AiDRC: Accelerating Detailed Routing by AI-Driven Design Rule Violation Prediction and Checking

YIFAN LI*, Pengcheng Laboratory, China
RUIZHI LIU*, SKLP, Institute of Computing Technology, Chinese Academy of Sciences, China
ZHISHENG ZENG, Pengcheng Laboratory, China
ZENGRONG HUANG, Pengcheng Laboratory, China
ZHIPENG HUANG, Beijing Institute of Open Source Chip, China
DONGBO BU, SKLP, Institute of Computing Technology, Chinese Academy of Sciences, China
XINGQUAN LI†, Pengcheng Laboratory, China

Design rule violation (DRV) evaluation and optimization constitute a critical challenge in modern VLSI physical design. Fast and accurate assessment of routability and DRV have gained significant research attention due to its pivotal role in improving design closure efficiency. Traditional detailed routing and design rule checking (DRC) are computationally expensive. To address the above challenges, this work leverages AI models for the specific location prediction of DRVs at the pre-detailed routing and the checking of DRVs during detailed routing, which achieves fast and precise DRV evaluation. By integrating the crisscross attention mechanism and channel transformer within a ResNet backbone, our proposed DRV prediction and checking model gains the capability to learn non-local and cross-layer relationships across different features and mitigates the negative impact of data imbalance. Experiments demonstrate the effectiveness of our prediction model and checking model, with area under curve (AUC) of 0.987 and F1-score of 0.934, respectively. Remarkably, being integrated into an open-source detailed routing tool, our DRV prediction model achieves 16× acceleration, while our DRV checking model achieves a 293× acceleration over the conventional DRC engine. By applying the predicted DRVs to the detailed routing tool, our framework eliminates an average of 44% DRVs of the initial detailed routing results, effectively bridging the gap between data-driven predictions and rule-based detailed routing workflows.

Additional Key Words and Phrases: Design rule violation, accelerating detailed routing, routability evaluation, neural network, prediction and checking

*Yifan Li and Ruizhi Liu contribute equally and are co-first authors.
†Corresponding author.

This work is supported in part by the Major Key Project of PCL (No. PCL2025AS04, PCL2025AS05) and the NSF of Fujian Province under Grants (No. 2024J09045).
Authors' addresses: Yifan Li, Pengcheng Laboratory, Shenzhen, China, liyf03@pcl.ac.cn; Ruizhi Liu, SKLP, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, liuruizhi19s@ict.ac.cn; Zhisheng Zeng, Pengcheng Laboratory, Shenzhen, China, zengzhsh@pcl.ac.cn; Zengrong Huang, Pengcheng Laboratory, Shenzhen, Guangdong, China, huangzr@pcl.ac.cn; Zhipeng Huang, Beijing Institute of Open Source Chip, Beijing, China, huangzhipeng@bosc.ac.cn; Dongbo Bu, SKLP, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, dbu@ict.ac.cn; Xingquan Li, Pengcheng Laboratory, Shenzhen, Guangdong, China, lixq01@pcl.ac.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1084-4309/2024/XX-ARTXXX
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Yifan Li, Ruizhi Liu, Zhisheng Zeng, Zengrong Huang, Zhipeng Huang, Dongbo Bu, and Xingquan Li. 2024. AiDRC: Accelerating Detailed Routing by AI-Driven Design Rule Violation Prediction and Checking. *ACM Trans. Des. Autom. Electron. Syst.* XX, X, Article XXX (XX 2024), 29 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The physical design of very large scale integration (VLSI) circuits and systems has become increasingly challenging. As chip sizes continue to shrink, the intricacy of chip design has significantly increased. Additionally, advanced process nodes aimed at improving yield and avoiding manufacturing issues have led to the continuous expansion of design rules, creating a bottleneck [1]. Routing is one of the most critical and complicated steps in the physical design of VLSI circuits [2]. Due to the intricacy of design rules, the routing process is typically divided into global routing (GR) and detailed routing (DR), with track assignment (TA) serving as the connection. After the routing task is completed, design rule checking (DRC) will be executed to accurately validate the circuit. The number of violations is expected to remain below a certain limit before the final timing closure. When numerous design rule violations (DRVs) occur in the circuit, designers must effectively adjust routing settings to resolve the violations and re-run the routing [3, 4]. DRC-based routing is an iterative process that is frequently executed during design to minimize violations [5]. However, time-consuming DRC checks can lead to slow routing convergence.

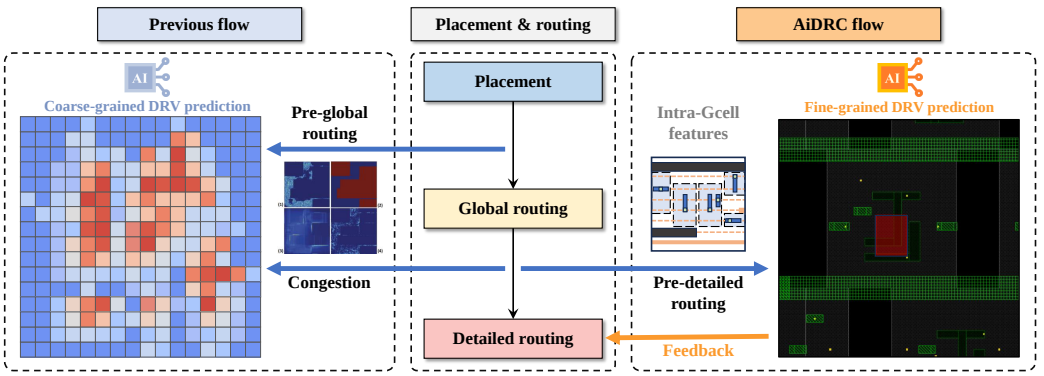
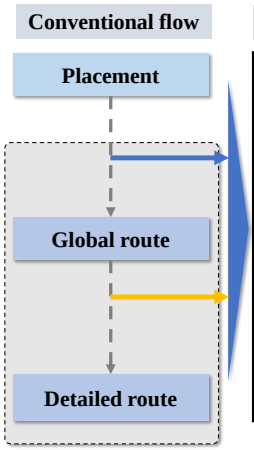


Fig. 1. Coarse-grained and fine-grained DRV predictions.

To address this challenge, numerous experts have proposed faster DRC solutions, introducing predictive models based on machine learning and artificial intelligence techniques to identify DRVs [6–8]. In this context, significant research has been conducted to predict DRVs in the early physical design [9]. The objective is to detect potential violations as early as possible and provide accurate corrective solutions. This approach helps reduce physical design time and improve the efficiency of chip design. In general, as shown in Fig. 1, DRV predictions can be classified into two types.

- The first is coarse-grained prediction, which mainly predicts the number of DRVs or determines the existence of DRVs within a GCell. The GCell-level DRV prediction is generally conducted before detailed routing (mainly placement or global routing). This prediction estimates DRV based on congestion and then provides feedback for quick routability evaluation before detailed routing.



DRC hotspot prediction situation

	[5]	[6]	[7]	[8]	[9]	Ours
AI method	GCN&CNN	FCN	CNN	RF	SVM	ResCCA-CT
Stage	Placement	Placement	Global route	Placement	Placement	Pre-detailed route
Feature	Pin Proximity Graph etc.	Rudy etc.	Surrounding Congestion etc.	Cell Utilization etc.	Cell Density etc.	Pre-detailed route etc.
Level*	Gcell	Standard cell	Gcell	Grid (45 metal track)	Bin (4μm*4μm)	Intra-Gcell
Feedback	N	N	N	N	Y	Y

*Output at different granularities, with or without DRC

Fig. 2. Current status of DRV prediction models and solutions.

- The second is fine-grained prediction, which identifies the specific locations and types of DRC violations within a GCell. This detailed level of prediction can offer more precise routability evaluation and interact with detailed routing. This prediction can accelerate detailed routing and improve the quality of detailed routing.

In Fig. 2, we summarize the current state of DRV prediction models and solutions [10–14]. Traditional approaches primarily extract relevant features from the placement or global routing stage for DRV prediction. Since the placement stage involves fewer elements, DRV prediction and design optimization can be executed more quickly [11]. Additionally, some studies have focused on predicting DRVs during the global routing stage. Global routing is commonly used to predict routability, as it provides a comprehensive congestion map [15]. Most of these studies have concentrated on the binary classification of GCells, emphasizing feature extraction. Liang et al. investigated the complex effects of pin accessibility and routing congestion on DRV prediction, while Hung et al. considered the influence of the surrounding environment [12, 16].

However, as the technology node size decreases, DRVs predicted solely from the GR congestion map no longer correlate well with those that appear after detailed routing. This is due to the many complex design rules applied to the layout to ensure manufacturability [17]. These DRVs, most of which are not visible in the GR routing model, significantly limit the performance of the detailed router. Therefore, GR-based methods are no longer reliable prediction tools for assessing overall design routability or for identifying potential DRVs before DR [18]. Moreover, the approaches mentioned above typically use a coarse-grained solution, relying on grid-based data structures that divide the design into grid cells (GCells). These DRV prediction models generally determine whether a violation will occur within a fixed-size grid [12]. However, this coarse-grained method is ineffective at accurately predicting the spatial distribution of DRVs during the DR stage. DR is generally the most time-consuming stage in the physical design process, as it must account for all design rules to generate precise interconnections. For current commercial tools, the runtime of GR is typically 10% or less of the runtime of detailed routing [13]. Therefore, while significant progress has been made in predicting DRVs, early-stage quick predictions offer limited acceleration for the overall physical design. Furthermore, little research has been conducted on how to effectively use these predictions to resolve DRVs [19].

To accelerate detailed routing and improve the quality of detailed routing, developing an effective, fast, and accurate prediction method is essential. This work introduces a fine-grained DRV prediction model based on ResNet and a customized feature fusion module, aiming to offer in-depth routability evaluation and optimize detailed routing. Specifically, this prediction model quickly estimates potential DRVs that could arise after routing is completed, without performing the detailed routing process. Unlike existing models that predict GCell violations during the placement stage, our approach evaluates DRV based on track assignment results within each GCell. This provides a finer level of granularity and enables more accurate predictions. The proposed DRV prediction method directly outputs location-based distribution information at a finer resolution. This distinction allows it to accurately locate violations during the design iteration process, effectively providing feedback to the detailed routing stage. In addition, we utilize AI-based DRV checking for detailed routing iterations. Note that we do not intend to replace the traditional DRC verification; our goal is to accelerate the tedious iterative checking process using the AI model. To the best of our knowledge, the pre-detailed routing DRV prediction model proposed in this paper is the first to utilize track assignment results for predicting DRC conditions. The unique contributions of the proposed AiDRC are summarized as follows:

- To the best of our knowledge, this is the first work that predicts the specific location of intra-GCell design rule violations at detailed routing using AI model. On one hand, unlike existing DRV prediction works, which focus on the coarse-grained prediction of the number of DRVs or the existence of DRVs, AiDRC can provide more refined prediction and checking at detailed routing, thereby more accurately reflecting potential violations. On the other hand, compared to traditional complex DRC check methods, this AI-based prediction method can greatly reduce the design rule check time and further improve the overall design efficiency.
- We propose a novel network architecture for our AiDRC framework that employs ResNet as its backbone, integrating crisscross attention and channel transformer mechanism. This architecture is specifically developed to learn non-local and cross-channel feature relationships from the diverse set of pre-detailed routing features we extracted, which precisely describe the intra-GCell states. Experimental results show an AUC of 0.987, and an FPR of 0.5%, indicating that our model is effective in identifying true violations while minimizing false positives.
- Compared to the traditional DRC tool, experiment results show that our AiDRC prediction achieves 16× speed-up on DRV estimation, and our AiDRC checking achieves 293× speed-up on DRV checking. Furthermore, we integrate AiDRC into detailed routing to iteratively guide the subsequent detailed routing and avoid unnecessary DRVs. Experimental results show that our AiDRC achieves 44% violation reduction at the initial detailed routing. This outcome demonstrates the significant effect of our AiDRC on reducing the number of DRVs during detailed routing.

