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

Lancheng Zou, Su Zheng, Peng Xu, Siting Liu, Bei Yu, *Senior Member, IEEE*, and Martin D. F. Wong, *Life Fellow, IEEE*

***Abstract*—Congestion modeling is crucial for enhancing the routability of VLSI placement solutions. The underutilization of netlist information constrains the efficacy of existing layout-based congestion modeling techniques. We devise a novel approach that grafts netlist-based message passing (MP) into a layout-based model, thereby achieving a better knowledge fusion between layout and netlist to improve congestion prediction performance. The innovative heterogeneous MP paradigm more effectively incorporates routing demand into the model by considering connections between cells, overlaps of nets, and interactions between cells and nets. Leveraging multiscale features, the proposed model effectively captures connection information across various ranges, addressing the issue of inadequate global information present in existing models. Using contrastive learning and mini-Gnet techniques allows the model to learn and represent features more effectively, boosting its capabilities and achieving superior performance. Extensive experiments demonstrate a notable performance enhancement of the proposed model compared to existing methods. Our code is available at: https://github.com/lanchengzou/congPred.**

***Index Terms*—Design automation, representation learning.**

I. Introduction

PLACEMENT is a critical and time-intensive phase in the electronic design automation (EDA) flow [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. Effective modeling and optimization of routing congestion during placement significantly impact the quality of results (QoR) [11], [12], [13]. To achieve accurate congestion modeling, placers often incorporate routing engines [14], [15], [16], [17] or analytical models [18], [19], [20], [21] for congestion estimation. However, routing-based methods tend to incur substantial runtime overhead, while model-based approaches often struggle with accuracy issues.

Deep-learning-based approaches have been developed as alternatives to traditional routing engines and analytical models in congestion modeling. These approaches mitigate the substantial overhead of invoking global routing while maintaining high accuracy. Utilizing placement features like the rectangular uniform wire density (RUDY) [18], various image-to-image translation models have been employed to predict routing congestion, such as fully-convolutional networks (FCNs) [22], [23], generative adversarial networks (GANs) [24], and J-Net [25]. Neural architecture search (NAS) further enhances congestion prediction by enabling the automatic and flexible design of models [26]. LACO [27] introduces a look-ahead mechanism as a plugin to address the distribution shift problem in congestion modeling. In these approaches, the circuit's layout is divided into grid cells, with each cell represented by a pixel on an image. Leveraging netlist information, graph neural networks (GNN) [28], [29], [30] are designed to predict the congestion on the circuit cells. LHNN [31] models grid cells and nets in placement as hypergraphs, using a lattice hypergraph neural network to exploit connection information for improved performance. In addition, PGNN [32] employs pin-based GNN to model routing demand effectively.

The abovementioned methods leverage vision models based on geometric features and graph models based on connections to enhance congestion prediction. However, several common issues persist in these approaches. First, the multimodal fusion of layout and netlist features has not been thoroughly explored. Current models struggle to combine information from cell locations and net connectivity effectively. Second, most methods predominantly utilize local information, overlooking long-range routing demands. Precisely, as illustrated in Fig. 1(a), vision-based models predict congestion by extracting local features through convolutional layers, which lack a global perspective on routing demand. Graph-based methods also face limitations due to the over-smoothing problem of GNNs [33], which limits the aggregation of long-range information. Third, existing GNN models often ignore the routing demand created by net overlaps, a crucial factor in routing congestion. A heterogeneous graph is proposed in LHNN [31] which includes cell-to-cell connections and cell-to-net connections. The cell-to-cell connections reflect the logical relationships among the circuit components, and the cell-to-net connections denote the relationships between nets and cells. Even though long-range connections can be established according to the netlist, the cell-to-cell or cell-to-net connections in current approaches fail to directly model the physical routing demand within GNNs as illustrated in Fig. 1(b). These limitations highlight the need for a novel multimodal congestion prediction model that can address these challenges.

Received 8 July 2024; revised 27 October 2024 and 23 December 2024; accepted 30 December 2024. Date of publication 8 January 2025; date of current version 20 June 2025. This work was supported in part by the AI Chip Center for Emerging Smart Systems (ACCESS), Research Grants Council of Hong Kong, SAR, under Grant CUHK14210723 and Grant CUHK14211824, and in part by the MIND Project under Grant MINDXZ202404. This article was recommended by Associate Editor L. Behjat. *(Lancheng Zou and Su Zheng contributed equally to this work.) (Corresponding author: Bei Yu.)*

Lancheng Zou, Su Zheng, Peng Xu, Siting Liu, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Martin D. F. Wong is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR.

Digital Object Identifier 10.1109/TCAD.2025.3527379

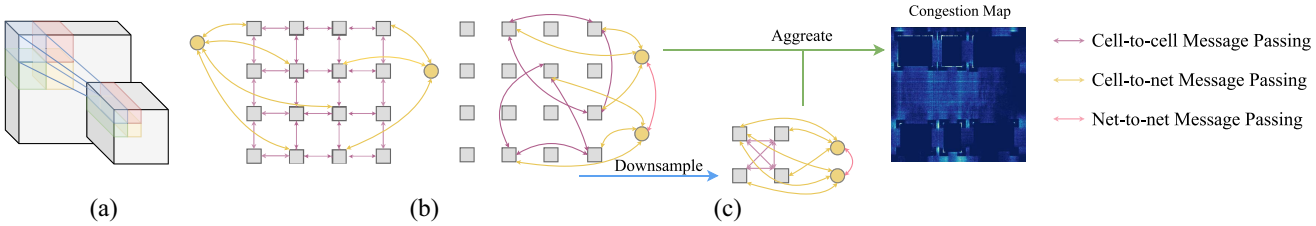

Fig. 1. Comparison between existing methods and Lay-Net. (a) Models based on convolutional layers suffer from the lack of a global view. (b) Lattice graph models can only aggregate local information from neighbors due to the over-smoothing problem. The cell-to-net MP does not directly model the routing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

To address the shortcomings of existing models, we introduce a novel approach that incorporates feature pyramids for extracting multiscale information and designs a heterogeneous message-passing (MP) paradigm to model routing demand directly. Our proposed model, Lay-Net, can graft netlist-based knowledge on a layout-based model to enhance congestion prediction performance. Lay-Net merges the strengths of vision-based and graph-based networks while overcoming the limitations of current models. Fig. 1(c) illustrates the basic principles of Lay-Net. Lay-Net effectively captures both local and global information through the use of multiscale features. To mitigate the issue of information loss during feature extraction, we propose mini-Gnet, which maintains the feature information and makes the features used by Lay-Net more closely associated with the global routing phase. Lay-Net explicitly represents routing demand using a net-to-net MP mechanism combined with cell-to-cell and cell-to-net connections. Relying on the powerful representation learning capability of contrastive learning, Lay-Net can learn better feature embedding to further improve congestion prediction performance. In summary, the major contributions of this article are as follows.

1) We propose a multimodal congestion prediction model to 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.
2) To address the limitation of local information aggregation in existing methods, at the model level, Lay-Net employs hierarchical feature maps in its vision-based components and enables multiscale MP in its graph-based components, enabling it to capture long-range relationships; at the feature level, we propose global routing-aware mini-Gnet to maintain global feature information to reduce feature loss during the feature extraction.
3) Lay-Net integrates a heterogeneous GNN structure that enables cell-to-cell, cell-to-net, and net-to-net MP. 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.
4) To the best of our knowledge, we are the first to leverage the capabilities of contrastive learning for congestion prediction. Contrastive learning is utilized to enhance the power of representation learning of the proposed methods. This is achieved by ensuring that feature embeddings of different congestion levels are pushed further apart while feature embeddings of the same congestion level are getting closer together in the embedding space.
5) Extensive experiments verify the effectiveness of the novel Lay-Net, which outperforms the existing congestion prediction models.

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

II. Preliminaries

A. Congestion Modeling for Placement

In the placement problem, the circuit is typically represented using a netlist hypergraph, denoted as $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Here, $\mathcal{V}$ represents the set of circuit cells, including both standard cells and macros. The set $\mathcal{E}$ consists of hyperedges corresponding to the circuit's nets. Each net $e \in \mathcal{E}$ connects multiple pins, with each pin belonging to a cell in $\mathcal{V}$. The global placement (GP) process then adjusts the locations of the cells to optimize the objectives, such as wirelength and density.

In [23] and [27], congestion prediction models are incorporated into the GP process by introducing a congestion penalty term into the placement objective function. Consequently, the objective function is defined as

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

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 underscores the importance of congestion modeling but also showcases the practical application of deep learning in placement, a significant advancement in the field.

