Routability-driven Power/Ground Network Optimization Based on Machine Learning

PING-WEI HUANG, Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan
YAO-WEN CHANG, Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan

The dynamic IR drop of a power/ground (PG) network is a critical problem in modern circuit designs. Excessive IR drop slows down circuit performance and causes potential functional failures. Most industrial practices tend to over-design the PG network for the dynamic IR drop constraints, reducing routing resources and incurring routing congestion. Existing machine-learning-based approaches target only dynamic IR drop prediction without considering the routability affected by the P/G network. This paper develops a machine-learning-based method to solve the dynamic IR drop and routing resources trade-offs. Our model can predict the two targets accurately by adopting a multi-task learning scheme, achieving a 0.99 high correlation coefficient. We show that our trained model is generalizable by testing different placement results. Our algorithm also achieves significant speedups of up to 29X compared to the time-consuming dynamic IR drop simulation by a leading commercial tool. Experimental results show that our algorithm can save about 13% routing resources without worsening the dynamic IR drop peak value.

CCS Concepts: • **Computer-Aided Engineering** → Computer-Aided Design (CAD).

Additional Key Words and Phrases: Physical design, power grid design, Power estimation and optimization, Machine learning, Neural Networks

1 INTRODUCTION

A power/ground (PG) network is designed to supply ideal voltage to various devices in the chip and thus plays an essential role in signal integrity. A typical PG network consists of power pads, power stripes, stacked vias, and power rails. A power pad is used to supply voltage to the chip, and the power stripes draw current from the power pads. The power stripes are in the higher metal layers and connect to the power rails with stacked vias. The PG stripes in higher metal layers are usually wider than those in lower layers because a wider stripe width results in lower parasitic resistance. A stacked via is used to connect a PG stripe to a PG rail. Then, the power rails are connected to the PG pins of the cell instances. Figure 1 shows an example of a PG network structure.

The configuration of a PG network strongly affects the reliability issues such as IR drop and electro-migration (EM). Building the desired PG network for an advanced technology node needs to satisfy the IR-drop and EM constraints to achieve better performance or reliability [4]. Moreover, the topology of a given PG network also affects the routability and wirelength of the subsequent global routing stage because the PG wires in the intermediate layer may compete for resources with the signal routing. The PG wires may also occupy redundant routing tracks if the widths of power stripes are not properly chosen [3].

The PG network design, or power planning, is performed before placement in the conventional IC design flow. Without the information of cell instances, current industrial approaches usually tend to reserve sufficiently larger PG metal areas at an early stage to ensure reliability [3]. However, this over-design in the PG network would inevitably waste signal routing resources. The over-designed PG vias and wires in the intermediate metal layers

This work was partially supported by the National Natural Science Foundation of China under Grants 61672005 and 61977017, and by AnaGlobe, MediaTek, TSMC, MOST of Taiwan under Grant No's MOST 104-2221-E-002-132-MY3, MOST 105-2221-E-002-190-MY3, MOST 106-2911-I-002-511, and NTU under Grant No. NTU-ERP-104R8951.
Authors' addresses: Ping-Wei Huang, Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, pwhuang@ntu.edu.tw; Yao-Wen Chang, Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan, ywchang@ntu.edu.tw.

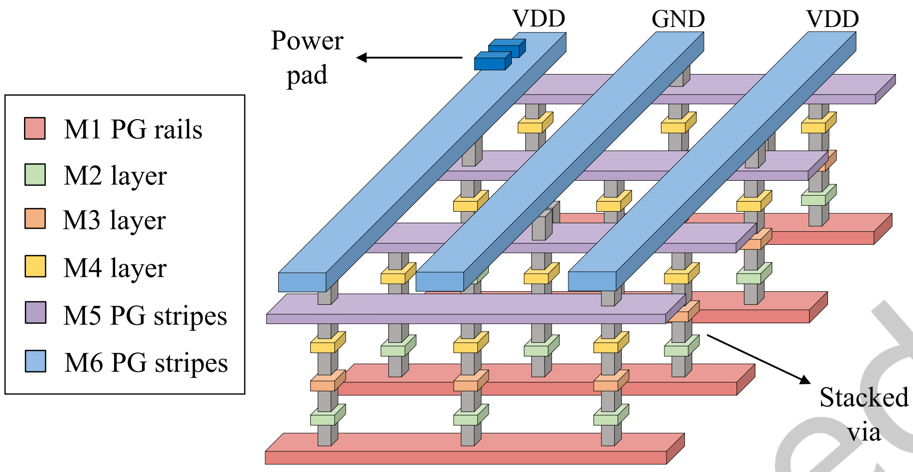


Fig. 1. The structure of a power ground network.