The remainder of this paper is organized as follows: Section 2 introduces the problem, some basic knowledge, and our AiDRC framework. Subsequently, Section 3 provides a detailed description of selected features. Section 4 shows our neural network model. In Section 5, a series of experiments are introduced to validate the feasibility and efficacy of the proposed methods. Finally, Section 6 summarizes the core contributions of the paper and offers perspectives on future research directions.

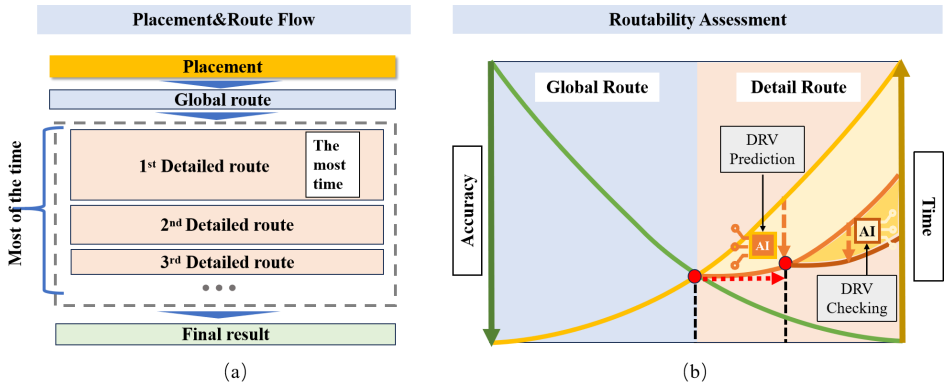


Fig. 3. (a) The runtime proportion of steps in physical design. (b) Diagram of the trade-off between accuracy and time of routability evaluation in the routing process.

2 PROBLEM AND FRAMEWORK

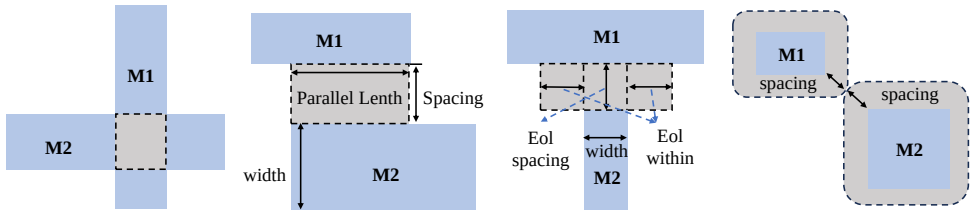
2.1 Problem and Motivation

With the continuous advancement of VLSI circuit design technologies, routability in physical design has become increasingly critical. Routability refers to the ability to complete all interconnections within a design area while satisfying various design constraints, and it remains one of the major challenges in VLSI backend design. In IC physical design, metal wires are used not only to connect nodes for optimal circuit performance but also to strictly adhere to design rules, as violations may lead to unmanufacturable designs. It is widely recognized that the quality of cell placement directly impacts routing feasibility. A well-optimized placement can reduce routing difficulties, whereas poor placement may compromise overall routability or even lead to circuit failures. Therefore, predicting routability quickly in the design process and avoiding numerous DRC conflicts during detailed routing is a critical problem for accelerating physical design efficiency.

In current EDA backend design flows, placement and routing are the most complex steps, with routing further divided into global routing and detailed routing stages. Fig. 3 shows our motivation. As shown in the left diagram, the iterative process in DR is the most time-consuming and resource-intensive phase. To mitigate the runtime losses associated with design modifications, traditional flows typically generate a congestion map during placement using a rapid global routing test. This approach combines congestion information from the global router with layout data, attempting to transform global routing results into predictions that better correlate with the actual distribution of DRVs, thereby aiding placement optimization and reducing subsequent DRC issues during routing.

On the one hand, as shown in Fig. 3(b), while traditional global routing–based routability evaluation strikes a balance between time and accuracy, it is essentially a compromise. Before the time-consuming detailed routing is completed, the actual DRVs in the chip layout remain unknown. Accurate DRC checks can only be performed after a full global-plus-detailed routing process. Recent studies have found that the gap between congestion maps generated by global routers and the actual conditions during detailed routing is widening [18]. Some DRVs are not reflected in the global routing stage, resulting in an overall less accurate routability evaluation. This discrepancy can mislead routability optimization engines that rely on such information, forcing designers to repeatedly modify the layout and thereby extending the design cycle.

On the other hand, DR is typically the most time-consuming stage in the physical design process, as it must account for all design rules across numerous nets to ensure accurate interconnections. In contrast, GR is significantly faster, and predictions based solely on GR provide limited benefits



(a) Short (b) ParallelRunLenth Spacing (c) EndofLine Spacing (d) Minimum Spacing

Fig. 4. Examples of different design rules.

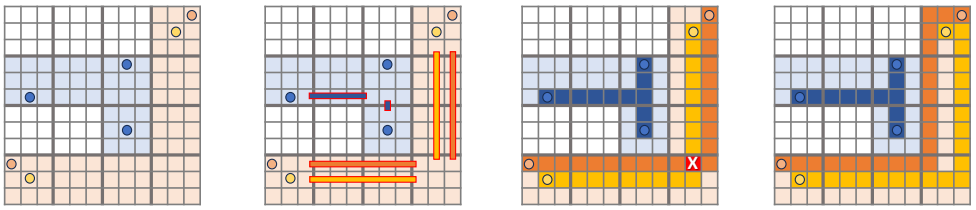
for accelerating iterations at this stage. Predicting DRVs before DR is crucial, as it can significantly reduce design time and resource consumption. In addition, during the design process, routing requires frequent feedback iterations with the DRC engine. As shown in Fig. 3(a), initial iterations often experience a high number of violations, leading to slow DRC checks that dominate the routing iteration time. Therefore, accelerating the DR phase is essential for shortening the overall physical design cycle and improving chip design efficiency.

To address this issue, an alternative approach is to use the design layout before and during the DR stage as an image input and leverage the inherent patterns of DRVs with machine learning techniques for prediction, and further DRV checking. Machine learning methods have shown promising prospects in this field, potentially extending routability evaluation from the early global routing stage to the more precise detailed routing phase, and achieving more accurate predictions to support routability assessment and more efficient routing iteration optimization.

In light of these challenges, this paper proposes a machine learning–based approach integrated into the routing flow to enhance prediction accuracy and avoid the time-consuming detailed routing process. The proposed model takes as input key information available before detailed routing and predicts potential DRVs that may occur during detailed routing, thereby assisting in the evaluation of layout routability. At the same time, fast and accurate DRC checks based on AI can significantly save design time and resources for DR iterations. As depicted in the diagram, compared to traditional flows, the AI-based prediction method establishes a new optimal balance between time and accuracy, providing an effective evaluation of routing feasibility without the need for executing the resource-intensive DR process. This, in turn, enables more informed layout design decisions. Additionally, the model can accurately check DRV locations, offering feedback during routing iterations and accelerating the convergence of the physical design process.

2.2 Preliminaries

2.2.1 Design Rule. In chip design, DRVs refer to regions that violate established design rules or layout constraints. These rules are formulated to ensure the chip's functionality, manufacturability, and long-term reliability. Each technology node defines a specific set of design rules imposed on the circuit layout to constrain the geometric properties of the patterns. Violations of these design rules can result in the failure of the design tape-out. DRC rules cover several critical aspects, including circuit layout, interconnection structures, size constraints, spacing limitations, and relationships between different layers. These rules arise from the limitations imposed by semiconductor manufacturing processes and physical constraints, acting as an interface and agreement between designers and process engineers. By executing DRC, design engineers can identify and correct potential violations in time, ensuring that the chip design meets manufacturing requirements [20].



(a) Global route (b) Track Assignment (c) Detailed route (d) Re-route

Fig. 5. Illustration of the different stages in the routing process.

When considering routing optimization to address DRVs, common approaches include adjusting the distribution of nets, primarily to resolve violations related to shorts and spacing. To provide a more focused and effective analysis and optimization, this work selects several key DRC rules from the 28nm standard CMOS process's process design kit (PDK). As shown in Fig. 4, the related DRC rules include: metal-to-metal shorts, spacing rules corresponding to parallel run lengths, forbidden regions extending from line ends, and minimum spacing requirements around the edges.

2.2.2 Routing. In the physical design process, routing is a crucial step. The specific routing flow is illustrated in Fig. 5. In the GR stage, the global router assigns routing paths for each net onto a grid structure composed of GCells, based on the macro layout of the chip, ensuring that all pins in each net are covered [21]. GR primarily uses a coarse grid graph and guides subsequent detailed routing. To make better use of GR information and bridge the gap between global and detailed routing, the core idea of track assignment is to plan long-distance routing paths across GCells during global routing. This reduces the search space for detailed routing and confines it within individual GCells, accelerating the DR process and improving overall routing efficiency. Once the track assignment is completed, DR is carried out within each GCell to interconnect all the pins [22].

During DR, the nets within the same GCell are connected. In Fig. 5(c), the pink global routing guidance areas illustrate four track assignment (TA) wires. Two orange and two yellow wires belong to different nets. The routing results for different nets can cause DRVs, such as short circuits, which are highlighted in red in the figure. The initial routing introduces DRVs. Therefore, these two nets need to be rerouted. In routing iterations, an A*-based routing algorithm guides the path based on cost. The CostDRC metric quantifies DRVs, and routing paths can be strategically adjusted based on this to mitigate DRVs during the detailed routing stage. Fig. 5(d) shows the routing solution as adjusted in the next iteration. The paths of the orange and yellow nets have been modified, effectively eliminating the previous DRV.

2.2.3 ResNet. DRVs are typically calculated at the grid level, which closely resembles pixel-level classification problems in image segmentation, where classic convolutional neural networks (CNNs) are often used as the backbone for machine learning models in these tasks. In the domain of image segmentation, ResNet [23] has emerged as one of the most influential architectures since its introduction in 2016. The core innovation of ResNet lies in its residual learning framework, which addresses the vanishing gradient problem through skip connections while enabling the training of substantially deeper networks. These characteristics make ResNet particularly suitable for segmentation tasks, as it can effectively preserve spatial information through its hierarchical feature extraction process. Building upon this foundation, we adopt ResNet-34 as the backbone architecture for our DRV prediction model.

Fig. 6 illustrates the ResNet-34 architecture for image classification and consists of several key components: an initial convolutional layer followed by max pooling, and several main residual

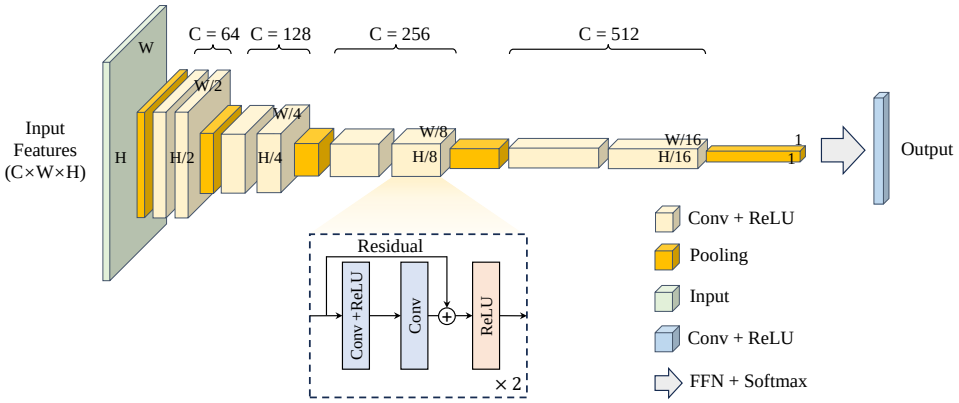


Fig. 6. The vanilla ResNet-34 architecture.

blocks with varying numbers of residual units (e.g. 2-6). The structure of ResNet used in image segmentation also adheres to the deep convolutional neural network design, while eliminating the final fully connected layer and incorporating the integration of upsampling layers in the latter part of the network to restore spatial dimensions. Each residual block contains 2-6 residual units, where each unit implements skip connections that bypass two or three convolutional layers. These skip connections enable the network to learn residual functions rather than direct mappings, effectively addressing the vanishing gradient problem. The architecture maintains spatial information through its hierarchical feature extraction process while progressively increasing the number of filters in deeper layers. This design allows ResNet to capture both low-level and high-level features effectively. As a result, ResNet is particularly well-suited for DRV prediction tasks in chip physical design, as it can effectively model complex spatial patterns while maintaining computational efficiency through its depth and residual learning framework.

