Lay-Net: Grafting Netlist Knowledge on Layout-Based Congestion Prediction

Su Zheng*[1], Lancheng Zou*[2], Peng Xu[1], Siting Liu[1], Bei Yu[1], Martin Wong[1]
[1]Chinese University of Hong Kong [2]Wuhan University

***Abstract*—Congestion modeling is a key point for improving the routability of VLSI placement solutions. The underutilization of netlist information limits the performance of existing layout-based congestion modeling methods. Combining the knowledge from netlist and layout, we graft netlist-based message passing on a layout-based model to achieve better congestion prediction performance. The novel heterogeneous message-passing paradigm better embeds the routing demand into the model by considering both connections between cells and overlaps of nets. With the help of multi-scale features, the proposed model can effectively capture connection information across different ranges, overcoming the problem of insufficient global information in existing models. Based on the advancements, the proposed model achieves significant improvement compared with existing methods.**

I. Introduction

Placement is a crucial and time-consuming stage in the electronic design automation (EDA) flow [1]–[10]. Modeling and optimization of routing congestion in placement can greatly influence the quality of results (QoR) [11]–[13]. To accurately model the congestion, placers commonly integrate routing processes [14]–[17] or analytical models [18]–[21] to estimate the congestion. However, the routing-based methods are plagued by considerable runtime overhead while the model-based approaches suffer from low accuracy.

To avoid the large overhead of invoking global routing while keeping high accuracy, deep-learning-based approaches have been proposed to replace the time-consuming routing engines in congestion modeling. Given the placement features like the rectangular uniform wire density (RUDY) [18], various image-to-image translation models have been utilized to predict the routing congestion, such as fully-convolutional networks (FCN) [22], [23], generative adversarial networks (GAN) [24], and J-Net [25]. Neural architecture search (NAS) allows for the automatic and flexible design of congestion prediction models [26]. LACO [27] proposes a look-ahead mechanism, which can serve as a plugin to mitigate the distribution shift problem in congestion modeling. In these methods, the layout is decomposed into grid cells, each of which is represented by a pixel on an image. Utilizing the information from netlists, graph neural networks (GNN) [28]–[30] are designed to predict the congestion on the circuit cells. In LHNN [31], the grid cells and nets in placement are modeled by hypergraphs, and the novel lattice hypergraph neural network can utilize the connection information to achieve better performance. PGNN [32] employs pin-based GNN to model the routing demand.

The above methods exploit vision models based on geometric features or graph models based on connections to improve congestion prediction. Nevertheless, several common problems exist in previous methods. First, the multimodal fusion of layout and netlist features has not been extensively explored. Existing models cannot effectively aggregate the information given by cell locations and net connectivity. Second, most methods can only utilize local information, ignoring long-range routing demand. Precisely, as illustrated in Fig. 1(a), vision-based models predict congestion by extracting local features with convolutional layers, which lacks a global view of the routing demand. Regarding graph-based methods, the well-known over-smoothing problem of GNN [33] limits the collection of long-range information. Third, existing GNN models overlook the routing demand arising from the overlaps of nets, which is a crucial factor contributing to routing congestion. As shown in Fig. 1(b), even though the long-range connections can be established according to the netlist, the cell-to-cell or cell-to-net links in existing approaches cannot directly model the physical routing demand in GNNs. These limitations call for a novel multimodal congestion prediction model to tackle them.

To conquer the weaknesses of existing models, we not only consider feature pyramids to extract multi-scale information but also design a heterogeneous message-passing paradigm to directly model the routing demand. The proposed model Lay-Net can graft netlist-based knowledge on layout-based model to achieve better congestion prediction performance. Lay-Net combines the strengths of vision-based and graph-based networks while overcoming the limitations of existing models. Fig. 1(c) illustrates the basic principles of Lay-Net. It utilizes multi-scale features to effectively capture both local and global information. The routing demand is explicitly represented by the net-to-net message-passing mechanism. In summary, the major contributions of this paper are as follows:

- We propose a multimodal congestion prediction model that can exploit the geometric features from post-placement layout and the connection information from circuit netlists. The novel network architecture boosts the performance by gathering diverse information that can indicate routing congestion.
- To address the limitation of local information aggrega-

This work is partially supported by AI Chip Center for Emerging Smart Systems (ACCESS) and Research Grants Council of Hong Kong SAR (No. CUHK14208021).

* These authors contributed equally to this paper.

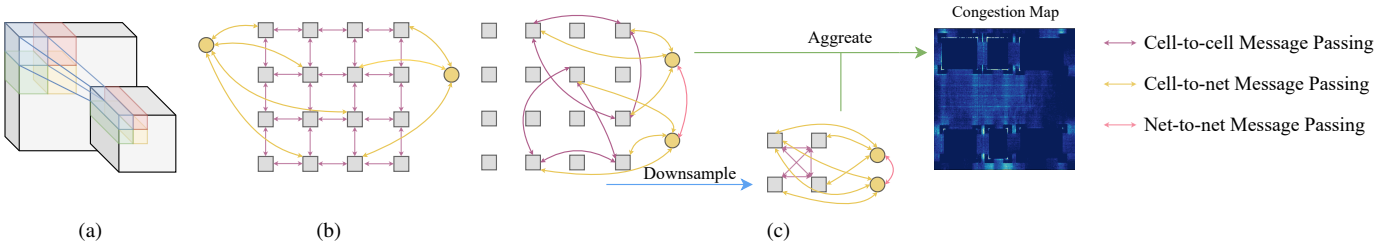

Fig. 1 Comparison between existing methods and Lay-Net. (a) Models based on convolutional layers suffer from the lack of a global view. (b) Lattice graph models can only aggregate local information from neighbors due to the over-smoothing problem. The cell-to-net message passing does not directly model the routing demand. (c) Lay-Net enables global information aggregation by utilizing hierarchical feature maps and explicitly models the routing demand via net-to-net message passing.

tion in existing methods, Lay-Net employs hierarchical feature maps in its vision-based components and enables multi-scale message passing in its graph-based components, enabling it to capture long-range relationships.

- Lay-Net integrates a heterogeneous graph neural network structure that enables cell-to-cell, cell-to-net, and net-to-net message passing. Cell-to-cell and cell-to-net connections can reflect the logical relationships between the circuit components. Net-to-net connections can imply the physical relationships between the nets, which explicitly models the routing demand.
- Extensive experiments verify the effectiveness of the novel Lay-Net, which outperforms the existing congestion prediction models.

The rest of our article is organized as follows. Section II introduces the preliminaries. Section III shows the details of the proposed method. Section IV presents various experimental results that can prove the effectiveness of Lay-Net. The conclusion is shown in Section V.

II. Preliminaries

A. Congestion Modeling for Placement

In a placement algorithm, we commonly represent the circuit with a netlist hypergraph, $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$. $\mathcal{V}$ is the set of circuit cells, including standard cells and macros. The set $\mathcal{E}$ contains the hyperedges that represent the nets in the circuits. Each net $e \in \mathcal{E}$ connects multiple pins, each of which belongs to a cell in $\mathcal{V}$. The global placement (GP) process adjusts the cell locations to optimize the objectives.