B. Placement Features for Congestion Prediction

In existing congestion prediction methods, RUDY-based features are commonly employed to model routing demand [22],

[26], [27], [31]. In addition, macro-based features are used to differentiate macros from standard cells [22], [23], [27]. Notable features include RUDY, PinRUDY, and MacroRegion, which are used for congestion prediction [23]. To calculate RUDY, the bounding box of each net is first determined, which is formulated by

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

where p_e represents a pin in the net e, located at (x_{p_e}, y_{p_e}). The RUDY for net e within the region $x \in [x_e^l, x_e^h]$ and $y \in [y_e^l, y_e^h]$ is then defined as

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

The value of $\mathbf{RUDY}_e(\boldsymbol{x}, \boldsymbol{y})$ is zero outside the region $[x_e^l, x_e^h] \times [y_e^l, y_e^h]$. Thus, RUDY is defined as

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

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

PinRUDY, inspired by RUDY, represents the pin density map. To compute PinRUDY, the layout is divided into an $M \times N$ grid, and the pin density of each grid cell $b_{k,l}$ is estimated. The PinRUDY value for a pin is calculated as follows:

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

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

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

where p_e denotes the pins covered by the grid cell $b_{k,l}$ and e is the net that p is incident to.

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

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

C. Contrastive Learning

Contrastive learning is typically a self-supervised learning approach that aims to learn valuable representations by distinguishing between similar and dissimilar pairs of data points [34]. The central idea is to bring similar (positive) samples closer together in the feature space while pushing dissimilar (negative) samples further apart, as shown in Fig. 2. Although collecting labeled data are expensive and time-consuming, training with large amounts of labeled data can significantly improve the model's performance. Thus, some researchers proposed supervised contrastive learning by leveraging label information which is applied for diverse domains successfully [35], [36], [37]. It demonstrates its effectiveness in learning robust and discriminative representations.

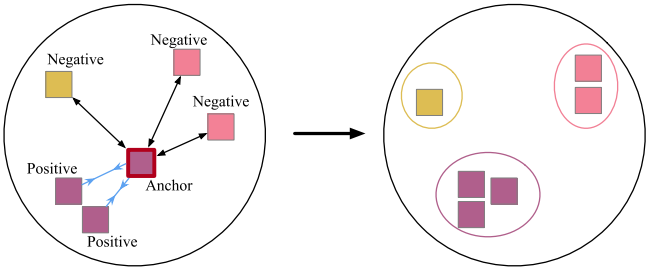

Fig. 2. Illustration of the idea of contrastive learning. Squares with the same color represent the feature embedding with the same class. During the training phase of contrastive learning, the feature embeddings of the same class are pulled together, whereas those of different classes are pushed away in the embedding space.

III. Methods for Congestion Prediction

The proposed model, Lay-Net, learns a mapping from layout-netlist fused information to a congestion heatmap. It enhances congestion prediction performance by grafting netlist-based knowledge into a layout-based model. Lay-Net maintains multiscale feature maps to leverage layout-based information, enabling the utilization of both short-range and long-range relationships. Lay-Net performs MP on the multiscale feature maps using heterogeneous GNN models to incorporate netlist-based knowledge. In this section, we first present the problem formulation for our congestion prediction method. We then detail the multiscale feature extraction, heterogeneous MP, neural network architecture, and input features.

A. Problem Formulation

The circuit layout is typically divided into $M \times N$ grid cells in congestion prediction. Each grid cell is analogous to a pixel in an image $\boldsymbol{X} \in \mathbb{R}^{C \times M \times N}$, where C represents the number of placement features, such as RUDY, PinRUDY, and MacroRegion. The routing overflow $\boldsymbol{Y} \in \mathbb{R}^{2 \times M \times N}$ for the grid cells is provided by a router, with the two channels corresponding to the horizontal and vertical routing overflow.

Consequently, image-to-image translation models like FCNs 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

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

Image-to-image translation models primarily focus on the geometric information of placement results. However, since congestion is induced by excessive routing demand, incorporating connectivity information into neural networks is beneficial. 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 article, we design a multimodal model that can fuse netlist and layout information

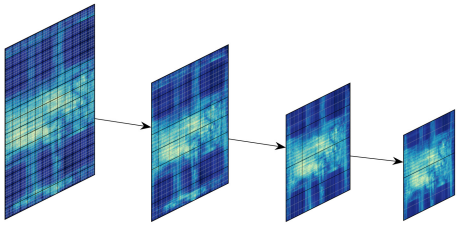

Fig. 3. Multiscale features from the Swin transformer backbone. Patch merging mechanism and Swin Transformer block are utilized to get the feature map at a lower scale.

to learn a function $\boldsymbol{f}_H(\mathcal{G}_H, \boldsymbol{X})$ that minimizes

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

B. Multiscale Feature Extraction

As shown in Fig. 3, Lay-Net extracts multiscale features via four stages, which are based on Vision Transformer (ViT) [38] and Swin Transformer [39]. ViT divides an image into fixed-size patches, treating each patch as a token in sequential data. The multihead self-attention mechanism [40] enables ViT to learn the relationships between different parts of the input and output data, regardless of their distance or position. This capability allows ViT to capture global information in the early stages of the model, achieving better accuracy than previous convolutional neural networks (CNNs). As illustrated in Fig. 5(a), a multihead self-attention layer involves the query (Q), key (K), and value (V), which are derived from linear transformations of the layer's input. These Q, K, and V vectors are then projected to multiple heads via additional linear transformations. Each head is processed by the attention mechanism, which can be formulated as follows:

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

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

The Swin Transformer models an image at various scales using patch merging and captures local features with self-attention in shifted windows. These advantages enable the Swin Transformer to outperform the vanilla ViT. In addition, due to the local attention mechanism, Swin Transformer's computational and memory requirements grow linearly with image size, rather than quadratically. This makes Swin Transformer more efficient than ViT. By using a shifted window, only the patches in the same window will interact with each other. Consequently, Lay-Net is designed based on the Swin Transformer architecture. As illustrated in Fig. 3, Swin Transformer uses four stages to extract the multiscale features. In the first stage, the input is divided into nonoverlapping patches, each typically sized 4×4. A linear embedding layer is then applied to these raw-valued features, projecting them to a specified dimension. To downscale the features, the patch merging mechanism and Swin Transformer blocks are utilized. Specifically, a patch merging layer concatenates the features of each group of 2×2 neighboring patches and applies a linear layer to the concatenated features. As shown in Fig. 4(a), a Swin Transformer block contains the following layers.

1) Multihead self-attention in nonoverlapped windows (W-MSA). As illustrated in Fig. 5(b), the feature patches are grouped by windows. A multihead self-attention layer is applied within each window.
2) Multilayer linear perceptrons (MLP), consisting of two fully-connected layers.
3) Multihead self-attention in shifted windows (SW-MSA). As shown in Fig. 5(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.
4) MLPs.

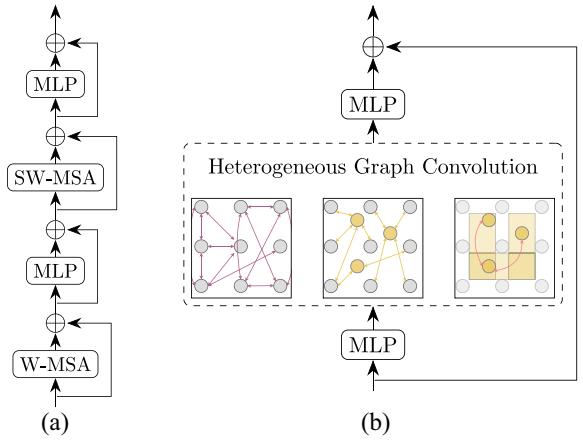

Fig. 4. Detailed structures of (a) Swin transformer block and (b) heterogeneous GNN block.

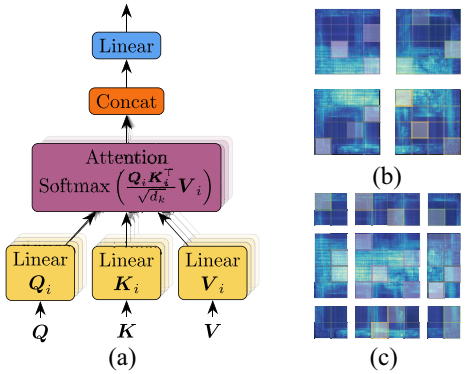

Fig. 5. Illustration of (a) multihead self-attention layer, (b) self-attention in nonoverlapped windows, and (c) self-attention in shifted windows. Different opacities indicate that different levels of attention are paid to the patches.