2.2.4 RCCA. The recurrent crisscross attention (RCCA) module enhances long-range dependency modeling through a bidirectional attention mechanism that computes orthogonal affinities in horizontal and vertical directions in feature space [24]. This dual-path architecture enables each spatial position to interact with all others through attention chains, overcoming the locality constraints of traditional convolutions. Its recurrent nature further strengthens this capability through iterative attention computations, making RCCA particularly effective in capturing complex spatial patterns, which has been demonstrated in previous research on PCB thermal distribution prediction [25]. For DRV prediction, RCCA proves particularly valuable by modeling the sparse yet spatially correlated nature of DRVs. It effectively detects distributed violation patterns beyond standard convolutional receptive fields, addressing a key limitation in traditional DRV prediction approaches.

2.2.5 Channel Transformer. Transformer architectures, originally developed for natural language processing, have shown remarkable success in capturing long-range dependencies and modeling complex relationships through self-attention mechanisms [26]. Building upon this foundation, the channel transformer introduces a novel approach to model cross-channel feature interaction by establishing dynamic relationships across different feature channels through attention-based mechanisms. In contrast to conventional methods that process information across different channels either independently or through linear weighting, the Channel Transformer employs the multi-head attention mechanism to compute global dependencies among all channels, enabling comprehensive and cross-channel information exchange. The multi-head attention mechanism captures

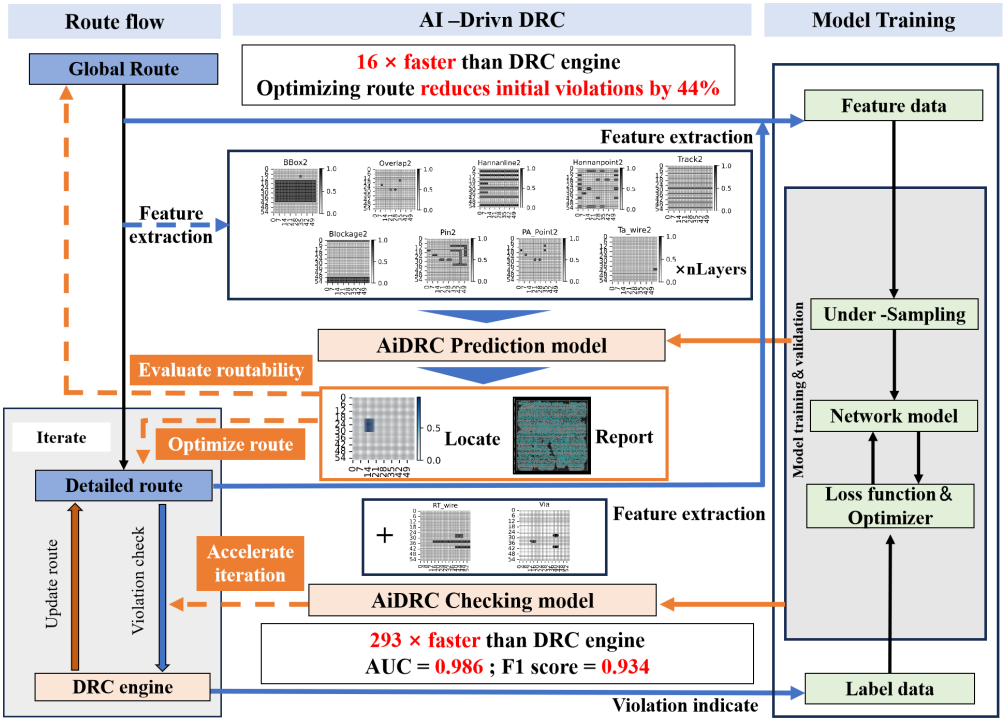


Fig. 7. The overall flow framework of AiDRC, which composes of two parts: 1) AiDRC prediction at initial detailed routing; 2) AiDRC checking during detailed routing.

both local and global channel interactions, while the parallel processing architecture ensures efficient computation of complex relationships. This capability has been demonstrated in various studies, such as time-series forecasting [27], medical image segmentation [28], and 3D Object Detection [29]. It is also crucial for DRV prediction, where diverse features from layout and net need to be effectively synthesized. By facilitating comprehensive information fusion across different feature representations, the channel transformer overcomes the limitations of incomplete feature integration in traditional DRV prediction approaches.

2.3 AiDRC Framework

Our overall framework is illustrated in Fig. 7. The AiDRC framework consists of two modules: 1) AiDRC prediction at initial detailed routing; and 2) AiDRC checking during detailed routing. For the AiDRC prediction module, we feed some features extracted from pre-detailed routing into AiDRC, and then we predict the DRV results of the first round of detailed routing. The predicted DRV results can be used to guide the first round of detailed routing or offer routability evaluation for the design steps before detailed routing. For the AiDRC checking module, we feed the features extracted from the results of the previous round of detailed routing, and then we can obtain the DRV checking results from AiDRC checking module. The DRV results obtained by the AiDRC checking module can be used to guide the next round of detailed routing instead of calling the traditional DRC engine.

In the data preparation phase, we first use an open-source routing tool (the routing tool in iEDA [30]) to perform automated routing on the collected designs. The LEF and DEF files are processed as inputs for the tool. Initially, the entire layout area is divided into a grid of GCells. In

this work, we focus on the DRV prediction during the detailed routing, and the features used in this work are extracted from the track assignment (TA) results after global routing (GR). These features are directly obtained from the post-TA layout results. Some basic features among GCell can be directly obtained, such as pin shapes, pin access points, routing blockages, TA wires, and routing tracks. Furthermore, to achieve a better prediction result, we extract more features, including the bounding box of each net, the overlap region between two different bounding boxes, and the points and lines of the Hanan grid. Details of the feature extraction process are discussed in Section 3. In addition, the DRVs reported by the traditional DRC tool are used as labels, which correspond to the extracted feature images. In summary, some information extracted from the TA stage and the post-routing DRC results serves as the input features and labels, respectively, for the predictive model.

In the model construction and training phase, the obtained labels and corresponding features are processed to create the specified dataset. The whole training flow is divided into two stages: the classification problem of the DRV existence, and the fitting problem of the DRV location. Since binary classification for DRV existence is a well-established technique, the focus here is on selecting data with violations, allowing the model to concentrate on predicting their spatial distribution [10]. This improves model accuracy while reducing training time. The labeled dataset is randomly split into training and validation datasets. The training set is used for model learning, guiding the training process of the DRV predictor, while the validation set is used to evaluate the performance of the trained predictor. The model parameters are iteratively adjusted and optimized until the model performance reaches an acceptable level and meets the expected convergence criteria.

During the inference process, the same features are extracted from the target design and fed into a fine-grained model trained to focus on violation locations. The model predicts the specific distribution of DRVs, allowing designers to estimate DRC status before detailed routing. On one hand, the trained AiDRC model can be used to predict accurate routability before detailed routing. On the other hand, integrating the trained AiDRC model into routing iterations enhances detailed routing. Using the trained AiDRC model in detailed routing can improve the routing quality and reduce time spent on subsequent routing iterations.

3 FEATURE SELECTION

Different features capture various aspects of the design layout, such as size and spacing, each of which may have varying impacts on DRV prediction results. Therefore, extracting and selecting proper features related to DRV is critical, which provides better insightful information within the design layout [31]. To apply the prediction model for DRV detection, we preprocess the extracted features into feature images. High-resolution feature images capture the exact location and shape of each pattern. These features within the GCells are represented using a 2D-pixel matrix with values of 0, 1. If a pattern falls within the GCell area, the corresponding pixels are set to 1; otherwise, they remain 0. For DRV labels, similarly, the violation areas within the GCell are marked with pixels set to 1, while other regions remain 0. Designing effective features significantly influences prediction accuracy. The key idea is to extract as much valuable information as possible from the layout results. In addition to directly extracted features like pins or blockages, the status of routing paths has a strong correlation with DRVs.

However, the TA stage alone does not provide comprehensive net information, making additional feature extraction necessary. Therefore, we incorporate more pre-routing information into our features. To account for the correlation between wiring layers, the lower metal layer serves as the foundation for upper metal layer routing. For example, in M2 layer prediction and check, the characteristics of M1 and M2 layers are taken into account. All extracted and selected features are shown in Fig. 8, and the detailed descriptions are listed as follows:

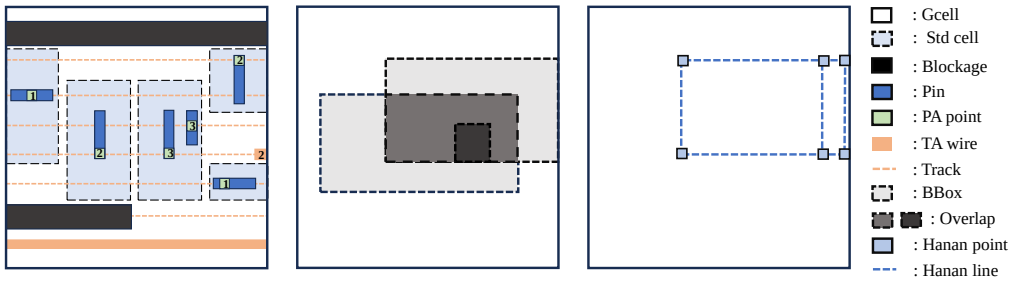


Fig. 8. (a) Features directly extracted in a GCell (including three nets: No. 1, 2, 3). (b) BBox and overlap features extracted in a GCell (including three nets: No. 1, 2, 3). (c) Hanan line and point features extracted in a GCell (corresponding to net No. 2).

(1) **Pin**: We traverse and retrieve the pins contained within the GCell, generating a pin feature map. The distribution and shape of the pins are crucial for determining access points and potential routing blockages. The complex joint effects of pin shapes and nearby blockages is an important cause of DRVs.
(2) **Pin Access Point (PA point)**: PA point refers to the coordinate generated by the pin access tool. We create a square centered on each PA point, with the side length of the square determined by the minimum width of the metal line, as shown in Fig. 8(a). Each square is then mapped to its corresponding GCell.
(3) **Blockage**: Blockages refer to obstacles in the environment that are key factors in causing DRVs during routing. The generation of blockage feature maps follows the same method as the pin feature maps described above, which helps identify areas where routing is constrained.
(4) **Track Assignment Wire**: After track assignment, pre-routed wires across GCells are determined, directly affecting the number of available routing tracks. It actively participates in subsequent routing processes and directly impacts the quality of the routing results.
(5) **Track**: During the routing process, a track is a path on which routing must occur. Routing must be performed along the track, meaning the track serves as a routing resource.
(6) **Bounding Box (BBox)**: The bounding box of a net represents the minimum enclosing rectangular area containing all pins of this net. The BBox of a net outlines its local routing area, providing potential areas where routing may lead to violations.
(7) **Overlap**: Overlap of bounding boxes refers to the overlapping part of bounding boxes corresponding to different nets. Multiple routing paths from different nets may overlap in this region, indicating a high-risk violation region.
(8) **Hanan Point**: Hanan points represent potential corner points, which are likely locations for vias. The six points in Fig. 8(c) are the Hanan points of net 2 in Fig. 8(a).
(9) **Hanan Line**: The Hanan lines connect various points within the routing area, forming the basis for future routing. Hanan line is either horizontal or vertical, and we extend it into a rectangle that meets the minimum line width requirement. Both Hanan points and Hanan lines provide clues for potential routing.