In [23], [27], congestion prediction models are integrated into GP by adding a congestion penalty term into the placement objective function. The objective function becomes:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{e\in\mathcal{E}} W_e(\boldsymbol{x},\boldsymbol{y}) + \lambda D(\boldsymbol{x},\boldsymbol{y}) + \eta L(\boldsymbol{x},\boldsymbol{y}), \tag{1}$$

where $W_e(\boldsymbol{x},\boldsymbol{y})$ is the wirelength of net e, λ is the density penalty weight, $D(\boldsymbol{x},\boldsymbol{y})$ is the density penalty function, η is the congestion penalty weight, and $L(\boldsymbol{x},\boldsymbol{y})$ is the routing congestion penalty obtained from the congestion prediction model. The integration of prediction models not only shows the practical application of deep learning in placement but also highlights the importance of congestion modeling.

B. Placement Features for Congestion Prediction

In existing congestion prediction methods, RUDY-based features are usually adopted to model the routing demand [22], [26], [27], [31]. Macro-based features are utilized to distinguish macros from standard cells [22], [23], [27]. RUDY, PinRUDY, and MacroRegion are three representative features [23]. To calculate RUDY, we first get the bounding box of each net, which is formulated by:

$$x_e^h = \max_{p_e} x_{p_e},\ x_e^l = \min_{p_e} x_{p_e},\ y_e^h = \max_{p_e} y_{p_e},\ y_e^l = \min_{p_e} y_{p_e}, \tag{2}$$

where p_e denotes a pin in the net e, whose location is (x_{p_e}, y_{p_e}). Next, the RUDY for net e in the region $x \in [x_e^l, x_e^h], y \in [y_e^l, y_e^h]$ is defined as:

$$\mathbf{RUDY}_e(\boldsymbol{x},\boldsymbol{y}) = \left(\frac{1}{x_e^h - x_e^l} + \frac{1}{y_e^h - y_e^l}\right). \tag{3}$$

We have $\mathbf{RUDY}_e(\boldsymbol{x},\boldsymbol{y}) = 0$ outside the region $[x_e^l, x_e^h] \times [y_e^l, y_e^h]$. Finally, RUDY can be defined as,

$$\mathbf{RUDY}(\boldsymbol{x},\boldsymbol{y}) = \sum_{e\in\mathcal{E}} \mathbf{RUDY}_e(\boldsymbol{x},\boldsymbol{y}). \tag{4}$$

In practice, the RUDY map is divided into $M \times N$ grid cells. The RUDY of a grid cell $b_{k,l}$ is calculated by summing up the RUDY values of the nets that cover it.

PinRUDY is the pin density map inspired by RUDY. To compute the PinRUDY, we need to split the layout into a $M \times N$ grid and estimate the pin density of each grid cell $b_{k,l}$. The PinRUDY of a pin is calculated with,

$$\mathbf{PinRUDY}_{p_e}(k,l) = \left(\frac{1}{x_e^h - x_e^l} + \frac{1}{y_e^h - y_e^l}\right), (x_{p_e}, y_{p_e}) \in b_{k,l}. \tag{5}$$

Finally, the PinRUDY of the grid cell $b_{k,l}$ is defined as,

$$\mathbf{PinRUDY}(k,l) = \sum_{p_e \in b_{k,l}} \mathbf{PinRUDY}_{p_e}(k,l). \tag{6}$$

MacroRegion indicates whether a region is covered by a macro cell or not. For a grid cell $b_{k,l}$, the MacroRegion feature is defined as,

$$\mathbf{MacroRegion}(k,l) = \begin{cases} 1, & \text{if } b_{k,l} \text{ is in a macro cell,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

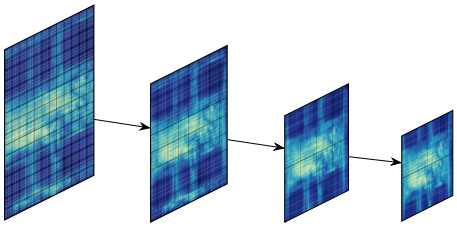

Fig. 2 Multi-scale features from the Swin Transformer backbone. Patch merging mechanism and Swin Transformer block are utilized to get the feature map at a lower scale.

III. Method

The proposed model Lay-Net learns a mapping from the layout-netlist information to the congestion heatmap. It achieves better congestion prediction performance by grafting netlist-based knowledge on a layout-based model. To exploit the layout-based information, Lay-Net maintains multi-scale feature maps so that both short-range and long-range relationships can be utilized. To incorporate netlist-based knowledge, Lay-Net carries out message passing (MP) on the multi-scale feature maps based on heterogeneous GNN models. In this section, we first show the problem formulation of our congestion prediction method, and then describe the details about the multi-scale feature extraction, heterogeneous message passing, neural network architecture, and input features.

A. Problem Formulation

In congestion prediction, we usually divide the circuit layout into $M \times N$ grid cells. Each grid cell is analogized to a pixel in an image $\boldsymbol{X} \in \mathbb{R}^{C \times M \times N}$ that contains C placement features, such as RUDY, PinRUDY, and MacroRegion. The routing overflow $\boldsymbol{Y} \in \mathbb{R}^{2 \times M \times N}$ of the grid cells can be given by a router. The two channels correspond to the horizontal and vertical routing overflow. Thus, image-to-image translation models like FCN and GAN can be employed to learn a mapping $\boldsymbol{f}_I : \mathbb{R}^{C \times M \times N} \mapsto \mathbb{R}^{2 \times M \times N}$ that minimizes:

$$L_I(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{NM} \| \boldsymbol{f}_I(\boldsymbol{X}) - \boldsymbol{Y} \|_2^2. \tag{8}$$

Image-to-image translation models focus on geometric information of the placement results. However, since congestion is induced by excessive routing demand, incorporating connection information into the neural networks is conducive. To model the relationships between grid cells and nets, we design a heterogeneous graph $\mathcal{G}_H =< \mathcal{V}_C, \mathcal{V}_N, \mathcal{E}_{CC}, \mathcal{E}_{CN}, \mathcal{E}_{NN} >$. Each vertex $v_C \in \mathcal{V}_C$ represents a grid cell $X_{i,j} \in \boldsymbol{X}$. Similarly, a vertex $v_N \in \mathcal{V}_N$ corresponds to a net in the netlist. $\mathcal{E}_{CC}$, $\mathcal{E}_{CN}$, and $\mathcal{E}_{NN}$ stand for cell-to-cell, cell-to-net, and net-to-net connections, respectively. Section III-C discusses the connections in details. In this paper, we design a multimodal model that involves netlist and layout information to learn a function $\boldsymbol{f}_H(\mathcal{G}_H, \boldsymbol{X})$ that minimizes:

$$L_H(\mathcal{G}_H, \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{NM} \| \boldsymbol{f}_H(\mathcal{G}_H, \boldsymbol{X}) - \boldsymbol{Y} \|_2^2. \tag{9}$$

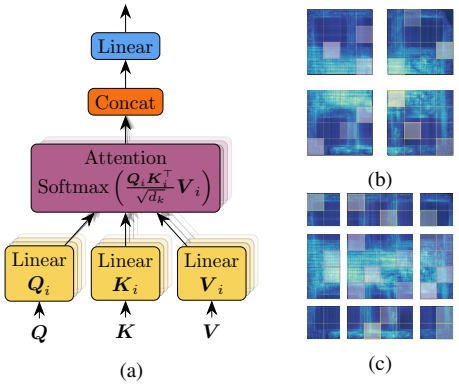

Fig. 3 Illustration of (a) multi-head self-attention layer; (b) self-attention in non-overlapped windows; (c) self-attention in shifted windows. Different opacities indicate that different levels of attention are paid to the patches.

