A Machine Learning Framework to Predict Design Violations Based on Placement Information

Yuxuan Pan
School of Electrical and
Computer Engineering
the University of British Columbia
Vancouver, British Columbia
Email: yuxuanpan@ece.ubc.ca

Steve Wilton
School of Electrical and
Computer Engineering
the University of British Columbia
Vancouver, British Columbia

***Abstract*—The routing stage is one of the most time-consuming, and thus the most important steps in modern VLSI design. Being able to predicate the design rule violations in early stage of routing, or even before actual routing, can guide the router to avoid possible violations, improving the performance and speed of the router. In this paper we proposed a machine learning based algorithm to predict the design rule violations, including wire short, via short, wire spacing and via spacing violations, based on placement information. Two machine learning frameworks were tested on industry level data sets. Even though the prediction results were not very satisfying, it still outperformed [1] in terms of accuracy when the number of violations was greater than 500. Moreover, reasons causing such poor performance have been looked into and possible solutions were proposed as future work. This work also cast a doubt on feature vector based machine learning violation prediction frameworks, which have gained a lot of popularity in recent years.**

I. Introduction

Nowadays, as the fabrication technology improves and the area of SoC increases, the challenging that routers are facing also increases. To eliminate the violations, *e.g.* short violations and spacing violations, a technique called rip-up reroute (RRR) is adopted in most state-of-the-art routers, meaning the routers rip-up the entire or just part of the design if violations existed, and reroute the design based on the guidance of last route. Intuitively, if the violations could be predicted before routing take place, it could save a lot of time and improve the performance of router, because the predicted results, if accurate, could guide the router before and in the middle of RRR iterations.

Many efforts have been put into this field of design rule violation prediction. Our work is inspired by A. Tabrizi *et al.*[1], who proposed a machine learning framework to do the prediction based on a placed netlist. A design is first divided into many tiles, each of which is assigned to a feature vector containing the features of that tile, such as number of pins, number of global nets, number of local nets, etc. The biggest advantage of this work is that the prediction can be done based on a placed netlist, which means before actual routing, which makes it practical. However, as many other previous works, it is performed on toy-level data sets, and suffers from low accuracy. In some designs the accuracy is merely 55%. The reason is that the data is too imbalanced, because there are too many negative samples while a little portion of positive samples. Thus the model has to over estimate violations to mitigate the imbalance of data, resulting in a low true negative rate. In some large designs, the true negative rate is about 50%, which is more like a blind guess.

Our work, similar to [1], also uses a machine learning framework to do the prediction. However, different from all previous work, which were more like explorations on the feasibility of machine learn based violation prediction. Our work is, to the best of our knowledge, the first to perform neural network based violation prediction on industry level designs like ISPD18 and ISPD19. Different machine learning frameworks, like neural network and SGD were tested and tuned. Although the neural network based framework outperformed [1] in large designs, the accuracy was still not very satisfying. From the results of experiments and analysis based on that, we tend to doubt the effectiveness of such feature vector based violation prediction methods.

II. Methodology

This work is not a simple re-implement of [1] on another data set. Two works vary a lot in many aspects like data set, feature extraction, types of violations, machine learning framework and training method. The work flow of proposed framework is shown in Fig. 1. First, a placer does placement on a netlist, the output, which is a placed netlist, will be fed into a router, to finish the global and detailed routing. The violations in the routed results will be saved and later be used to train the machine learn model as the target outputs. The proposed work takes four categories of violations into consideration: wire short violation, via short violation, wire spacing violation and via spacing violation[3], while [3] is only able to predict wire violations. Meanwhile, the placed netlist is partitioned into many tiles based on the resolution of global router. Each tile is assigned to a feature vector, which describe the features of that tile, such as number of pins, number of nets, etc. And that feature vector is the input of a machine learning model. Afterwards, the training of machine learn model is completed by the inputs, which are feature vectors, and target outputs, which are violation results.

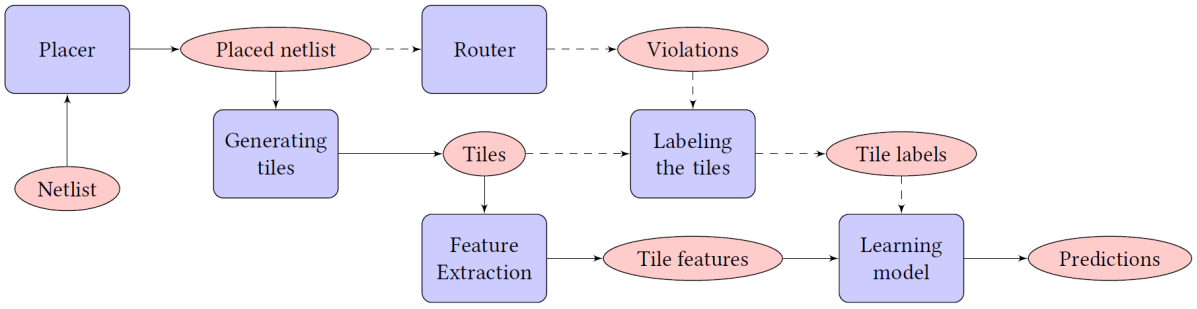

Fig. 1: The flow of the proposed violation detection framework and its integration in physical design flow

A. Feature Extraction

Seven different features are extracted, forming a feature vector containing 23 elements.