To address the specific needs of our AiDRC prediction model, we examine the factors that lead to DRVs during the DR stage. To further investigate the importance of different features on recognition of DRVs, we calculate the absolute gradient values of model outputs for each input channel

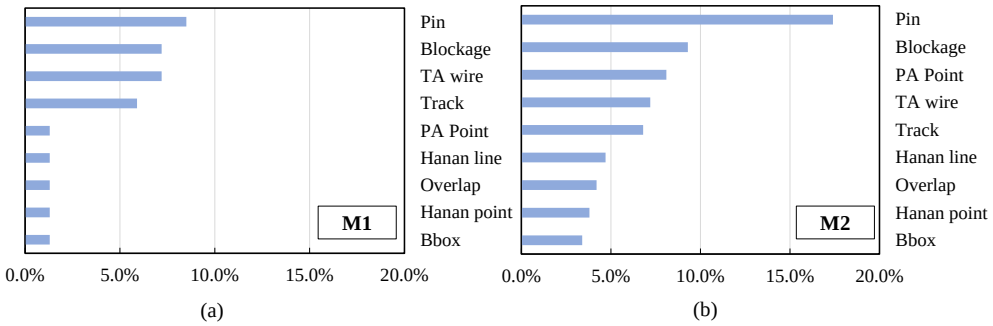


Fig. 9. Comparison of the impact of all features.(a) Case of first metal layer. (b) Case of second metal layer.

to measure the contribution of each channel of feature to DRV recognition. We accumulate and average gradients across all batches of GCell data to obtain overall importance scores, and normalization is also performed. As shown in Fig. 9, the contribution ratio of different input features is reflected. It can be observed that the features of both M1 and M2 layers contribute to DRV prediction. In contrast, the information on the M2 layer itself is more important. From the comparison of different feature types, it can be found that the influence of the features extracted from the layout itself is significantly higher than the further expanded features. This is because, compared with the inherent features, the expanded features are an estimate of the possible location of the wire. Among them, the pin of a net is the most important feature causing DRV.

Furthermore, to train our AiDRC checking model during detailed routing, two features of the routed wires and vias are extracted and fed into the model. Geometric attributes, such as wire width, spacing, jogs, and bends, can lead to issues like shorts, spacing violations, and width-related constraints. The intricate metal traces within the constrained layout plane play a crucial role in determining design rule compliance. Additionally, vias typically expand in both directions along the routing path due to enclosure rules, which define the required metal overlap around the via for reliable electrical connectivity. This expansion, while necessary for manufacturability, can lead to spacing violations when vias are placed too close to adjacent interconnects or other vias.

4 MODEL ARCHITECTURE

4.1 Network Overview

For the DRV prediction model in Fig. 10, we employ a modified ResNet as the backbone of our model and combine it with the feature fusion block, including the recurrent crisscross attention module (RCCA) and channel transformer (CT), to build the prediction network. It receives 18 channels of different GCell features as input, while the output is a single-channel map indicating the positions of the predicted DRV region. To retain more accurate spatial details of feature maps from the GCell, we only apply one down-sampling layer and one up-sampling layer in the ResNet backbone. Every two residual blocks are grouped in one pair. Using 5×5 convolution, the input layer first expands the 18-dimensional features to 32 channels, then progressively increases the channels to 256 by doubling at each residual block, thereby encoding rich information from the input features. After four pairs of residual blocks, a convolutional layer is applied to obtain the feature map X of dimension reduction. Then, X is fed into the feature fusion block, which consists of one RCCA module and one CT module. In the RCCA module, X is fed into two sequential crisscross attention (CCA) modules to generate a new feature map Y, aggregating non-local spatial information. In

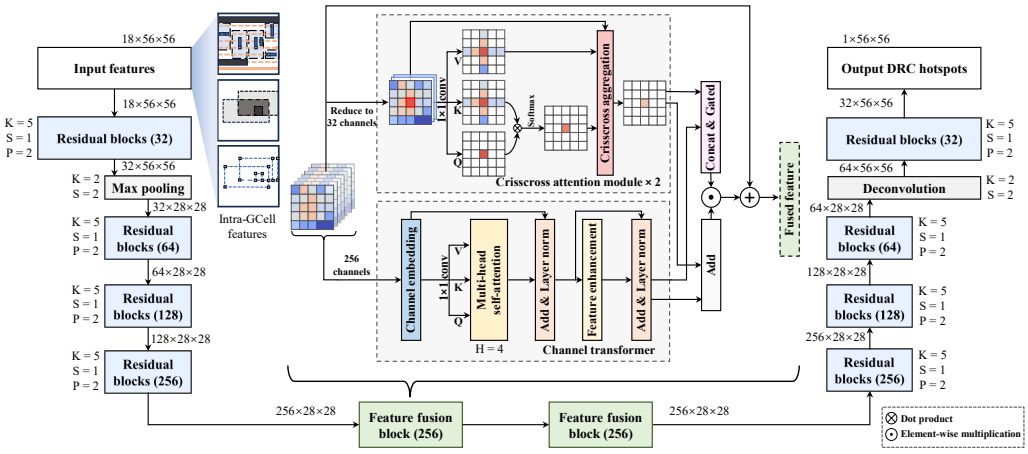


Fig. 10. The architecture of AiDRC prediction model.

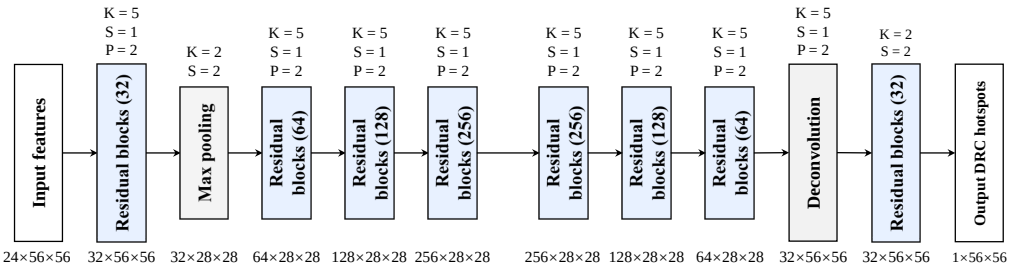


Fig. 11. The architecture of AiDRC checking model.

the CT module, the multi-head attention mechanism captures local and global cross-channel interactions from the channel embedding vectors, while the parallel processing architecture ensures efficient computations. The output Z of the channel transformer and the Y from RCCA will be fused through a learnable gating mechanism, with their combined features further enhanced via a residual connection. Two feature fusion blocks are added and followed by another four pairs of residual blocks with 5×5 convolution. The number of channels gradually decreases from 256 to 32 by halving at each residual block, culminating in a final reduction to 1 by the output layer for prediction. The output feature map undergoes a sigmoid operation to obtain the final predicted DRV map.

For the DRV checking model in Fig. 11, we employ the ResNet-34 architecture [23] and apply only one down-sampling layer and one up-sampling layer to balance computing efficiency and accurate spatial details. It receives 18 channels of features from different layers of GCell as input, and the output is a single-channel map indicating the positions of the predicted post-DR intra-GCell DRV region. This model architecture ensures efficient inference while achieving accurate DRV checking with the help of ResNet design.

4.2 Recurrent Crisscross Attention

Spatial Attention Mechanism: The Recurrent crisscross attention (RCCA) mechanism is adopted in our network to capture long-range dependencies in feature maps and aggregate full-GCell spatial information by sequentially computing two CCA along both horizontal and vertical directions for each pixel [24]. As illustrated in Fig. 10, the RCCA module first reduces the input channel dimension through a 5×5 convolution layer with batch normalization, followed by the core CCA operations.

Given an input feature map $X \in \mathbb{R}^{C \times W \times H}$, where C is the number of channels, W and H are the width and height respectively, the CCA starts by computing queries, keys, and values using 1×1 convolutions with weight matrices W_q, W_k, and W_v:

$$Q = W_q * X, \quad K = W_k * X, \quad V = W_v * X \tag{1}$$

where $*$ denotes the convolution operation and $Q, K \in \mathbb{R}^{C' \times W \times H}$ and $V \in \mathbb{R}^{C \times W \times H}$ with $C' = C/8$ for computational efficiency.

Crisscross Spatial Aggregation: For each pixel (i, j) in the input feature map X, we use $\mathbb{K}_{(i,j)}$ to denote the set of feature vectors extracted from K which are in the same row or column with position (i, j) and $\mathbb{K}_{(i,j)} \in \mathbb{R}^{(W+H-1) \times C'}$, and compute the horizontal attention weights $A^H_{i,j}$ by applying the softmax function to the dot product of the query at row i and the key at position (i, j):

$$d_{(i,j),k} = Q_{(i,j)} \cdot \mathbb{K}^T_{(i,j),k} \tag{2}$$

where $d_{(i,j),k} \in D$ is the degree of correlation between $Q_{(i,j)}$ and $\mathbb{K}_{(i,j),k}$, $k = [1, ..., W+H-1]$. Note that $D \in \mathbb{R}^{(H+W-1) \times (W \times H)}$, we compute the attention map A by applying the softmax function to D. We also use $\mathbb{V}_{(i,j)}$ to denote the collection of feature vectors extracted from V that are in the same row or column with position (i, j), then the crisscross spatial information can be aggregated by:

$$Y_{(i,j)} = \sum_{k=0}^{W+H-1} A_{(i,j),k} \mathbb{V}_{(i,j),k} + X_{(i,j)} \tag{3}$$

Recurrent Processing: The output feature map $Y \in \mathbb{R}^{C \times H \times W}$ is then returned as the result of the CCA mechanism. As shown in Fig. 10, the CCA operation is applied recurrently for two iterations to ensure comprehensive spatial information aggregation. After two CCA operations, the information of positions in different rows and columns from the coordinates (i, j) can be aggregated, thus allowing the learning of long-range spatial correlation. The final output is processed through another 5×5 convolution with batch normalization and a dropout rate of 0.1 for regularization.

4.3 Channel Transformer

Inter-Channel Attention: Beyond RCCA, we apply a modified channel transformer mechanism to enable inter-channel feature fusion [28]. As depicted in the right branch of Fig. 10 and detailed in the channel transformer architecture diagram, this module processes channel-wise correlations through a multi-head self-attention mechanism with convolutional feature enhancement.

Given the output feature map $Y \in \mathbb{R}^{C \times W \times H}$ from CCA, channel-wise correlations are captured through the multi-head self-attention mechanism. We first compute the query, key, and value projections using learnable parameters:

$$Q_c = W_{cq} * X, \quad K_c = W_{ck} * X, \quad V_c = W_{cv} * X \tag{4}$$

where $W_{cq}, W_{ck}, W_{cv} \in \mathbb{R}^{C\times C}$ are 1×1 convolution kernels. The input tensor is reshaped to $\hat{X} \in \mathbb{R}^{(W\times H)\times B\times C}$ (where B is batch size) for multi-head attention computation.

Multi-Head Attention Processing: The scaled dot-product attention computes channel affinity matrices:

$$\text{Attention}(\hat{Y}) = \text{Softmax}\left(\frac{Q_c K_c^T}{\sqrt{d_k}}\right) V_c \in \mathbb{R}^{(W\times H)\times B\times C} \tag{5}$$

where $d_k = C/\text{num\_heads}$ is the dimension scaling factor with num_heads = 4. This computation is performed in parallel across h attention heads, whose outputs are concatenated and reprojected to form the final channel attention features. The normalized residual connection is implemented as:

$$Z = \text{LayerNorm}(X + \text{Reshape}(\text{Attention}(\hat{X}))) \tag{6}$$

Convolutional Feature Enhancement: A convolutional feature enhancement block (ConvFE) with channel expansion replaces the standard feed-forward network, as shown in the FFN component of the architecture:

$$\text{ConvFE}(Z) = W_{conv2} * \sigma(W_{conv1} * Z) \quad \text{where} \begin{cases} W_{conv1} \in \mathbb{R}^{4C\times C\times 1\times 1} \\ W_{conv2} \in \mathbb{R}^{C\times 4C\times 1\times 1} \end{cases} \tag{7}$$

$$Z' = \text{LayerNorm}(Z + \text{ConvFE}(Z)) \tag{8}$$

Dual Attention Feature Fusion: The dual attention features (Y from RCCA and Z' from channel transformer) are then fused through parametric gating G, as illustrated in the fusion mechanism of Fig. 10:

$$F = \text{Concat}(Y, Z') \in \mathbb{R}^{2C\times W\times H} \tag{9}$$

$$G = \text{Sigmoid}(W_g * F) \quad \text{with } W_g \in \mathbb{R}^{C\times 2C} \tag{10}$$

$$Z_{output} = X + G \odot (Y + Z') \tag{11}$$

where $\odot$ denotes element-wise multiplication. This architecture effectively integrates spatial patterns from CCA with channel-wise dependencies learned through transformer operations, establishing comprehensive feature representations for DRV prediction as demonstrated in the overall AiDRC architecture.