B. Multi-scale Feature Extraction

As shown in Fig. 2, Lay-Net extracts multi-scale features via four stages, which are based on Vision Transformer (ViT) [34] and Swin Transformer [35]. ViT splits an image into fixed-size patches, each of which is regarded as a token in sequential data. The multi-head self-attention mechanism [36] enables ViT to learn the relationships between different parts of the input and output data, regardless of their distance or position. Thus, it can capture global information in the early stages of the model and achieve better accuracy than previous convolutional neural networks (CNN). As illustrated by Fig. 3(a), a multi-head self-attention layer involves the query (Q), key (K), and value (V), which are obtained by linear transformations of the layer's input. After that, Q, K, and V are further projected to multiple heads via linear transformations. Each head is processed by the attention mechanism, which can be formulated by:

$$\text{Attention}(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i) = \text{Softmax}\left(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^\top}{\sqrt{d_k}} \boldsymbol{V}_i\right), \tag{10}$$

where d_k is a normalization factor to avoid abnormal gradients. The heads are finally concatenated and linearly transformed as the output.

Swin Transformer models an image in various scales with patch merging and utilizes local features with self-attention in shifted windows. These advantages help Swin Transformer to outperform the vanilla ViT. Thus, Lay-Net is designed based on Swin Transformer. As illustrated in Fig. 2, Swin Transformer uses four stages to get the multi-scale features. The features at the first stage are obtained by splitting the input into non-overlapping patches, each of which typically has a size of 4×4. A linear embedding layer is applied to these raw-valued features to project them to a specified dimension. Patch merging mechanism and Swin Transformer block are used to downscale the features. A patch merging layer concatenates the features of each group of 2×2 neighboring patches and applies a linear layer on

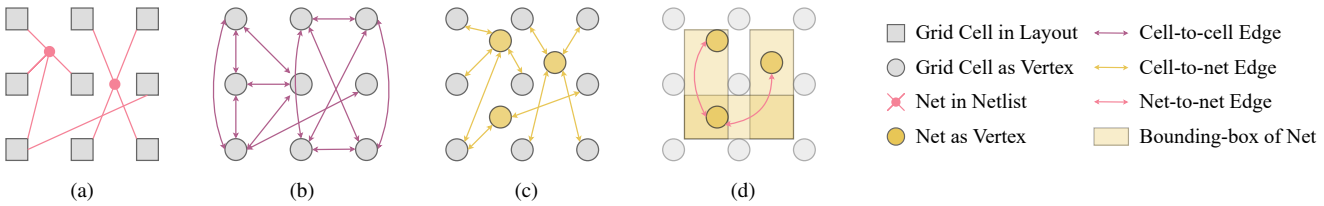

Fig. 4 Illustration of the novel heterogeneous message-passing mechanism. (a) The original grid cells and nets. (b) Cell-to-cell edges, each of which connects a pair of vertices linked by a net. (c) Cell-to-net edges between the nets and the grid cells that they connect. (d) Net-to-net edges, constructed according to the overlaps between net bounding boxes.

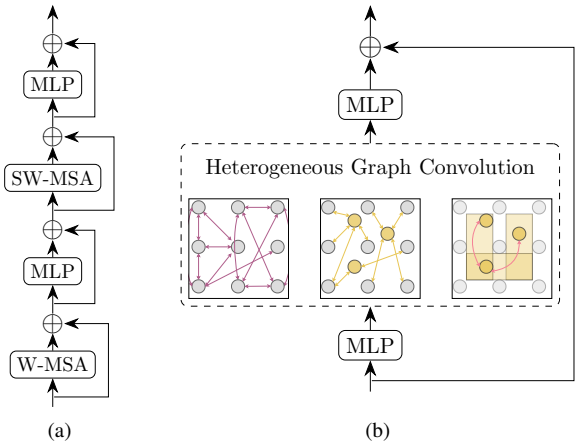

Fig. 5 Detailed structures of (a) Swin Transformer block and (b) heterogeneous GNN block.

the concatenated features. As shown in Fig. 5(a), a Swin Transformer block contains the following layers:

- Multi-head self-attention in non-overlapped windows (W-MSA). As illustrated in Fig. 3(b), the feature patches are grouped by windows. A multi-head self-attention layer is applied within each window.
- Multi-layer linear perceptrons (MLP), consisting of two fully-connected layers.
- Multi-head self-attention in shifted windows (SW-MSA). As shown in Fig. 3(c), W-MSA uses a regular window partitioning strategy that starts from the top-left, while SW-MSA displaces the windows by half of the window size.
- Multi-layer linear perceptrons (MLP).

C. Heterogeneous Message Passing

Graph neural networks (GNN) process graph data via message passing, which iteratively updates the features of vertices or edges by exchanging information with their neighbors. Designing a heterogeneous message-passing mechanism is the key point for handling heterogeneous graphs in GNN. As discussed in Section III-A, we design a heterogeneous graph $\mathcal{G}_H =< \mathcal{V}_C, \mathcal{V}_N, \mathcal{E}_{CC}, \mathcal{E}_{CN}, \mathcal{E}_{NN} >$ to represent the netlist knowledge. The vertex sets $\mathcal{V}_C$ and $\mathcal{V}_N$ correspond to the grid cells and nets, respectively. $\mathcal{E}_{CC}$ contains the edges that connect the grid cells according to the netlist. The edges in $\mathcal{E}_{CN}$ indicate the relationships between grid cells and nets. $\mathcal{E}_{NN}$ is designed to reflect the interplay between different nets. In this section, we describe how to model the routing demand with the edge sets and the heterogeneous message-passing paradigm.

