Routability-Driven Macro Placement with Embedded CNN-Based Prediction Model

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

[1]National Taiwan University of Science and Technology, Taipei, Taiwan
[2]Duke Univeristy, Durham, NC, USA
[3]Nvidia Corporation, Austin, TX, USA
[4]Texas A&M University, College Station, TX, USA
e-mail: m10507422@mai.ntust.edu.tw, syfang@mail.ntust.edu.tw

***Abstract*—With the dramatic shrink of feature size and the advance of semiconductor technology nodes, numerous and complicated design rules need to be followed, and a chip design can only be taped-out after passing design rule check (DRC). The high design complexity seriously deteriorates design routability, which can be measured by the number of DRC violations after the detailed routing stage. In addition, a modern large-scaled design typically consists of many huge macros due to the wide use of intellectual properties (IPs). Empirically, the placement of these macros greatly determines routability, while there exists no effective cost metric to directly evaluate a macro placement because of the extremely high complexity and unpredictability of cell placement and routing. In this paper, we propose the first work of routability-driven macro placement with deep learning. A convolutional neural network (CNN)-based routability prediction model is proposed and embedded into a macro placer such that a good macro placement with minimized DRC violations can be derived through a simulated annealing (SA) optimization process. Experimental results show the accuracy of the predictor and the effectiveness of the macro placer.**

I. Introduction

With the dramatic shrink of feature size and the advance of semiconductor technology nodes, numerous and complicated design rules need to be followed, and a chip design can only be taped-out after passing design rule check (DRC). The high design complexity seriously deteriorates design routability, which can be measured by the number of DRC violations (DRVs) after the detailed routing stage. In addition, a modern large-scaled design typically consists of many huge macros due to the wide use of intellectual properties (IPs). The number of macros has been dramatically increased, and these huge macros can occupy more than 70% of chip area [11].

Empirically, the placement of macros greatly determines routability, which can be observed by the example layouts shown in Fig. 1. Figs. 1(a) and (b) give two different macro placements of the same circuit, and Figs. 1(c) and (d) respectively show the layouts of the two macro placements after cell placement and routing, which are accomplished by the commercial EDA Tool $Encounter$ [3]. The green blocks are the macros, and the white crosses indicate the locations of DRVs, which are caused by routed nets violating the DRC rules such as shorts and spacing violations. As reported by $Encounter$, the layout in Fig. 1(c) has 4892 violations, while the layout in Fig. 1(d) has 65868 violations, which clearly shows that macro placement could have tremendous influence on design routability. Since it usually takes a long time to accomplish cell placement and routing for modern large scale designs, it is unrealistic to enumerate all macro placements of a given design and find the best one after running the whole design flow for each macro placement.

This work was partially supported by MOST of Taiwan under Grant No. MOST 107-2636-E-011-002 and NSF (CCF-1525749, CNS-1618824, 1615475).

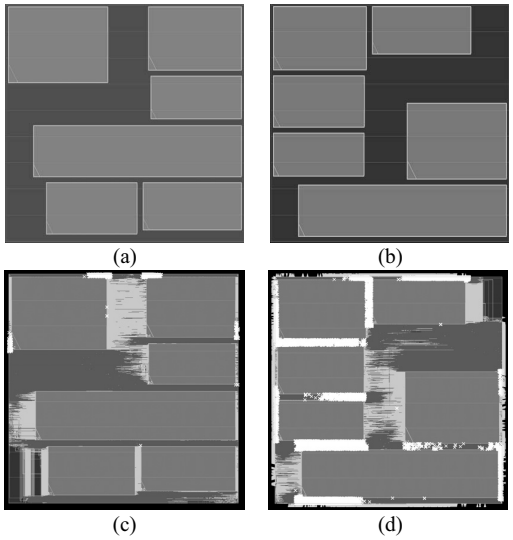

Fig. 1. Two different macro placements (a) and (b) cause significantly different routing results (c) One macro placement causes 4892 violations. (d) Another macro placement causes 65868 violations after cell placement and routing.

