# Pin Accessibility and Routing Congestion Aware DRC Hotspot Prediction using Graph Neural Network and U-Net

Kyeonghyeon Baek[1], Hyunbum Park[2], Suwan Kim[3], Kyumyung Choi[4] and Taewhan Kim[5]
Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
{[1]khbaek, [2]hbpark, [3]suwankim, [5]tkim}@snucad.snu.ac.kr, [4]kmchoi@snu.ac.kr

## ABSTRACT

An accurate DRC (design rule check) hotspot prediction at the placement stage is essential in order to reduce a substantial amount of design time required for the iterations of placement and routing. It is known that for implementing chips with advanced technology nodes, (1) *pin accessibility* and (2) *routing congestion* are two major causes of DRVs (design rule violations). Though many ML (machine learning) techniques have been proposed to address this prediction problem, it was not easy to assemble the aggregate data on items 1 and 2 in a unified fashion for training ML models, resulting in a considerable accuracy loss in DRC hotspot prediction. This work overcomes this limitation by proposing a novel ML based DRC hotspot prediction technique, which is able to accurately capture the combined impact of items 1 and 2 on DRC hotspots. Precisely, we devise a graph, called *pin proximity graph*, that effectively models the spatial information on cell I/O pins and the information on pin-to-pin disturbance relation. Then, we propose a new ML model, called PGNN, which tightly combines GNN (graph neural network) and U-net in a way that GNN is used to embed pin accessibility information abstracted from our pin proximity graph while U-net is used to extract routing congestion information from grid-based features. Through experiments with a set of benchmark designs using Nangate 15nm library, our PGNN outperforms the existing ML models on all benchmark designs, achieving on average 7.8~12.5% improvements on F1-score while taking 5.5× fast inference time in comparison with that of the state-of-the-art techniques.

## 1 INTRODUCTION

In modern integrated circuit designs, the introduction of new complex design rules at the advanced technology nodes makes implementing chips very challenging. Particularly, the complex design rules put so much burden on physical design, demanding lots of iterations on the time-consuming process of cell placement and net routing to clean up all DRVs (design rule violations) before tapping out. Thus, at the placement stage, if we were able to identify, with high confidence, DRC (design rule check) hotspots that would be likely to occur at the routing stage, we can pay more attention on the cell placements in those DRC hotspots, so that the iteration process of placement and routing should quickly converge to DRV-clean.

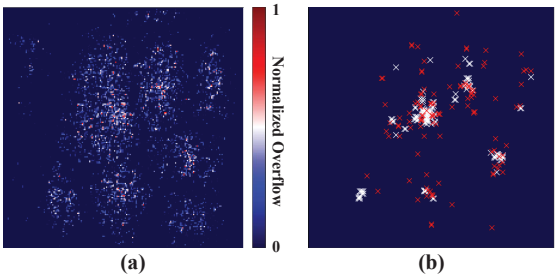



**Figure 1: Comparison between GR congestion map and actual DRC hotspots on ECG circuit. (a) GR congestion. (b) An overlay of actual hotspots (white crosses) and predicted hotspots (red crosses) by GR congestion in (a).**

Traditionally, commercial place-and-route tools have used *congestion map* delivered from initial global routing (GR) or trial routing as an early DRC hotspot estimator at the placement. However, as the technology node shrinks, the DRC hotspots predicted by GR congestion map alone is not fully correlated with the post-route DRC hotspots, as shown in Fig. 1. At the advanced node, it is known that *I/O pin inaccessibility* in standard cells is one of the main causes of the DRC hotspot misprediction. The high increase of pin inaccessibility in routing is attributed by the cell size reduction and the reduced number of routing tracks over the cells. Furthermore, as the routing complexity considering complex design rules increases, routers expose more often unpredictable behaviors, which is also one of main causes of the increase of DRC hotspot misprediction.

To sum up, *routing congestion* and *pin accessibility* are two major causes of DRVs. The prior works of DRC hotspot prediction previously had considered routing congestion, but paid little or no attention on pin accessibility, then gradually taking into account pin accessibility as the technology node shrinks.

The shortage of routing capacity in the area of routing congested region ends up committing design rule violations. In order to discover the routing congestion related DRVs, Chan *et al.* [1, 2] extracted, as features, the number of pins, the incoming and outgoing hyper-edges, and others from a local window, and used a SVM (support vector machine) model to predict occurrence of DRVs in the window. Hung *et al.* [3] used a CNN (convolutional neural network) model to capture routing congestion around a local window based on the congestion map (extracted from global routing information) of adjacent grids. Xie *et al.* (RouteNet) [4] utilized FCN (fully convolutional network) structure. They divided the entire placement region into grids of small size, and concatenated


*ICCAD '22, October 30-November 3, 2022, Gainesville, FL, USA*

ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549346

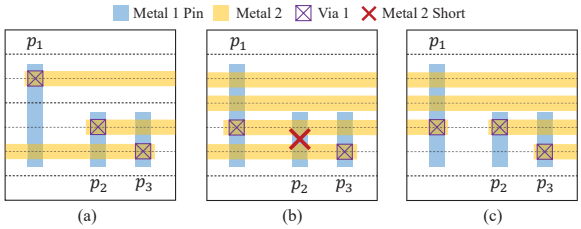



**Figure 2: Pin accessing situations.**

various DRV related features for each grid into three-dimensional feature map to maintain spatial information. One common feature of the aforementioned works is that they all focused on developing ML (machine learning) models of routing congestion aware DRC hotspot prediction and never incorporated pin accessibility into their models.

Recently, noticeable studies addressing pin accessibility aware DRC hotspot prediction have emerged. Yu *et al.* [5, 6] exploited CNN to predict DRVs mainly induced by low pin accessibility in a local window. They used fine-grained pin pattern images as input feature to express the shape of the pins in detail. Liang *et al.* [7] suggested a novel neural networks structure called J-net, which is a sort of modified model of U-net, adding a series of down-sampling modules in front of the encoding path of U-net to accommodate not only high-resolution pin pattern images but also grid-based feature maps as input feature. Although Yu *et al.* [5, 6] and J-net [7] both have tried to express pin information with fine-grained pin pattern images, these works have two critical limitations.

