uteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network

Zhiyao Xie[1], Yu-Hung Huang[2], Guan-Qi Fang[2], Haoxing Ren[3], Shao-Yun Fang[2], Yiran Chen[1], Jiang Hu[4]

[1] Duke University, Durham, NC, USA
[2] National Taiwan University of Science and Technology, Taipei, TW
[3] Nvidia Corporation, Austin, TX, USA
[4] Texas A&M University, College Station, TX, USA

zhiyao.xie@duke.edu jianghu@tamu.edu

ABSTRACT

Early routability prediction helps designers and tools perform preventive measures so that design rule violations can be avoided in a proactive manner. However, it is a huge challenge to have a predictor that is both accurate and fast. In this work, we study how to leverage convolutional neural network to address this challenge. The proposed method, called RouteNet, can either evaluate the overall routability of cell placement solutions without global routing or predict the locations of DRC (Design Rule Checking) hotspots. In both cases, large macros in mixed-size designs are taken into consideration. Experiments on benchmark circuits show that RouteNet can forecast overall routability with accuracy similar to that of global router while using substantially less runtime. For DRC hotspot prediction, RouteNet improves accuracy by 50% compared to global routing. It also significantly outperforms other machine learning approaches such as support vector machine and logistic regression.

1 INTRODUCTION

Every chip design project must complete routing without design rule violation before tapeout. However, this basic requirement is often difficult to be satisfied especially when routability is not adequately considered in early design stages. In light of this fact, routability prediction has received serious attention in both academic research and industrial tool development. Moreover, routability is widely recognized as a main objective for cell placement.

In industrial designs, fast trial global routing is often employed for routability prediction at placement stage [24]. The "fast" here is relative to full-fledged global router that generates solutions for further detailed routing. Such trial global routing is still too slow from the routability prediction point of view, as it is called many times within placement engine. Probabilistic prediction [15, 25] and other fast alternatives [21] have been developed. However, their sacrifice on accuracy is quite significant and trial global routing is still the de facto standard despite its costly runtime [24].

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240843

In addition to forecasting overall routability, one also needs to predict locations of Design Rule Checking (DRC) hotspots where routability optimization engines can be applied to fix them. Evidently, predicting hotspot locations is much more difficult than forecasting overall routability, which is often indicated by Design Rule Violation (DRV) count. In this case, even global routing is not accurate enough [4] due to complicated design rules imposed upon design layout for manufacturing. Overall, global routing is neither fast enough for overall routability forecast nor accurate enough for pinpointing DRC hotspots.

To find accurate yet fast routability prediction approach, people recently explore machine learning techniques, which have exhibited exciting progress and have been investigated in several EDA applications [11], including lithography hotspot detection [7], structured design placement [23] and NoC router modeling [10]. In [18, 26], Multivariate Adaptive Regression Spline (MARS) is applied for forecasting detailed routing routability. However, global routing solution is still required as an input feature to the learning here so that there is no benefit for runtime reduction. MARS and Support Vector Machine (SVM)-based routability forecast without using global routing information is proposed in [3]. However, this technique does not indicate how to handle macros, which prevail in modern chip designs and considerably increase the difficulty of routability prediction [26]. In [4], an SVM-based method is introduced for predicting locations of DRC hotspots. It shows accuracy improvement compared to global routing on testcases without macros. Cases without macros allow this method to use a set of small regions in a circuit as a training set and predict routability of other regions of the same circuit. Such convenience disappears when macros present, as a large region (often entire layout) of a circuit needs to be "receptive" to the machine learning model.

In this work, we attempt to solve two problems: 1. fast routability forecast for cell placement in terms of the number of Design Rule Violations (#DRV) such that a few relatively routable placement solutions can be identified among many candidate solutions; 2. the prediction of DRC hotspot locations such that the few identified solutions can be proactively modified to prevent design rule violations. In both of the problems, we will consider macros, which are prevalent in modern industrial designs. Our approach is built upon Convolutional Neural Network (CNN), which largely contributed to the recent success and popularity of the machine learning field but has not been investigated for routability prediction. A key rationale is that CNN has been demonstrated very effective for image recognition while chip layout information, such as pin density, can be treated as images. In overall routability forecast, we make use of

RUDY [21], a previous work of fast routability estimation, which can also be easily represented as images. The main differences between our method, called RouteNet, and previous machine learning-based works are summarized in Table 1.

Table 1: Comparison among different machine learning techniques."GR" indicates global routing.

Methods	Use GR?	Predict #DRV?	Predict hotspot?	Handle macros?
[18] (Qi, et al., ICCD14)	Y	Y	N	Y
[26] (Zhou, et al., ASQED15)	Y	Y	N	N
[3] (Chan, et al., ICCD16)	N	Y	N	N
[4] (Chan, et al., ISPD17)	Y	N	Y	N
RouteNet #DRV prediction	N	Y	N	Y
RouteNet hotspot prediction	Y	N	Y	Y

The main contribution of our work includes:

- Our work provides the first systematic study on CNN-based routability prediction. Intuitively, this is a promising direction but has not been well studied in the past.
- Our method, RouteNet, can quickly forecast overall routability in terms of DRV count considering macros. It achieves similar accuracy to that of global routing but is orders of magnitude faster even if training time is counted. To the best of our knowledge, this is the first routability predictor that has both such high accuracy and high speed.
- For predicting DRC hotspot locations considering macros, RouteNet makes a large progress of 50% accuracy improvement versus global routing. It also remarkably outperforms SVM and logistic regression-based prediction.