- **Pin density:** The number of pins in a tile.
- **Neighborhood pin density:** the average number of pins in surrounding tiles.
- **Net density:** Given a two-pin net bounding box, two L-shape edge paths are possible along the boundaries of that area. The total wire length of these two routes reflects the density over a given layout region. This feature mimics how a pattern routing performs the congestion estimation. Another approach of calculating the net density is called RUDY [].
- **Number of local nets:** Number of the nets that have at least two pins in the tile.
- **Number of global nets:** Number of the nets that have at least one pin in the tile and at least one pin outside the tile.
- **Wire capacity:** Given a tile, the wire capacity of that tile on a specific layer is defined as the maximum number of wires that can go through the tile on that layer. It's also equal to the number of tracks of the tile on that layer. Because each design has 9 layers, each tile has 9 wire capacities.
- **Via capacity:** Given a tile, the via capacity of that tile on a specific layer is defined as the number of vias of the tile on that layer, which is equal to the number of cross points of the tracks of that tile on target layer and tracks on neighboring layers. Each tile has 9 via capacities for the same reason as wire capacity.

B. Machine Learning Models

Two different machine learning frameworks were tested, Stochastic gradient descent (SGD) and neural network. The reason we chose these two frameworks was that the training set was too large that it must be trained in batches. Therefore many mahine learning methods like SVM couldn't be used.

One problem about the data is its imbalance. In real designs, there usually can't be too many violations, which makes them difficult to predict. Machine learning models may predict all samples as negative, and still achieve a high accuracy. But being able to predict violations could be critical because we want to eliminate all violations. If we underestimated, the predictor would make no sense by predicting all samples as negative. On the other hand, if we overestimated, the classification accuracy would be very low because of the false alarms.

Weighted loss is used to solve the imbalance problem. Weighted loss is defined as follow:

$$H(p,q) = -\sum_{x} a_x p(x) \log q(x) \tag{1}$$

where a_x is a weight, which is inversely proportional to the number of sample x in training set. The idea of weighted loss is that make minority class contribute more to the loss.

III. Experiments

A. Benchmarks

The benchmarks used in our work, which were ISPD 2018 and ISPD 2019 benchmarks, are different with that in [1], which was ISDP 2015 in terms of size, complexity and the ratio of negative and positive samples.

- **Size.** The 16 designs in ISPD 2015 have an average number of nets of 301, which is much smaller than that of ISPD 2018 and ISPD 2019, which have 209099 nets in average for each design. In our experiment, the resolution of tiles was the resolution of global routing, which means the number of input samples was equal to the number of global routing tiles, which was more than 10 million. This number was significant larger than that of [1], which was 241123.
- **Complexity.** The designs in ISPD 2015 are mostly not real circuits, they are just designed deliberately to be some shapes to test the routers, which can be very impractical. ISPD 2018 and ISPD 2019, however, contains some complicated circuits, like some Quad-core designs with 16 memory blocks, using 32nm technology.
- **Imbalance.** The ratio of positive and negative samples is 15, while this number in ISPD 2018 and ISPD 2019 is 200. For a classifier, the more imbalanced the data is, the harder to give accurate predictions.
- **Data division.** In our experiments, all benchmarks (18 designs) were divided into a training set of 12 designs and a test set of 6 designs. While in [1], when doing

prediction on a specific design, all designs besides the target design were used as training set, so the model needed to be retrained for each design.

To summarize, ISPD 2015 is more like a toy-level data set for research. ISPD 2018 and ISPD 2019 are closer to real circuits in industry. We believe achieving good results on a small data set could be useful to explore the feasibility of violation prediction. However, as many works have already proved it, it is time to make it more practical.

B. Router

The ISPD 2018 and ISPD 2019 data sets provide with global routing guides. Therefore, a global router was not needed in our experiments. Dr. CU [3] was used as the detailed router. Dr. CU is a detailed router proposed in 2019, and it is one of the state-of-the-art routers.

C. Machine Learning Models Setup

1) SGD: A Pyhton library: Scikit-learn was used for SGD. Different hyper-parameters, including learning rate, number of iterations, penalty were tested. However, we failed to find a sweet point and the model didn't converge.

2) Neural Network: The neural network was implemented based on Python library: Pytorch. The input of this neural network, which is a feature vector of a tile, has 23 elements. This network has two layers, the first layer has 25 neurons, relu was used as the activation function, and the second layer, which is the output layer, has two neurons, with softmax being the activation function. The learning rate was set to $\alpha = 0.1$. The weights of positive and negative loss were 216.3440 and 1 respectively.