C. Heterogeneous Message Passing

GNNs process graph data through MP, which iteratively updates the features of vertices or edges by exchanging information with their neighbors. Designing a heterogeneous MP mechanism is crucial for effectively handling heterogeneous graphs in GNNs. 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.

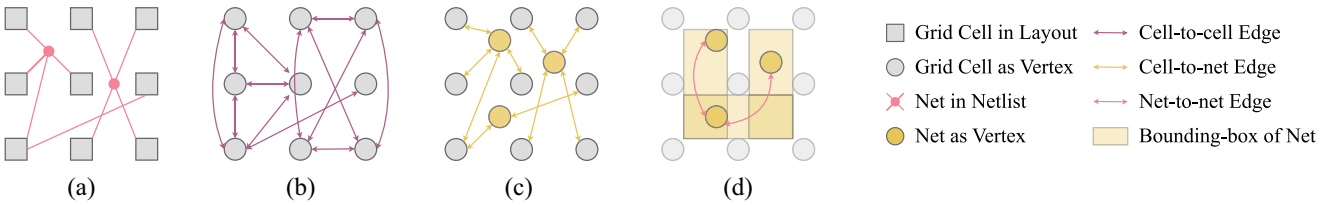

Fig. 6. Illustration of the novel heterogeneous MP mechanism. (a) Original grid cells and nets. (b) Cell-to-cell edges, each of which connects a pair of vertices linked by a net. (c) Cell-to-net edges between the nets and the grid cells that they connect. (d) Net-to-net edges, constructed according to the overlaps between net bounding boxes.

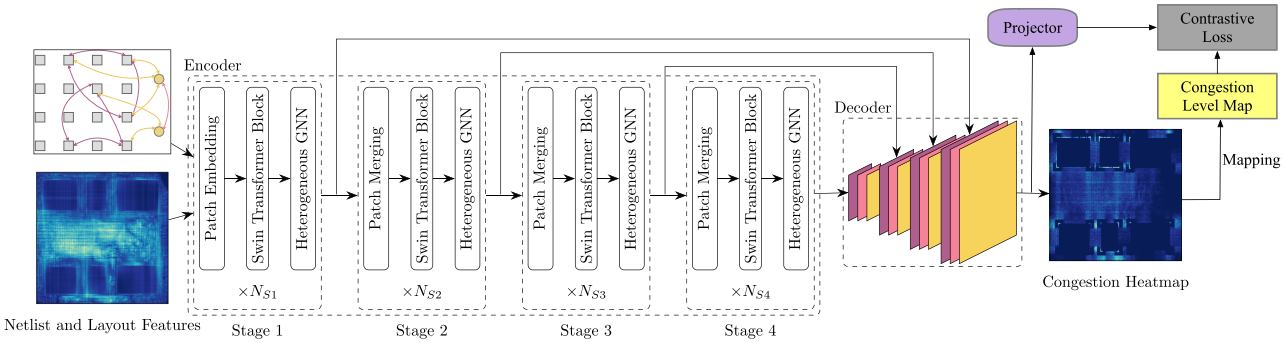

Fig. 7. Overview of Lay-Net, which consists of four stages. At each stage, the patch merging layer and Swin transformer block extract features from the previous stage's output. The heterogeneous GNN layer conducts MP on the output of the Swin transformer block. The Lay-Net's representation learning capability is enhanced by leveraging supervised contrastive learning. It is achieved by the projector after the decoder and the synthetic congestion level map obtained from the original congestion heatmap.

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. This section describes how to model the routing demand using these edge sets and the heterogeneous MP paradigm.

Cell-to-Cell Connections: Each vertex in $\mathcal{V}_C$ represents a grid cell on the layout, which may correspond to one or more cells in the netlist. Fig. 6(a) and (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 $\mathcal{E}_{CC}$. Consequently, $\mathcal{E}_{CC}$ can reflect the logical connections between grid cells. MP along $\mathcal{E}_{CC}$ facilitates the exchange of routing demand information between grid cells, which is beneficial 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 illustrated in Fig. 6(c), these edges represent the relationships between grid cells and nets. More importantly, $\mathcal{E}_{CN}$ bridges the gap between cell-to-cell and net-to-net MP, effectively fusing logical and physical information.

Net-to-Net Connections: As presented in Fig. 6(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 typically optimize the half-perimeter wire lengths (HPWL) of nets as one of their objective functions, which reflects the assumption that most routing demand of a net lies within its bounding box. Therefore, overlaps between bounding boxes can indicate conflicting routing demands from different nets. These net-to-net connections directly model the physical routing demand, distinguishing Lay-Net from existing GNN-based methods that only utilize logical connections from the netlist.

Given the multiscale features from the layout-based backbone network, we apply a GNN block at each scale to fuse the netlist knowledge into the feature maps. For the ith scale, we construct a heterogeneous graph $\mathcal{G}_H^{(i)} = <\mathcal{V}_C^{(i)}, \mathcal{V}_N^{(i)}, \mathcal{E}_{CC}^{(i)}, \mathcal{E}_{CN}^{(i)}, \mathcal{E}_{NN}^{(i)}>$, where each grid cell corresponds to an element on the feature map. At each scale, we construct an independent graph to record the connection information. As Fig. 7 illustrates, there are four stages in total. Thus, four graphs for four different scales are constructed to utilize the mutli-scale features from the Swin Transformer blocks. The heterogeneous graphs keep the relationships among grid cell and grid net. The cell-to-cell, cell-to-net, and net-to-net connections guide the heterogeneous MP, effectively integrating the netlist information into the multiscale feature representations.

MP Paradigm: For a grid cell $v \in \mathcal{V}_C^{(i)}$, whose feature is $\boldsymbol{h}_v^{(i)}$, we first transform it with MLP

$$\boldsymbol{h}_v^{(i)\prime} = f_{C1}^{\text{MLP}}\left(\boldsymbol{h}_v^{(i)}\right). \tag{11}$$

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

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

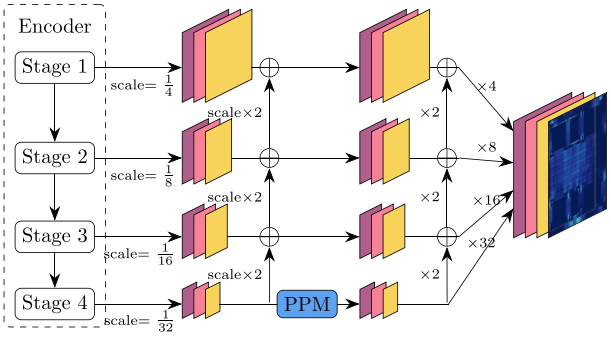

Fig. 8. UPerNet-based decoder, which employs upscaling functions and residual connections to combine the multiscale features. The quadrangles represent convolutional layers for extracting local features. PPM enables the utilization of global contextual information.

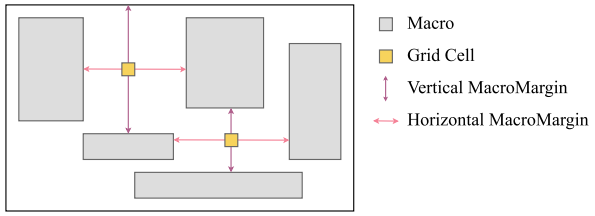

Fig. 9. Illustration of the horizontal/vertical MacroMargin. For a grid cell, MacroMargin measures the distance between its neighboring macros. If a grid cell has no neighboring macro, we use the layout boundary to calculate the distance.

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

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

Note that we omit the ReLU activation functions in these formulas for simplicity. We employ 3-layer MLPs in heterogeneous MP 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. 4(b).

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

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

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

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

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. 7 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 embeds the input features from the patches. After that, the transformed layout features and the inital input netlist features are fed to the heterogeneous GNN block, which carries out MP 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 Swin Transformer and heterogeneous GNN blocks process the downscaled features and embedding of $v_N \in \mathcal{V}_N^{(i-1)}$. Using the heterogeneous GNN, the layout features associated with cells and the netlist features associated with nets are fused and interact. The refined layout and netlist features are then propagated to the next stage. Note that $\mathcal{G}_H^{(i)}$ is used in the heterogeneous GNN block at the ith stage and $\mathcal{V}_N^{(i-1)}$ is the vertexes of grid nets at $i-1$th stage. Combining feature pyramids [41] with CNNs, we employ UPerNet [42] as a decoder to aggregate the multiscale features and predict the congestion heatmap. As shown in Fig. 8, the UPerNet-based decoder employs upscaling functions and residual connections to combine the features from different stages. The feature maps at different stages form the feature pyramid. Pyramid pooling module (PPM) [43] captures global contextual information, while convolutional layers extract local features. For further details on feature pyramids, UPerNet and PPM, we refer the reader to [41], [42], and [43].