Firstly, local pin accessibility cannot be accurately modeled by pin pattern image alone, which represents only the detailed shape and location of the pins. Rather, DRC hotspots heavily rely on how metal wires approach to their target pins and how many metal wires go through over the pins together with the situation of accessing neighboring pins even though the pin information such as shape or location are identical. For example, Fig. 2 illustrates the different pin accessing situations with the same pin layouts, in which pins $p_1$, $p_2$, and $p_3$ are easily accessible if nothing tries to obstruct the connections as shown in Fig. 2(a). However, as illustrated in Fig. 2(b), if other metal wires expand metal tracks near the pins, a DRV occurs since $p_2$ is not accessible by being blocked by other metals. Even with the same pin layouts, if $p_1$ and $p_3$ are accessed through different directions, the inaccessibility to $p_2$ can be resolved as shown in Fig. 2(c). Secondly, using high-resolution pin pattern images incur significant additional run-time as well as memory overhead to the prediction models. J-net [7] used pin pattern images which required more than thousands of pixels for representing pin layouts in a single grid. However, as mentioned above, the pin layout itself does not accurately describe the pin accessibility problem. Furthermore, the models consume most of the computation time for processing the massive size of images.

To overcome these limitations, this work proposes a novel methodology of predicting DRC hotspots with a combined model of GNN (graph neural network) and U-net to accurately and efficiently capture the compound impact of pin accessibility and routing congestion on DRC hotspot prediction. The main contributions of our work can be summarized as follows:

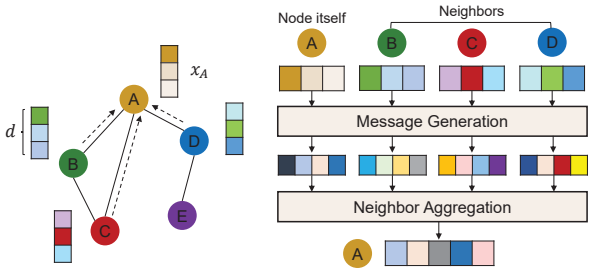



**Figure 3: An illustration of processing on a single graph convolutional layer in GNN.**

- We propose a *pin proximity graph* that effectively models not only the spatial information of each pin but also the information on how each net accesses to its target pin. To the best of our knowledge, this is the first work that uses a graph for modeling pin information.
- We propose *GNN architecture compatible with pin proximity graph*. The suggested architecture aggregates the information of the neighboring pins from the reference pin to consciously formulate pin accessibility.
- We propose a novel deep ML model, called PGNN (Pin accessibility aware GNN and U-Net), which is a combined model of GNN and U-net. PGNN can adopt pin proximity graph as well as grid-based feature map as input feature. GNN in PGNN embeds pin accessibility information of each grid taken from the pin proximity graph, and U-net in PGNN extracts routing congestion information from the grid-based features.
- In comparison with the prior works, which have utilized high-quality pin shape images for the construction of pin accessibility metrics, our graph formulation of PGNN tremendously saves the runtime for both training and inference.

## 2 PRELIMINARY

### 2.1 Graph Neural Network

GNN is a powerful deep ML model specialized for graph data. Since many EDA problems can be formulated into optimization problems on graphs, GNN has been widely adopted in this field. (e.g. [8, 9])

A graph $G(V, E)$ is defined as a set, $V$, of vertices and a set, $E$, of edges. A vertex matrix $V \in \mathbb{R}^{n \times d}$ is a two-dimensional matrix, in which $n = |V|$ and $d$ is the size of initial feature vector of each node in $V$. The edges in $E$ are represented by an adjacency matrix $A \in \{0, 1\}^{n \times n}$. A GNN takes $G(V, E)$ with $A$ as input and generates the embedding vector of every node in $G$. Similar to the convolutional layer of CNN, a graph convolutional layer is a main component of GNN, which iteratively updates node features considering the influence of the neighboring nodes.

The process on a graph convolutional layer consists of two main operations, which are *message generation* and *neighbor aggregation*. Fig. 3 illustrates the operation at a single graph convolutional layer. For each node, its neighboring nodes generate *messages* by applying learnable fully connected layers to its node feature. These messages are aggregated into one reduced message, and the feature of the node is updated by using the aggregated message. The graph

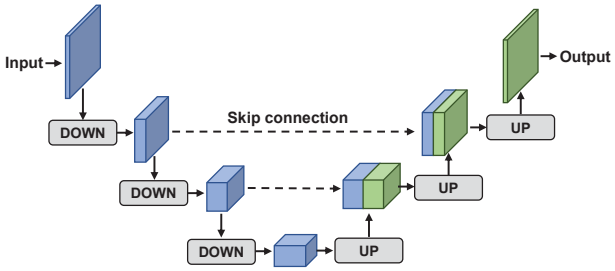



**Figure 4: U-Net architecture [13].**

convolutional layer operation can be expressed as follows [10]:

$$x_i^{(l)} = AGG^{(l)}(\{MSG^{(l)}(x_j^{(l-1)}),\ j \in \{N(i) \cup i\}\}) \quad (1)$$

where $x_i^{(l)}$ denotes a feature of node $i$ at the $l$-th graph convolutional layer, $N(i)$ denotes the neighboring nodes of $i$, and $MSG^{(l)}$ and $AGG^{(l)}$ indicate the message generation and neighbor aggregation operation in the $l$-th graph convolutional layer. $AGG^{(l)}$ is typically a sum, max or mean operation. Existing GNNs such as GCN [10] and GAT [11] differ with how graph convolutions are performed. The global pooling layer is often applied after completing the operations on GNN to obtain an embedding vector of the entire graph by aggregating the final features of all nodes in the graph.