2 BACKGROUND

2.1 CNN and Transfer Learning

CNN has demonstrated its impressive classification and object detection ability on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) since 2012 [13]. Many deep and complex variations have been proposed since then. As the winner of ILSVRC in 2015, ResNet is one of state-of-the-art CNN architectures [8]. Compared with traditional machine learning algorithms, CNN learns more abstract patterns from images. Our RouteNet transfers such state-of-the-art ability in image pattern recognition to circuits for capturing the patterns about routability.

In practice, it requires a huge dataset and a long time to train a network from scratch (from random initialization). Thus, it is common to download a network pretrained on a large image database like ImageNet [6] and then fine-tune it with the dataset related to the specific task. Such pretraining and fine-tuning processes are referred to as *transfer learning*. RouteNet takes the benefit of transfer learning. It predicts routability based on a pretrained ResNet architecture.

Figure 1 shows a typical structure of CNN, which is composed by convolutional (CONV) layers, pooling (POOL) layers and fully-connected (FC) layers. CONV layers and POOL layers perform downsampling and produce a three-dimensional feature map smaller than the input. FC layers, near the end of CNN, produce one-dimensional output. The final output is a single vector of class scores, whose length equals the number of classes.

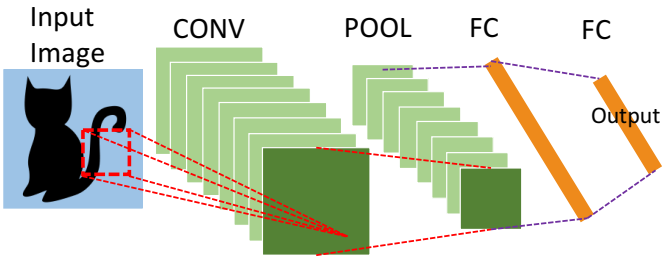

Figure 1: Convolutional Neural Network (CNN). CNN integrates convolutional layers (CONV), pooling layers (POOL) and fully-connected (FC) layers into an end-to-end structure.

2.2 Fully Convolutional Network

Compared with CNN targeting image classification, the CNN architecture without FC layers is firstly proposed to perform end-to-end semantic segmentation [14]. Such architecture, named Fully Convolutional Network (FCN), outputs an image with size equal to or smaller than input. To compensate the downsampling effect of CONV and POOL layers, transposed-convolutional (TRANS) layers are usually added at the end to upsample feature maps and control the size of final output. Such architecture is widely used in many computer vision problems, like crowd counting [20] and biomedical image segmentation [19]. Besides eliminating FC layers, many FCNs have both deep and shallow paths in one network. Some of them have multiple shortcuts, which concatenate feature maps in the front directly to feature maps near the end. As a result, both longer and shorter paths exist from input layer to final output layer. Such multi-path architecture reserves both shallow and deep information and ensures a high accuracy in both pixel segmentation and classification.

Unlike the overall routability (#DRV) prediction, which is a classification problem, DRC hotspot detection is more like an end-to-end object detection task, which is more difficult to solve. The nice property of FCN allows input to be any size and produces an output with exactly same size as input, indicating the existence of DRC hotspots at any region. The DRC hotspot detection method in RouteNet is based on such FCN architecture and adopts the path shortcut structure.

2.3 RUDY

RUDY (Rectangular Uniform wire DensitY) [21] is a pre-routing congestion estimator. It is employed as an input feature to our RouteNet as it partially correlates with routing congestion, is fast to obtain and can be directly represented as images that dovetail with RouteNet.

Given a cell placement, RUDY of a net is obtained by uniformly spreading the wire volume of this net into its bounding box. For the kth net with bounding box $\{x^k_{min}, x^k_{max}, y^k_{min}, y^k_{max}\}$, its RUDY at location (x, y) is defined as

$$w^k = x^k_{max} - x^k_{min},\ h^k = y^k_{max} - y^k_{min}$$

$$c^k = \begin{cases} 1 & x \in [x^k_{min}, x^k_{max}]\ ,\ y \in [y^k_{min}, y^k_{max}] \\ 0 & otherwise \end{cases}$$

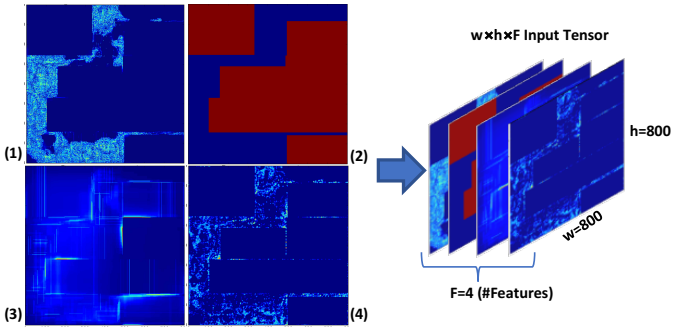

Figure 2: Three-dimensional input tensor constructed by stacking two-dimensional features including (1) pin density, (2) macro region, (3) long-range RUDY, (4) RUDY pins.