Many previous studies on macro placement have been proposed to address these design challenges, which can be classified into three categories: (1) one-stage mixed-size placement simultaneously placing macros and standard cells; for example: mPL6 [12], NTUplace3 [9], Hsu et al. [14], ComPLx [17], MAPLE [18], ePlace-MS [19], and FastPlace3.0 [27]. (2) Constructive mixed-size macro placement placing macros with partitioning and keeping macro overlap-free during iterations; for example, Capo [1] and FLOP [28]. (3) Three-stage mixed-size placement consisting of placement prototyping, macro placement, and standard-cell placement, as illustrated in Fig. 2; for example, XDP [10], CG [8], Floorist [21], MP-tree [11], CP-tee [7], and Chiou et al [5]. The three-stage approaches attract most attention due to its better performance and the easiness to be integrated into most of design flows. Recently, MP-tree is proposed to efficiently pack macros [11]. CP-tree inherits the strength of MP-tree and additionally considers pre-placed blocks [7]. Since the above two representations have the deficiency in handling macro overlapping, the circular-contour-based macro placement using corner sequence [20] is then developed [5]. However, all the three-stage macro placers rely on a placement prototype that is generated from a one-stage mixed-size placement. The placement prototyping step not only requires more runtime for the additional step, but it may also result in some deficiencies in

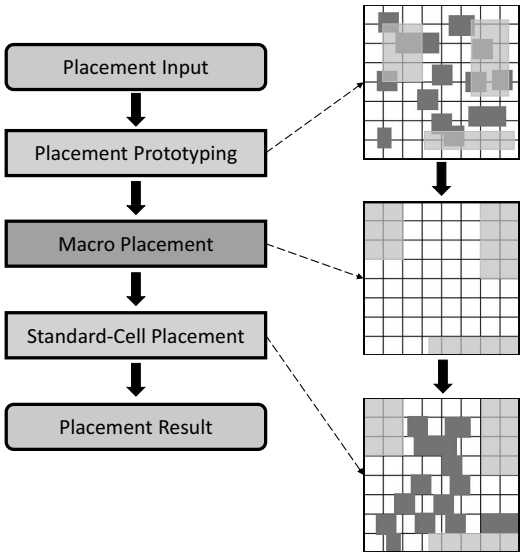

Fig. 2. Three-stage mixed-size macro placement.

routability optimization. Since the placers estimate the wirelength and routability costs according to the macro positions in the prototype and minimize the displacements of macros from the prototype in the objective function, the final macro placement could be quite similar to that in the prototype. In other words, the prototype may limit the optimization solution space and become the major factor to determine the outline of the optimized macro placement. In contrast, if the macro displacement is set as a secondary priority in the objective function such that the macro placers are more flexible to generate significantly different macro placements, the wirelength and routability estimations become inaccurate and the generated macro placement may not be the optimal one.

Since no effective cost metric exists to directly evaluate a macro placement due to the extremely high complexity and unpredictability of cell placement and routing, macro placement is an optimization problem that could reasonably appeal to the help of machine learning. There have been several works on DRV prediction during physical design. Some of them predict the number of DRVs and congested regions of a given cell placement by considering congestion maps after global routing or trial routing [23], [29]. However, the miscorrelation between congestion maps and final DRV maps has been shown because of more and more complicated design rules in advanced process nodes [4], [6]. To avoid being misled by improper feature and to save the runtime of global/trial routing, other works try to predict routability of a placement without congestion maps [4], [6], [25], [26]. [25] and [26] develop machine learning-based methods to predict whether short violations occur in a layout tile. The model proposed by [6] predicts whether a placement solution is routable. [4] proposes a support vector machine (SVM)-based method to predict the locations of DRVs. Nevertheless, all these previous works perform local routability prediction for a given cell placement, and the existing models are not applicable to the macro placement stage. In addition, as the example shown in Fig. 1, the routability of a design, quantified by the number of DRVs, could be mainly determined by macro placement. Therefore, developing a routability prediction model for macro placement and a macro placer guided by such model is desirable.

In this paper, we propose the first work, to the best of our knowledge, of routability-driven macro placement with deep learning. A convolutional neural network (CNN)-based routability prediction model is proposed and embedded into a macro placer such that a good macro placement with minimized DRC violations can be derived through a simulated annealing (SA) optimization process. The major contributions of this paper are summarized as follows:

- It is the first work of routability prediction with machine learning at the macro placement stage. A routability predictor is built to predict the number of DRVs. The input of the prediction model only requires a macro placement and does not need the information of cell placement and routing.
- Transfer learning is adopted to solve the problem of training data insufficiency. The pre-trained VGG16 [24] architecture is used and fine-tuned with our dataset of macro placements.
- The developed machine learning-based routability predictor is embedded into a macro placer. Besides, the simulated annealing optimization is used to search the optimal macro placement with the expected fewest DRVs.
- Experimental results demonstrate the accuracy of the prediction model and the effectiveness of our routability-driven macro placer.