## 2.2 Fully Convolutional Network

Fully convolutional network (FCN) [12] is proposed to solve semantic segmentation task, and all layers are composed of convolutional layers. Semantic segmentation in image recognition is to identify the class of every pixel in an input image. Thus, ML model for semantic segmentation should be able to take an arbitrary size of input images and outputs images of the same size as that of the inputs. DRC hotspot prediction problem can be seen as semantic segmentation if a kind of grid-based prediction is tried.

U-net [13] is an FCN based network and has achieved a great success in semantic segmentation. It adopts encoder-decoder structure like that shown in Fig. 4. Encoder gradually down-sample the input by applying a series of convolutional layers and pooling layers. Decoder is also comprised of the multiple repetitive up-sampling units, which takes a compacted feature map from the encoder as input, restoring them into the original size of input. Each up-sampling unit first receives the intermediate feature map from the encoder on the same level through the *skip connections*, and concatenates it with the output of the previous up-sampling unit. It typically applies transposed convolutional layers to scale up the input. The skip connection plays an important role in U-net, enabling the network to generate a high-quality prediction.

# 3 PROPOSED PREDICTION METHODOLOGY

## 3.1 Overall Flow

The overall flow of our proposed prediction methodology is depicted in Fig. 5. First, the entire placement region is divided into two-dimensional ($W \times H$) grids whose width and height are equal to that of G-cell. For the training data preparation, we conducted placement and routing on reference circuits with various placement

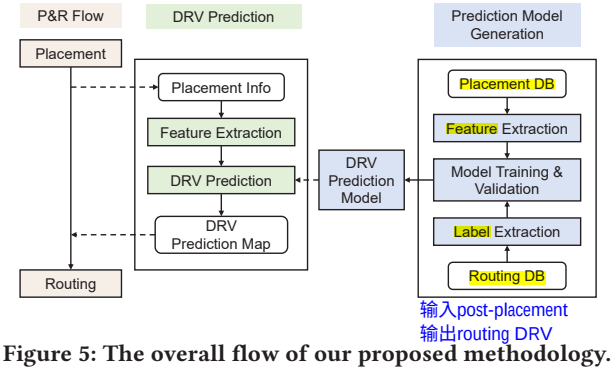



**Figure 5: The overall flow of our proposed methodology.**

settings. Input features are extracted at the placement and DRVs are extracted as the ground-truth after detailed routing. Output label is also a two-dimensional binary map with the same size of input feature map, indicating which grids are DRC hotspot.

Trained model can be adopted in the commercial place-and-route flow. After placement, input features are extracted from the placement results, and DRC hotspot prediction is generated by our prediction model, which can be utilized to optimize the placement before routing.

## 3.2 Pin Proximity Graph

There are four factors that affect pin accessibility; (1) shape (or length) of pins, (2) approaching direction of nets to connect pins, (3) pin accessing disturbance caused by neighboring pins, and (4) number of nets passing through the grid. To accommodate this information, we formulate the pin information in a local grid as *pin proximity graph* where each node indicates a distinct pin and an influence between two pins is represented by the existence of edge between the two nodes corresponding to the pins.

For each grid $g_{(x,y)}$, suppose that there is a set of pins $P = \{p_1, p_2, ..., p_i\}$ in $g_{(x,y)}$. In pin proximity graph $\mathcal{G}_{(x,y)}(V, E)$, every node $v_i \in V$ corresponds to a distinct pin $p_i \in P$, and the node feature of $v_i$ is formulated into $\overrightarrow{v_i} = (x_{p_i}, \overrightarrow{d_{p_i}})$ whose details are as follows:

- **Average $x$-coordinate pin access point $x_{p_i}$ :** This feature is the average value of the $x$-coordinates of all access points in pin $p_i$. In this case, the $x$-coordinate is a relative coordinate based on grids. For example, if the access point is located on the left edge of the grid, the $x$-coordinate is set to 0, and if it is located on the right edge, the $x$-coordinate is set to 1. This feature indicates the $x$-directional location of $p_i$.
- **Pin digit vector $\overrightarrow{d_{p_i}}$ :** The more the number of access points on a pin that span metal 2 tracks is, the better the pin accessibility is. Pin digit vector of pin $p_i$ is a binary vector that indicates if the individual metal 2 tracks cross an access point on $p_i$ or not. For example, the cell in Fig. 6(a) has three pins $p_1$, $p_2$, and $p_3$ in the grid. The yellow dotted lines indicate metal 2 tracks, and there are 7 metal 2 tracks inside the grid, in which $p_3$ does not have an access point on both top and bottom metal 2 tracks, thus, $\overrightarrow{d_{p_3}} = [0\ 1\ 1\ 1\ 1\ 1\ 0]$.

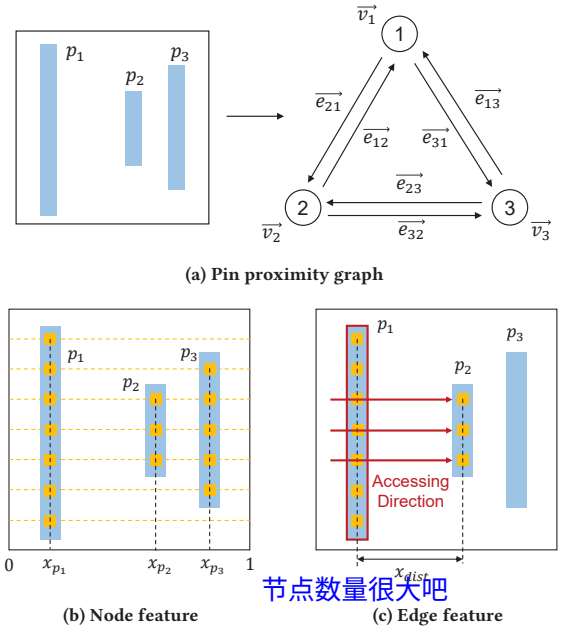



**Figure 6: Overall picture of pin proximity graph.**

Edge $e_{ij} \in E$ is a directed edge from $n_j \in V$ to $n_i \in V$ and indicates a disturbance caused by the existence of $p_j$ when accessing $p_i$. If pins $p_i$ and $p_j$ are located within a certain short distance, which implies the two pins involve a nontrivial amount of interaction, $e_{ij}$ exists. We extract diverse features that affect pin accessibility between two pins, and formulate the features into $\overrightarrow{e_{ij}} = (\overrightarrow{d_{ij}}, x_{dist}, po_{ij}, h_d)$ whose details are as follows:

- **Relative position of $p_j$ with respect to accessing direction ($\overrightarrow{d_{ij}}$) of $p_i$ :** For pin $p_i$, we apply the method presented in [14] to estimate the approaching direction of the nets among four possible directions (left, right, up, down), in which it uses FLUTE [15] and edge shifting [16] techniques for fast wirelength and congestion driven Steiner tree generation. We call the estimated direction *primary* direction, the opposite *last* direction, and the rest *secondary* direction. If $p_j$ locates in the primary direction, it greatly affects the $p_i$ accessibility. $\overrightarrow{d_{ij}}$ is a one-hot vector that represents the location of $p_j$ with respect to $p_i$ and its approaching direction. For example, in Fig. 6(c), FLUTE estimates that $p_2$ is accessed from the left direction. Since $p_1$ is located on the left of $p_2$ (i.e. primary direction), $\overrightarrow{d_{21}}$ = [1 0 0]. Otherwise, $p_3$ is located on the right of $p_2$ (i.e. last direction), $\overrightarrow{d_{23}}$ = [0 0 1].
- **Distance ($x_{dist}$) between $p_i$ and $p_j$ :** The closer two pins are located, the greater the amount of influence on each other is. Especially, end-of-line or spacing rule violation can possibly occur if the two pins are closely located.
- **Overlapped access point track ratio $po_{ij}$ :** This feature refers to the ratio of the number of metal-2 tracks on which both $p_i$ and $p_j$ have access points to the number of metal-2 tracks on which $p_i$ has access points. For example, pin $p_1$ in Fig. 6(b) has access points on 7 metal-2 tracks, and both pin $p_1$ and $p_2$ has access points on 3 metal-2 tracks in the center, from which $po_{12} = 3/7$. This feature represents the amount of horizontal influence of $p_j$ on $p_i$.
- **Horizontal routing congestion on grid $h_d$ :** A high horizontal routing congestion on the grid implies a high difficulty in accessing the pins in the grid due to the influence of nets passing through the grid. Hence, we adopt horizontal net density as a feature of the edge of the pin proximity graph. The detailed calculation of horizontal net density will be described in Sec. 3.3.

With these features, pin proximity graph can efficiently represent the complex interaction among the pins in the grid. Each grid generates an independent pin proximity graph, and there is no external connection between two graphs that correspond to two distinct grids. We transform the graphs of all grids into a three-dimensional pin proximity graph map while preserving location information.

## 3.3 Grid-based Features

The performance of prediction model is greatly affected by the features extracted from the grid. Previous works [1–7] have proposed various kinds of grid-based features. Through intensive experiments with many input features, we chose a set of representative features that significantly affect the model performance. Those features are described below.

- **Pin density:** This feature indicates the ratio of the area occupied by pins to the area of the grid. Since most pins are on metal 1 layer, the density is extracted individually for metal 1 pins and metal 2 pins. A high value of pin density implies an existence of a large number of pins on the grid, which leads to cause a dense routing congestion around the grid.
- **Global/Local net:** Global net is an incoming or outgoing edge that connects a pin inside the grid, and local net connects pins inside the grid. We count the number of these nets and use them as input feature. A large number of those nets on the grid means a high possibility of occurring DRV.
- **Long/Short RUDY:** RUDY [17] is a fast routing congestion estimator from a cell placement result. It assumes that net wire is equally distributed in its bounding box. Given bounding box of net $n$ of size $w \times h$ grid, RUDY of $n$ is defined as the expected number of nets passing through the unit grid inside the bounding box, that is, $RUDY_n = \frac{w+h}{w \times h}$. This value is added to every grid overlapped with the bounding box. We generate two different RUDYs: Long RUDY for nets with a large bounding box and short RUDY for the short nets. If the half parameter of bounding box of a net is longer than 15 grids, it is classified as long net.
- **Horizontal/Vertical routing capacity:** If there are many metal tracks available, routing can be successfully performed without generating DRV even in areas with high routing demand. Since routing is performed in the horizontal and vertical directions, we use the total number of horizontal/vertical metal tracks per grid for each direction as feature.
- **Horizontal/Vertical net density:** Similar to RUDY, it indicates the expected number of nets passing in the horizontal/vertical direction, which was firstly proposed in [18]. Given bounding box of net $n$ of size $w \times h$ grid, horizontal density is $\frac{1}{w}$ and vertical density is $\frac{1}{h}$ for unit grids of each row and column. These values are added to the grid overlapped with the bounding box.

