembles. *arXiv preprint arXiv:1811.04151* (2018).