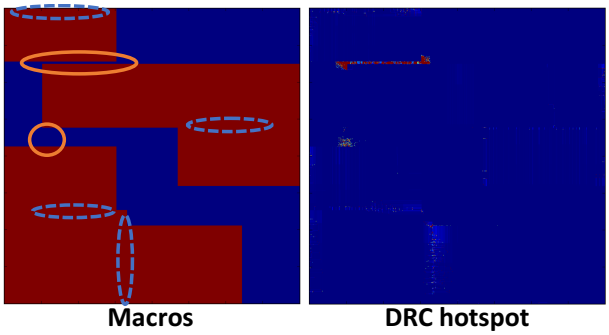

Figure 3: Macros and DRC hotspot distribution. All macros are red rectangles. Orange circles indicate regions with high density of DRC hotspots. Blue dashed circles indicate the remaining sparsely distributed hotspots.

$$RUDY^k(x,y) \propto c^k \frac{w^k + h^k}{w^k \times h^k}$$

RUDYs of all K nets are calculated and superimposed on top of each other to provide a rough estimation of routing congestion as

$$RUDY(x,y) = \sum_{k=1}^{K} RUDY^k(x,y)$$

3 PROBLEM FORMULATION

We aim at solving two problems: 1. early forecast of overall routability; 2. the prediction of DRC hotspot locations. In problem 1, the routability of a placement is evaluated by its DRV count denoted as #DRV. The task is to fit a function $f_{\#DRV}$ that estimates ground-truth DRV count of a placement. In applications, $f_{\#DRV}$ is employed to select a few placements with relatively low #DRV from many candidate placement solutions. In problem 2, DRC hotspots mean the specific locations with high density of DRVs. The task is to find a function $f_{hotspot}$ that detects most DRC hotspots in a placement. RouteNet solves problem 1 without performing any routing while its solution to problem 2 uses global routing as a feature. Since problem 2 is performed on a few relatively routable designs, its runtime constraint is less tight than problem 1. In both tasks, the informative features about placements are input and DRV information is the prediction target. The ground-truth DRV information is also referred to as label.

To evaluate routability in terms of design rule violations, a layout (or placement solution) is tessellated into an array of grid cells, each of which is an $l \times l$ square. Then, a rectangular layout with size $W \times H$ is divided into $w \times h$ grid cells, where $w = W/l$ and $h = H/l$. In our experiments, l is set to be the height of standard cells.

After design rule checking, the location and area of all DRC violations are reported. The overall DRV count for the ith placement solution is recorded as $y_i \in \mathbb{N}$. The density of DRV is calculated at a grid-level granularity. The DRV density in each grid cell is a summation of contributions from all violations covering this grid cell. As a result, for the ith placement with $w \times h$ grid cells, its DRV density is a two-dimensional matrix $Y_i \in \mathbb{R}^{w \times h}$. When the density of violations in a grid cell is higher than a threshold ϵ, this grid cell is labeled as a DRC hotspot. Existence of DRC hotspots is a Boolean matrix $V_i \in \{0,1\}^{w \times h}$, where $V_{i_{mn}} = \mathbb{1}(Y_{i_{mn}} > \epsilon)$ for grid cell (m,n).

Similar to DRV density, densities of different informative features are calculated as the input of RouteNet. The jth feature of ith placement is $X_{ij} \in \mathbb{R}^{w \times h}$. If F different features are generated, the input for the ith placement is $X_i \in \mathbb{R}^{w \times h \times F}$. Such X_i is constructed by stacking all two-dimensional features $\{X_{ij} \mid j \in [1,F]\}$ together in a third dimension as shown in Figure 2. Inputs for the two problems are not exactly the same. For example, since #DRV prediction starts early, global routing information is not included in input $X_i^{\#DRV}$. The two problems are formally stated as follows.

Problem 1 (#DRV prediction). *Find an estimator $f^*_{\#DRV}$ of DRV count in a placement:*

$$f_{\#DRV} : X_i^{(\#DRV)} \in \mathbb{R}^{w \times h \times F_1} \rightarrow y_i \in \mathbb{N}$$

$$f^*_{\#DRV} = \arg\min_f Loss(f(X_i^{(\#DRV)}),\ y_i)$$

Problem 2 (Hotspot prediction). *Find a detector $f^*_{hotspot}$ of hotspots. It reports locations of all DRC hotspots in a placement.*

$$f_{hotspot} : X_i^{(hotspot)} \in \mathbb{R}^{w \times h \times F_2} \rightarrow V_i \in \{0,1\}^{w \times h}$$

$$f^*_{hotspot} = \arg\min_f Loss(f(X_i^{(hotspot)}),\ V_i)$$

4 THE CHALLENGE OF MACROS

In this section, the influence of macros on DRV is discussed. Figure 3 shows the distribution of DRC hotspots on a benchmark circuit with macros. The orange circles in Figure 3 indicate a strong tendency for hotspots to aggregate at the small gap between neighboring macros. The remaining small number of hotspots, indicated by blue dashed circles, also locate sparsely around the edges of some macros. To detect hotspots more precisely, such distribution pattern has to be captured. This requires global information about neighboring macros. When estimating DRV of a grid cell, one needs to consider how large its neighboring region is receptive to the estimator. Only when such receptive region is large enough can the estimator decide whether such grid cell actually locates inside a gap between two macros.

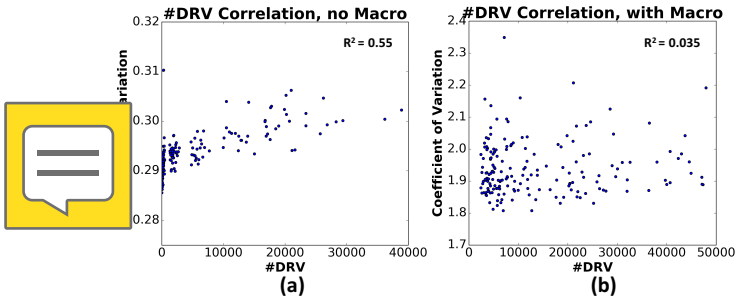

Figure 4: Correlation between #DRV and coefficient of variation of pin density.