E. Input Features

Lay-Net utilizes the following layout features.

1) RUDY, defined by (4).
2) PinRUDY, defined by (6).
3) MacroRegion, defined by (7).
4) Horizontal/vertical MacroMargin. As shown in Fig. 9, 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.

The Horizontal MacroMargin can be formulated as follows:

$$B = \left\{x_i^{\text{left}}, x_i^{\text{right}} | y \in \left[y_i^{\text{bottom}}, y_i^{\text{top}}\right]\right\} \vee \{0, W\} \tag{17}$$

$$x^l = \max_{x_i \in B}\{x_i | x_i < x\} \tag{18}$$

$$x^r = \min_{x_i \in B}\{x_i | x_i > x\} \tag{19}$$

$$\text{Horizontal MacroMargin}(x, y) = x^r - x^l \tag{20}$$

where x and y is the position of the grid cell. B is the set of the left and right boundaries of macros and layout, and y should be in the range of the bottom and right boundaries of the macro. W denotes the rightmost x-coordinate of layout. x^l is the nearest left boundary and x^r is the nearest right boundary. The vertical MacroMargin can be calculated in a simliar way.

As a result, the shape of the input tensor for Lay-Net is $5 \times M \times N$. Note that MacroMargin is the novel feature proposed

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

Method	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 [44]	Path-finding on grid cells	Precise tracking of signal transmission, less useful for congestion modeling
HybridNet [45]	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

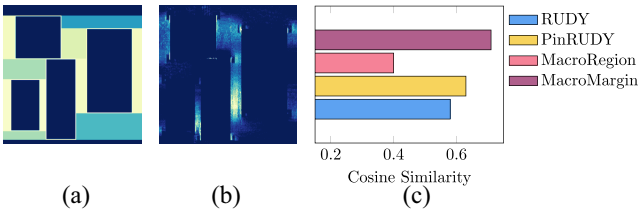

Fig. 10. Illustration of MacroMargin. (a) Shows the horizontal MacroMargin of the `mgc_des_perf_a` testcase. (b) Is the horizontal congestion heatmap. (c) Compares the average cosine similarities between the features and the congestion heatmap on our dataset.

in this article. Fig. 10(a) and (b) visualize the MacroMargin and congestion heatmap of the `mgc_des_perf_a` testcase. It can be seen that congestion usually occurs in regions with high MacroMargin values. Fig. 10(c) illustrates how well the features match the congestion heatmap by measuring their average cosine similarities on our dataset. MacroMargin has the highest cosine similarity with the ground truth among the features.

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 highlights the differences between Lay-Net and existing models for routability prediction, including congestion and design rule violation (DRV) prediction. Most methods are RUDY-aware because routability-based features are crucial for congestion prediction. Macro-aware features are also important, as congestion often occurs around macros. In addition, many methods are routing-free, which does not depend on the time-consuming trial global routing process.

Lay-Net's multiscale features allow it to aggregate global information without losing local details, setting it apart from existing methods. Lay-Net models routing demand logically and physically by combining cell-to-cell, cell-to-net, and net-to-net MP. Furthermore, the multiscale heterogeneous GNNs integrate local and global information without over-smoothing, enhancing Lay-Net's performance. These advancements collectively contribute to Lay-Net's superior capabilities in routability prediction.

Combining layout features and netlist information is a multimodal fusion mechanism for congestion prediction. Various methods exist in related fields, like DRV and timing prediction, which combine multimodal features. Table II compares different multimodal fusion schemes.

Lay-Net's multiscale heterogeneous MP captures both local and global routing demands. The alternation of layout-based and netlist-based blocks ensures full interaction and fusion between different modalities, enhancing Lay-Net's effectiveness in congestion prediction.

IV. EXTENDED METHODS FOR CONGESTION PREDICTION

In this section, we present the mini-Gnet technique, which reduces the loss of long-range information and resolves the issue induced by the large-scale grid net. At the model architecture level, Lay-Net is endowed with the ability to extract global information. Mini-Gnet, in contrast, provides Lay-Net with more comprehensive and less biased global information in terms of the features themselves. We then describe the methodology of supervised contrastive learning we used to improve the feature learning capability of the proposed model. The utilization of contrastive learning facilitates the differentiation of feature embedding across varying congestion levels, thereby enhancing the predictive performance. Compared to our preliminary work [46], the extended methods are proposed to optimize Lay-Net at feature and feature learning level, which complement each other

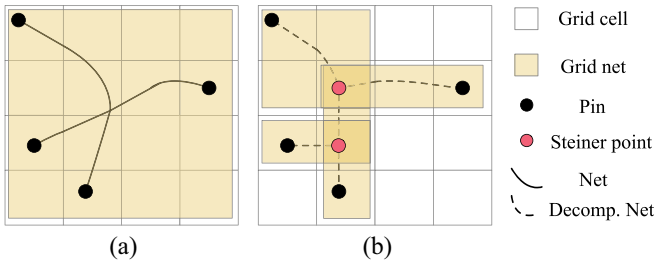

Fig. 11. Illustration of the proposed mini-Gnet. (a) Original Gnet with large scale and (b) multipin net is decomposed into five two-pin nets which form five mini-Gnets, the mini-Gnets perverse the relationship by net-to-net connections.

to further enhance the superiority of Lay-Net in congestion prediction. In addition, we propose a congestion optimization method using the prediction result.

A. Mini-Gnet

Gnet was first introduced in LHNN [31]. Gnet is a set of grid cells that can fully contain the bounding box of the net. In the cell-to-net connections and net-to-net connections, we see the Gnet as a vertex for the net part. For further details about Gnet, the readers can refer to [31]. To prevent the neighbor sampling phase from being dominated by large Gnets, both LHNN and the conference version of Lay-Net [46] excluded all Gnets containing more than 0.25% of the total grid cell count for each circuit. Large Gnets inherently contain a significant amount of long-range information. If they are directly ignored, it can cause much global information loss during feature extraction, leading to potential biases in the model's predictions.

Our proposed method, the mini-Gnet, is designed to minimize the impact of large Gnets on sampling while preserving global information as much as possible. This reassures that the model's predictions will not be biased due to global information loss. In global routing, multipin nets are typically decomposed into a collection of two-pin nets using Steiner tree construction techniques [47]. This decomposition facilitates the application of well-established methods for single-source single-sink shortest path searching. Thus, we utilize multipin net decomposition for mini-Gnet construction.

As shown in Fig. 11, the large Gnet will be decomposed into several smaller-scale Gnets called mini-Gnets. Mini-Gnet reduces the presence of large Gnets, thereby alleviating the problem of their dominance over neighbor sampling. Concurrently, mini-Gnets are still capable of maintaining their original long-range information by establishing connections through overlaps between each other. In the implementation, all Gnets containing more than 0.25% grid cells will be decomposed into mini-Gnets. As an approach inspired by global routing, it enables the model to match the behavior of global routing more closely, thus reducing the bias of congestion prediction.

B. Congestion Prediction With Contrastive Learning

During the training process, another branch after the UPerNet-based decoder uses a convolutional projector to obtain feature embedding for contrastive learning as shown in Fig. 7. In general, contrastive learning does not require any label information. However, supervised contrastive learning leverages label information to enhance the capability of representation learning. Pixel-wise label-based supervised contrastive loss is proposed for semantic segmentation, which results in considerable performance gains [36].

The congestion heatmap prediction task is a regression task, not a classification task which is often handled by contrastive learning. To extend supervised contrastive learning to congestion heatmap prediction, we need to first define the class of each pixel or each grid cell. The congestion heatmap reflects horizontal and vertical routing overflow. These values are distributed between 0 and 1. Different values of overflow implicitly show different congestion levels. For instance, the closer the overflow value is to 1, the higher the congestion level is, and vice versa, the lower the congestion level. The overflow value is therefore divided into several intervals, each of which is assigned to the same class as follows:

$$\boldsymbol{Y}^L = \text{Round}\left(\boldsymbol{Y} \cdot N^L\right) \tag{21}$$

where $\boldsymbol{Y}^L \in \mathbb{R}^{2\times M\times N}$ represents the labels of horizontal and vertical congestion levels. N^L denotes the number of congestion levels set as 5 in our implementation. Round is the rounding function with rounding-to-nearest scheme to obtain the integer label of congestion level.