Cell-to-cell Connections. Each vertex in $\mathcal{V}_C$ represents a grid cell on the layout, which may correspond to one or multiple cells in the netlist. Fig. 4(a) and Fig. 4(b) illustrate the construction of $\mathcal{E}_{CC}$. For vertices $v_{C,i}, v_{C,j} \in \mathcal{V}_C$, if a cell in $v_{C,i}$ and a cell in $v_{C,j}$ are connected by a net, we add an edge $(v_{C,i}, v_{C,j})$ to $\mathcal{E}_{CC}$. As a result, $\mathcal{E}_{CC}$ can reflect the logical connections between grid cells. The message passing according to $\mathcal{E}_{CC}$ can exchange the routing demand between grid cells, which is helpful for congestion prediction.

Cell-to-net Connections. If a net $v_{N,k} \in \mathcal{V}_N$ connects grid cells $v_{C,1}, v_{C,2}, \ldots v_{C,l} \in \mathcal{V}_C$, we add the edges $(v_{C,1}, v_{N,k}), (v_{C,2}, v_{N,k}), \ldots, (v_{C,l}, v_{N,k})$ to $\mathcal{E}_{CN}$. As shown in Fig. 4(c), these edges indicate the relationships between grid cells and nets. More importantly, $\mathcal{E}_{CN}$ bridges the gap between cell-to-cell and net-to-net message passing, fusing logical and physical information.

Net-to-net Connections. As presented in Fig. 4(d), if the bounding boxes of two nets $v_{N,i}, v_{N,j} \in \mathcal{V}_N$ are overlapped, we add an edge $(v_{N,i}, v_{N,j})$ to $\mathcal{E}_{NN}$. Placement algorithms commonly optimize the half perimeter wire lengths (HPWL) of nets. This objective implies an assumption that most routing demand of a net lies within its bounding box. Therefore, the overlaps between bounding boxes can indicate the conflicting routing demand from different nets. The net-to-net connections directly model the physical routing demand, distinguishing Lay-Net from existing GNN-based methods that only utilize the logical connections from netlists.

Given the multi-scale features from the layout-based backbone network, we apply a GNN block at each scale to embed the netlist knowledge into the feature maps. We construct a heterogeneous graph $\mathcal{G}_H^{(i)} =< \mathcal{V}_C^{(i)}, \mathcal{V}_N^{(i)}, \mathcal{E}_{CC}^{(i)}, \mathcal{E}_{CN}^{(i)}, \mathcal{E}_{NN}^{(i)} >$ for the ith scale, where a grid cell corresponds to an element on the feature map. The cell-to-cell, cell-to-net, and net-to-net connections can guide the heterogeneous message passing.

Messasge Passing Paradigm. For a grid cell $v \in \mathcal{V}_C^{(i)}$, whose

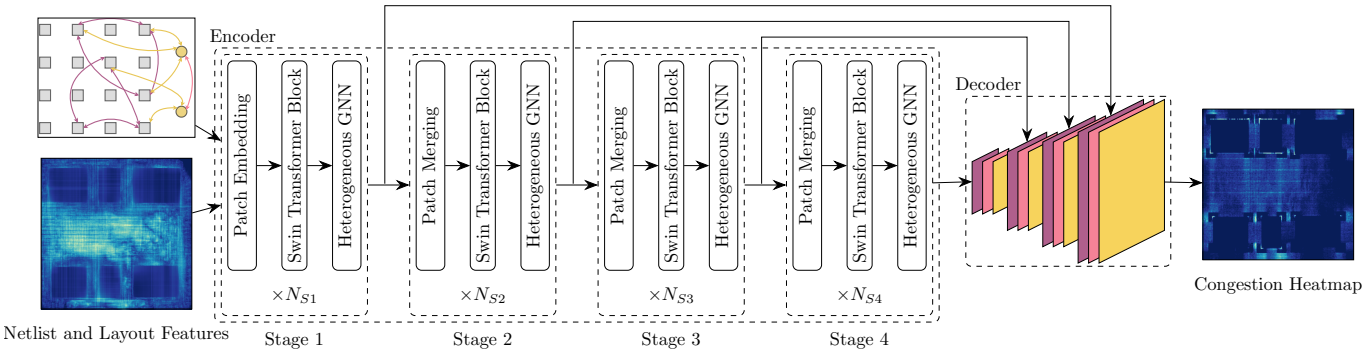

Fig. 6 Overview of Lay-Net, which consists of 4 stages. At each stage, the patch merging layer and Swin Transformer block extract features from the output of the previous stage. The heterogeneous GNN layer conducts message passing on the output of the Swin Transformer block.

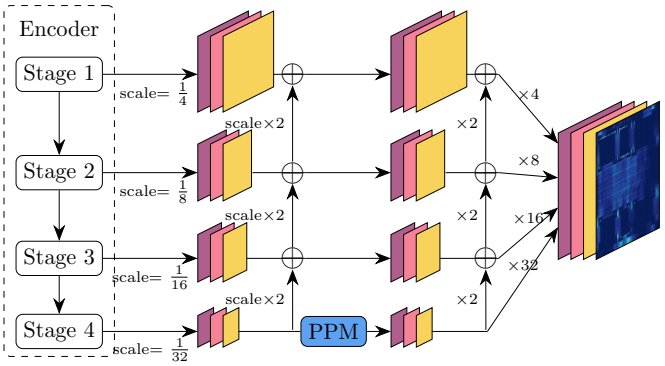

Fig. 7 UPerNet-based decoder, which employs upscaling functions and residual connections to combine the multi-scale features. The quadrangles represent convolutional layers for extracting local features. Pyramid pooling module (PPM) enables the utilization of global contextual information.

feature is $\boldsymbol{h}_v^{(i)}$, we first transform it with MLP:

$$\boldsymbol{h}_v^{(i)'} = f_{C1}^{MLP}(\boldsymbol{h}_v^{(i)}). \tag{11}$$

After that, we apply a heterogeneous graph convolution operation, which can be formulated as:

$$\boldsymbol{h}_v^{(i)''} = \sum_{u \in \mathcal{N}_{CC}(v)} \frac{\mathbf{W}_{CC}}{c_{uv}} \boldsymbol{h}_u^{(i)'} + \sum_{u \in \mathcal{N}_{CN}(v)} \frac{\mathbf{W}_{CN}}{c_{uv}} \boldsymbol{h}_u^{(i)'}, \tag{12}$$

where $\mathcal{N}_{CC}(v)$ and $\mathcal{N}_{CN}(v)$ denote the neighbors of vertex v in cell-to-cell and cell-to-net connections, respectively. The weight matrices $\mathbf{W}_{CC}$ and $\mathbf{W}_{CN}$ are designed for these two types of connections. The normalization factor c_{uv} is calculated according to the vertex degrees, i.e. $c_{uv} = \sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}$. Finally, we obtain the output features of the current scale with the residual MLP defined as:

$$\boldsymbol{h}_v^{(i)'''} = \boldsymbol{h}_v^{(i)} + f_{C2}^{MLP}(\boldsymbol{h}_v^{(i)''}). \tag{13}$$

Note that we omit the ReLU activation functions in these formulas for simplicity. We employ 3-layer MLPs in heterogeneous message passing with the same hidden layer dimension as the preceding Swin Transformer block. The overall structure of a heterogeneous GNN block can be summarized in Fig. 5(b).