4.4 Model Training

The loss function significantly impacts the effectiveness of model training. Since this work transforms the DRV prediction problem into a violation prediction problem, simply using a loss function designed for classification problems cannot meet the design requirements. As shown in Eq. (14), our loss function comprises the focal loss, the dice loss, and a specially designed loss that contributes to precision improvement.

We adopt focal loss ℓ_{focal} [32] to mitigate the imbalance between positive and negative class samples and the imbalance between simple and complex samples in the training data. Let p_t be the predicted probability for the ground truth class t, ℓ_{focal} is defined as:

$$\ell_{focal} = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{12}$$

where α_t is the weighting factor while γ is the parameter that focuses on reducing the loss of ill-classified samples. In this work, we use $\alpha_t = 0.75$ and $\gamma = 2.0$.

We also adopt the dice loss ℓ_{dice} to measure the similarities between the predicted candidate points and the ground truth. Using p_{xy} to represent the probability of a pixel at position (x, y) predicted as a candidate point and g_{xy} to represent the ground truth, ℓ_{dice} can be expressed as:

$$\ell_{dice} = 1 - \frac{2\sum_{x,y} p_{xy} g_{xy} + \epsilon}{\sum_{x,y} p_{xy} + \sum_{x,y} g_{xy} + \epsilon} \tag{13}$$

where ϵ is a small constant added to avoid division by zero.

Additionally, we introduce a custom loss ℓ_{pre} specifically designed to improve the precision of the model's predictions. Precision is a critical metric in tasks where minimizing false positives is essential. This loss penalizes the model for predicting violation in the GCell where the ground truth is negative, thereby encouraging the model to be more conservative and accurate in its predictions. Given the predicted probability map p_{xy} and the ground truth g_{xy}, the custom loss ℓ_{pre} is defined as:

$$\ell_{pre} = \frac{\sum_{x,y} p_{xy} \cdot (1 - g_{xy}) + \epsilon}{\sum_{x,y} p_{xy} + \epsilon}$$

where ϵ is a small constant added to avoid division by zero.

Then the trainable model θ is determined at the training stage by minimizing the loss function as follows:

$$\mathcal{L}(\theta) = c_{fl} \cdot \ell_{focal} + c_{di} \cdot \ell_{dice} + c_{pre} \cdot \ell_{pre} \tag{14}$$

where c_{fl}, c_{di}, c_{pre} represent the weights of corresponding loss items. In this work, we use $c_{fl} = 5.0$, $c_{di} = 1.0$ and $c_{pre} = 2.0$.

5 EXPERIMENTAL RESULTS

5.1 Dataset Generation

Table 1. Statistics on the design set "ysyx" (28nm).

Benchmark	Statistics			
	#Instances	#Nets	#GCells	#Util
ysyx_0	88860	86072	168510	0.65
ysyx_1	95581	92630	169332	0.61
ysyx_2	97207	98065	191406	0.65
ysyx_3	145160	141757	231842	0.60
ysyx_4	283285	275313	573806	0.70
ysyx_5	244476	239401	510510	0.50
ysyx_6	350494	341425	718256	0.70
ysyx_7	379679	371241	771762	0.50
ysyx_8	469977	457585	1161006	0.50
ysyx_9	453875	444934	912980	0.50

• ysyx is a RISC-V processor chip talent plan.

Training and testing data were generated from the "ysyx" chip dataset. Our experiments were conducted using data from a total of 10 chip designs based on 28nm technology. Table 1 provides detailed information on the 10 "ysyx" designs, including various specifications such as the number of standard cells, nets, GCells, and overall resource utilization. For each design, we used the track assignment tool from the open-source EDA platform iEDA to generate iroute data, which is stored in the corresponding GCells as input features for the model [33]. The routing areas are covered

by the same GCell across different routable layers as a whole. Multichannel feature maps were generated for each GCell to facilitate training. The feature extractor was implemented using both PyTorch and the Boost Geometry library in C++ and Python. Subsequently, the detailed router and DRC tools from iEDA were employed to complete the detailed routing tasks within each GCell [30]. The resulting violation distribution data are used to generate the corresponding label data by calling AiEDA (an open-source AI for EDA library) [34].

To ensure fairness and avoid data leakage, design cases generated from the same original design were not used for both training and testing, as they may be very similar. To address this, we implemented a k-fold cross-validation scheme to separate the training and testing data. Specifically, we employed a 10-fold cross-validation approach, which can be formally defined as:

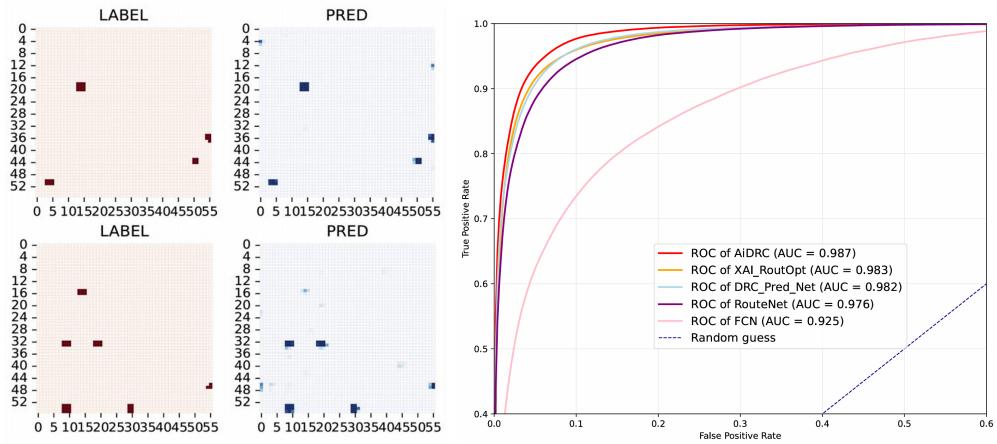
$$\text{For } k = 10, \text{ fold } i \in 1, 2, ..., k: \quad \mathcal{D}_{\text{test}}^{(i)} = C_i, \quad \mathcal{D}_{\text{train}}^{(i)} = \bigcup_{j \neq i} C_j \tag{15}$$

where Ci represents the dataset from the i-th benchmark circuit, $\mathcal{D}_{\text{train}}^{(i)}$ is the training set for fold i, and $\mathcal{D}_{\text{test}}^{(i)}$ is the corresponding test set. In this k-fold validation setup, data from 9 of the 10 benchmark circuits are used for training, while the remaining 1 circuit is used for testing. We repeat the experiment 10 times, each time using data from a different benchmark circuit for testing, with the rest used for training. This k-fold cross-validation ensures that the testing circuits are completely unseen by the training data and provides a robust evaluation of our model's generalization capability across different chip designs.

5.2 Evaluation on DRV Prediction

The design rule violation prediction model was implemented using Python and Pytorch on a Linux machine with two NVIDIA A100 GPUs, 4 Intel Xeon Platinum 8380 CPUs at 2.3 GHz, and 1TB of RAM. The model was built with the following configurations: We used the AdamW optimizer with β coefficients of $(0.95, 0.999)$ and a weight decay coefficient of 0.001. A cosine annealing scheduler [35] was employed to control the learning rate, which decayed from 5e-4 to 1e-6. For model training, each model was trained for 100 epochs. The batch size was set to 128. In our model, ReLU was used as the activation function.

After training the model, evaluation of its performance is especially important in the context of DRV prediction because the DRV dataset is highly imbalanced; only a small fraction of regions are DRVs [35]. Performance metrics such as accuracy are no longer effective measures of predictive performance due to the significant imbalance between the number of positive samples (violations) and negative samples (non-violations). As long as the model predicts most samples as non-DRV samples, the accuracy metric can easily be high [35]. Therefore, selecting a suitable evaluation metric is crucial to effectively reflecting the predictive ability of the model. The DRV prediction task can be transformed into an image detection or segmentation task in the field of computer vision because the output is an activation tensor map of the same size as the original input feature map [31]. Each element in the output binary tensor map can be regarded as a binary classification, of which the commonly used metrics are derived from the classification confusion matrix [17]. The confusion matrix contains true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). In our experiments, precision and recall are used to evaluate the effectiveness of the learning model. Precision is calculated as TP / (TP + FP). Precision focuses on the correctness of samples predicted as positive, rather than the proportion of all correctly predicted samples. Recall is calculated as TP / (TP + FN). Recall emphasizes the model's ability to correctly identify positive samples.



(a) (b)

Fig. 12. (a) Examples of design rule violation labels and corresponding prediction results in GCells. (b) Comparison of average ROC curves of different models.

Since the issue of false alarms (i.e., non-violations predicted as violations, or violations predicted as non-violations) needs to be carefully considered, we evaluate the performance of the prediction model using the area under the ROC curve (AUC). The AUC-ROC curve plays an important role in assessing the performance of classification methods [35]. ROC is a probability curve that describes the trade-off between the true positive rate (TPR) and the false positive rate (FPR) across various classification thresholds. TPR is the proportion of correctly classified positive samples among all actual positive samples (i.e., recall), while FPR is the proportion of negative samples incorrectly predicted as positive among all actual negative samples. AUC represents the model's ability to distinguish between categories. An AUC of 1 on the ROC curve indicates perfect prediction, while random guessing results in an AUC of 0.5. The strength of a prediction model is reflected in its higher AUC, indicating that the model can predict both true positives and true negatives with higher accuracy.

In the evaluation experiment for the DRV prediction task, we compared traditional machine learning models, such as Fully Convolutional Networks (FCN), based on classic convolution architectures. The FCN method is one of the popular approaches for the DRV prediction problem [11]. It does not include a fully connected (FC) layer at the end of the model. FCN can adapt to input images of various sizes and generate outputs with a size identical to the input. It outputs an image of the same size as the original input, making it suitable for pixel-level binary classification problems. We also conducted a comprehensive evaluation by incorporating three state-of-the-art baselines: RouteNet [11], DRC_Pred_Net [19], and XAI_RoutOpt [7]. These models represent recent advances in learning-based prediction of DRV and routing optimization, providing a more robust benchmark for evaluation. We excluded the original U-Net baseline since RouteNet and DRC_Pred_Net were developed upon the original U-Net architecture, and their performances effectively represent the potential of modified U-Net. To ensure a fair comparison, we adopted the original network architectures of these baselines or with minimal modifications (removing only the final flattening and fully connected layers to adapt to our task's input-output requirements), while maintaining consistent loss functions and training parameters across all models. Table 2 summarizes the performance of the models across ten benchmark designs. Four objective evaluation

Table 2. The comparison results of different DRV prediction models FCN, RouteNet [11], DRC_Pred_Net [19], XAI_RoutOpt [7] and AiDRC.