We then extend supervised contrastive learning for pixel-level congestion heatmap prediction. The contrastive loss is defined as

$$L^{CL} = -\frac{1}{N^S}\sum_{p=1}^{N^S}\frac{1}{N^S_{y_p}}\sum_{q=1}^{N^S}\mathbb{1}_{pq}\log\frac{e_{pq}}{\sum_{k=1}^{N^S} e_{pk}} \tag{22}$$

where N^S is the number of sampled pixels. Significant memory overhead will be introduced if all the pixels are used for contrastive loss computation. Let y_p denotes the class label of pixel p and $N^S_{y_p}$ represents the number of pixels in sampled pixels with class label y_p. f_p is the unit-normalized feature embedding of pixel p, which is the output of the contrastive learning projector. Let $\mathbb{1}_{pq} = \mathbb{1}[y_p = y_q]$ and $e_{pq} = \exp([(f_p \cdot f_q)/\tau])$ where τ is the temperature parameter.

Thus, the training loss function of Lay-Net is formulated as

$$L = L_H + \lambda \cdot L^{CL} \tag{23}$$

where λ is the weight of the supervised contrastive loss.

V. Prediction Guided Placement Refinement

A valuable downstream task of the congestion prediction is to integrate the model to help refine the placement solution [23], [27]. In this article, we propose a method to refine the placement solution using the congestion prediction model. A poor placement solution with congestion will introduce routing detour or even failure which will degrade the performance of routing and result in long turn-around time. Thus, prerouting congestion optimization is critical.

Given a placement solution, we can use the prediction model to estimate the congestion issues in the layout. We add a partial

TABLE III
INFORMATION OF THE EVALUATED BENCHMARK

Benchmark	#Cells	#Nets	Benchmark	#Cells	#Nets
des_perf_1	113k	113k	des_perf_a	109k	110k
des_perf_b	113k	113k	edit_dist_a	130k	131k
fft_1	35k	33k	fft_2	35k	33k
fft_a	34k	32k	fft_b	34k	32k
matrix_mult_1	160k	159k	matrix_mult_2	160k	159k
matrix_mult_a	154k	154k	matrix_mult_b	151k	152k
matrix_mult_c	151k	152k	pci_bridge32_a	30k	30k
pci_bridge32_b	29k	29k	superblue11_a	954k	936k
superblue12	1.3m	1.3m	superblue14	634k	620k
superblue16_a	698k	697k	superblue19	522k	512k

placement blockage in the congested region. The target density can be set as $D_{\text{current}} - D\%$, where D_{current} is the current density in the congested region and D is a hyperparameter to control the extent of cell spreading. In our implementation, D is set as 5. The partial placement blockages are added by the TCL scripts, then we will evoke the incremental placement to spread the cells using Innovus.

VI. EXPERIMENTS

A. Experiment Settings

We implement Lay-Net with DGL [48] and Pytorch. The hardware platform is equipped with Intel Gold 6326 CPU and RTX 3090 GPU. We conduct the experiments on ISPD 2015 benchmark [49] whose information is listed in Table III. We generate 600 placement solutions for each design using Cadence Innovus v17.1 with varying parameters. To enrich the training set, we permit Innovus to adjust macro locations. The congestion ground truths are derived from the global routing solutions provided by Innovus. FLUTE [47] is utilized to generate the rectilinear Steiner minimum tree for each multipin net to obtain mini-Gnets.

To evaluate our method, we conduct two experiments to demonstrate its effectiveness.

1) *Exp1 (Experiment With Seen Designs):* All designs will be included in the training and test sets. The dataset is divided into a training set and a test set in the ratio of 7:3, as usual.
2) *Exp2 (Experiment With Unseen Designs):* It is a much more challenging setting. It should be noted that the design will not be repeated in the training and test sets. For instance, all placement solutions of `mgc_des_perf_a` are included in the training set. Consequently, the test set will have no `mgc_des_perf_a`-related data. We conducted this experiment five times, randomly dividing the 20 designs in the dataset into two parts each time: 10 designs for part-A and 10 for part-B. We first use part-A as the training set and part-B as the test set, then switch, using part-B as the training set and part-A as the test set to evaluate performance. The evaluation results will only be computed when the design is in the test set and the design will not be in the training set at the same time.

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], [50]. Structural similarity (SSIM) measures the similarity between two images, which is defined as

$$\text{SSIM}\left(\overline{\boldsymbol{Y}}, \boldsymbol{Y}\right) = \frac{\left(2\mu_{\boldsymbol{Y}}\mu_{\overline{\boldsymbol{Y}}} + C_1\right)\left(2\sigma_{\boldsymbol{Y},\overline{\boldsymbol{Y}}} + C_2\right)}{\left(\mu_{\boldsymbol{Y}}^2 + \mu_{\overline{\boldsymbol{Y}}}^2 + C_1\right)\left(\sigma_{\boldsymbol{Y}}^2 + \sigma_{\overline{\boldsymbol{Y}}}^2 + C_2\right)}. \quad (24)$$

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

$$\text{NRMS}\left(\overline{\boldsymbol{Y}}, \boldsymbol{Y}\right) = \frac{\|\overline{\boldsymbol{Y}} - \boldsymbol{Y}\|_2}{(Y_{\max} - Y_{\min})\sqrt{N_Y}} \quad (25)$$

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

$$\text{Score}\left(\overline{\boldsymbol{Y}}, \boldsymbol{Y}\right) = \frac{\text{SSIM}\left(\overline{\boldsymbol{Y}}, \boldsymbol{Y}\right)}{\text{NRMS}\left(\overline{\boldsymbol{Y}}, \boldsymbol{Y}\right)}. \quad (26)$$

To measure the prediction performance in the regions where congestion matters most, we define $\text{MSE}_{x\%}$ which is the mean square error of the most critical $x\%$ grid cells between groud truth and the prediction results. In the experiment, we use $\text{MSE}_{2\%}$, $\text{MSE}_{5\%}$ and $\text{MSE}_{10\%}$ to evaluate the performances of different models.

In Table IV, we compare our model Lay-Net with RouteNet [22], GAN [24], LHNN [31], and Lay-Net [46] of our conference version on ISPD 2015 benchmark with seen designs. RouteNet and GAN are layout-only methods, and their experimental results show that utilizing only local-range information is insufficient for congestion prediction. LHNN achieves better results than layout-only methods, indicating the validity of incorporating netlist-based knowledge. Lay-Net can outperform the previous methods regarding SSIM, NRMS, and $\text{MSE}_{2\%}$. It can improve SSIM, NRMS, and $\text{MSE}_{2\%}$ by 6.0%, 8.6%, and 33% compared with LHNN. The average $\text{MSE}_{5\%}$ and $\text{MSE}_{10\%}$ of ours are 0.005 and 0.002, while the average $\text{MSE}_{5\%}$ and $\text{MSE}_{10\%}$ of Lay-Net in conference version are 0.006 and 0.004. Thus, our enhanced version can improve the performance in the most critical regions. It demonstrates that multiscale features and net-to-net connections in Lay-Net help improve the performance of congestion prediction. The comparison with our conference version will be detailed in Section VI-D. It is also observed that the model does not accurately predict the congestion heatmap for all designs, even in the case of seen designs. This discrepancy is attributed to the significant variations in size and congestion distribution among different designs.