The rest of this paper is organized as follows: Section II introduces the problem formulation and the preliminaries of CNN. In Section III, the proposed model development and macro placement methods are detailed in each subsection. Experimental results are presented in Section IV. Finally, we conclude our work in Section V.

II. Problem Formulation and Preliminaries

In this section, a formal problem formulation is given in II-A. Then, we introduce the architecture and preliminaries of CNN in II-B. After that, the motivation of applying transfer learning is explained in II-C.

A. Problem Formulation

The objective of our work is to solve two problems: (1) Early routability forecast for a given macro placement; (2) Finding the best macro placement with minimized design rule violations (DRVs). The formal problem formulations are given as follows:

Problem 1: In the first problem, the task is to find a model predicting the number of DRVs (denoted as #DRVs) of a given macro placement, which can be expressed as a function $f_{\#DRVs}$ as follows:

$$f_{\#DRVs} : S_i \in \mathbb{R}^{W \times H} \rightarrow y_i \in \mathbb{N} \tag{1}$$

The input is a macro placement $S_i \in \mathbb{R}^{W \times H}$, which is a two-dimensional rectangular layout with width W and height H. The output is the predicted #DRVs denoted as y_i after cell placement and routing. Our goal is to find an optimal function $f^*_{\#DRVs}$ minimizing the loss of #DRVs prediction:

$$f^*_{\#DRVs} = \arg\min_{f} Loss(f_{\#DRVs}(S_i) - y_i) \tag{2}$$

Problem 2: In the second problem, the input P_i is a placement instance composed of macros, standard cells, and chip area $W \times H$. Our goal is to find a macro placement S_i of a design with minimized predicted #DRVs after cell placement and routing; that is, performing macro placement with the following objective function:

$$\min \quad f^*_{\#DRVs}(S_i) \quad | \quad S_i \in \mathbb{R}^{W \times H} \tag{3}$$

B. Convolutional Neural Network

CNN is a neural network that is a system of interconnected artificial neurons with convolution operations and exchanging messages among each other. The numeric weights of connections are tuned during the training process. CNN has been extensively applied on many different research fields. Fig. 3 is a common CNN architecture for image processing. A typical CNN is a feed-forward network composed of convolutional layers, pooling layers and fully-connected layers.

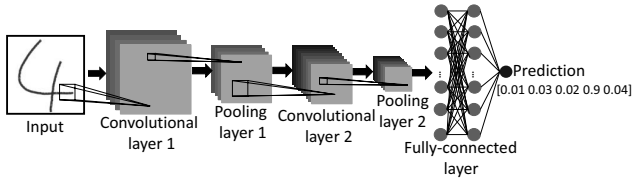

Fig. 3. The architecture of a common CNN for recognizing handwritten digits.

The convolutional layer is a critical part of CNN, which applies convolution operations to the input and passes the result to the next layer. The purpose of convolution operations is to extract specific features of the input image. A convolutional layer consists of a set of trainable filters (kernels), each of which is a small matrix of values sliding over the whole input image. In the sliding process, each filter convolves with the input data and extracts local feature, such that a complete feature map can be generated. Finally, an activation function is applied, which can be non-linear in order to approximate any non-linear function.

Fig. 4 illustrates the convolution operation with two 3×3 filters. The input data is an 8×8 matrix with an L-shape pattern. Filter 1 is a feature detector to detect vertical edges by computing horizontal gradients. After completing the convolution operation, a 6×6 Feature Map 1 is generated, which indicates the occurrence of vertical edges. Similarly, as illustrated in Feature Map 2, the horizontal edges are successfully detected by using Filter 2. The filters in CNN are not necessary to be initialized but are automatically trained during the model training process for performance optimization.

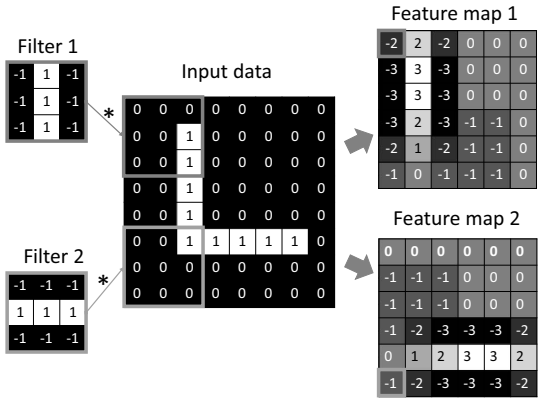

Fig. 4. A convolutional layer with 3×3 filters for image processing.