Similarly, the heterogeneous message-passing paradigm for a net $v \in \mathcal{V}_N^{(i)}$ can be formulated by:

$$\boldsymbol{h}_v^{(i)'} = f_{N1}^{MLP}(\boldsymbol{h}_v^{(i)}), \tag{14}$$

$$\boldsymbol{h}_v^{(i)''} = \sum_{u \in \mathcal{N}_{CN}(v)} \frac{\mathbf{W}_{NC}}{c_{uv}} \boldsymbol{h}_u^{(i)'} + \sum_{u \in \mathcal{N}_{NN}(v)} \frac{e_{uv}\mathbf{W}_{NN}}{c_{uv}} \boldsymbol{h}_u^{(i)'}, \tag{15}$$

$$\boldsymbol{h}_v^{(i)'''} = \boldsymbol{h}_v^{(i)} + f_{N2}^{MLP}(\boldsymbol{h}_v^{(i)''}), \tag{16}$$

where $\mathbf{W}_{NC}$ and $\mathbf{W}_{NN}$ denote the weight matrices for cell-to-net and net-to-net connections, respectively. $\mathcal{N}_{NN}(v)$ contains the neighbors of vertex v in net-to-net connections. The weight e_{uv} models the routing conflict between nets u and v, which is computed according to the overlapping area between their bounding boxes.

D. Network Architecture

Fig. 6 presents the network architecture of Lay-Net, which consists of four stages. We input the layout features and netlist information into the network. At the first stage, a patch embedding layer partitions the layout features into 4×4 patches and applies a linear transformation to each patch. The Swin Transformer block extracts meaningful information from the patches. After that, the transformed layout features are fed to the heterogeneous GNN block, which carries out message passing on the graph $\mathcal{G}_H^{(1)}$. At each following stage, a patch merging layer concatenates the features of each group of 2×2 neighboring patches. The downscaled features are processed by the Swin Transformer and heterogeneous GNN blocks. Note that $\mathcal{G}_H^{(i)}$ is used in the heterogeneous GNN block at the ith stage. Combining feature pyramids [39] with convolutional neural networks, we employ UPerNet [40] as a decoder to aggregate the multi-scale features and predict the congestion heatmap. As shown in Fig. 7, the UPerNet-based decoder employs upscaling functions and residual connections to combine the features from different stages.

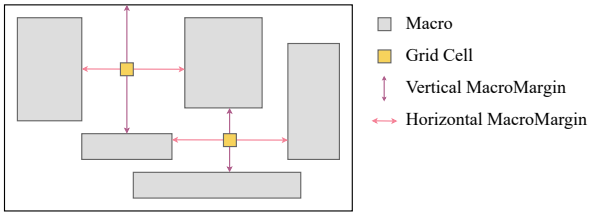

Fig. 8 Illustration of the horizontal/vertical MacroMargin. For a grid cell, MacroMargin measures the distance between its neighboring macros. If a grid cell has no neighboring macro, we use the layout boundary to calculate the distance.

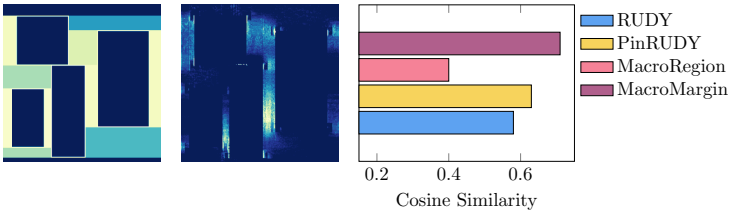

Fig. 9 Illustration of MacroMargin. (a) shows the horizontal MacroMargin of the `mgc_des_perf_a` testcase. (b) is the horizontal congestion heatmap. (c) compares the average cosine similarities between the features and the congestion heatmap on our dataset.

TABLE I Comparison Between Prediction Methods

Characteristic	RUDY-Aware*	Macro-Aware	Routing-Free	Global Info.	Cell-to-cell	Cell-to-net	Net-to-net	Multi-scale Graphs
RouteNet [22]	✓	✓	✗	✗	✗	✗	✗	✗
GAN [24]	✓	✓	✓	✗	✗	✗	✗	✗
NAS [26]	✓	✓	✓	✗	✗	✗	✗	✗
Cross-Graph [29]	✗	✗	✓	✗	✓	✗	✗	✗
LHNN [31]	✓	✗	✓	✗	✓	✓	✗	✗
PGNN [32]	✓	✗	✓	✗	✓	✗	✗	✗
CircuitGNN [30]	✓	✗	✓	✗	✓	✓	✗	✗
Lay-Net	✓	✓	✓	✓	✓	✓	✓	✓

*: Any network that is aware of routability features are considered as RUDY-Aware.

TABLE II Comparison Between Multimodal Fusion Methods

	Fusion Scheme	Characteristics
PGNN [32]	Pin-based GNN inside grid cells	Accurate local modeling, but complex and in lack of global information
LHNN [31], CircuitGNN [30]	Cell-to-cell MP between grid cells	Simple and computationally efficient, but with little global information
TimingPred [37]	Path-finding on grid cells	Precise tracking of signal transmission, less useful for congestion modeling
HybridNet [38]	Topological + geometric GNNs	Little information exchange between modalities
Lay-Net	GNN on multi-scale layout features	Full interaction between modalities, aware of global&local routing demand

Pyramid pooling module (PPM) [41] is utilized to capture global contextual information, while convolutional layers are used to extract local features.

E. Input Features

Lay-Net utilizes the following layout features:

1) RUDY, defined by Equation (4).
2) PinRUDY, defined by Equation (6).
3) MacroRegion, defined by Equation (7).
4) Horizontal/vertical MacroMargin. As shown in Fig. 8, it measures the distance between the margins of two adjacent macros. For a grid cell with no adjacent macro, we use the layout boundary to calculate the distance.

As a result, the input tensor of Lay-Net is $5 \times M \times N$. Note that MacroMargin is the novel feature proposed in this paper. Fig. 9(a) and Fig. 9(b) visualize the MacroMargin and congestion heatmap of the `mgc_des_perf_a` testcase. It can be seen that congestion usually occurs in the region with high MacroMargin values. Fig. 9(c) illustrates how well the features match the congestion heatmap by measuring their average cosine similarities on our dataset. Among the features, MacroMargin has the highest cosine similarity with the ground truth.

In the heterogeneous GNN block, each vertex $v_C \in \mathcal{V}_C^{(i)}$ utilizes the features of the corresponding grid cell output by the Swin Transformer block. For $v_N \in \mathcal{V}_N^{(i)}$, we use the horizontal span, vertical span, and area of a net as the features of the corresponding vertex.