Figure 4 shows the impact of macros on the correlation between pin density and #DRV. R^2 here measures the strength of linear relationship between two variables on a 0 to 1 scale. The coefficient of variation is calculated by $\frac{\sigma}{\mu}$, where σ and μ are the standard deviation and mean value of pin density across all grid cells. It measures how evenly pins are distributed. The work of [3] indicates the existence of correlation between pin density and #DRV, which is confirmed in Figure 4(a). But with macros, such correlation largely disappears in Figure 4(b).

Overall, a layout without macros is much more homogeneous than that with macros. The homogeneity implies resemblance among different regions of a chip layout. As such, the DRC hotspot detection work of [4] can use individual small regions as training data. By contrast, mixed-size designs with macros must have a much larger region, often entire layout, as a single training case, in order to capture the global view. The homogeneity also allows the DRV count forecast [3] to use only a small number of chip statistics, such as the distribution of pin density and the worst negative slack, ut features to the model. However, the information carried h statistics is not sufficient for cases with macros as shown ure 4, and more detailed local information is necessary.

THE ROUTENET ALGORITHM

In this section, we first describe all input features extracted at different stages of physical design flow. Then, both the #DRV prediction method and the hotspot detection method of RouteNet are presented.

5.1 Feature Extraction

Figure 5 shows the input features extracted at different layout stages. Here is a detailed description:

- **Macro**: After floorplanning, the locations of all macros are fixed. Several features about macro are extracted:
  - the region occupied by macros.
  - density distribution of macro pins in each metal layer.
- **Global cell & global RUDY**: After global placement, cell locations become available and then RUDY is calculated. Features from this stage are denoted by *global cell* and *global RUDY*. After detailed placement, the information from global cell and global RUDY are recalculated by refined cell locations, denoted as *cell* and *RUDY*.
- **Cell**: For both global cell and cell, two features are extracted:
  - density distribution of cells.

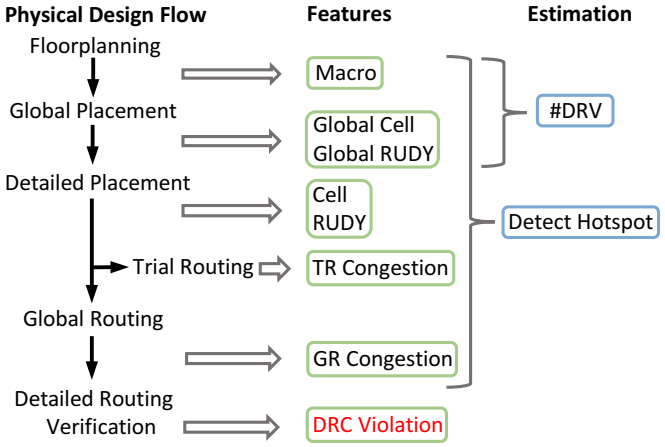

Figure 5: Feature extraction in physical design flow.

  - density distribution of cell pins.
- **RUDY**: For both global RUDY and RUDY, three features are calculated:
  - long-range RUDY.
  - short-range RUDY.
  - RUDY pins.

  The original RUDY feature is decomposed into long-range and short-range RUDY. Long-range RUDY is from nets covering a distance longer than a threshold. Similarly, short-range RUDY is for nets shorter than this threshold. Such decomposition is due to a stronger correlation between long-range RUDY and DRV than the shorter one. To further capture such effect, RouteNet uses another feature named *RUDY pins*. It is similar to pin density, while the contribution of each pin equals the long-range RUDY of the net it connects to.
- **Congestion**: Two types of congestion are generated by placement and routing tool. The trial global routing, also denoted as trial routing, can be performed at the end of detailed placement. It produces an estimation of routing congestion, named *TR congestion*. Compared with it, the full-fledged global routing generates a more detailed congestion map, named *GR congestion*.
- **DRC violation**: DRC violations from all metal layers are included. It is the prediction target and the label for training RouteNet. Its density is calculated in the same way as all other features. Most DRC violations only occupy very small regions.

For the ith placement, all above F features $X_{ij} \in \mathbb{R}^{w \times h}$ are calculated independently and then combined together as one input tensor $X_i \in \mathbb{R}^{w \times h \times F}$.

5.2 #DRV Prediction

As an early routability forecast, the #DRV prediction by RouteNet is performed before detailed placement starts. Details are shown in Algorithm 1. In order to convert #DRV prediction to an image classification task, both input features and prediction target are preprocessed.

The range of #DRV is broad for different placements of the same design. At this early stage, two placements with a slight difference

Algorithm 1 Algorithm of RouteNet for #DRV Prediction

Input: Number of training placements: N, Features: $\{X_i \in \mathbb{R}^{w\times h\times 3} \mid i \in [1, N]\}$, Targets: $\{y_i \in \mathbb{R} \mid i \in [1, N]\}$

Preprocess:

1: **for** each int $i \in [1, N]$ **do**
2: Resize $X_i \in \mathbb{R}^{w\times h\times 3}$ into $X_i^{\#DRV} \in \mathbb{R}^{224\times 224\times 3}$
3: Find 25%, 50%, 75% quantizes of y_i: q_1, q_2, q_3
4: **for** each int $i \in [1, N]$ **do**
5: $C_i \leftarrow 0$
6: **for** each int $t \in [1, 3]$ **do**
7: **if** $y_i > q_t$ **then**
8: $C_i \leftarrow t$, **break**
9: Form dataset $\{(X_i^{\#DRV}, C_i) | i \in [1, N]\}$
10: Training set $\{(X_i^{\#DRV}, C_i) \mid C_i = 0 \text{ or } C_i = 3\}$

Training:

1: Get pretrained ResNet18 $f_{Res} : \mathbb{R}^{224\times 224\times 3} \rightarrow \mathbb{R}^{1000}$
2: Replace output layer, s.t. $f_{\#DRV} : \mathbb{R}^{224\times 224\times 3} \rightarrow \mathbb{R}$
3: Choose MSE as loss function, SGD for optimization
4: Train $f_{\#DRV}$ with preprocessed dataset for ~30 epoches

Output: $f_{\#DRV}$ estimating #DRV level

in #DRV may not have substantially different patterns in their features, especially when routing information is absent. As a result, RouteNet can wrongly capture such minor difference if the exact V is used as label. To avoid this, we group placements into #DRV levels, referred to as c_0, c_1, c_2, c_3, respectively, where c_0 sponds to the class of placements with the least #DRV and c_3 sponds to class with the most #DRV.