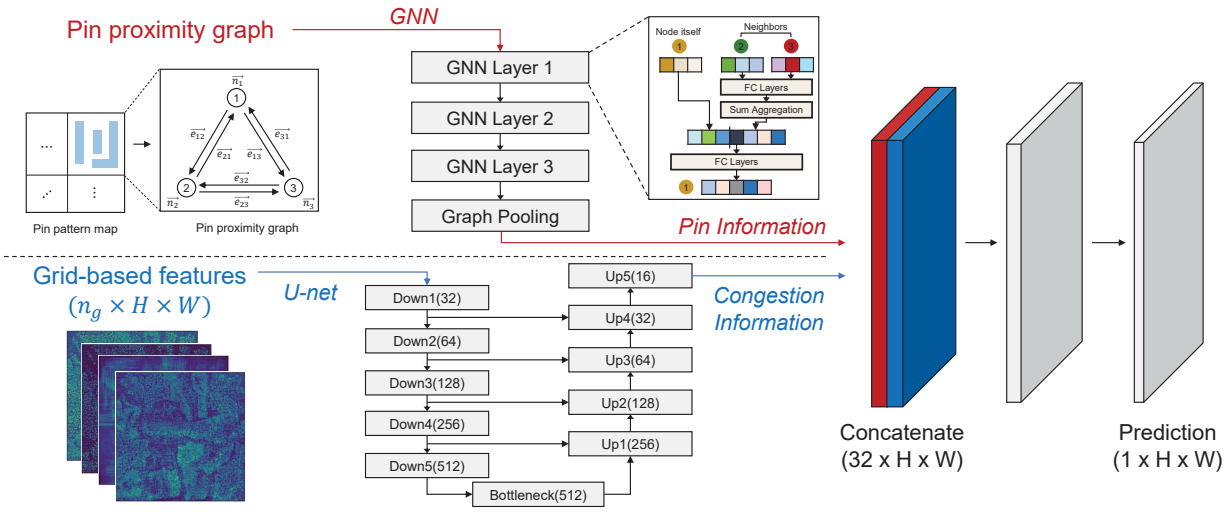



**Figure 7: The overall architecture of our proposed model.**

## 3.4 Overall Architecture of PGNN

Our proposed model called PGNN is composed of three modules: GNN module for modeling pin information inside grid, U-net module for modeling routing congestion, and final prediction module that accepts the outputs of the two modules and produces prediction. Fig. 7 shows our PGNN architecture. The GNN and U-Net modules accommodate different types of input feature. The input feature of GNN is the pin proximity graph, which captures local pin accessibility inside grid. On the other hand, U-net takes a set of grid-based features as input and generates representation of each grid while preserves routing congestion from the global point of view. Then, the outputs of the two modules are concatenated, and the final prediction module produces DRC hotspot prediction map. If the prediction value exceeds a given threshold, we report the corresponding grid to be DRC hotspot.

## 3.5 GNN Architecture in PGNN

Since our pin proximity graph has edge features, conventional GNNs such as GCN [10] and GAT [11] are not able to be directly applied to our graph. Therefore, we develop a novel graph neural network architecture that is highly applicable to our graph.

In our problem, the node features obtained from GNN express the accessibility of the pins representing the corresponding nodes. Our GNN layer iteratively updates the node features by considering the information of the neighboring nodes. Fig. 8 shows the process

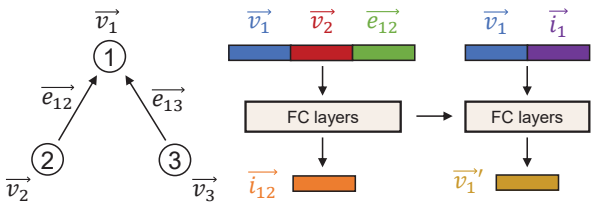



**Figure 8: Overall picture of single GNN layer in PGNN**

of updating the node features of the graph in a single GNN layer. Note that each node $j$ represents pin $p_j$ in the grid. For updating the feature of node $j$ (e.g., $j = 1$ in Fig. 8), GNN layer first calculates $\overrightarrow{i_{jk}}$ which is the disturbance of the neighboring pin $p_k$ when accessing pin $p_j$. Since $\overrightarrow{i_{jk}}$ is affected by edge feature as well as node feature, it can be calculated as follows:

$$\overrightarrow{i_{jk}} = f_\theta(\overrightarrow{v_j} \| \overrightarrow{v_k} \| \overrightarrow{e_{jk}}) \tag{2}$$

where $f_\theta$ is a set of learnable parameters of fully-connected layers. The total disturbance by all neighboring pins is the sum of all individual disturbance $\overrightarrow{i_{jk}}$:

$$\overrightarrow{i_j} = \sum_{k \in n(j)} \overrightarrow{i_{jk}} \tag{3}$$

where $n(j)$ is the set of neighboring nodes of node $j$.
Finally, the node feature $\overrightarrow{v_j}$ is updated through the fully-connected layer after concatenating the original node feature and the total disturbance $\overrightarrow{i_j}$:

$$\overrightarrow{v_j}' = f_\phi(\overrightarrow{v_j} \| \overrightarrow{i_j}) \tag{4}$$

where $f_\phi$ is a set of learnable parameters of fully-connected layers.

For the pin proximity graph of each grid, the node features are obtained by processing three consecutive GNN layers. To process them efficiently, we express the graphs of all grids as a single supergraph, so that parallel processing can be applied. To conduct grid-based prediction, we transform all node features into global graph feature by applying graph average pooling, which simply takes an average of all node features. We then convert this graph feature vector generated for each grid into a three-dimensional matrix to restore spatial information.

## 3.6 U-net Architecture in PGNN

Fig. 9 shows the detailed architecture of U-Net in PGNN. The number in parenthesis in each block of Fig. 9 indicates the number of output kernels. The input of U-net is grid-based features described in Section 3.3 with the size of $10 \times H \times W$ where 10 is the number of

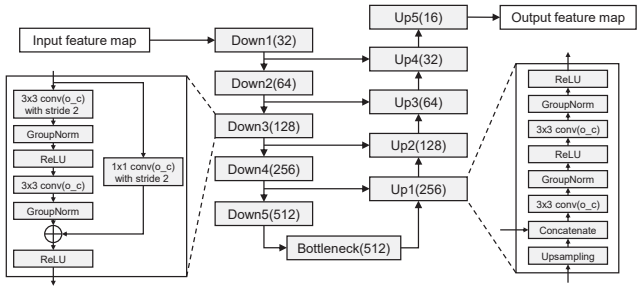



**Figure 9: Block-level details of our U-net architecture.**

grid-based features. For the encoding path of the U-net, we adopt ResNet [19] backbone architecture, which consists of five down-sampling blocks adopting shortcut connection structure. Each block applies a pair of 3x3 convolutional layers to its input. The first convolutional layer uses stride 2 to halve the size of input. Shortcut connection applies 1x1 convolutions with stride 2 to control the output dimension. Therefore, the encoder reduces the input size by 32 times.