A pooling layer is usually inserted in-between successive convolutional layers in order to reduce the amount of parameters and computations in a network. Pooling which is also called subsampling, is used to filter off noise and keep representative input feature. In addition, overfitting can also be controlled with the adoption of pooling layers. There are many different pooling operations, such as max pooling, average pooling, and L2-norm pooling. In this work, max pooling is adopted due to its simplicity and its sufficient ability to keep representative features. Fig. 5 demonstrates a max pooling operation with a 2×2 pooling window, which is performed by taking the maximum value in each pooling window.

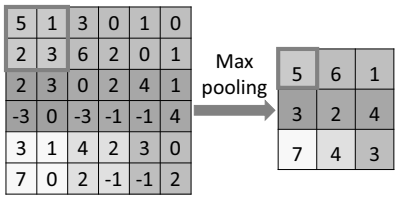

Fig. 5. Max pooling operation with pooling size 2×2.

C. Transfer Learning

Transfer learning has been extensively used in the machine learning filed, which transfers the knowledges from a well-studied problem to a different but related problem [22]. Since it is very difficult to train a model from scratch (with random initialization) when available dataset is not sufficient to train a full network, using an existing network that has been well-trained with a huge dataset (such as ImageNet [13]) and fine-tuning it with the limited dataset of the target task is more realistic.

In our work, in order to predict #DRVs, the whole design flow needs to be accomplished to get the final routing result, and thus it is extremely time-consuming to get huge training data. Table I shows the average runtime in seconds to generate a single training data of each circuit from the 2015 ISPD benchmark [2], where GP, DP, GR, and DR separately denote global placement, detailed placement, global routing, and detailed routing. It can be observed from the table that generating a data costs 7–38 minutes. Therefore, we adopt the VGG network architecture as the base of our CNN model and fine-tune it with the limited training data we can derive in reasonable time [22].

TABLE I
THE AVERAGE RUNTIME OF ACCOMPLISHING PLACEMENT AND ROUTING FOR FIVE CIRCUITS.

Circuit Name	Runtime (s)				
	GP	DP	GR	DR	Total
des_perf_a	103.01	471.83	306.09	1404.97	2285.89
matrix_mult_a	402.87	524.49	324.42	876.06	2127.85
matrix_mult_b	301.87	520.03	299.86	770.86	1892.62
fft_a	24.62	90.26	169.28	185.06	469.23
fft_b	23.77	89.56	174.88	153.11	441.32
Average	171.23	339.23	254.91	678.01	1443.383

III. Macro Placement with Embedded Routability Prediction Model

In this section, we introduce our CNN-based routability prediction model and the macro placer that can find an optimized macro placement with the fewest predicted #DRVs. Section III-A first introduces feature extraction for model training. Then, Sections III-B and III-C respectively present the prediction model and SA-based macro placer.

A. Feature Extraction

In this work, the routability of a macro placement is quantified by the #DRVs after cell placement and routing, and thus a CNN model that can accurately predict the number of DRVs for a given macro placement is required. As the input of the CNN model, we regard the layout of a macro placement as a three-dimensional

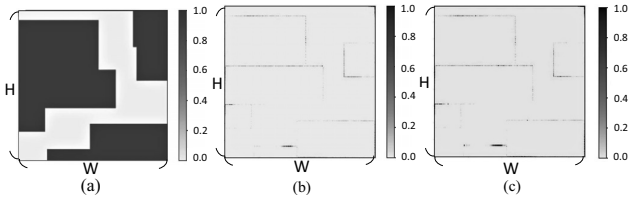

Fig. 6. The Illustration of three two-dimensional feature maps. (a) Macros. (b) Pin density. (c) Connectivity.

tensor, which is constructed by stacking two-dimensional feature maps. The three feature maps used in this work are as follows:

- **Macro density map:** Given a macro placement generated from a macro placer, the locations of macros are fixed such that a density map can be derived by computing the macro coverage rate of each pixel.
- **Pin density map:** At the macro placement stage, the locations of cells are undetermined. Hence, only the density distribution of macro pins are calculated.
- **Connectivity map:** The connectivity of a macro pin indicates the number of nets connecting to the pin. The connectivity map gives the connectivities of macro pins at the corresponding pixels.

Fig. 6 illustrates the three two-dimensional feature maps served as the input of the CNN model. The three feature maps with size $W \times H$ are generated independently by calculating the three features of a macro placement S_i . After that, a three-dimensional input tensor $X_i \in \mathbb{R}^{W \times H \times 3}$ is constructed by stacking these three feature maps. Note that feature normalization is applied to each feature map. It is because if a large variation exists among feature maps, some features could have deterministic influence on the training model and the others could not. Therefore, we normalize each feature map such that the values are between 0 and 1. The normalization also contributes to the faster convergence of the training model.