Design	FCN				RouteNet [11]				DRC_Pred_Net [19]				XAI_RoutOpt [7]				AiDRC			
	P	R	FPR	AUC	P	R	FPR	AUC	P	R	FPR	AUC	P	R	FPR	AUC	P	R	FPR	AUC
ysyx_0	0.441	0.425	0.008	0.925	0.644	0.610	0.006	0.977	0.675	0.623	0.005	0.983	0.636	0.609	0.007	0.980	**0.709**	**0.635**	**0.004**	**0.984**
ysyx_1	0.433	0.414	0.009	0.923	0.617	0.587	0.006	0.975	0.655	0.606	0.006	0.979	0.572	0.670	0.005	0.982	**0.701**	**0.656**	**0.004**	**0.986**
ysyx_2	0.443	0.410	0.008	0.905	0.623	0.608	0.005	0.968	0.681	0.672	0.005	0.983	0.642	0.678	0.006	0.983	**0.717**	**0.680**	**0.003**	**0.990**
ysyx_3	0.432	0.410	0.009	0.927	0.645	0.599	0.005	0.976	0.658	0.614	0.007	0.980	0.629	0.635	0.006	0.981	**0.687**	**0.651**	**0.006**	**0.988**
ysyx_4	0.421	0.437	0.010	0.925	0.602	0.614	0.007	0.981	0.644	0.624	0.006	0.981	0.590	**0.647**	0.006	0.984	**0.676**	0.615	**0.005**	**0.985**
ysyx_5	0.415	0.400	0.009	0.931	0.581	0.613	0.007	0.984	0.676	0.651	0.004	0.982	0.620	**0.701**	0.004	0.988	**0.684**	0.686	**0.004**	**0.989**
ysyx_6	0.404	0.388	0.009	0.921	0.606	0.563	0.005	0.971	0.695	0.650	0.005	0.985	0.648	**0.675**	0.006	0.984	**0.705**	0.657	**0.005**	**0.987**
ysyx_7	0.392	0.368	0.009	0.928	0.578	0.594	0.007	0.973	**0.684**	0.669	0.005	0.982	0.611	**0.685**	0.005	0.985	0.682	0.679	**0.005**	**0.987**
ysyx_8	0.436	0.405	0.008	0.938	0.604	0.633	0.008	0.976	0.667	0.690	**0.005**	**0.985**	0.620	0.692	0.007	0.982	**0.687**	**0.694**	0.006	0.984
ysyx_9	0.398	0.371	0.009	0.925	0.621	0.600	0.005	0.981	0.701	0.675	0.005	0.984	0.628	0.673	0.006	0.984	**0.712**	**0.682**	**0.004**	**0.990**
Average	0.422	0.403	0.008	0.925	0.612	0.602	0.006	0.976	0.674	0.647	0.005	0.982	0.620	0.667	0.006	0.983	**0.706**	**0.663**	**0.005**	**0.987**

metrics are used: "Precision (P)", "Recall (R)", "FPR", and "AUC of ROC". These metrics are compared based on the balance between recall and precision, which is further elaborated in Section 5.3. The important results in the table are highlighted in bold.

As shown in Table 2, our AiDRC method demonstrates superior performance across all evaluation metrics compared to all baselines. Specifically, AiDRC achieves the highest average AUC of 0.987, outperforming XAI_RoutOpt (0.983), DRC_Pred_Net (0.982), RouteNet (0.976), and FCN (0.925). The ROC curves in Fig. 12(b) clearly illustrate that AiDRC maintains consistently higher true positive rates across all false positive rate ranges, indicating better discrimination capability. Moreover, AiDRC shows the most balanced performance across different design cases (ysyx_0 to ysyx_9), with consistently high precision (0.706 average) and recall (0.663 average) values, demonstrating its robustness and generalizability, meaning it produces the most hits and the fewest missed detections. At the same threshold of 0.5, our model also has a lower FPR, indicating a lower rate of incorrect violation classifications.

Fig. 12(a) shows the comparison between the DRV prediction results based on our model and the labels. It can be observed that the DRV regions correspond one-to-one with the violations in the labels, proving that the model can accurately locate the violation regions. Fig. 12(b) shows the AUC–ROC curve of the model comparison. Our model has the largest area under the curve with an average value of 0.987, indicating that our model has better classification performance.

5.3 Accelerating Initial Detailed Routing by AiDRC Prediction

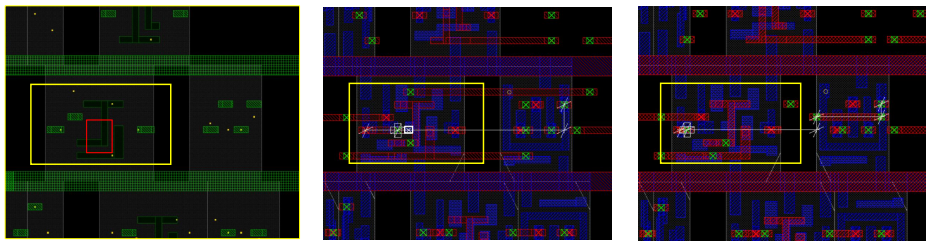
To further investigate the role of our AiDRC model, we integrated it into the routing tool (iRT) [22] of iEDA to accelerate the detailed routing. The core of iRT is based on the A* algorithm, where the cost map is generated by combining known costs and estimated costs. The formulation is as follows:

$$\begin{aligned} Cost_{Total} = {} & Cost_{Node} + Cost_{KnownWire} + Cost_{KnownVia} \\ & + Cost_{KnownSelf} + Cost_{EstimateWire} + Cost_{EstimateVia} \end{aligned} \tag{16}$$

$$Cost_{Node} = Cost_{FixedRect} + Cost_{RoutedRect} + Cost_{Violation} \tag{17}$$

The total cost is composed of several components. $Cost_{Node}$ captures the intrinsic properties of a node and consists of three parts: $Cost_{FixedRect}$, which accounts for the influence of obstacles; $Cost_{RoutedRect}$, which reflects the impact of already routed regions; and $Cost_{Violation}$, which represents DRC violations. Predicted violations are also incorporated into $Cost_{Violation}$, thereby guiding the router to avoid regions with a high likelihood of violations. In addition, $Cost_{KnownWire}$ captures the effects of wire length and direction preferences, $Cost_{KnownVia}$ represents the cost of inserting vias when switching layers, and $Cost_{KnownSelf}$ accounts for special cases, such as discouraging short non-preferred segments that may lead to DRC violations. Finally, $Cost_{EstimateWire}$ is based on the Manhattan distance to approximate the theoretical shortest wire length, while $Cost_{EstimateVia}$ considers the layer difference to represent the minimum required via cost.

In addition, for the iterative updates of cost map, the detailed routing functionality of iRT updates the net after completing the routing of each net. It then recalculates factors such as non-preferred direction routing, vias (layer changes), proximity to fixed obstacles, and overlaps with existing routed wires. Moreover, DRVs are updated globally for each checking window. Therefore, the cost map is dynamically updated based on the routing order within each checking window and the movement of checking window. The method used to resolve DRVs in detailed routing is based on redistributing routing resources with a cost penalty. For instance, methods like splitting and rerouting are based on cost parameters. By optimizing cost parameters, iRT can improve the routing quality. Currently, based on the prediction model, the DRC cost parameters are incorporated



(a) (b) (c)

Fig. 13. (a) Design rule violation prediction (red box) before DR. (b) Actual DRV generated after initial DR. (c) Initial DR after iteration based on DRV prediction.

Table 3. Statistics on four new chip designs.

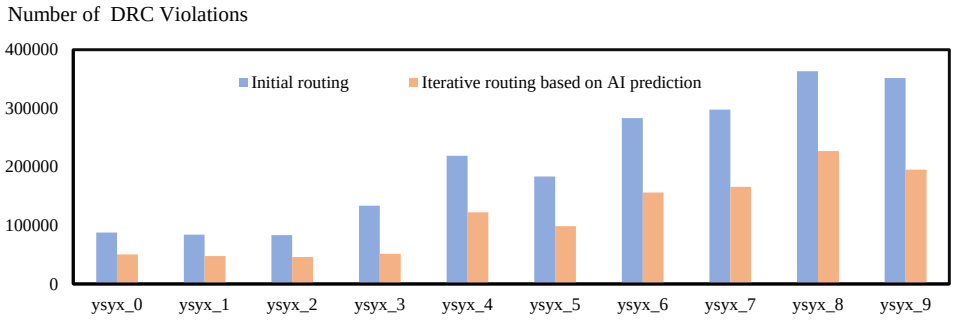
Benchmark	#Instances	#Nets	Description	Source
s38584	6977	6540	A digital processing unit	ISCAS89
AES	19071	18007	The AES block cipher core	OpenLane2
JPEG	27507	28996	A JPEG encoder module	OpenROAD
Eth	41905	38178	Ethernet MAC (media access control) module	OpenROAD

according to the prediction results. This evaluation is provided as DRC cost feedback to iRT. The subsequent iterations of detailed routing can be done more strategically, thereby reducing DRVs.

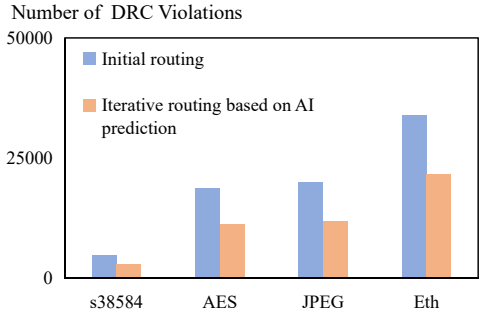
As shown in Fig. 13, we further illustrate the predicted DRV map obtained from our prediction model, the actual DRV map of the initial detailed route without iteration, and the DR map after iteration based on the prediction results, from left to right, to provide an intuitive comparison. The scale of each map is the same. The prediction result of our model in Fig. 13(a) is marked with a red rectangular box. The small white box with a cross in the corresponding figure of the violation generated after direct routing in Fig. 13(b) achieves coverage matching with the violation position after the actual detailed routing. We integrate the predicted DRV result by our prediction model into the initial DR, then we obtain the result without DRV as shown in Fig. 13(c). This is because the router updates the wire net to avoid violating the DRC.

To further validate the effectiveness of our model, we compared the initial routing quality of the traditional design flow with the routing quality after iteration based on the predictive model. Each design instance underwent routing twice: first, to obtain the DRVs from the initial routing, and second, to generate a higher-quality solution guided by DRV predictions. The comparison is based on the balance of precision and recall. The figure illustrates the difference in the number of DRVs. As shown in Fig. 14(a), our predictive model, when combined with routing-related optimizations, effectively reduces DRVs in each design, significantly improving the initial routing quality. Overall, the predicted routing iteration eliminates an average of 44% DRVs in the initial routing.

To further demonstrate the generalizability of the proposed method in detailed routing scenarios, we selected four representative designs from ISCAS'89, OpenLane2 and OpenROAD for experimentation as shown in Table 3. By applying our AI-based prediction model during iterative routing on these open-source benchmarks, we achieved a significant reduction in the number of initial routing violations—on average by 38.8% as shown in Fig. 14(b). These results validate the robustness and effectiveness of the AI prediction approach across different design contexts.



(a) Comparison on "ysyx" design set.



(b) Comparison on four new designs from ISCAS'89, OpenLane2 and OpenROAD.

Fig. 14. Comparison of the number of initial detailed routing DRVs based on DRC engine and AI prediction.

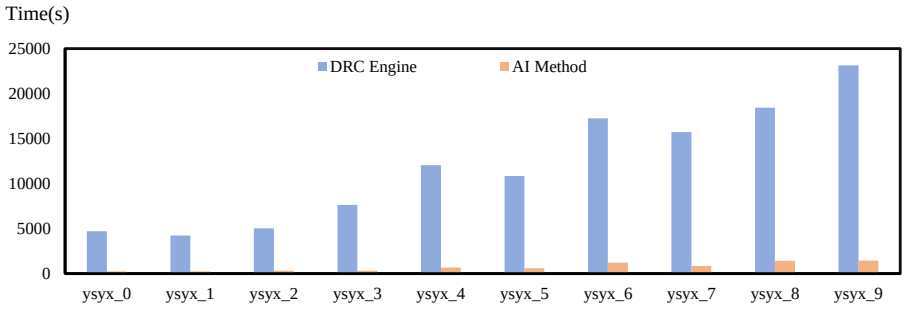


Fig. 15. Comparison of the time consumption for obtaining DRVs based on DRC engine and AI prediction.