Subsequently, the bottleneck block in Fig. 9 updates the output of the encoder by 3x3 convolutions, and then the decoder restores the compressed feature map to the original size through processing five up-sampling blocks. Each up-sampling block doubles the input size by using a transposed convolutional layer, and concatenates the doubled input with the feature map produced by the down-sampling block in the same block level (counting from the bottleneck block) of the up-sampling block, followed by performing 3x3 convolutions. We adopt group normalization [20] and ReLU activation layers in each block. The output size of the decoder is $16 \times H \times W$.

### 3.7 Final Prediction in PGNN

Our final prediction module concatenates the output feature maps of GNN and U-net, resulting in a feature size of $32 \times H \times W$. The module then processes two 3x3 convolutional layers, one with 16 kernels and the other with 1 kernel, to generate a final prediction map of size $H \times W$. Finally, the sigmoid layer is applied to make [0, 1] probability score. If the score exceeds a given threshold, the corresponding grid is classified as DRC hotspot grid.

## 4 EXPERIMENTAL RESULTS

### 4.1 Experimental Setup

We implemented the feature extractor in C++ and PGNN in *Python* with *Pytorch* and *Pytorch Geometric* library [21]. We generated training and test data from 10 Opencore [22] circuits. Table 1 shows the statistics of the benchmarks used in our experiments. We obtained a total of 604 different placement data with different placement parameter settings such as chip utilization, clock period, and number of routing layers. With Nangate 15nm library [23], we repeatedly applied logic synthesis, clock tree synthesis, placement, and routing to collect ground-truth labels. We used *Synopsys Design Compiler* for synthesizing all benchmarks, and used *Synopsys IC Compiler 2* for placement and routing. Experiments were conducted on a linux machine with Intel i7-9700K 3.6GHz CPU, 32GB memory and RTX 2080Ti GPU. The size of unit grid for prediction was $768nm \times 768nm$

**Table 1: Design circuit statistics**

| Designs | #Gates | #Nets | #Grids | #Placements |
|---|---|---|---|---|
| AES_128 | 124,699 | 115,518 | 83,507 | 60 |
| B18 | 38,166 | 33,556 | 21,267 | 60 |
| B19 | 75,145 | 66,499 | 43,687 | 58 |
| ECG | 148,039 | 125,325 | 114,893 | 60 |
| ETH | 45,910 | 46,012 | 56,665 | 43 |
| JPEG | 291,460 | 244,078 | 164,686 | 74 |
| LDPC | 57,956 | 76,921 | 40,549 | 60 |
| NOVA | 208,536 | 152,845 | 121,483 | 72 |
| TATE | 314,515 | 256,753 | 222,333 | 57 |
| VGA_LCD | 118,608 | 76,464 | 115,718 | 60 |

使用不同的参数进行布局的次数

which is equal to the height of standard cell. We trained PGNN by using AdamW optimizer [24] with initial learning rate 0.001 and weight decay 0.01. Cosine annealing scheduler [25] was utilized to control the learning rate. Each model was trained with 150 training iterations.

We used two different schemes to separate training and test set data, as follows.

- **Seen setting:** We randomly divide the total of 604 placement data produced from 10 benchmark circuits into 10 groups. Then, we conduct 10-fold cross-validations on the groups i.e., at each fold, 9 groups are used for training and the remaining 1 group for test. That is, we repeat experiments 10 times, each using a distinct group for test and the rest for training. In this setting, both the training and test sets may include data generated from the same benchmark circuit.
- **Unseen setting:** We use the data from 9 benchmark circuits among 10 for training and the remaining 1 for test. We repeat experiments 10 times, each using the data from a distinct benchmark circuit for test and the rest for training. This setting ensures that the test circuit is completely unseen from the training data.

In both settings, 10% of training set is used as a validation set to determine the classification threshold.

In binary classification problem, prediction results are classified into four groups i.e., true positive (TP), true negative (TN), false positive (FP), and false negative (FN) according to its prediction and ground-truth. TP/TN are the positive/negative samples that predict correctly, and FN/FP are the positive/negative samples that mispredict. Based on the four groups, we use the following five metrics to evaluate the performance of the model in experiments.

- **Accuracy** = $\frac{TP+TN}{TP+FP+FN+TN}$
- **Precision** = $\frac{TP}{TP+FP}$
- **Recall** = $\frac{TP}{TP+FN}$
- **FPR (false positive rate)** = $\frac{FP}{FP+TN}$
- **F1-score** = $2 \times \frac{Precision \times Recall}{Precision+Recall}$

Models with better performance shows higher accuracy, precision, recall, and F1-score with lower FPR. Especially in DRC hotspot prediction problem, since our dataset is highly imbalanced i.e., only 2.4% of grids are DRC hotspot, F1-score is the most reliable metric for evaluating the model performance.

**Table 2: Performance analysis, in terms of F1-score in seen data setting, of our PGNN by varying the usage of input features.**

| Module | Features | Model setting | | | | | | |
|---|---|---|---|---|---|---|---|---|
| GNN | Pin proximity graph | ✓ | | | | | | ✓ |
| U-Net | Pin density | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Global/Local net | | | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Routing capacity | | | | ✓ | ✓ | ✓ | ✓ |
| | Long/Short RUDY | | | | | ✓ | ✓ | ✓ |
| | H/V net density | | | | | | ✓ | ✓ |
| F1-score (%) | | **54.59** | 46.80 | 48.26 | 55.18 | 64.67 | **66.66** | **71.91** |

## 4.2 Analysis on PGNN Performance

Table 2 summarizes the F1-scores of our PGNN produced by varying the usage of input features. The GNN in PGNN is used to model pin accessibility in each local grid, and the U-Net in PGNN is used to model routing congestion considering the information of neighboring grids.

Firstly, We conduct experiments on the effectiveness of grid-based features by selectively including the features to the U-Net. It is seen from Table 2 that pin density, routing capacity, and RUDY feature play a key role on the U-Net performance. In the routing congestion perspective, a high pin density in a grid implies a large number of pins, which indicates a large number of nets are likely to be derived from the grid. Although global/local net feature tells the exact number of nets on the grid, pin density feature seems to sufficiently infer this information. On the other hand, RUDY feature provides a coarse-grained routing congestion estimation, revealing a high leap of F1-score if included to the U-net. For H/V net density feature, though it estimates congestion in horizontal and vertical directions, the calculation method is similar to RUDY. Consequently, it reveals little performance improvement. Finally, for routing capacity feature, it helps to distinguish the situations of DRV variations by the number of routing layers used in the same placement.