In Table V, we compare our model Lay-Net with previous methods on congestion prediction with the setting of unseen designs. According to Section VI-A, we randomly divide the testcases into part-A and part-B. We show the results of

TABLE IV
COMPARISON BETWEEN LAY-NET AND PREVIOUS METHODS ON ISPD 2015 BENCHMARK WITH SEEN DESIGNS

Benchmark	RouteNet [22]			GAN [24]			LHNN [31]			Lay-Net [46]			Ours		
	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$
des_perf_1	0.336	0.094	0.057	0.357	0.086	0.024	0.378	0.083	0.022	0.415	0.083	0.020	0.430	0.079	0.020
des_perf_a	0.698	0.042	0.077	0.787	0.029	0.028	0.791	0.025	0.017	0.905	0.027	0.017	0.903	0.022	0.011
des_perf_b	0.119	0.042	0.000	0.731	0.036	0.000	0.733	0.027	0.000	0.815	0.026	0.000	0.800	0.026	0.000
edit_dist_a	0.653	0.042	0.033	0.726	0.043	0.048	0.742	0.037	0.037	0.749	0.041	0.032	0.768	0.034	0.020
fft_1	0.444	0.065	0.024	0.454	0.063	0.016	0.493	0.063	0.016	0.508	0.058	0.012	0.540	0.055	0.011
fft_2	0.339	0.067	0.011	0.507	0.066	0.010	0.494	0.063	0.012	0.517	0.065	0.010	0.552	0.061	0.008
fft_a	0.735	0.034	0.045	0.893	0.031	0.049	0.899	0.035	0.027	0.912	0.030	0.028	0.914	0.023	0.019
fft_b	0.788	0.043	0.086	0.857	0.037	0.074	0.886	0.030	0.040	0.889	0.039	0.038	0.905	0.026	0.024
matrix_mult_1	0.375	0.074	0.016	0.362	0.078	0.020	0.367	0.076	0.021	0.394	0.075	0.018	0.396	0.074	0.017
matrix_mult_2	0.372	0.081	0.016	0.317	0.087	0.015	0.318	0.086	0.017	0.335	0.085	0.016	0.351	0.084	0.015
matrix_mult_a	0.391	0.023	0.000	0.667	0.019	0.000	0.854	0.015	0.000	0.930	0.016	0.000	0.956	0.014	0.000
matrix_mult_b	0.848	0.012	0.000	0.846	0.011	0.000	0.978	0.015	0.000	0.982	0.012	0.000	0.984	0.008	0.000
matrix_mult_c	0.204	0.060	0.000	0.458	0.034	0.000	0.683	0.021	0.000	0.776	0.014	0.000	0.873	0.014	0.000
pci_bridge32_a	0.738	0.021	0.007	0.747	0.018	0.010	0.916	0.014	0.008	0.971	0.018	0.006	0.965	0.012	0.005
pci_bridge32_b	0.442	0.031	0.012	0.838	0.039	0.014	0.845	0.018	0.011	0.921	0.017	0.008	0.900	0.016	0.007
superblue11_a	0.830	0.015	0.000	0.945	0.016	0.000	0.965	0.014	0.000	0.971	0.016	0.000	0.972	0.013	0.000
superblue12	0.641	0.030	0.014	0.794	0.028	0.015	0.793	0.027	0.012	0.822	0.027	0.012	0.810	0.026	0.010
superblue14	0.693	0.011	0.000	0.700	0.019	0.000	0.968	0.008	0.000	0.978	0.008	0.000	0.972	0.008	0.000
superblue16_a	0.791	0.023	0.021	0.804	0.024	0.025	0.924	0.023	0.047	0.926	0.026	0.028	0.931	0.021	0.021
superblue19	0.755	0.019	0.010	0.880	0.018	0.010	0.893	0.018	0.010	0.897	0.018	0.010	0.900	0.018	0.010
Average	0.560	0.041	0.021	0.683	0.039	0.018	0.746	0.035	0.015	0.781	0.035	0.013	**0.791**	**0.032**	**0.010**
Ratio	0.708	1.281	2.100	0.865	1.219	1.800	0.943	1.094	1.500	0.987	1.094	1.300	**1.000**	**1.000**	**1.000**

TABLE V
COMPARISON BETWEEN LAY-NET AND PREVIOUS METHODS ON ISPD 2015 BENCHMARK WITH UNSEEN DESIGNS

Benchmark	RouteNet [22]			GAN [24]			LHNN [31]			Lay-Net [46]			Ours		
	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$	SSIM	NRMS	$MSE_{2\%}$
des_perf_1	0.170	0.142	0.129	0.114	0.196	0.113	0.188	0.155	0.089	0.220	0.141	0.036	0.226	0.174	0.033
des_perf_a	0.310	0.075	0.253	0.761	0.072	0.254	0.654	0.072	0.220	0.718	0.072	0.239	0.761	0.071	0.242
des_perf_b	0.012	0.103	0.000	0.496	0.032	0.000	0.243	0.093	0.000	0.442	0.109	0.000	0.502	0.038	0.000
edit_dist_a	0.310	0.084	0.164	0.539	0.090	0.164	0.501	0.088	0.121	0.571	0.086	0.151	0.533	0.090	0.147
fft_1	0.318	0.100	0.149	0.218	0.133	0.145	0.353	0.104	0.105	0.387	0.099	0.110	0.333	0.108	0.080
fft_2	0.156	0.087	0.019	0.403	0.108	0.017	0.235	0.086	0.008	0.459	0.083	0.019	0.482	0.101	0.016
fft_a	0.273	0.063	0.107	0.787	0.046	0.108	0.703	0.049	0.089	0.743	0.047	0.073	0.812	0.046	0.075
fft_b	0.368	0.065	0.210	0.632	0.065	0.211	0.802	0.061	0.206	0.830	0.064	0.208	0.791	0.065	0.206
matrix_mult_1	0.151	0.098	0.020	0.208	0.121	0.018	0.236	0.092	0.017	0.228	0.097	0.015	0.235	0.115	0.013
matrix_mult_2	0.143	0.105	0.011	0.190	0.128	0.010	0.220	0.098	0.013	0.213	0.104	0.012	0.223	0.125	0.009
matrix_mult_a	0.042	0.060	0.000	0.309	0.041	0.000	0.616	0.043	0.000	0.627	0.079	0.000	0.550	0.026	0.000
matrix_mult_b	0.307	0.025	0.000	0.727	0.020	0.000	0.850	0.019	0.000	0.935	0.018	0.000	0.899	0.019	0.000
matrix_mult_c	0.018	0.190	0.000	0.255	0.108	0.000	0.378	0.099	0.000	0.601	0.100	0.000	0.637	0.068	0.000
pci_bridge32_a	0.195	0.046	0.016	0.623	0.034	0.016	0.502	0.051	0.015	0.437	0.090	0.015	0.653	0.037	0.013
pci_bridge32_b	0.121	0.220	0.016	0.447	0.058	0.016	0.492	0.039	0.016	0.576	0.039	0.017	0.600	0.032	0.016
superblue11_a	0.301	0.024	0.000	0.709	0.020	0.000	0.825	0.018	0.000	0.914	0.018	0.000	0.857	0.018	0.000
superblue12	0.229	0.035	0.014	0.524	0.032	0.014	0.476	0.039	0.013	0.456	0.041	0.013	0.603	0.031	0.010
superblue14	0.159	0.029	0.000	0.556	0.017	0.000	0.885	0.011	0.000	0.793	0.012	0.000	0.755	0.014	0.000
superblue16_a	0.309	0.036	0.050	0.741	0.032	0.050	0.879	0.032	0.035	0.825	0.032	0.046	0.694	0.033	0.042
superblue19	0.266	0.028	0.012	0.506	0.025	0.010	0.774	0.020	0.006	0.809	0.020	0.009	0.705	0.021	0.007
Average	0.208	0.081	0.058	0.487	0.069	0.057	0.541	0.063	0.048	0.589	0.068	0.048	**0.593**	**0.062**	**0.045**
Ratio	0.351	1.306	1.289	0.821	1.113	1.267	0.912	1.016	1.067	0.993	1.097	1.067	**1.000**	**1.000**	**1.000**

the models trained on part-B for a test case in part-A and vice versa. We conduct this setting five times and obtain the average performance. This setting presents a challenging task for evaluating the generalization of the proposed methods. The aim is to ascertain whether the model is capable of learning the routing overflow inherently. The worse results for the unseen design compared to previous experiments illustrate the difficulty of making congestion predictions for the unseen designs. LHNN and Lay-Net still perform better than other layout-only methods. However, LHNN lags behind Lay-Net in terms of SSIM and NRMS. It can indicate the importance of netlist-layout feature fusion and long-range information for congestion prediction. In this experiment, Lay-Net can surpass LHNN in terms of SSIM, NRMS and $MSE_{2\%}$ by 9.6%, 1.6%, and 6.3%. The average $MSE_{5\%}$ and $MSE_{10\%}$ of ours are 0.021 and 0.010, while the average $MSE_{5\%}$ and $MSE_{10\%}$ of Lay-Net in conference version are 0.023 and 0.012. Thus, our enhanced version can also improve the performance in the most congestion areas, even in the unseen settings. The experimental results also demonstrate that there is still a large gap between seen setting and unseen setting. For a more practical implementation, we need to utilize placement solutions from the same design for training.