D. Results

Because the SGD never converged, only the performance of neural network will be discussed in this section. The true positive rate (TPR), true negative rate (TNR), and overall accuracy is shown in Tab. I. As we can see in the table, neither TPR nor TNR were satisfying. The overall accuracy was merely 68.05%. One the over hand, the average accuracy of [1] was 91% and 93% when the number of shorts were 100 to 500 and less than 100 respectively. However, it is unfair to compare our work with predictor on such simple benchmarks. As shown in Tab. II, when the number of shorts was over 500, our work outperformed [1], with the accuracy being 65%.

TABLE I: Results of Proposed Nueraul Network

	TPR	TNR	Overall Accuracy
Rate (%)	63.92	68.07	68.05

TABLE II: Comparison of Accuracy between Two Networks when Number of Violations Are Greater than 500

	Proposed Network	A. Tabrizi's Network
Accuracy (%)	68.5	65

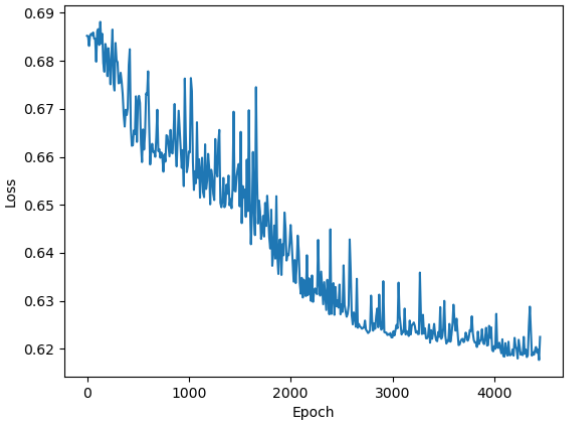

Fig. 2: Learning Rate of Proposed Neural Network

E. Analysis

Reasons resulting in such poor performance have been looked into. Based on the data we obtained from the experiments, several possible explanations and their solutions have been proposed:

1) Too Many Samples: Lacking of data has always been a huge problem in the field of routing algorithms. But in this work, too many of them may also be a problem. Due to the limitation of input size, this is a very small neural network. Training a small neural network using so many samples may result in not converging of the network. With over 10 million training samples, it's possible that similar input vectors have opposite outputs, which confuses the model. As shown in Fig. 2, after 4000 epochs, the loss decreased from 0.69 to 0.62, which was not a big decrease. Larger learning rate was test, but the loss didn't go lower either.

One way to solve this problem is to decrease the resolution, which could decrease the number of sample significantly and lower the difficulty of prediction at the same time. However, this may make violation prediction less useful for routers. As higher definition may guide routers to avoid violations better.

2) Lacking Global Information: The way we were doing prediction was that we assigned a vector to each tile, meaning that each tile is irrelevant to other tiles. However, in reality, whether a tile will have violation or not could be influenced by the availability of neighboring tiles, e.g., a tile surrounded by congested tiles are more likely to have violations.

Global information can be introduced to solve this problem. For example, each feature vector also contains features of neighboring tiles, making the model be aware of the big picture of a design, not just a single tile. Moreover, this could increase the input size, and make a larger neural network possible.

3) Different Data Sets: Two different data sets: ISPD 2018 and ISPD 2019 were used. Similar these two data sets are, there may be some differences between them. Due to time limit, we didn't divide these two data sets.

F. Questions

The more we dug into this field of violation prediction, the more we became to doubt the feasibility and practicality of feature vector based violation predictions.

Many current works are similar to our work, using a feature vector which doesn't contain global information to represent each tile. There are two main advantages of this: fast to run and easy to implement. However, most feature vector based predictors failed to work well on large benchmarks.

If the resolution is fine, a single modern VLSI design may have millions of tiles to predict, the limited information provided by feature vectors can hardly handle such complicated situations.

IV. Future Work

In the future, we plan to use global information to improve the performance of this work. A larger neural network will be adopted to deal with a larger input. Also, ISPD 2018 and ISPD 2019 will be trained and tested separately for better performance.

V. conclusion

In this work we proposed a machine learn framework to predict the design rule violations based on a placed netlist. Experiments have been down to test the performance of proposed framework. Dr. CU was used as the router, and four different design rule violations were taken into consideration. Two machine methods: SGD and neural network were tested. The neural network based framework outperformed [1] in terms of accuracy in large designs. Moreover, drawbacks of our work have been looked into and some possible solutions have been proposed.

References

[1] Tabrizi A F, Rakai L, Darav N K, et al. A machine learning framework to identify detailed routing short violations from a placed netlist[C]//2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018: 1-6.

[2] Alawieh M B, Li W, Lin Y, et al. High-definition routing congestion prediction for large-scale FPGAs[C]//2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2020: 26-31.

[3] Gengjie Chen, Chak-Wa Pui, Haocheng Li, and Evangeline F.Y. Young, "Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 39, no. 9, pp. 1902–1915, 2020.