In summary, Table 2 shows that GNN and U-Net alone achieve 54.59% and 66.66% F1-score whereas the combined model achieves 71.91%. This means the prior works that have used pin density feature only for describing pin accessibility do not work well. By devising the pin proximity graph, formulating it into GNN, and integrating the GNN with U-Net, we are able to achieve a high F1-score of DRC hotspot prediction.

## 4.3 Comparison with Previous Works

To assess the performance of PGNN, we implement a set of representative DRV prediction models from existing studies. It includes global routing congestion (denoted as GR-Cong) from the representative commercial EDA tool, and representative research results, such as RouteNet [4] and J-Net [7]. GR-Cong is obtained from *ICC2* after global routing stage, and grids with high routing congestion are classified as DRC hotspot. To implement J-Net similar to [7], pin pattern of the grid is represented by 32×32 pixels, and input feature maps are cropped with 80 × 80 size of window. Thus, the size of pin pattern image for input becomes 2560 × 2560. The model architecture and features of RouteNet and J-Net used in our experiment are the same as that in [4] and [7].

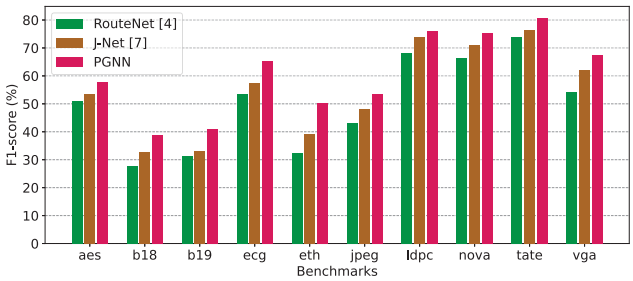



**Figure 10: F1-score comparison on all 10 benchmarks in seen setting.**

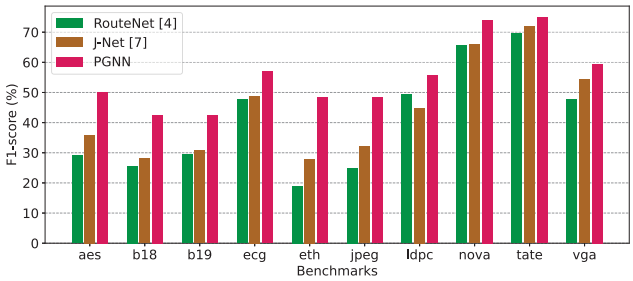



**Figure 11: F1-score comparison on all 10 benchmarks in unseen setting.**

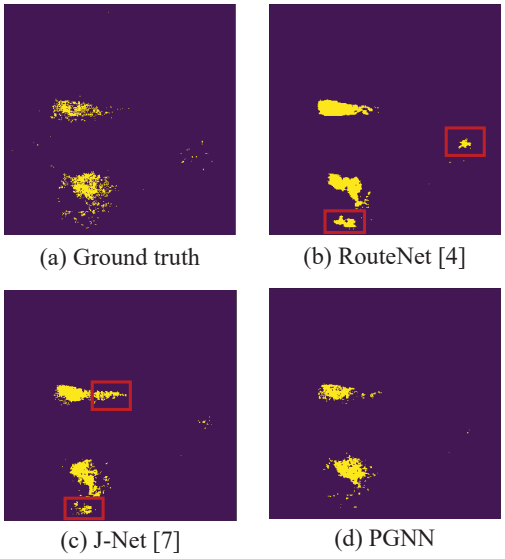
• **Comparing model performance**: Table 3 summarizes the comparison of model performance of PGNN with that of the previous works. PGNN shows superior performance over other models even without using GR congestion as a feature, achieving 7.8%, 12.5% of improvements on F1-score over J-Net in seen and unseen settings, respectively. Figs. 10 and 11 show the comparison of F1-scores of RouteNet, J-Net and PGNN on individual circuits in both settings. It is shown that PGNN outperforms other models on all 10 benchmarks.

One main reason of the inferior performance of RouteNet is that it does not consider pin accessibility since it just uses grid-based features. J-Net expresses pin information as pin pattern image. However, pin pattern image alone is not sufficient to provide detailed information on how each net accesses to its target pins. Furthermore, since pin pattern image requires a massive amount of parameters, J-Net should perform input cropping to mitigate GPU memory overhead for inferencing, losing neighboring information which limits J-Net performance. On the other hand, the pin proximity graph of PGNN can effectively model not only the spatial information of pins but also information on the direction of the pin signals by using much fewer parameters than that of J-Net. In addition, our PGNN integrating GNN architecture abstracting pin proximity graph exhibits a superior performance to J-Net.

Fig. 12 visualizes the DRC hotspot prediction results of the models for circuit ECG. The left-top figure shows ground-truth whose yellow dots indicate DRC hotspot grid. Red boxes in Figs. 12(b) and (c) represent grid region that is misclassified as DRC hotspot grid. Since RouteNet and J-Net do not consider pin accessibility in depth,

**Table 3: Comparison of the performance of DRC hotspot prediction in seen and unseen setting.**

| Model | Use GR? | Seen setting | | | | | Unseen setting | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Accuracy | Precision | Recall | FPR | F1-score | Accuracy | Precision | Recall | FPR | F1-score |
| GR-Cong | Yes | 97.50% | 40.39% | 54.39% | 1.53% | 43.28% | 97.50% | 40.39% | 54.39% | 1.53% | 43.28% |
| RouteNet [4] | Yes | 98.28% | 62.39% | 62.92% | 0.89% | 62.65% | 98.08% | 59.16% | 52.62% | 0.85% | 55.70% |
| J-Net [7] | No | 98.42% | 66.61% | 66.77% | 0.81% | 66.69% | 97.67% | 61.67% | 48.42% | 0.88% | 57.48% |
| PGNN | No | 98.70% | 71.28% | 72.55% | 0.69% | **71.91%** | 98.18% | 58.21% | 72.80% | 1.23% | **64.69%** |



(a) Ground truth (b) RouteNet [4] (c) J-Net [7] (d) PGNN

**Figure 12: Visualization of DRC hotspot prediction. Yellow dots represent DRC hotspot grids.**

they have relatively more misclassified regions over that by our PGNN.