Fig. 12 presents examples of congestion prediction results. Although both RouteNet and Lay-Net can estimate the congested regions, RouteNet incorrectly identifies some uncongested areas as congested and tends to be overly pessimistic in predicting congestion, resulting in lower

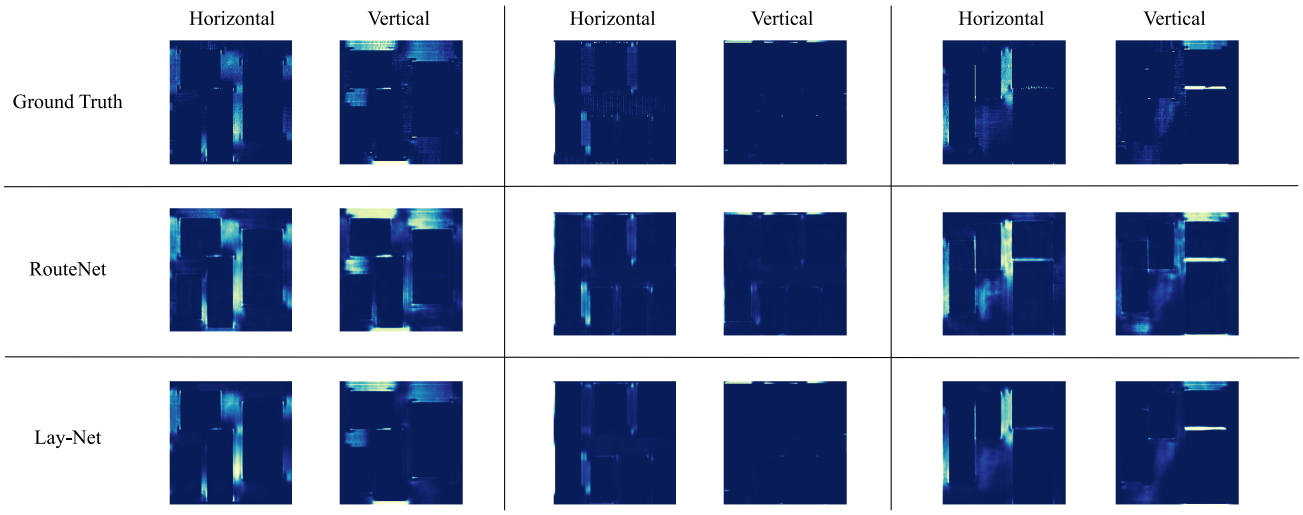

Fig. 12. Visualization of congestion prediction results on some test designs. RouteNet incorrectly identifies some noncongested areas as congested. Lay-Net mitigates the issue of over-pessimism in congestion prediction compared to previous methods.

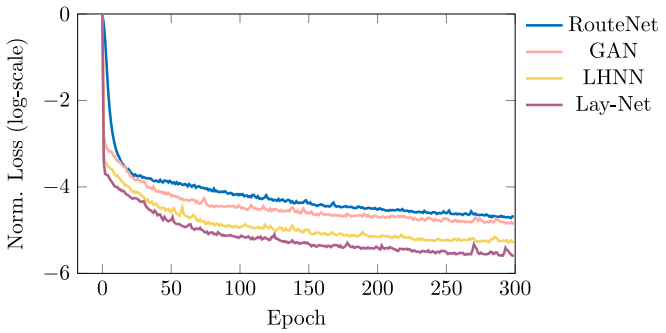

Fig. 13. Normalized loss curves of the models in log-scale in Exp1. Lay-Net achieves faster convergence compared to other models.

performance than Lay-Net. Moreover, as shown in Fig. 13, Lay-Net achieves faster convergence than other models due to the easy propagation of gradients to the earlier stages.

For Exp1, the training time is about 10 h. As for Exp2, the training is nearly 8 h. The model inference time is negligible as shown in Fig. 14(d). However, the time for feature extraction is critical for practical usage. The average time for feature extraction of our proposed model is 32.98 s which includes the time for parsing the DEF/LEF files, feature generation. Thus, the end-to-end runtime for practical use is much faster than evoking the global routing engine of the commercial tool.

C. Experiment of Prediction Guided Placement Refinement

In this experiment, we use RouteNet [22] and Lay-Net to predict the congestion based on 5 superblue designs in the ISPD 2015 benchmark. We integrate the prediction models into the proposed refinement method mentioned in Section V. We employ three metrics: wire length (WL), horizontal and vertial congestion rate (H-CR and V-CR) after global routing to evaluate the quality of the refined placement solution. H-CR and V-CR are used in [23] to measure the level of congestion. H-CR is defined as

$$\text{H-CR} = \frac{\sum_i^H \sum_j^W Lk^H(i,j)}{HW} \tag{27}$$

where Lk^H is the number of demand resources exceeds what is available in the horizontal direction. V-CR is defined in a similar manner.

In Table VI, we compare our placement refinement results with original placement solutions and RouteNet [22] guided placement refinement results. The experiment results show two prediction guided placement refinements can obtain placement solutions with better routability. Our Lay-Net can outperform RouteNet [22] in terms of WL, H-CR and V-CR by 0.2%, 2.9% and 14%. This reflects the effect of accurate prediction by the models. As illustrated in Fig. 12, RouteNet predicts noncongested regions as congested regions. The over-pessimistic prediction will degrade the placement refinement in case that some cells in the noncongested regions will also be spread out. This will result in longer WL and new congested region.

D. Ablation Study

In this section, ablation studies are conducted to demonstrate the effectiveness of the proposed techniques compared with our conference version [46]. We compare the following schemes.

1) *Lay-Net:* This scheme is the conference implementation of Lay-Net, including multiscale layout features from Swin Transformer, heterogeneous MP mechanism, and horizontal/vertical MacroMargin techniques.
2) *Lay-Net+M:* It employs the proposed mini-Gnet method based on Lay-Net.
3) *Lay-Net+C:* It utilized the extension of supervised contrastive learning based on Lay-Net.
4) *Lay-Net+MC:* It adds the mini-Gnet and supervised contrastive learning techniques concurrently, which is our final implementation.

The schemes are compared by SSIM, NRMS, score, and average runtime in Fig. 14. As illustrated in Fig. 14(a)–(c), Lay-Net with mini-Gnet can predict congestion heatmap values more accurately with the help of more global information. With the help of supervised contrastive learning, the feature

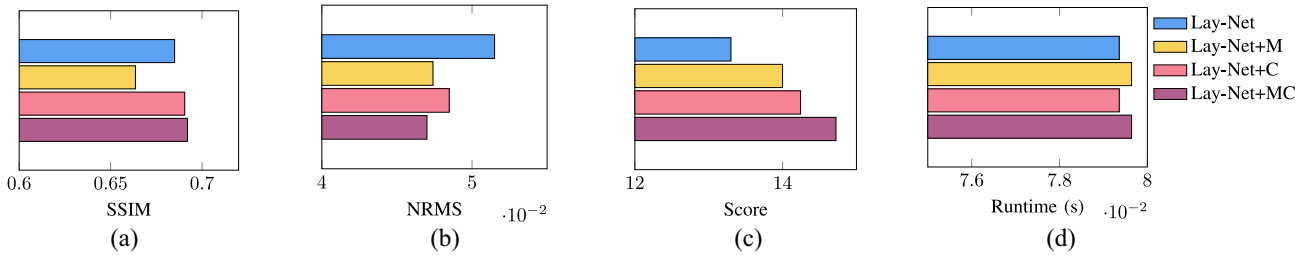

Fig. 14. Comparison between different schemes by (a) SSIM, (b) NRMS, (c) score, and (d) average runtime, where SSIM, NRMS, and score are the average performance on the two experiments.

TABLE VI
EXPERIMENT RESULTS OF PREDICTION GUIDED PLACEMENT REFINEMENT ON SUPERBLUE DESIGNS OF ISPD 2015 BENCHMARK

Benchmark	Origin			RouteNet [22]			Ours		
	WL ($\times 10^6 \mu m$)	H-CR	V-CR	WL ($\times 10^6 \mu m$)	H-CR	V-CR	WL ($\times 10^6 \mu m$)	H-CR	V-CR
superblue11_a	489.08	0.011	0.001	509.07	0.007	0.002	515.22	0.006	0.002
superblue12	520.22	0.150	0.264	497.55	0.086	0.188	488.04	0.087	0.163
superblue14	303.93	0.006	0.005	304.58	0.006	0.004	303.07	0.005	0.003
superblue16_a	357.13	0.060	0.006	358.08	0.046	0.008	358.39	0.042	0.008
superblue19	219.31	0.030	0.049	215.33	0.026	0.048	215.61	0.023	0.038
Average	377.934	0.051	0.065	376.922	0.034	0.050	**376.066**	**0.033**	**0.043**

learning ability can be improved more comprehensively, further improving congestion prediction performance. Moreover, combining mini-Gnet with supervised contrastive learning, better features and more powerful feature learning methods strengthen the original Lay-Net, which improves 1.0%, 8.7% and 10.7% on SSIM, NRMS and score. As for the runtime, leveraging contrastive learning will not introduce extra runtime overhead because the contrastive learning branch will be passed or frozen during the inference. However, using more effective global information will increase the runtime overhead slightly. Although more global information is fed into the model, but the smaller-scale mini-Gnet can reduce some heterogeneous connections to reduce computations.