may compete for routing resources with signal nets. Take Figure 2 for example; initially, the orange pins can be connected with single straight wires. However, if there is a PG via between the pins, this net would require a detoured connection, reducing the routing resource. Thus, there are trade-offs between power integrity and signal routing resources [4, 14]. Increasing the density of PG wires may incur signal routing congestion, while decreasing the density of PG wires may cause severe IR drop violations.

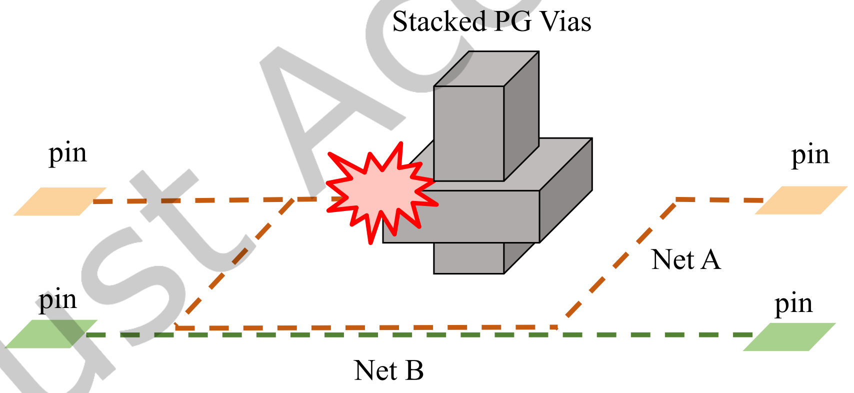


Fig. 2. The power/ground vias may act as obstacles for signal routing.

As technologies continue to scale down, the IR-drop problem becomes harder and harder to tackle. IR drop, or voltage drop, results from the resistance of the interconnect in the power grid. When a large current flows through the power ground network, the voltages of some regions might be lower than the ideal. There are two types of IR drop: static IR drop and dynamic IR drop. The average current calculates static IR drop in the power ground network. Dynamic IR drop considers the current waveform within clock cycles and the RC transient behavior in the power ground network [15]. Thus, the analysis of dynamic IR drop is more complicated than the static one. Excessive IR drop reduces circuit performance and might incur functional failures [15]. Moreover, as the wire width shrinks, the parasitic resistance increases as a result. Therefore, the IR-drop problem has become a critical issue for modern circuit designs. Usually, designers have to perform many IR drop analyses throughout different design stages to ensure the IR drop is within a predefined threshold. However, simulation-based IR-drop analysis is extremely time-consuming because the size of the conductance matrix increases quadratically to that of the power network [13]. We propose a machine-learning-based optimization framework to consider various objectives simultaneously for a fast yet sufficiently accurate estimation of the dynamic IR drop and routing congestion.

1.1 Previous Works

With the advent of the AI era, many researchers have applied machine learning to predict IR drop or other objectives related to the PG network. The machine-learning (ML) approach can provide a shorter estimation time and better scalability compared with the simulation-based approach. Their works usually include using a commercial analyzer to get golden IR drop values and feature engineering to get representative features such as current, resistance, and timing. Many works have adopted xgboost [5] or convolutional neural networks (CNN) as their machine learning models. Xgboost is a gradient-boosting algorithm that is scalable and computationally fast for various machine learning problems. Convolutional neural networks (CNN) consist of filters to capture two-dimensional features and succeed in various computer vision tasks.