F. Comparison with Previous Models

TABLE I presents the difference between Lay-Net and existing models for routability prediction, including congestion prediction, design rule violation (DRV) prediction, etc. Most methods are RUDY-aware since routability-based features are essential for congestion prediction. Macro-aware feature is important because we observe that congestion frequently appears around macros. Routing-free is also a common feature, which means that a method does not rely on the time-consuming trial global routing process. The multi-scale features enable Lay-Net to aggregate global information without losing local details, distinguishing it from existing methods. Combining cell-to-cell, cell-to-net, and net-to-net message passing, Lay-Net models the routing demand logi-

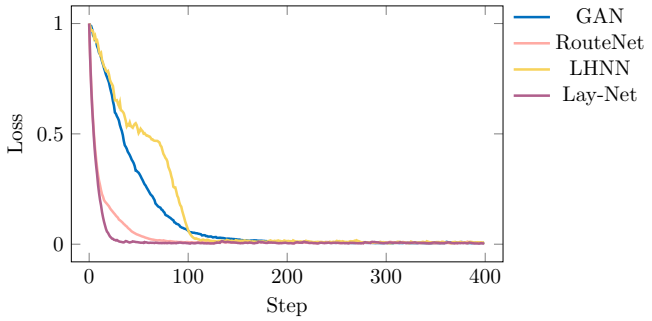

Fig. 10 Loss curves of the models. Lay-Net converges faster than other models.

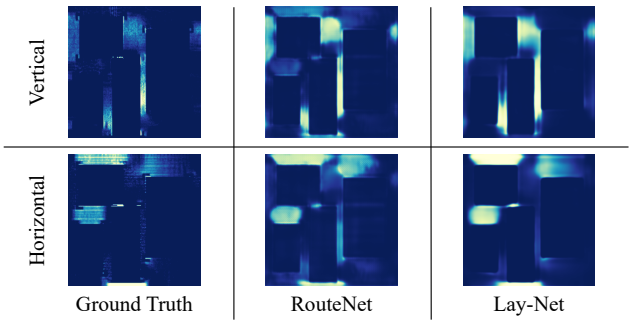

Fig. 11 Visualization of congestion prediction results. RouteNet incorrectly identifies some non-congested areas as congested.

cally and physically, which is one of Lay-Net's advantages. Moreover, the multi-scale heterogeneous GNNs can combine local and global information without over-smoothing. These advancements can boost the performance of Lay-Net.

The combination of layout features and netlist information serves as the multimodal fusion mechanism in congestion prediction. In related fields such as congestion prediction, DRV prediction, and timing prediction, there exist several ways to combine multimodal features. TABLE II compares different multimodal fusion schemes. Multi-scale heterogeneous message passing enables Lay-Net to capture both local and global routing demand. The alternation of layout-based and netlist-based blocks in Lay-Net achieves a full interaction between the modalities. These characteristics make Lay-Net more suitable for congestion prediction.

IV. Experiments

A. Experiment Settings

We implement Lay-Net with DGL [42] and Pytorch. The hardware platform is equipped with Intel Gold 6326 CPU and RTX 3090 GPU. We conduct the experiments on ISPD 2015 benchmark [43]. For each design, we generate 600 placement solutions using Cadence Innovus v17.1 with different parameters. To enrich the training set, we allow Innovus to adjust the macro locations. The congestion ground truths are generated based on the global routing solutions from Innovus.

For the training and test data, we employ a challenging setting. We randomly divide the 20 designs in the dataset into 10 designs as part-A and 10 designs as part-B. To evaluate the performance of our model, we conduct two experiments: exp-AB and exp-BA. In exp-AB, the training and test sets are part-A and part-B, respectively. Conversely, in exp-BA, we use part-B for training and part-A for testing.

B. Comparison with Previous Methods

To compare previous methods with our Lay-Net on congestion prediction, we employ the commonly used metrics, SSIM and NRMS [27], [44]. Structural similarity (SSIM) measures the similarity between two images, which is defined as:

$$\text{SSIM}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{(2\mu_{\boldsymbol{Y}}\mu_{\overline{\boldsymbol{Y}}} + C_1)(2\sigma_{\boldsymbol{Y},\overline{\boldsymbol{Y}}} + C_2)}{(\mu_{\boldsymbol{Y}}^2 + \mu_{\overline{\boldsymbol{Y}}}^2 + C_1)(\sigma_{\boldsymbol{Y}}^2 + \sigma_{\overline{\boldsymbol{Y}}}^2 + C_2)}. \quad (17)$$

Given the ground truth $\boldsymbol{Y}$ and predicted congestion $\overline{\boldsymbol{Y}}$, $\mu_{\boldsymbol{Y}}$ and $\mu_{\overline{\boldsymbol{Y}}}$ are their mean values, $\sigma_{\boldsymbol{Y}}^2$ and $\sigma_{\overline{\boldsymbol{Y}}}^2$ are their variances. The correlation coefficient between the ground truth and predicted result is $\sigma_{\boldsymbol{Y},\overline{\boldsymbol{Y}}}$. C_1 and C_2 are two constants that stabilize the division with a weak denominator.

Normalized root mean square error (NRMS) measures the quality of the predicted image, which can be defined as:

$$\text{NRMS}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{\|\overline{\boldsymbol{Y}} - \boldsymbol{Y}\|_2}{(Y_{\max} - Y_{\min})\sqrt{N_Y}}, \quad (18)$$

where N_Y is a number of grid cells. $Y_{\max}$ and $Y_{\min}$ are the maximum and minimum values of $\boldsymbol{Y}$, respectively.

Note that a larger SSIM is better, while a smaller NRMS is preferred. To get a unified metric, we score the models by:

$$\text{Score}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{\text{SSIM}(\overline{\boldsymbol{Y}}, \boldsymbol{Y})}{\text{NRMS}(\overline{\boldsymbol{Y}}, \boldsymbol{Y})}. \quad (19)$$

In TABLE III, we compare RouteNet [22], GAN [24], LHNN [31], and our model Lay-Net on congestion prediction. According to Section IV-A, we randomly divide the testcases into part-A and part-B. For a testcase in part-A, we show the results of the models trained on part-B, and vice versa. Lay-Net outperforms the models in terms of average SSIM, NRMS, and score. Lay-Net and LHNN, which incorporate netlist-based knowledge, achieve higher SSIM than the rest of the models. However, LHNN lags behind Lay-Net in NRMS by a large margin. This indicates that the multi-scale features and net-to-net connections in Lay-Net contribute to the improvement in NRMS. Compared to LHNN, Lay-Net improves the SSIM, NRMS, and score by 1.0%, 27.5%, and 38.9%, respectively. As shown in Fig. 10, Lay-Net achieves faster convergence than other models, due to the easy propagation of gradients to the earlier stages. Fig. 11 presents the examples of congestion prediction results. Although both RouteNet and Lay-Net can roughly estimate the congested regions, RouteNet incorrectly identifies some non-congested areas as congested, resulting in lower performance than Lay-Net.