For all CNN models pretrained on dataset ImageNet, the dimension of input image X_i is fixed to be $224 \times 224 \times 3$. This means the input images have 224×224 pixels and 3 channels (RGB). To utilize such pretrained model, the original input tensor $X_i \in \mathbb{R}^{w\times h\times F}$ needs to be resized into $X_i^{\#DRV} \in \mathbb{R}^{224\times 224\times 3}$. To accomplish this, three features (macro, global long-range RUDY, global RUDY pins) are firstly selected to construct 3-channel input tensor in $\mathbb{R}^{w\times h\times 3}$. We choose them because they intuitively contain more global and general information than feature like pin density. After that, the 3-channel input is resized into $\mathbb{R}^{224\times 224\times 3}$ by interpolation. Figure 6 shows visualizations of preprocessed input, (a)(b) and (c)(d) are different placements with different levels of #DRV for two benchmark circuits.

After the preprocessing step, transfer learning is applied to a pretrained 18-layer ResNet. The output layer is replaced to produce a single score. During the fine-tuning process, the weights in every layer are changeable. Mean Square Error (MSE) is used as loss function and Stochastic Gradient Descent (SGD) is used for optimization.

The dataset is randomly split into the training set and the validation set. For the training set, all data in classes c_1 and c_2 are removed, only classes c_0 and c_3 are kept. That is, only placements in highest #DRV or lowest #DRV levels will be used for training. Such removal proves to give better results than keeping all four classes.

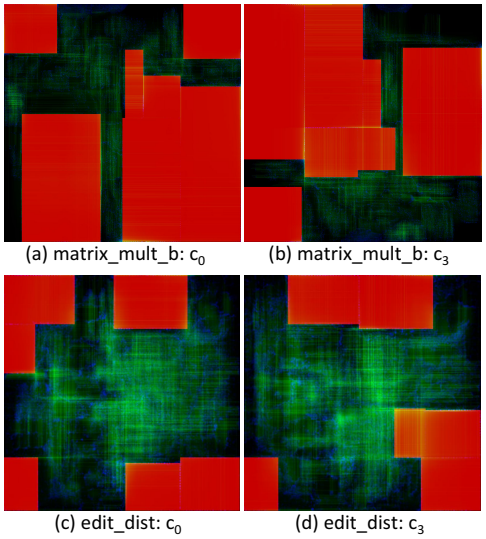

Figure 6: Input features for #DRV prediction. Red: macro region; Green: global long-range RUDY; Blue: global RUDY pins.

5.3 DRC Hotspot Detection

Different from #DRV prediction, hotspot detection is more like an object detection task. The output can no longer be a simple score value or vector. Instead, we make it a two-dimensional density map, directly reflecting the existence of all hotspots in a placement. In this case, the size of output equals circuit size $f_{hotspot}(X_i) \in \mathbb{R}^{w\times h}$. FCN enables such function format and accepts input with different w, h. As a result, different designs can be used for training and inference on exactly the same model.

Figure 7 shows the FCN architecture. It accepts input tensor with size $\mathbb{R}^{w\times h\times F}$ and produces a two-dimensional $\mathbb{R}^{w\times h}$ output. In this structure, a shortcut directly connects the 2nd layer to the 7th layer, providing a shorter path from input to output. Two POOL layers downsize feature maps from $h \times w$ to $\frac{h}{4} \times \frac{w}{4}$ in the front, then two TRANS layers upsample the size back to $h \times w$. Strides of kernels in CONV and TRANS layers are set to 1 and 2, respectively. The kernel sizes are indicated by the number in parentheses, ranging from 3 to 9 grid cells. Such pooling structure and large kernel size substantially enlarge the regions receptive to each grid cell in output. Compared with previous methods capturing features within a small region, more neighboring or global information is available for FCN and the result function $f_{hotspot}$ will be more complex.

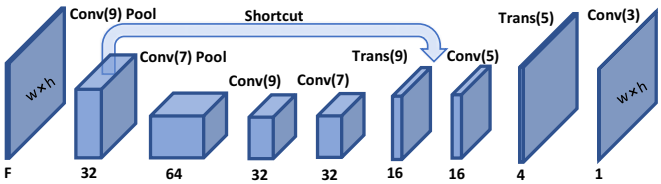

Figure 7: FCN architecture for hotspot detection.

During the training process, the DRV density Y_i is used as label. DRV density is clipped by a threshold c in Equation (1) to reduce the dominating effect of a few grid cells with very high DRV density. Batch normalization [9] is applied to accelerate convergence in training. The Adam method [12] is used for optimization. The loss function is defined by a summation of pixel-wise Euclidean distance and L2 regularization in Equation (2),

$$Y_{i_{mn}}^{clip} = min(Y_{i_{mn}}, c) \quad (1)$$

$$Loss = \sum_{i=1}^{N} \sum_{m=1}^{w} \sum_{n=1}^{h} ||f_{hotspot}(X_{i_{mn}}) - Y \quad W||_2 \quad (2)$$

where λ is regularization coefficient and W denotes all weights in FCN. By adding such L2 norm into loss function, all weights are forced to decay towards zero. As a result, it reduces unnecessary weights and avoids overfitting.