B. Routability Prediction

As mentioned in Section II-C, due to insufficient training data, we adopt transfer learning to train our prediction model. VGG16, a deep convolutional neural network, is adopted in our work, where "16" means the depth of the configuration. Fig. 7 shows the architecture of VGG16, which can be divided into two parts: the convolution part and the fully-connected part. The convolution part is constructed by stacking convolutional layers and max-pooling layers, and "$m, n \times n$ CONV" in each convolution layer means that the layer uses m $n \times n$ filters. In the fully-connected part, the neurons in a fully-connected layer have full connections to all activations in the previous layer. Finally, VGG16 outputs 1000 floating values, each of which is the probability that the input image belongs to one of the 1000 classes.

To generate required data, a macro placer is implemented to derive a set of different macro placements and the EDA tool $Encounter$ is used for cell placement and routing [3] . After design rule check, the #DRVs of each case is reported. The generated macro placements are our training data and testing data, and the corresponding #DRVs are served as the labels. Since the dimension of an input image of VGG16 is fixed as 224×224, each input tensor $X_i \in \mathbb{R}^{W \times H \times 3}$ is reshaped to $X_i \in \mathbb{R}^{224 \times 224 \times 3}$. In addition, the labels are also scaled such that their values fall between 1 and m, where m is a user-defined number. The major reason can be explained with Fig. 8, where the #DRVs of 300 cases for each circuit are sorted and plotted. It can be observed from the figure that the distributions of #DRVs are highly diverse among different circuits, which is understandable as larger circuits usually suffer from more DRVs than smaller

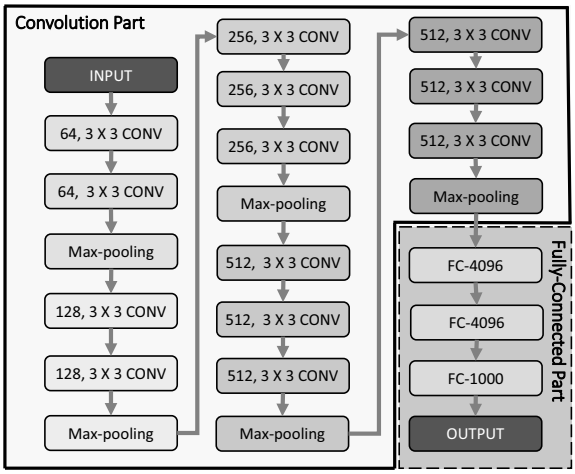

Fig. 7. The architecture of VGG16 [24]

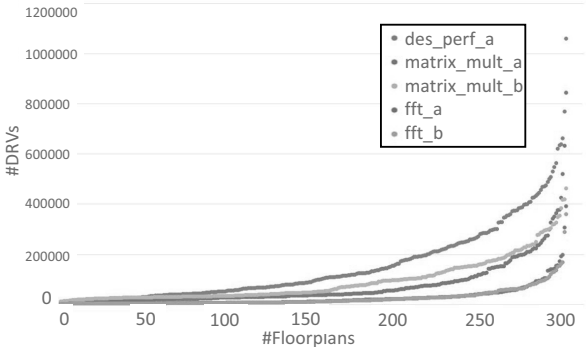

Fig. 8. The distribution of violations with five circuits.

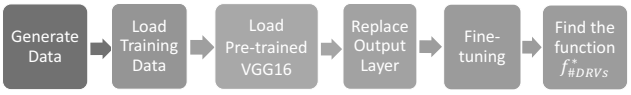

Fig. 9. The training flow of the routability predictor

circuits do. This phenomenon implies the difficulties in accurate #DRVs prediction for all circuits with a single prediction model. In addition, considering the objective of a macro placer, it is more important to find a good macro placement with relatively low #DRVs than accurately predict actual number of DRVs. Therefore, the labels (i.e., #DRVs) are scaled to $[1, m]$ first such that the model can be applicable to any input circuit.