TABLE III Comparison Between Lay-Net and Previous Methods on ISPD 2015 Benchmark

Benchmark	#Cells	#Nets	Part	RouteNet [22]			GAN [24]			LHNN [31]			Lay-Net		
				SSIM	NRMS	Score	SSIM	NRMS	Score	SSIM	NRMS	Score	SSIM	NRMS	Score
des_perf_1	113k	113k	B	0.364	0.087	4.183	0.442	0.076	5.815	0.716	0.100	7.159	0.721	0.068	10.60
des_perf_a	109k	110k	A	0.499	0.072	6.930	0.542	0.081	6.691	0.789	0.079	9.987	0.778	0.061	12.75
des_perf_b	113k	113k	A	0.499	0.069	7.231	0.531	0.085	6.247	0.863	0.064	13.48	0.851	0.053	16.05
edit_dist_a	130k	131k	A	0.464	0.091	5.098	0.491	0.109	4.504	0.777	0.089	8.730	0.772	0.068	11.35
fft_1	35k	33k	A	0.432	0.087	4.965	0.482	0.102	4.725	0.753	0.079	9.531	0.755	0.060	12.58
fft_2	35k	33k	A	0.465	0.083	5.602	0.494	0.100	4.939	0.775	0.085	9.117	0.771	0.063	12.23
fft_a	34k	32k	A	0.470	0.105	4.476	0.489	0.114	4.289	0.651	0.113	5.761	0.826	0.094	8.787
fft_b	34k	32k	B	0.337	0.096	3.510	0.494	0.085	5.811	0.814	0.074	11.00	0.801	0.059	13.57
matrix_mult_1	160k	159k	B	0.325	0.091	3.571	0.383	0.088	4.352	0.526	0.112	4.696	0.530	0.092	5.760
matrix_mult_2	160k	159k	B	0.375	0.083	4.518	0.435	0.077	5.649	0.669	0.105	6.371	0.676	0.070	9.657
matrix_mult_a	154k	154k	B	0.391	0.089	4.393	0.451	0.085	5.305	0.599	0.092	6.510	0.603	0.088	6.852
matrix_mult_b	151k	152k	B	0.422	0.092	4.586	0.493	0.081	6.086	0.708	0.173	4.092	0.715	0.070	10.21
matrix_mult_c	151k	152k	B	0.366	0.090	4.066	0.443	0.081	5.469	0.660	0.112	5.892	0.664	0.079	8.405
pci_bridge32_a	30k	30k	B	0.301	0.102	2.950	0.356	0.095	3.747	0.675	0.115	5.869	0.530	0.092	5.760
pci_bridge32_b	29k	29k	A	0.425	0.093	4.569	0.471	0.102	4.617	0.730	0.101	7.227	0.734	0.077	9.532
superblue11_a	954k	936k	B	0.445	0.074	6.013	0.521	0.070	7.442	0.675	0.115	5.869	0.740	0.066	11.21
superblue12	1.3m	1.3m	B	0.323	0.111	2.909	0.392	0.096	4.083	0.638	0.093	6.860	0.641	0.084	7.630
superblue14	634k	620k	A	0.476	0.083	5.734	0.498	0.099	5.030	0.793	0.083	9.554	0.783	0.063	12.42
superblue16_a	698k	697k	A	0.385	0.095	4.052	0.458	0.084	5.452	0.653	0.108	6.046	0.661	0.068	9.720
superblue19	522k	512k	A	0.454	0.116	3.913	0.488	0.105	4.647	0.800	0.078	10.25	0.783	0.064	12.23
Average	-	-	-	0.411	0.090	4.566	0.468	0.091	5.142	0.713	0.099	7.202	**0.717**	**0.072**	**9.958**
Ratio	-	-	-	0.57	1.25	0.46	0.65	1.26	0.52	0.99	1.38	0.72	**1.00**	**1.00**	**1.00**

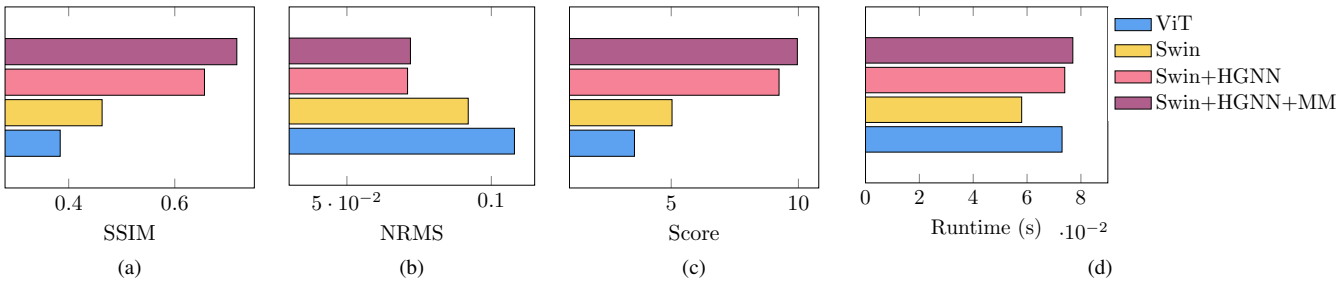

Fig. 12 Comparison between different schemes by (a) SSIM, (b) NRMS, (c) score, and (d) runtime per sample. ViT with a single-scale feature has the worst performance. Swin improves the results with the multi-scale modeling ability. Swin+HGNN incorporates the netlist features to achieve better performance. The heterogeneous GNN block brings significant improvement with some runtime overhead. Swin+HGNN+MM can surpass other models in terms of SSIM and score.

C. Ablation Study

In this section, ablation studies are conducted to demonstrate the effectiveness of the proposed techniques. We compare the following schemes:

1) ViT: It uses single-scale layout features based on RUDY, PinRUDY, and MacroRegion.
2) Swin: It can utilize multi-scale layout features from Swin Transformer.
3) Swin+HGNN: It applies the proposed heterogeneous message-passing mechanism to each stage of Swin Transformer, utilizing the netlist information.
4) Swin+HGNN+MM: In addition to Swin+HGNN, it adds horizontal/vertical MacroMargin to the input features. This scheme is the final implementation of Lay-Net.

The schemes are compared by SSIM, NRMS, score, and runtime per sample in Fig. 12. As shown in Fig. 12(a), Fig. 12(b), and Fig. 12(c), the most significant improvement is brought by the netlist information, which underscores the value of multimodal fusion. The utilization of multi-scale features also contributes to the superiority of Lay-Net. Swin+HGNN+MM further enhances the performance with the proposed MacroMargin feature. According to Fig. 12(d), the runtime of Swin+HGNN+MM is not much larger than the baselines. To conclude, the ablation study validates the benefits of using multi-scale features, netlist information, and horizontal/vertical MacroMargin, which highlights the improvements achieved by Lay-Net.