6 EXPERIMENTAL RESULTS

6.1 Experiment Setup

Table 2: Circuit Designs Used in Experiment

Circuit Name	#Macros	#Cells	#Nets	Width (μm)	#Placements
des_perf	4	108288	110283	900	600
edit_dist	6	127413	131134	800	300
fft	6	30625	32088	800	300
matrix_mult_a	5	149650	154284	1500	300
matrix_mult_b	7	146435	151614	1500	300

Five designs from ISPD 2015 benchmarks [1] are used in the experiment. Table 2 shows their basic information. The shapes of all five designs are squares[1], whose sizes range from 800 μm to 1500 μm. For each circuit design, at least 300 different floorplans are generated by placing macros at different locations with the "obstacle-aware macro placement" algorithm [5]. Though placed differently, macros all tend to locate near the chip boundary in order to leave plenty of space at chip center region, where routing demand tends to be high. Then, each floorplan is placed and routed by Cadence Encounter v14.20 [2]. DRC violation information for each layout is recorded as label.

Both #DRV prediction and hotspot detection methods of RouteNet are tested on all five designs. When each design is tested, the machine learning model is trained only on data from the other four designs. This ensures that the tested design is totally *unseen* to the corresponding model, which eliminates the possibility of information leak from the testing dataset to the training dataset.

All algorithms are implemented in Python. CNN is implemented based on PyTorch [16]. As references for comparison, SVM and Logistic Regression (LR)-based methods are implemented based on scikit-learn [17]. Hyperparameters are carefully tuned. Training and inference of all methods are performed on a machine with 2.40 GHz CPU and one NVIDIA GTX 1080 graphics card.

6.2 Overall #DRV Prediction

For comparison, the previous method [3] is directly transferred to our benchmark as a reference. In this method, only the maximum value (acros s) and the coefficients of variation of features are ext ch placement. The same preprocessing is performed. SVM with Radial Basis Function kernel are tested as classifiers. Compared with RouteNet, the input of this method $X_i^{Ref} \in \mathbb{R}^{2\times F}$ contains much less feature information.

[1]Not a requirement. RouteNet accepts rectangular circuit design in any size.

Table 3: #DRV Prediction Comparison

Circuit Name	$c_0/c_1+c_2+c_3$ accuracy (%)					Best rank in top 10				
	SVM	LR	TR	GR	Route Net	SVM	LR	TR	GR	Route Net
des_perf	63	74	80	77	80	87^{th}	15^{th}	2^{nd}	1^{st}	2^{nd}
edit_dist	69	68	78	77	76	17^{th}	17^{th}	3^{rd}	3^{rd}	2^{nd}
fft	66	62	73	70	75	6^{th}	6^{th}	2^{nd}	33^{rd}	1^{st}
matrix_mult_a	66	65	78	74	72	30^{th}	5^{th}	1^{st}	1^{st}	5^{th}
matrix_mult_b	63	62	76	73	76	22^{nd}	93^{rd}	4^{th}	1^{st}	4^{th}
Average	65	66	77	74	76	32^{nd}	27^{th}	2^{nd}	8^{th}	3^{rd}

The goal of #DRV prediction is to select a small set of placements with low #DRV from many candidates. Table 3 shows the performance in #DRV prediction. The "$c_0/c_1+c_2+c_3$" accuracy checks the binary classification accuracy by treating all placements in c_1, c_2, c_3 as one class and c_0 as the other. As indicated in Algorithm 1, c_0 means the lowest #DRV level. This accuracy evaluates how different methods recognize placements with the lowest #DRV level. The result shows that RouteNet has similar accuracy as trial routing (TR) and global routing (GR).

We are also interested in the quality of placements selected by each method. To evaluate this, we first rank all placements from same design in ascending order of #DRV. Then for each method, the top ten placements predicted to have least #DRV are selected. Among such ten placements, the rank of the placements with least ground truth #DRV is reported as "Best rank in top 10" in Table 3. On average, RouteNet finds the 3^{rd} best placement from hundreds of candidates within 10 selections. Again, it shows comparable performance with trial routing and global routing, and is much better than both LR and SVM.

When Table 3 only shows the rank of the best placement in 10 selections, Figure 8 further indicates the gap between such "best in ten" and the actually 1st-ranked placement with least #DRV. Such

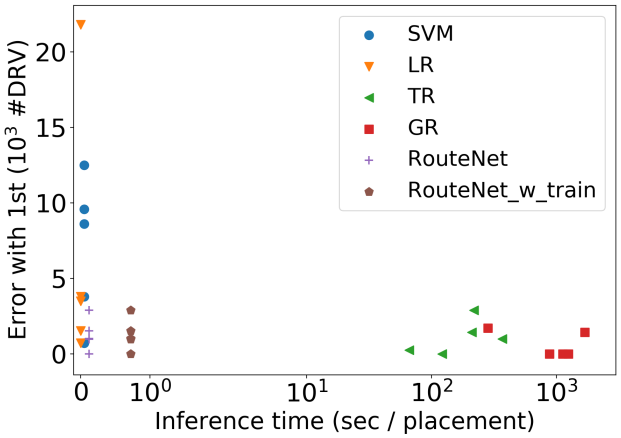

Figure 8: Trade-off between error with actually 1st-ranked placement and inference time in #DRV prediction.

gap is denoted as error in #DRV value. Each point represents the result of one design. Besides accuracy, runtime is another essential factor to consider. Figure 8 also shows the inference time for each method. Inference time is the overall time taken to predict one placement, starting at the end of global placement. In practice, RouteNet is trained in advance with other designs, so the training process costs no extra time during inference. But for reference, we still provide "RouteNet_w_train", which includes the training time of RouteNet. In Figure 8, the results for RouteNet aggregate at the lower left corner, which means low inference time and high accuracy are achieved at the same time. By contrast, trial routing and global routing take substantially longer runtime to reach similar accuracy. LR and SVM, however, cannot guarantee low error though they are quite fast. Our RouteNet is the only fast and accurate method in #DRV prediction. Even with training time included, the average inference time for one placement is still less than one second.