Fig. 9 shows the training flow of the routability predictor. After the data preparation, the pre-trained VGG16 model is loaded. Since the volume of our training data is relatively small (compared with ImageNet used to train VGG16), we fix the convolution part of VGG16 that has strong ability of image processing and only fine-tune the fully-connected part. We replace the output layer from a 1000-classes classifier to a regressor that outputs one single number representing the scaled #DRVs. All hidden layers are equipped with the activation function–Rectified Linear Unit (ReLu) [15]. Mean squared logarithmic error (MSLE) is used as the loss function and Adam is used for optimization [16]. Besides, a convolution operation will shrink an input image, while losing the information of chip boundaries is undesirable because macros are often placed close to the boundaries. Therefore, padding is adopted such that the output has the same dimension

as the original input. Furthermore, the dropout layers are set with dropout fraction 0.25 to prevent overfitting.

C. Simulated Annealing Optimization

The ultimate goal of our work is to find the best macro placement with the fewest predicted #DRVs for a given circuit. Therefore, we embed the prediction model developed in Section III-B into a macro placer and apply the simulated annealing (SA) algorithm to globally search the optimal solution. We implement the circular-contour-based macro placer, which uses a corner sequence [20] as the representation of a macro placement [5]. Given a corner sequence CS of the current solution, three perturbation operations are adopted during SA:

- **OP1:** Exchange the i-th and j-th macros in CS.
- **OP2:** Insert the i-th macro after the j-th macro in CS.
- **OP3:** Assign a different corner for each macro.

The perturbation operations are iteratively performed until a legal macro placement is derived. After deriving a new solution of macro placement $S' \in \mathbb{R}^{W \times H}$, its reshaped tensor $X' \in \mathbb{R}^{224 \times 224 \times 3}$ is fed into the prediction model. If the predicted #DRVs of S' is fewer than that of the previous solution S, S' will be accepted. In contrast, S' could also be accepted with a probability if it is worse than S. The output will be the best macro placement found by the SA algorithm before the termination condition is met.

IV. Experimental Results

The ISPD 2015 placement benchmarks are used to generate the training and testing data, and the circuits with similar numbers of macros are adopted. The statistics of the circuits are shown in Table II. According to the chip sizes, we classify these five circuits into three types, which are small (fft_a and fft_b), medium (des_perf_a), and large (matrix_mult_a and matrix_mult_b) circuits. For each circuit, we generate at least 300 significantly different macro placements by using the implemented macro placer and guarantee 600 macro placements of each circuit type. Then, the EDA tool Cadence $Encounter$ is used to accomplish cell placement and routing [3], and the reported numbers of DRVs after DRC are treated as the labels. The routability prediction model is implemented in Python, and the SA-based optimization and dataset generation are implemented with the C++ programming language. The model training process is implemented on a machine with 3.7GHz CPU and an Nvidia GeForce GTX 1080 Ti graphics card, and the other experiments are conducted on a 2.3GHz machine with 64GB memory.

A. Routability Prediction

To verify the prediction ability of the proposed model, the model is trained by four of the circuits and tested by the other one. This guarantees that the testing circuit is an unseen design for the trained machine learning model to meet the requirement in industry dealing with many new and unseen designs. We sort all the macro placements of each testing circuit by its true #DRVs in ascending order, denoted as T, and we also sort the macro placements by its predicted #DRVs in ascending order, denoted as P. Since we are interested in whether those top macro placements (with fewer #DRVs) predicted by the machine learning model are really good solutions, we report the best true rank in the top n macro placements of P. As an example shown in Fig. 10, one of the top 10 macro placements predicted by the trained model is the real best one with the lowest true #DRVs. Table III shows the experimental results. It can be observed that the 1st or 2nd best macro placement of each circuit can be found by the predictor in its Top-30 candidates, and most of them can be found in Top-10 or Top-20 candidates. The results show that the predictor has certain ability to predict the quality of a macro placement in terms of routability.

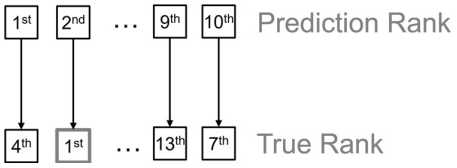

Fig. 10. The illustration of #DRVs prediction comparison.