V. Conclusion

In this paper, we propose Lay-Net, a multimodal neural network for congestion prediction that aggregates both layout and netlist information. The utilization of multi-scale features and the novel heterogeneous message-passing mechanism enable Lay-Net to achieve up to 38.9% improvement over existing methods. The effectiveness of the proposed techniques is further demonstrated by the ablation studies. The superiority of Lay-Net highlights the importance of layout-netlist information fusion and multi-scale feature extraction in congestion prediction.

REFERENCES

[1] B. Hu and M. Marek-Sadowska, "Fine granularity clustering-based placement," *IEEE TCAD*, vol. 23, no. 4, pp. 527–536, april 2004.

[2] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang, "NTUplace: a ratio partitioning based placement algorithm for large-scale mixed-size designs," in *Proc. ISPD*, 2005, pp. 236–238.

[3] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ISPD*, 2005, pp. 185–192.

[4] A. B. Kahng and Q. Wang, "A faster implementation of APlace," in *Proc. ISPD*, 2006, pp. 218–220.

[5] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proc. ASPDAC*, 2007, pp. 135–140.

[6] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: placement based on novel rough legalization and refinement," in *Proc. ICCAD*, 2013, pp. 357–362.

[7] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng *et al.*, "ePlace-MS: Electrostatics-based placement for mixed-size circuits," *IEEE TCAD*, vol. 34, no. 5, pp. 685–698, 2015.

[8] F.-K. Sun and Y.-W. Chang, "BiG: A bivariate gradient-based wirelength model for analytical circuit placement," in *Proc. DAC*, 2019.

[9] H. Szentimrey, A. Al-Hyari, J. Foxcroft, T. Martin, D. Noel, G. Grewal, and S. Areibi, "Machine learning for congestion management and routability prediction within FPGA placement," *ACM TODAES*, vol. 25, no. 5, 2020.

[10] L. Liu, B. Fu, M. D. F. Wong, and E. F. Y. Young, "Xplace: An extremely fast and extensible global placement framework," in *Proc. DAC*, 2022, p. 1309–1314.

[11] T. Taghavi, C. Alpert, A. Huber, Z. Li, G.-J. Nam, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in *Proc. ICCAD*, 2010, pp. 621–624.

[12] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, "NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE TCAD*, vol. 33, no. 12, pp. 1914–1927, 2014.

[13] C. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE TCAD*, vol. 38, no. 9, pp. 1717–1730, 2019.

[14] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," in *Proc. ICCAD*, 2011, pp. 67–73.

[15] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai, and E. F. Y. Young, "Ripple 2.0: High quality routability-driven placement via global router integration," in *Proc. DAC*, 2013, pp. 152:1–152:6.

[16] W.-H. Liu, C.-K. Koh, and Y.-L. Li, "Optimization of placement solutions for routability," in *Proc. DAC*, 2013, pp. 153:1–153:9.

[17] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, "NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *IEEE TCAD*, vol. 37, no. 3, pp. 669–681, 2018.

[18] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. DATE*, 2007.

[19] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller, and S. S. Sapatnekar, "GLARE: Global and local wiring aware routability evaluation," in *Proc. DAC*, 2012, pp. 768–773.

[20] X. He, T. Huang, L. Xiao, H. Tian, and E. F. Y. Young, "Ripple: A robust and effective routability-driven placer," *IEEE TCAD*, vol. 32, no. 10, pp. 1546–1556, 2013.

[21] J.-M. Lin, C.-W. Huang, L.-C. Zane, M.-C. Tsai, C.-L. Lin, and C.-F. Tsai, "Routability-driven global placer target on removing global and local congestion for vlsi designs," in *Proc. ICCAD*, 2021.

[22] 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 *Proc. ICCAD*, 2018, pp. 80:1–80:8.

[23] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, "Global placement with deep learning-enabled explicit routability optimization," in *Proc. DATE*, 2021.

[24] C. Yu and Z. Zhang, "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets," in *Proc. DAC*, 2019.

[25] R. Liang, H. Xiang, J. Jung, J. Hu, and G.-J. Nam, "A stochastic approach to handle non-determinism in deep learning-based design rule violation predictions," in *Proc. ICCAD*, 2022.

[26] C.-C. Chang, J. Pan, T. Zhang, Z. Xie, J. Hu, W. Qi, C.-W. Lin, R. Liang, J. Mitra, E. Fallon, and Y. Chen, "Automatic routability predictor development using neural architecture search," in *Proc. ICCAD*, 2021.

[27] S. Zheng, L. Zou, S. Liu, Y. Lin, B. Yu, and M. D. F. Wong, "Mitigating distribution shift for congestion optimization in global placement," in *Proc. DAC*, 2023.

[28] R. Kirby, S. Godil, R. Roy, and B. Catanzaro, "CongestionNet: Routing congestion prediction using deep graph neural networks," in *Proc. VLSI-SoC*, 2019, pp. 217–222.

[29] A. Ghose, V. Zhang, Y. Zhang, D. Li, W. Liu, and M. Coates, "Generalizable cross-graph embedding for gnn-based congestion prediction," in *Proc. ICCAD*, 2021.

[30] Z. Yang, D. Li, Y. Zhang, Z. Zhang, G. Song, J. Hao *et al.*, "Versatile multi-stage graph neural network for circuit representation," *Proc. NeurIPS*, vol. 35, pp. 20 313–20 324, 2022.

[31] B. Wang, G. Shen, D. Li, J. Hao, W. Liu, Y. Huang, H. Wu, Y. Lin, G. Chen, and P. A. Heng, "LHNN: Lattice hypergraph neural network for VLSI congestion prediction," in *Proc. DAC*, 2022, pp. 1297–1302.

[32] K. Baek, H. Park, S. Kim, K. Choi, and T. Kim, "Pin accessibility and routing congestion aware DRC hotspot prediction using graph neural network and U-Net," in *Proc. ICCAD*, 2022.

[33] C. Yang, R. Wang, S. Yao, S. Liu, and T. Abdelzaher, "Revisiting over-smoothing in deep GCNs," *arXiv preprint arXiv:2003.13663*, 2020.

[34] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. ICLR*.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. CVPR*, 2021, pp. 10 012–10 022.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. NeurIPS*, vol. 30, 2017.

[37] Z. Wang, S. Liu, Y. Pu, S. Chen, T.-Y. Ho, and B. Yu, "Realistic sign-off timing prediction via multimodal fusion," in *Proc. DAC*, 2023.

[38] Y. Zhao, Z. Chai, Y. Lin, R. Wang, and R. Huang, "HybridNet: Dual-branch fusion of geometrical and topological views for VLSI congestion prediction," *arXiv preprint arXiv:2305.05374*, 2023.

[39] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, 2017, pp. 2117–2125.

[40] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proc. ECCV*, 2018, pp. 418–434.

[41] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017, pp. 2881–2890.

[42] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[43] 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. ISPD*, 2015, pp. 157–164.

[44] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. A. Iyer, and D. Z. Pan, "High-definition routing congestion prediction for large-scale FPGAs," in *Proc. ASPDAC*, 2020, pp. 26–31.