• **Comparing training and inference time**: Model training and inference time is another important factor for model adoption. Table 4 compares the training and inference time for the largest circuit TATE among the benchmarks.

The inference task consists of two stages: feature extraction and model prediction. Since the model prediction stage can be performed quickly by using GPU, the feature extraction stage accounts for the majority of total inference time. Note that since RouteNet uses GR congestion as input feature, the time taken by global routing (8.7 mins) is included in the total inference time of RouteNet. Since RouteNet uses grid-based features only, its feature extraction time is the shortest. Besides grid-based features, J-net uses pin pattern image while PGNN uses pin proximity graph. For J-Net, pin pattern image requires a massive number of parameters and input cropping is essential, demanding expensive feature extraction time. For PGNN, generating pin proximity graph includes the execution of FLUTE [15] to estimate approaching direction and graph construction. (FLUTE is a look-up table based method to provide a fast runtime.) The inference time comparison in Table 4 shows that PGNN achieves 5.5× and 4.6× faster inference time than J-Net and RouteNet, respectively.

**Table 4: Model training and inference time comparison on circuit TATE.**

| Task | RouteNet [4] | J-Net [7] | PGNN |
|---|---|---|---|
| Training Time | 0.7 hours | 31.7 hours | 3.8 hours |
| Global routing | 8.7 mins | N/A | N/A |
| Feature extraction | 29.9s | 640.0s | 119.8s |
| Prediction | 1.9s | 12.5s | 2.9s |
| **Tot. Inference Time** | 9.2 mins | 10.9 mins | **2.0 mins** |

Regarding training time, early convolutional layers for J-Net to process pin pattern image require a huge amount of computation. PGNN can achieve 8.3× faster training time than J-Net by expressing core pin information using graph. Though model training is a one-time-only process, PGNN shows a better scalability over J-Net.

## 5 CONCLUSION

This work addressed the problem of DRC hotspot prediction at the placement stage. In comparison with the conventional ML (machine learning) based models, which invariably revealed ineffectiveness on assembling the aggregate data on (1) pin accessibility and (2) routing congestion, this work proposed a novel ML based DRC hotspot prediction model called PGNN, which was able to accurately capture the combined impact of items 1 and 2 on DRC hotspots by devising a new graph representation so-called *pin proximity graph* and developing a tightly combined ML model of GNN and U-net. In the meantime, through experiments using Nangate 15nm library, it was confirmed that our PGNN consistently outperformed the existing ML models, achieving on average 7.8~12.5% improvements on F1-score while taking 5.5× fast inference time over the existing state-of-the-art technique. In the future, we want to develop a placement optimization methodology which installs PGNN as a core engine.

## ACKNOWLEDGMENTS

This work was supported in part by Samsung Electronics Company, Ltd. under IO201216-08205-01 and FOUNDRY-2020-10DD003F, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) under 2020-R1A4A4079177 and 2021-R1A2C2008864, in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by Korea government (MSIT) under 2021-0-00754, Software Systems for AI Semiconductor Design), and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University. The EDA tool was supported by the IC Design Education.

## REFERENCES


[1] 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 *Proceedings of International Conference on Computer Design (ICCD)*, 2016.

[2] 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 *Proceedings of International Symposium on Physical Design (ISPD)*, 2017.

[3] W.-T. Hung, J.-Y. Huang, Y.-C. Chou, C.-H. Tsai, and M. Chao, "Transforming global routing report into drc violation map with convolutional neural network," in *Proceedings of International Symposium on Physical Design (ISPD)*, 2020.

[4] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "Routenet: Routability prediction for mixed-size designs using convolutional neural network," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[5] T.-C. Yu, S.-Y. Fang, H.-S. Chiu, K.-S. Hu, P. H.-Y. Tai, C. C.-F. Shen, and H. Sheng, "Pin accessibility prediction and optimization with deep learning-based pin pattern recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[6] T.-C. Yu, S.-Y. Fang, H.-S. Chiu, K.-S. Hu, P. H.-Y. Tai, C. C.-F. Shen, and H. Sheng, "Lookahead placement optimization with cell library-based pin accessibility prediction via active learning," in *Proceedings of International Symposium on Physical Design (ISPD)*, 2020.

[7] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu, "Drc hotspot prediction at sub-10nm process nodes using customized convolutional network," in *Proceedings of International Symposium on Physical Design (ISPD)*, 2020.

[8] Y. Zhang, H. Ren, and B. Khailany, "Grannite: Graph neural network inference for transferable power estimation," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2020.

[9] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2020.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, 2015.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical image computing and computer-assisted intervention (MICCAI)*, 2015.

[14] S. Kim and T. Kim, "Pin accessibility-driven placement optimization with accurate and comprehensive prediction model," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.

[15] C. Chu, "Flute: Fast lookup table based wirelength estimation technique," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2004.

[16] M. Pan and C. Chu, "Fastroute: A step to integrate global routing into placement," in *Proceedings of IEEE/ACM international conference on Computer-aided design (ICCAD)*, 2006.

[17] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2007.

[18] J. Chen, J. Kuang, G. Zhao, D. J.-H. H. Huang, and E. F. Young, "Pros: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[20] Y. Wu and K. He, "Group normalization," in *Proceedings of European conference on computer vision (ECCV)*, 2018.

[21] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[22] Oliscience, "Opencores," 1999. [Online]. Available: https://opencores.org

[23] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, "Open cell library in 15nm freepdk technology," in *Proceedings of International Symposium on Physical Design (ISPD)*, 2015.

[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[25] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.