B. Simulated Annealing Optimization

We embed the routability predictor into the implemented macro placer and use the SA-based optimization method to find the best macro placement with the fewest #DRVs. For comparison, we implement a baseline circular-contour-based macro placer without the embedded model [5]. Because placement prototyping is not considered in our work, we implement the baseline macro placer by only considering the boundary cost during SA optimization, such that the placer focuses on keeping more space and the completeness of routing region for standard cells. Due to the stochastic scheme, we repeatedly perform the SA-based optimization to get 100 macro placements for each circuit and for both our placer and the baseline placer. The resulting #DRVs are compared with the dataset generated at the beginning for model training and testing. Note that since $Encounter$ license was changed recently and only provides services for academic designs with fewer than 100k cells, only the two smaller circuits are tested in this experiment because it was conducted after that. Table IV shows the solution quality of the two macro placers, where $x\%$ in the column "Top-y" reports the percentage of macro placements whose solution quality are comparable with the Top-y macro placements of the dataset. According to Table IV, the macro placements of fft_a generated by the baseline placer comparable to the Top-10, Top-20, and Top-30 of the dataset are only 2%, 22% and 34%, separately, while 75%, 82% and 84% solutions from our placer are comparable to the Top-10, Top-20, and Top-30 of the dataset. Similar trends can be found in the statistics of fft_b. Furthermore, 34% and 24% macro placements of fft_a and fft_b generated from our placer are even better than the best macro placements in the dataset, which implies that the prediction ability of the trained routability prediction model contributes to finding much better solutions that are never seen in the dataset. The layouts of the best macro placements (with the least #DRVs) respectively derived from our placer and the baseline macro placer are shown in Figs. 11(a) and (b). After accomplishing cell placement and routing, the baseline macro placement results in 2426 violations, while only 851 violations are generated from our macro placement. Besides, the average total wirelength (tWL) and the average runtime are also reported in Table IV. For the two circuits, our placer can achieve 9.16% and 5.35% tWL reduction compared to the baseline placer, which should be due to the better routability of macro placements. Note that our placer requires more runtime for routability prediction where placement reshaping, feature extraction, and CNN inference are performed in each SA iteration. Nevertheless, not only the runtime overhead is acceptable, but the proposed flow can greatly save the overall design time from running the whole back-end flow for hundreds of macro placements to run only tens of them.

V. Conclusions

This paper has proposed the first work of routability-driven macro placement with deep learning. A CNN-based routability predictor is built to predict #DRVs at the macro placement stage without cell placement and routing information. Besides, we embed the routability predictor into a macro placer to generate routability-driven macro placements. Moreover, the SA optimization is applied to find a macro placement minimizing

TABLE II
THE ISPD 2015 BENCHMARKS WITH FIVE CIRCUITS.

Circuit Name	ISPD 2015 benchmarks					
	#Macros	Size (μm^2)	Macro Coverage (%)	#Instances	#Nets	#Macro Placements
des_perf_a	4	900×900	50.46	108288	110281	600
matrix_mult_a	5	1500×1500	66.39	149650	154284	300
matrix_mult_b	7	1500×1500	62.45	146435	151612	300
fft_a	6	800×800	65.35	30631	32088	300
fft_b	6	800×800	65.35	30631	32088	300

TABLE III
#DRVS PREDICTION COMPARISON.

Circuit Name	Best Rank in		
	Top-10	Top-20	Top-30
des_perf_a	3rd	1st	1st
matrix_mult_a	1st	1st	1st
matrix_mult_b	13th	2nd	2nd
fft_a	2nd	2nd	2nd
fft_b	2nd	2nd	2nd

TABLE IV
COMPARE ROUTABILITY-DRIVEN MACRO PLACER WITH BASELINE MACRO PLACER.

Circuit Name	Baseline macro placer [5]					Our work					
	Top-10	Top-20	Top-30	tWL (mm)	Runtime (s)	Top-10	Top-20	Top-30	Better than Top-1	tWL (mm)	Runtime (s)
fft_a	2%	22%	34%	1409.19	1	75%	82%	84%	34%	1290.87	262.1
fft_b	8%	25%	33%	1393.97	1	79%	85%	86%	24%	1323.2	260.8

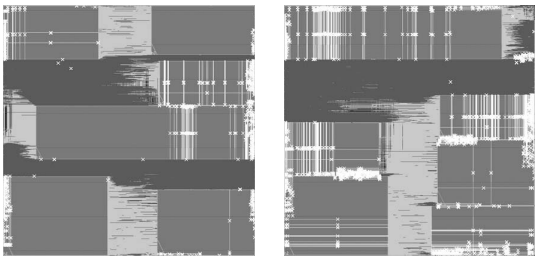

Fig. 11. The best macro placements derived from (a) our macro placer and (b) the baseline macro placer.

#DRVs. Experimental results show the accuracy of the routability predictor and the effectiveness of our macro placer.

REFERENCES

[1] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov. "Unification of partitioning, placement and floorplanning." In *Proc. of ICCAD*, pp. 550-557 2004.