6.3 DRC Hotspot Detection

For comparison, alternative methods similar to the previous work [4] are implemented. Features from each grid cell itself are extracted as its input, then grid cells are classified independently by either LR or SVM.

Table 4: Hotspot Detection Comparison

Circuit Name	FPR (%)	TPR (%)				
		TR	GR	LR	SVM	RouteNet
des_perf	0.54	17	56	54	42	74
edit_dist	1.00	25	36	38	28	64
fft	0.30	21	45	54	31	71
matrix_mult_a	0.21	13	30	34	12	49
matrix_mult_b	0.24	13	37	41	20	53
Average	0.46	18	41	44	27	62

Table 4 shows the accuracy in hotspot detection. TPR (True Positive Rate) and FPR (False Positive Rate) are used for evaluation. FPR describes the rate of grid cells being wrongly classified as hotspots. TPR, also named recall or sensitivity, describes the percentage of detected hotspots over all existing hotspots. By adjusting the decision threshold of prediction result, FPR and TPR change proportionally. For a fair comparison, we compare the TPR of all methods under the same FPR. The same decision threshold is used for all designs, which results in slightly different FPR among designs, but all under 1%. It ensures the number of errors is acceptable.

		Prediction Result: Positive	Prediction Result: Negative	Evaluation
Label	Positive	TP	FN	$TPR = \frac{TP}{TP+FN}$
	Negative	FP	TN	$FPR = \frac{FP}{FP+TN}$

As Table 4 shows, global routing is a much better hotspot detector than trial routing, although both methods have similar accuracy in overall #DRV prediction. LR demonstrates better accuracy than global routing. SVM, however, is inferior to global routing even with our best effort on hyperparameter tuning. RouteNet is superior

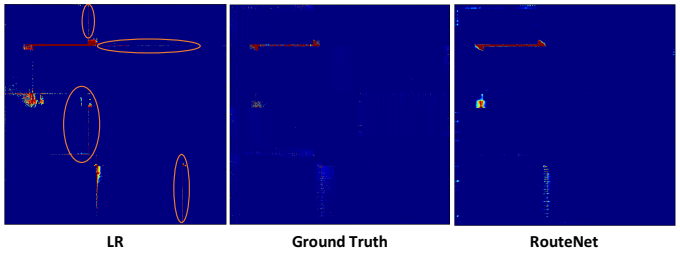

Figure 9: Visualization of hotspot detection results.

to all methods and improves global routing accuracy by 50%. Figure 9 provides an illustration of hotspot detection results. The result of RouteNet is closer to the ground truth. Orange circles indicate grid cells wrongly recognized as hotspots with high confidence by LR. These grid cells typically locate at the edges of macros. LR exaggerates the influence of macro on them.

7 DISCUSSION

To further explore the hotspot detection problem, some variations of both FCN and alternative methods are evaluated. Results are shown in Table 5 and 6. These variant methods are briefly described as follows.

- Infer seen: Training and inference are performed on different placements of the same circuit.
- Less data: Trained on less data. Only placements from two circuits are used for training instead of four.
- No short: No shortcut. The shortcut structure is removed from current FCN architecture.
- Less conv: Three convolutional layers (with channels 64, 32, 32) in the middle of shortcut are removed, resulting in a shallower network.
- No pool: Based on the shallow network above, the POOL layers are removed. TRANS layers are replaced by normal CONV layers.
- 5×5 LR: Using window size of 5×5 grid cells to capture neighboring features of each grid cell in LR. Similarly, 9×9 LR means 9×9 grid cells of window size.
- 5×5 SVM: The same as the 5×5 LR above in feature extraction, but for SVM.

The effect of training and inference on different designs is explored by "Infer seen" and "Less data" in Table 5. Difference in designs used for training and inference can be vital to the transferability of an algorithm. That is, if the distribution of DRC hotspots varies greatly among different designs, the pattern learned from training data may not be applicable to new "unseen" designs. Compared with original RouteNet, the better performance for "Infer seen" in Table 5 implies the existence of certain *pattern* unique to each design. But lower accuracy for "Less data" indicates that more training data from different designs can bridge such gap.