In addition, computational efficiency is a crucial factor in practical applications. We compared the speed of DRV prediction with the time required to run the same rule set in a traditional DRC engine during a detailed routing iteration. Our framework integrates multiple test sets, and Fig. 15 presents the inference time of our model alongside the checking time of the traditional DRC engine for each design example, which is the accumulation of the time of each thread under the same multi-threading. Among the design examples, the fastest inference time is only 5% of the DRC engine's checking time, achieving an average speed-up of 16×. While the traditional DRC engine

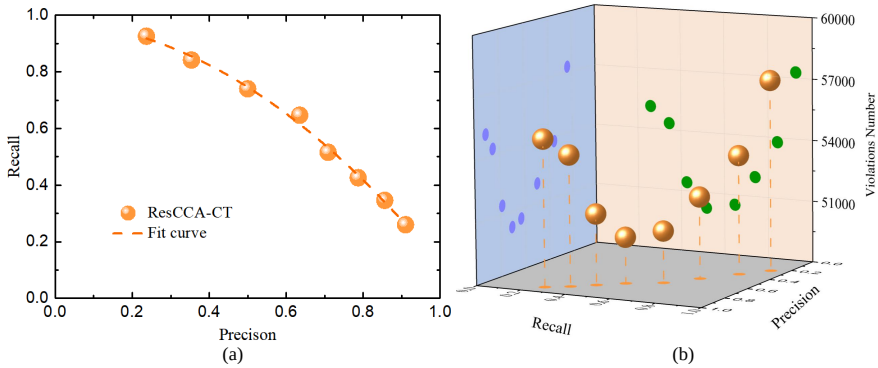


Fig. 16. (a) The trade-off between precision and recall of AiDRC prediction model. (b) The number of DRVs after initial routing iteration based on prediction results of different precision and recall.

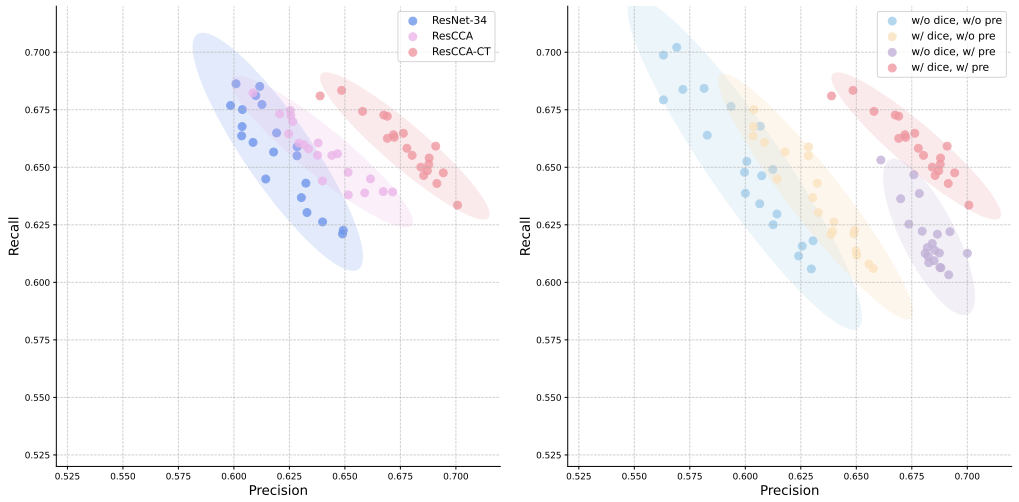
provides a more comprehensive and precise violation check, the significant speed advantage of the AI prediction model makes it suitable for early-stage detailed routing. Early detection and resolution of conflicts can help shorten the overall design cycle.

For model training, the parameters are influenced by the loss function, and the prediction metrics can vary significantly. Here, we take precision and recall as examples. A high recall rate indicates broad violation prediction coverage, but it may also lead to false positives. Conversely, a high precision rate suggests fewer false positives, but it may result in fewer predicted violations. Therefore, the impact of feedback-based routing iteration remains highly uncertain. To further explore the balance between these metrics, we adjust the weight ratio of the combined loss function. As shown in the Fig. 16(a), this produces a set of trade-off data for precision and recall. The fitted curve forms an arc, showing that as one metric increases, the other decreases. The corresponding generated model is then integrated into detailed routing, and an iteration is performed based on "ysyx_0". As illustrated in the Fig. 16(b), the number of violations is minimized when precision and recall are relatively balanced. When precision increases, prediction coverage decreases; when recall increases, the number of violations rises accordingly. From the actual feedback results, the highest optimization is achieved when a balance is maintained. Compared to extreme cases on both sides, the reduction in the middle reaches up to 12%, demonstrating that iterative violation reduction yields the greatest benefit.

5.4 Ablation Study of AiDRC Prediction

To evaluate the impact of individual components in our proposed network architecture, as well as the contribution of different terms in the loss function, we conducted a series of ablation studies.

Fig. 17(a) compares three variants: the original ResNet, ResNet with Crisscross Attention (ResCCA), and ResNet with both Crisscross Attention and Channel Transformer (ResCCA-CT). Experiments are conducted using 100,000 randomly sampled GCells from cases ysyx-5~9 for training and 10,000 GCells from ysyx-0~4 for testing. We report mean precision and recall on the test set, with confidence analysis over 20 independent runs (using different random seeds), visualized through 95% confidence ellipses to robustly assess performance. Results show that ResCCA-CT achieves superior performance with its distribution nearest to the top-right corner (higher precision & recall) and tighter clustering that demonstrates both higher accuracy and stability. CCA alone also provides measurable accuracy gains, which are valuable for DRV prediction. Using the same ResCCA-CT architecture, we examine the impacts of different loss functions in Fig. 17(b). The experimental results demonstrate that combining focal loss ℓ_{focal}, dice loss ℓ_{dice} and precision loss ℓ_{pre} (w/dice,



(a) (b)

Fig. 17. Ablation studies. (1) Network architecture. (b) Loss function.

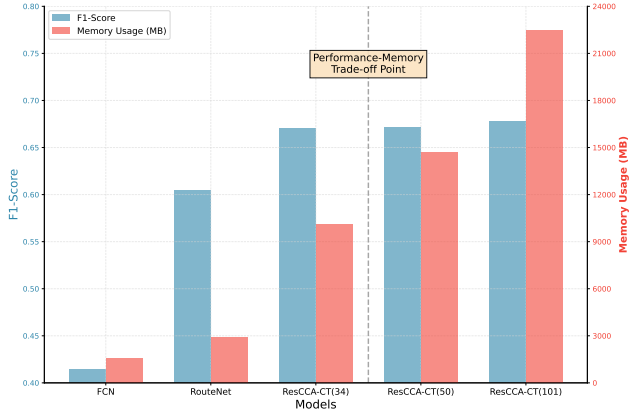


Fig. 18. Model performance vs GPU Memory Usage Comparison.

w/pre) yields optimal results (nearest to the top-right corner), while adding only ℓ_{dice} (w/dice, w/o pre) or only ℓ_{pre} (w/o dice, w/ pre) still improves performance over using solely ℓ_{focal} (w/o dice, w/o pre), which also reduces the stability of model. These findings confirm that both architectural refinements (Crisscross Attention / Channel Transformer) and loss function design critically impact model robustness and prediction quality.

To validate the rationality of backbone network depth selection in the ResCCA-CT model, we conducted a comparative analysis of performance and resource consumption across ResNet architectures with varying depths. As illustrated in Fig. 18, transitioning from FCN to U-Net and finally to ResCCA-CT(34), the model's F1-score improved from 0.412 to 0.671, while GPU memory usage increased from 1588 MB to 10125 MB, demonstrating a substantial performance-resource trade-off. However, further increasing the backbone depth to ResNet-50 and ResNet-101 led to a sharp rise in

Table 4. The results of AiDRC checking model.

Design	Metrics				
	Precision	Recall	FPR	AUC	F1-score
ysyx_0	0.941	0.925	0.001	0.997	0.932
ysyx_1	0.959	0.928	0.001	0.995	0.943
ysyx_2	0.953	0.910	0.001	0.985	0.931
ysyx_3	0.972	0.906	0.001	0.980	0.938
ysyx_4	0.935	0.917	0.001	0.995	0.926
ysyx_5	0.967	0.911	0.001	0.983	0.938
ysyx_6	0.964	0.922	0.001	0.995	0.942
ysyx_7	0.964	0.932	0.001	0.985	0.948
ysyx_8	0.973	0.884	0.001	0.971	0.927
ysyx_9	0.956	0.868	0.001	0.987	0.910
Average	**0.960**	**0.909**	**0.001**	**0.986**	**0.934**

memory consumption to 14714 MB and 22470 MB (representing increases of 45% and 122%, respectively), accompanied by only marginal F1-score improvements to 0.672 and 0.678 (gains of merely 0.15% and 1.04%). This observation indicates that ResCCA-CT(34) achieves the optimal point in the performance-resource tradeoff. Deeper architectures not only substantially increase memory overhead and training complexity but also potentially introduce overfitting risks, while offering diminishing performance returns. Therefore, employing ResNet-34 as the backbone for ResCCA-CT provides an optimal balance, ensuring robust prediction performance while maintaining reasonable computational resource requirements.

As previously noted, DRC checking is a frequent and time-intensive task during detailed routing iterations, with the DRC engine's computation serving as a significant bottleneck. To accelerate this stage, we embed an AI model into the detailed routing iteration process to enhance checking efficiency. After each detailed routing iteration, key features are extracted and fed into the AI model, which generates rectangular boxes of DRVs. These DRV results are then passed to the next iteration, guiding detailed routing decisions and helping to bypass violations, thereby optimizing the network. At this detailed routing stage, the AI model leverages more routing and via feature information to enhance DRV detection.

5.5 Accelerating Detailed Routing by AiDRC Checking

Our AiDRC model delivers exceptional checking accuracy and achieves a significantly higher acceleration ratio compared to traditional DRC engine-based checking. Table 4 summarizes the AiDRC checking model across ten benchmark designs. Five objective evaluation metrics are used: "Precision", "Recall", "FPR", "AUC of ROC", and "F1-score". The important results in the table are highlighted in bold. The experimental results in Table 4 show that our AiDRC checking achieves an impressive AUC of 0.986, F1-score of 0.96, along with high recall, and precision. A visual diagram of metrics is shown in Fig. 19(a). These metrics demonstrate that our AiDRC checking model can correctly detect most of the design rule violations. This reflects the effectiveness of our AiDRC checking in DRC detection during detailed routing iterations.

The runtime speed-up results on "ysyx_0" are shown in Fig. 19(b). Our AiDRC checking offers fast inference speed. Compared to traditional DRC engine-based checking, our AiDRC checking performs violation prediction 293× faster. This improvement demonstrates that AI-driven DRV

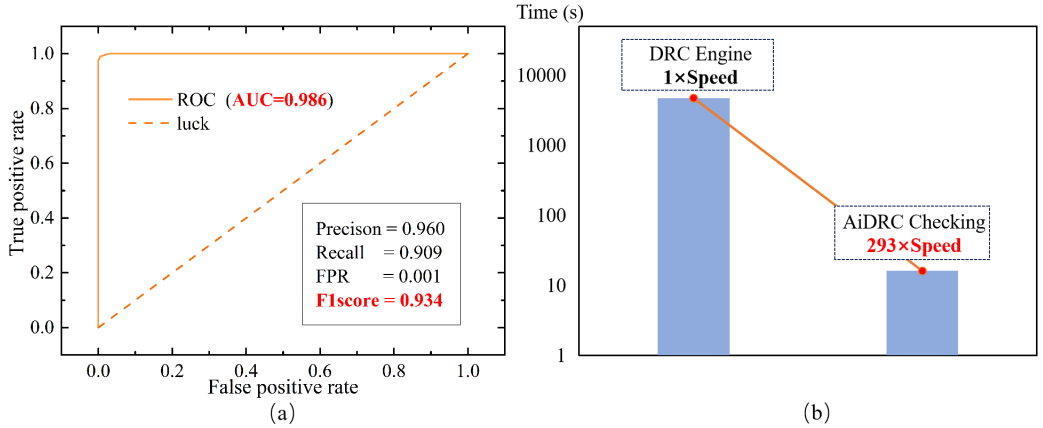


Fig. 19. (a) Performance indicators of our AiDRC checking in detailed routing iteration. (b) Runtime comparison between traditional DRC engine and our AiDRC checking.