[2] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis. "ISPD 2015 benchmarks with fence regions and routing blockages for detailed routing-driven placement." In *Proc. of ISPD*, 2015.

[3] Cadence encounter user guide. http://www.cadence.com

[4] W. T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena. "Routability optimization for industrial designs at sub-14nm process nodes using machine learning." In *Proc. of ISPD*, pp. 15-21, 2017.

[5] C. H. Chiou, C. H. Chang, S. T. Chen and Y. W. Chang. "Circular-contour-based obstacle-aware macro placement." In *Proc. of ASP-DAC*, pp. 172-177, 2016.

[6] W. T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi. "BEOL stack-aware routability prediction from placement using data mining techniques." In *IEEE ICCD*, pp. 41-48, 2016.

[7] Y. F. Chen, C. C. Huang, C. H. Chiou, Y. W. Chang and C. J. Wang. "Routability-driven blockage-aware macro placement." In *Proc. of DAC*, pp. 1-6., 2014.

[8] H.-C. Chen, Y.-L. Chuang, Y.-W. Chang, and Y.-C. Chang. "Constraint graphbased macro placement for modern mixed-size circuit designs." In *Proc. of ICCAD*, pp. 218-223, 2008.

[9] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints." In *IEEE TCAD*, 27(7):12281240, July 2008.

[10] J. Cong and M. Xie. "A robust mixed-size legalization and detailed placement algorithm." In *IEEE TCAD*, 27(8):13491362, August 2008.

[11] T. C. Chen, P. H. Yuh, Y. W. Chang, F. J. Huang and T. Y. Liu. "MP-Trees: A Packing-Based Macro Placement Algorithm for Modern Mixed-Size Designs." In *IEEE TCAD*, vol. 27, no. 9, pp. 1621-1634, 2008.

[12] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. "mPL6: enhanced multilevel mixed-size placement." In *Proc. of ISPD*, pp. 212-214, 2006.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database." In *IEEE CVPR*, 2009.

[14] M. K. Hsu and Y. W. Chang. "Unified analytical global placement for large scale mixed-size circuit designs." In *IEEE TCAD*, vol. 31, no. 9, pp. 3661378, 2012.

[15] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H S. Seung. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." In *Nature*, vol. 405, no.6789, pp. 947, 2000.

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

[17] M.-C. Kim and I. L. Markov. "ComPLx: A competitive primal-dual lagrange optimization for global placement." In *Proc. of DAC*, pp. 747-752, 2012.

[18] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji. "MAPLE: Multilevel adaptive placement for mixed-size designs." In *Proc. of ISPD*, pp. 193-200, 2012.

[19] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng. "ePlace-MS: Electrostatics based placement for mixed-size circuits." In *IEEE TCAD*, 34(5):685698, January 2015.

[20] J. M. Lin, Y. W. Chang, and S. P. Lin. "Corner sequence—A P-admissible floorplan representation with a worst case linear-time packing scheme." In *IEEE TVLSI*, vol. 11, no. 4, pp. 679-686, 2003.

[21] M. D. Moffitt, A. N. Ng, I. L. Markov, and M. E. Pollack. "Constraint-driven floorplan repair." In *Proc. of DAC*, PP. 1103-1108, 2006.

[22] H. Qassim, A. Verma and D. Feinzimer. "Compressed residual-VGG16 CNN model for big data places image recognition." In *IEEE CCWC*, pp. 169-175, 2018.

[23] Z. Qi, Y. Cai and Q. Zhou. "Accurate prediction of detailed routing congestion using supervised data learning." In *IEEE ICCD*, pp. 97-103, 2014.

[24] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." In *Visual Geometry Group, Department of Engineering Science, University of Oxford*, 2014.

[25] A. F. Tabrizi, N. K. Darav, S. Xu, L. Rakai, I. Bustany, A. Kennings and L. Behjat "A machine learning framework to identify detailed routing short violations from a placed netlist." In *Proc. of DAC*, pp. 48, 2018.

[26] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings and L. Behjat. "Detailed routing violation prediction during placement using machine learning." In *Proc. of VLSI-DAT*, pp. 1-4, 2017.

[27] N. Viswanathan, M. Pan, and C. Chu. "Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control." In *Proc. of ASP-DAC*, pp. 135-140, 2007.

[28] J. Z. Yan, N. Viswanathan, and C. Chu. "Handling complexities in modern large-scale mixed-size placement." In *Proc. of DAC*, pp. 436-441, 2009.

[29] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou, and Y. Cai. "An accurate detailed routing routability prediction model in placement." In *ASQED*, 2015.