VII. CONCLUSION

In this article, we propose Lay-Net, a multimodal neural network for congestion prediction that aggregates both layout and netlist information. Multiscale features and mini-Gnet reduce the loss of global information at both the model architecture and feature levels. The novel heterogeneous MP mechanism and supervised contrastive learning not only excel in fusing netlists and layout but also enhance the learning ability of the representations, enabling Lay-Net to achieve up to 9.6% improvement over existing methods. The ablation studies further demonstrate the effectiveness of the proposed techniques. The superiority of Lay-Net highlights the importance of layout-netlist information fusion and multiscale feature extraction in congestion prediction.

REFERENCES

[1] B. Hu and M. Marek-Sadowska, "Fine granularity clustering-based placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 527–536, Apr. 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. ACM Int. Symp. Phys. Design (ISPD)*, 2005, pp. 236–238.

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

[4] A. B. Kahng and Q. Wang, "A faster implementation of APlace," in *Proc. ACM Int. Symp. Phys. Design (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. IEEE/ACM Asia South Pac. Design Autom. Conf. (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. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2013, pp. 357–362.

[7] J. Lu et al., "ePlace-MS: Electrostatics-based placement for mixed-size circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 685–698, May 2015.

[8] F.-K. Sun and Y.-W. Chang, "Big: A bivariate gradient-based wirelength model for analytical circuit placement," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[9] H. Szentimrey et al., "Machine learning for congestion management and routability prediction within FPGA placement," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 5, pp. 1–25, 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. ACM/IEEE Design Autom. Conf. (DAC)*, 2022, pp. 1309–1314.

[11] T. Taghavi, Z. Li, C. Alpert, G.-J. Nam, A. Huber, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2010, pp. 621–624.

[12] M.-K. Hsu et al., "NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1914–1927, Dec. 2014.

[13] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1717–1730, Sep. 2019.

[14] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2011, pp. 67–73.

[15] X. He et al., "Ripple 2.0: High quality routability-driven placement via global router integration," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2013, pp. 1–6.

[16] W.-H. Liu, C.-K. Koh, and Y.-L. Li, "Optimization of placement solutions for routability," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2013, pp. 1–9.

[17] C.-C. Huang et al., "NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 3, pp. 669–681, Mar. 2018.
[18] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. IEEE/ACM Design, Autom. Test Eurpoe (DATE)*, 2007, pp. 1–6.
[19] Y. Wei et al., "GLARE: Global and local wiring aware routability evaluation," in *Proc. ACM/IEEE Design Autom. Conf. (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 Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1546–1556, Oct. 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. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–8.
[22] Z. Xie et al., "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–8.
[23] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, "Global placement with deep learning-enabled explicit routability optimization," in *Proc. IEEE/ACM Design, Autom. Test Europe (DATE)*, 2021, pp. 1821–1824.
[24] C. Yu and Z. Zhang, "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets," in *Proc. ACM/IEEE 56th Design Autom. Conf. (DAC)*, 2019, pp. 1–6.
[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. 41st IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–8.
[26] C.-C. Chang et al., "Automatic routability predictor development using neural architecture search," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–9.
[27] S. Zheng, L. Zou, S. Liu, Y. Lin, B. Yu, and M. Wong, "Mitigating distribution shift for congestion optimization in global placement," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.
[28] R. Kirby, S. Godil, R. Roy, and B. Catanzaro, "CongestionNet: Routing congestion prediction using deep graph neural networks," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (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. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2021, pp. 1–9.
[30] Z. Yang et al., "Versatile multi-stage graph neural network for circuit representation," in *Proc. 36th Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 20313–20324.
[31] B. Wang et al., "LHNN: Lattice hypergraph neural network for VLSI congestion prediction," in *Proc. 59th ACM/IEEE Design Autom. Conf. (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. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–9.
[33] C. Yang, R. Wang, S. Yao, S. Liu, and T. Abdelzaher, "Revisiting over-smoothing in deep GCNs," 2020, *arXiv:2003.13663*.
[34] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 3733–3742.
[35] P. Khosla et al., "Supervised contrastive learning," in *Proc. 34th Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 18661–18673.
[36] X. Zhao et al., "Contrastive learning for label efficient semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 10623–10633.
[37] Z. Pei, X. Yao, W. Zhao, and B. Yu, "Quantization via distillation and contrastive learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17164–17176, Dec. 2024.
[38] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–22.
[39] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 10012–10022.
[40] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1–11.
[41] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2117–2125.
[42] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 418–434.
[43] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2881–2890.
[44] Z. Wang, S. Liu, Y. Pu, S. Chen, T.-Y. Ho, and B. Yu, "Restructure-tolerant timing prediction via multimodal fusion," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.
[45] Y. Zhao, Z. Chai, Y. Lin, R. Wang, and R. Huang, "HybridNet: Dual-branch fusion of geometrical and topological views for VLSI congestion prediction," 2023, *arXiv:2305.05374*.
[46] S. Zheng, L. Zou, P. Xu, S. Liu, B. Yu, and M. Wong, "Lay-net: Grafting netlist knowledge on layout-based congestion prediction," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2023, pp. 1–9.
[47] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 70–83, Jan. 2008.
[48] M. Wang et al., "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2019, *arXiv:1909.01315*.
[49] 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. ACM Int. Symp. Phys. Design (ISPD)*, 2015, pp. 157–164.
[50] 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. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2020, pp. 26–31.

Lancheng Zou received the B.E. degree in electronic information engineering from Wuhan University, Wuhan, China, in 2023. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning for electronic design automation, efficient deep neural network, and hardware/software co-design.

Su Zheng received the B.Eng. and M.S. degrees from Fudan University, Shanghai, China, in 2019 and 2022, respectively. He is currently pursing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. Bei Yu and Prof. Martin D. F. Wong.

His research interest is to solve critical problems in electronic design automation with advanced artificial intelligence methods.

Peng Xu received the B.S. degree from Central South University, Changsha, China, and the M.S. degree from Harbin Institute of Technology (Shenzhen), Shenzhen, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. Bei Yu from Fall 2022.

His research interests include machine learning for analog physical design and optimization in EDA problems.

Siting Liu received the B.S. degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2020. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Her research interests include physical synthesis, machine learning application, and GPU acceleration in VLSI CAD algorithms.

Ms. Liu is a recipient of the Best Paper Award from DATE 2022 and the Best Paper Award Nomination from DATE 2021.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.

Dr. Yu received Eleven Best Paper Awards from ICCAD 2013, 2021, and 2024, IEEE TSM 2022, DATE 2022, ASPDAC 2012 and 2021, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, the IEEE CEDA Ernest S. Kuh Early Career Award in 2021, the DAC Under-40 Innovator Award in 2024, and the Hong Kong RGC Research Fellowship Scheme Award in 2024.

Martin D. F. Wong (Life Fellow, IEEE) received the B.Sc. degree in math from the University of Toronto, Toronto, ON, Canada, the M.S. degree in math and the Ph.D. degree in CS from the University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA.

He was a Bruton Centennial Professor of CS with The University of Texas at Austin, Austin, TX, USA, and a Edward C. Jordan Professor of ECE with UIUC. From August 2012 to December 2018, he was a Executive Associate Dean of the College of Engineering with UIUC. From January 2019 to August 2023, he was the Dean of Engineering and Choh-Ming Li Professor of Computer Science and Engineering with The Chinese University of Hong Kong, Hong Kong. Since August 2023, he joined Hong Kong Baptist University, Hong Kong, as the Provost and a Chair Professor of Computer Science. He has published around 500 papers and graduated over 50 Ph.D. students in electronic design automation (EDA). His main research interest is in EDA.

Dr. Wong is a Fellow of ACM.