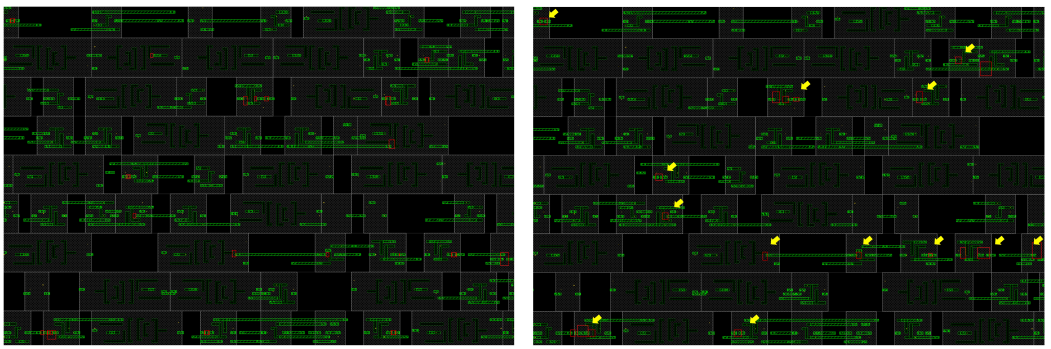


Fig. 20. (a) Actual DRV results from a DRC engine (displayed as a red box). (b) DRV results from our AiDRC cheking.

checking can significantly accelerate the detailed routing iteration process. However, the DRC engine remains essential as the final check to ensure full DRC verification before routing convergence. AI can work alongside the DRC engine to accelerate design convergence, reducing overall iteration time while maintaining accuracy.

At last, we show a comparison of design rule violation checking results as in Fig. 20. The ground truth DRV results detected by a traditional DRC engine are shown in Fig. 20(a), and the detection results by our AiDRC checking are shown in Fig. 20(b) (highlighted by yellow arrows). From the two figures, we can see that our AiDRC checking can achieve relatively high detection accuracy.

6 CONCLUSION

This study presents AiDRC, an advanced deep learning-based framework designed for precise fine-grained DRV prediction and accelerated DRV checking. We introduce an innovative DRV prediction model, which employs a ResNet-based architecture enhanced with non-local recurrent crisscross attention and cross-layer channel transformer mechanisms. This enables accurate and

efficient routability prediction in the physical design process. This model predicts the fine-grained intra-GCell DRVs after the completion of track assignment, successfully identifying initial DRVs and effectively overcoming the limitations of existing coarse-grained prediction methods based solely on global routing. The routing guidelines generated based on the prediction results significantly reduce the number of initial DRVs by 44%, thereby lowering design iteration costs. Beyond prediction capabilities, our framework introduces a neural network-based DRV checking paradigm that replaces traditional DRC engines in the detailed routing iteration. The implemented ResNet-based DRV checking model achieves an extraordinary 293× acceleration compared to the conventional rule-based DRC engine while maintaining equivalent accuracy in DRV identification. This work strikes a better balance between accuracy and computational efficiency and pioneers the application of learning-based methods into complex IC physical design workflows. Compared to traditional DRV checks performed after detailed routing, our method significantly reduces the checking time, enabling layout engineers to assess routability quickly and converge toward a DRV-free layout. The application of this approach is expected to advance routability prediction technology and provide a more efficient solution for IC design.

REFERENCES

[1] Yu-Hung Huang, Zhiyao Xie, Guan-Qi Fang, Tao-Chun Yu, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. Routability-driven macro placement with embedded cnn-based prediction model. In *Proc. of 2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 180–185, 2019.

[2] Aysa Fakheri Tabrizi, Nima Karimpour Darav, Logan Rakai, Ismail Bustany, Andrew Kennings, and Laleh Behjat. Eh?Predictor: A Deep Learning Framework to Identify Detailed Routing Short Violations From a Placed Netlist. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 39(6):1177–1190, 2020.

[3] Chen-Han Lu, Hsin-Hung Pan, Ting-Chi Wang, Po-Yuan Chen, and Chin-Fang Cindy Shen. Machine Learning based Routing Guide Generation and its Application to Design Rule Violation Reduction. In *Proc. of 2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA)*, pages 1–4, 2023.

[4] Ping Zhang, Pengju Yao, Xingquan Li, Bei Yu, and Wenxing Zhu. V-GR: 3D Global Routing with Via Minimization and Multi-Strategy Rip-up and Rerouting. In *Proc. of 2024 Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 963–968. IEEE, 2024.

[5] Daeyeon Kim, Jakang Lee, and Seokhyeong Kang. Routability prediction using deep hierarchical classification and regression. In *Proc. of 2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–2, 2023.

[6] Jingui Lin, Wenxiong Lin, Shiyan Liang, Peng Gao, Yan Xing, Tingting Wu, Xiaoming Xiong, and Shuting Cai. An efficient method of drc violation prediction with a serial deep learning model. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 29(6):1–16, 2024.

[7] Seonghyeon Park, Daeyeon Kim, Seongbin Kwon, and Seokhyeong Kang. Routability prediction and optimization using explainable ai. In *Proc. of 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8, 2023.

[8] Chen-Chia Chang, Jingyu Pan, Tunhou Zhang, Zhiyao Xie, Jiang Hu, Weiyi Qi, Chun-Wei Lin, Rongjian Liang, Joydeep Mitra, Elias Fallon, and Yiran Chen. Automatic routability predictor development using neural architecture search. In *Proc. of 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9, 2021.

[9] Jhen-Gang Lin, Yu-Guang Chen, Yun-Wei Yang, Wei-Tse Hung, Cheng-Hong Tsai, De-Shiun Fu, and Mango Chia-Tso Chao. DRC Violation Prediction with Pre-global-routing Features Through Convolutional Neural Network. In *Proc. of the Great Lakes Symposium on VLSI (GLSVLSI)*, page 313–319, 2023.

[10] Hyunbum Park, Kyeonghyeon Baek, Suwan Kim, Kyumyung Choi, and Taewhan Kim. Pin Accessibility and Routing Congestion Aware DRC Hotspot Prediction for Designs in Advanced Technology Nodes With Consolidated Practical Applicability and Sustainability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD)*, 43(12):4786–4799, 2024.

[11] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network. In *Proc. of the 37th IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2018.

[12] Wei-Tse Hung, Yu-Guang Chen, Jhen-Gang Lin, Yun-Wei Yang, Cheng-Hong Tsai, and Mango Chia-Tso Chao. DRC Violation Prediction After Global Route Through Convolutional Neural Network. *IEEE Transactions on Very Large Scale Integration Systems*, 31(9):1425–1438, 2023.

[13] Riadul Islam. Early Stage DRC Prediction Using Ensemble Machine Learning Algorithms. *IEEE Canadian Journal of Electrical and Computer Engineering*, 45(4):354–364, 2022.

[14] Ying-Yao Huang, Chang-Tzu Lin, Wei-Lun Liang, and Hung-Ming Chen. Learning Based Placement Refinement to Reduce DRC Short Violations. In *Proc. of 2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–4, 2021.

[15] Wei-Tse Hung, Jun-Yang Huang, Yih-Chih Chou, Cheng-Hong Tsai, and Mango Chao. Transforming Global Routing Report into DRC Violation Map with Convolutional Neural Network. In *Proc. of the 2020 International Symposium on Physical Design (ISPD)*, page 57–64, 2020.

[16] Rongjian Liang, Hua Xiang, Jinwook Jung, Jiang Hu, and Gi-Joon Nam. A stochastic approach to handle non-determinism in deep learning-based design rule violation predictions. In *Proc. of the 41st IEEE/ACM International Conference on Computer-Aided Design(ICCAD)*, pages 1–8, 2022.

[17] Luis Francisco, Tanmay Lagare, Arpit Jain, Somal Chaudhary, Madhura Kulkarni, Divya Sardana, W. Rhett Davis, and Paul Franzon. Design Rule Checking with a CNN Based Feature Extractor. In *Proc. of 2020 ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD)*, pages 9–14, 2020.

[18] Wei-Ting J. Chan, Pei-Hsin Ho, Andrew B. Kahng, and Prashant Saxena. Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning. In *Proc. of the 2017 ACM on International Symposium on Physical Design (ISPD)*, page 15–21, 2017.

[19] Hailiang Li, Yan Huo, Yan Wang, Xu Yang, Miaohui Hao, and Xiao Wang. A Lightweight Inception Boosted U-Net Neural Network for Routability Prediction. In *Proc. of the 2nd IEEE International Symposium of Electronics Design Automation (ISEDA)*, pages 648–653, 2024.

[20] Andrew B. Kahng, Lutong Wang, and Bangqi Xu. TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(TCAD)*, 41(4):1076–1089, 2022.

[21] Ruizhi Liu, Shizhe Ding, Jingyan Sui, Xingquan Li, and Dongbo Bu. NeuralSteiner: Learning Steiner Tree for Overflow-avoiding Global Routing in Chip Design. *Advances in Neural Information Processing Systems (NIPS)*, 37:127346–127368, 2025.

[22] Zhisheng Zeng, Jikang Liu, Zhipeng Huang, Ye Cai, Biwei Xie, Yungang Bao, and Xingquan Li. Net Resource Allocation: A Desirable Initial Routing Step. In *Proc. of the 61st ACM/IEEE Design Automation Conference (DAC)*, page 1–6, 2024.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[24] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. CCNet: Criss-Cross Attention for Semantic Segmentation . *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(06):6896–6908, 2023.

[25] Tinghuan Chen, Silu Xiong, Huan He, and Bei Yu. TRouter: Thermal-Driven PCB Routing via Nonlocal Crisscross Attention Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(10):3388–3401, 2023.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, 2017.

[27] Wenyong Han, Tao Zhu, Liming Chen, Huansheng Ning, Yang Luo, and Yaping Wan. MCformer: Multivariate Time Series Forecasting With Mixed-Channels Transformer. *IEEE Internet of Things Journal*, 11(17):28320–28329, 2024.

[28] Haonan Wang, Peng Cao, Jiaqi Wang, and Osmar R Zaiane. Uctransnet: rethinking the skip connections in u-net from a channel-wise perspective with transformer. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 2441–2449, 2022.

[29] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3D Object Detection With Channel-Wise Transformer. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2743–2752, October 2021.

[30] Xingquan Li, Simin Tao, Shijian Chen, Zhisheng Zeng, Zhipeng Huang, Hongxi Wu, Weiguo Li, Zengrong Huang, Liwei Ni, Xueyan Zhao, He Liu, Shuaiying Long, Ruizhi Liu, Xiaoze Lin, Bo Yang, Fuxing Huang, Zonglin Yang, Yihang Qiu, Zheqing Shao, Jikang Liu, Yuyao Liang, Biwei Xie, Yungang Bao, and Bei Yu. iPD: An Open-source intelligent Physical Design Toolchain. In *Proc. of 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 83–88, 2024.

[31] Lin Li, Yici Cai, and Qiang Zhou. An Efficient Approach for DRC Hotspot Prediction with Convolutional Neural Network. In *Proc. of 2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2021.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proc. of 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[33] Xingquan Li, Simin Tao, Zengrong Huang, Shijian Chen, Zhisheng Zeng, Liwei Ni, Zhipeng Huang, Chunan Zhuang, Hongxi Wu, Weiguo Li, et al. iEDA: An Open-source Intelligent Physical Implementation Toolkit and Library. *arXiv preprint arXiv:2308.01857*, 2023. https://github.com/OSCC-Project/iEDA.

[34] Yihang Qiu, Zengrong Huang, Weiguo Li, Xinhua Lai, Rui Wang, He Liu, Ping Zhou, Simin Tao, Junfeng Liu, Yifan Li, et al. AiEDA-2.0: An Open-source AI-Aided Design (AAD) Library for Design-to-Vector. In *Proc. of 2025 International Symposium of Electronics Design Automation (ISEDA)*, pages 5–6. IEEE, 2025. https://github.com/OSCC-Project/AiEDA.

[35] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network. In *Proc. of the 2020 International Symposium on Physical Design (ISPD)*, page 135–142, 2020.