The FCN structure in RouteNet has both shallow and deep paths connecting input and output layers. In order to check the effect of such two-path structure, two variations "No short" and "Less conv" are tested. "No short" removes the shorter path and "Less conv" removes the longer path. As expected, Table 5 shows accuracy degradation for both variations. More interestingly, by removing

Table 5: Hotspot Detection for FCN Variations

Circuit Name	FPR (%)	TPR (%)					
		Infer seen	Less data	No short	Less conv	No pool	Route Net
des_perf	0.54	77	71	71	73	68	74
edit_dist	1.00	68	61	63	62	55	64
fft	0.30	74	70	68	68	69	71
matrix_mult_a	0.21	51	46	45	45	45	49
matrix_mult_b	0.24	58	50	51	50	50	53
Average	0.46	66	60	60	60	57	62

Table 6: Hotspot Detection for Alternative Methods

Circuit Name	FPR (%)	TPR (%)					
		LR	5×5 LR	9×9 LR	SVM	5×5 SVM	9×9 SVM
des_perf	0.54	54	58	58	42	47	29
edit_dist	1.00	38	39	38	28	29	20
fft	0.30	54	56	54	31	41	23
matrix_mult_a	0.21	34	36	35	12	32	9
matrix_mult_b	0.24	41	44	42	20	39	16
Average	0.46	44	47	45	27	38	19

POOL and TRANS structures in "No pool", which leads to a large reduction in receptive region, overall accuracy further degrades. It supports our claim on the importance of receptive region and the global information in hotspot detection.

Table 6 shows how a larger receptive region affects other machine learning methods in hotspot detection. We tested several larger window sizes for feature extraction as 3×3, 5×5, 7×7, 9×9 grid cells. The 5×5 window size turns out to perform the best. Again, the large receptive region gives better results, but 5×5 is the upper limit in our experiment. An even larger window blurs the local information of the target grid cell. Compared with these alternative methods, RouteNet provides a better solution to obtain the benefit of large receptive region.

8 CONCLUSION AND FUTURE RESEARCH

This work advanced the state of the art of routability prediction at two fronts. For overall routability forecast of mixed-size designs, RouteNet achieves similar accuracy as global routing but is several orders of magnitude faster. This largely solves the challenge of having both accurate and fast routability prediction of general designs. For DRC hotspot detection with consideration of macros, RouteNet also makes an important step forward by improving global router's accuracy by 50%. We will further improve the prediction accuracy for both scenarios in our future research.

ACKNOWLEDGMENT

This work was partially supported by NSF (CCF-1525749, CNS-1618824, 1615475) and MOST of Taiwan (MOST 107-2636-E-011-002). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of grant agencies or their contractors.

REFERENCES

[1] Ismail S. Bustany, David Chinnery, Joseph R. Shinnerl, and Vladimir Yutsis. 2015. ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement. In *ACM International Symposium on Physical Design (ISPD)*.
[2] Cadence. 2017. Cadence encounter User Guide. (2017). http://www.cadence.com
[3] Wei-Ting J. Chan, Yang Du, Andrew B. Kahng, Siddhartha Nath, and Kambiz Samadi. 2016. BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques. In *International Conference on Computer Design (ICCD)*.
[4] 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 *ACM International Symposium on Physical Design (ISPD)*.
[5] Chien-Hsiung Chiou, Chin-Hao Chang, Szu-To Chen, and Yao-Wen Chang. 2016. Circular-Contour-Based Obstacle-Aware Macro Placement. In *IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*.
[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[7] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan. 2012. EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation. In *IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*.
[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[9] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*.
[10] K. Jeong, A. B. Kahng, B. Lin, and K. Samadi. 2010. Accurate Machine-Learning-Based On-Chip Router Modeling. *IEEE embedded systems letters (ESL)* (2010).
[11] Andrew B. Kahng. 2018. New Directions for Learning-based IC Design Tools and Methodologies. In *IEEE Asia and South Pacific Design Automation Conference (ASPDAC)*.
[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*.
[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*.
[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).
[15] Jinan Lou, Shankar Krishnamoorthy, and Henry S. Sheng. 2001. Estimating Routing Congestion using Probabilistic Analysis. In *ACM International Symposium on Physical Design (ISPD)*.
[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems Workshops (NIPS-W)*.
[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)* (2011).
[18] Zhongdong Qi, Yici Cai, and Qiang Zhou. 2014. Accurate Prediction of Detailed Routing Congestion using Supervised Data Learning. In *International Conference on Computer Design (ICCD)*.
[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.
[20] Vishwanath A. Sindagi and Vishal M. Patel. 2017. Generating High-Quality Crowd Density Maps using Contextual Pyramid CNNs. In *IEEE International Conference on Computer Vision (ICCV)*.
[21] Peter Spindler and Frank M. Johannes. 2007. Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
[22] Aysa Fakheri Tabrizi, Nima Karimpour Darav, Shuchang Xu, Logan Rakai, Ismail Bustany, Andrew Kennings, and Laleh Behjat. 2018. A machine learning framework to identify detailed routing short violations from a placed netlist. In *ACM/IEEE Design Automation Conference (DAC)*.
[23] S. Ward, D. Ding, and D. Z. Pan. 2012. PADE: A High-performance Placer with Automatic Datapath Extraction and Evaluation Through High Dimensional Data Learning. In *ACM/IEEE Design Automation Conference (DAC)*.
[24] Yaoguang Wei, Cliff Sze, Natarajan Viswanathan, Zhuo Li, Charles J. Alpert, Lakshmi Reddy, Andrew D. Huber, Gustavo E. Tellez, Douglas Keller, and Sachin S. Sapatnekar. 2012. GLARE: Global and Local Wiring Aware Routability Evaluation. In *ACM/IEEE Design Automation Conference (DAC)*.
[25] Jurjen Westra, Chris Bartels, and Patrick Groeneveld. 2004. Probabilistic Congestion Prediction. In *ACM International Symposium on Physical Design (ISPD)*.
[26] Quan Zhou, Xueyan Wang, Zhongdong Qi, Zhuwei Chen, Qiang Zhou, and Yici Cai. 2015. An Accurate Detailed Routing Routability Prediction Model in Placement. In *Asia Symposium on Quality Electronic Design (ASQED)*.