Yamato *et al.* [30] were the first to build a linear regression model to predict IR drop to our best knowledge. They observed that cell power is highly correlated with IR drop, so they built a model for each cell to estimate IR drop based on its average power within a cycle. However, as the design size continues to scale, building a model for each cell is impractical and time-consuming. Ye *et al.* [31] proposed a support vector machine (SVM) method to predict the timing delay induced by IR drop. However, their approaches require additional hardware implementation. Furthermore, their approach focuses solely on the target cell, ignoring the impact of neighboring cells. Fang *et al.* [8] proposed a framework to predict and optimize dynamic IR drop to shorten ECO turnaround time. Their features are related to timing, power, and physical location. They used XGBoost [5] and CNN as their model, but they may have over-pessimistic prediction results after ECO changes. Xie *et al.* [29] proposed a transferable CNN-based dynamic IR drop analyzer. They sliced a layout into uniform grids and measured each grid's total power, serving as a feature map to CNN. They also divided a clock cycle into N timestamps, measuring the maximum IR drop of a tile from the feature maps in all the timestamps. The main drawback is that their method needs to process multiple power maps, which requires high computation cost. Huang *et al.* [11] proposed a dynamic IR drop prediction and optimization flow by considering every timestamp in the current waveform. Their approach includes training a random forest model after placement and an ILP-based algorithm to perform cell relocation. Their model achieves high accuracy, but their approach requires getting the data from each timestamp in the current waveform and ignores the routability affected by cell relocation. Some other existing works only target static IR drop. In Incpird [10], they use an xgboost model to predict static IR drop. Their proposed flow can handle different types of incremental changes, such as power distribution network (PDN) modification or block movement. However, their method is limited to static IR drop analysis.

Some other works discussed the relationship between the PG network and routability. Chang *et al.* [3] studied the optimal PG stripe width to reduce the wastage in the number of routing tracks. Chang *et al.* [4] proposed a machine learning-based model to predict routing cost. They used the Gaussian process regression model to predict the global routing wirelength with a given power distribution network (PDN) configuration. Lin *et al.* [14] proposed a linear programming algorithm to minimize the PG network area and simultaneously consider routability. Lin *et al.* [14] proposed a row-style PG network to partition a layout into several rows considering the pre-placed macros. They proposed a linear programming (LP) based algorithm to minimize the PG network area and simultaneously consider routability. Wang *et al.* [27] proposed a mix-signal routing algorithm to minimize

Table 1. ML works on different objectives of the PG networks

Methods	Stage	Model	Objectives
[31]	Post-route	SVM	IR-aware timing
[4]	Placement	Gaussian process regression	Global Route Wirelength
[8]	Sign-off	XGBoost & CNN	Dynamic IR drop
[10]	Placement	XGBoost	Static IR drop
[11]	Placement	Random Forest	Dynamic IR drop
[29]	CTS	CNN	Dynamic IR drop
Ours	Placement	CNN	Dynamic IR drop & routability

the total metal usage of a PG network. Their algorithm can handle multiple nets with different IR drop constraints simultaneously. They adopted a tree structure with tree splitting and tapering to route power nets. Table 1 summarizes the recent machine learning-based prediction works.

1.2 Differences from the Previous Works

To our best knowledge, most of the previous machine learning works predict or optimize IR drop after incremental changes . Not much work used machine learning to predict multiple objectives simultaneously for the PG network analysis and design. The routability affected by the incremental modifications in the PG network also requires time-consuming calculations. Still, not much work adopted machine learning to predict dynamic IR drop and routability after incremental changes. These objectives must be considered simultaneously because a PG network configuration will affect dynamic IR drop and routability. To solve the trade-off between dynamic IR drop and routability, we propose a machine learning model to predict the two targets simultaneously. Because the feature set of the two tasks may overlap, a multi-task learning scheme can share the machine learning parameters and further reduce the model size. By combining similar tasks, the knowledge learned from different tasks can be shared, enabling our model to generalize better on our tasks [23].

Both dynamic IR drop and routing congestion require time-consuming calculations, and making IR drop and routing congestion converge usually requires many iterations. We try to develop machine learning techniques for this problem. Machine learning models can guide at an early design stage and help to reduce the number of design iterations [28]. Furthermore, the task of predicting IR drop and routing congestion may have some information that can be shared. We could develop a multi-objective machine-learning model to reduce the training time compared to training two separate models. The model can provide a faster evaluation at a lower cost than a commercial tool and can guide designers on these two objectives simultaneously. It can enable designers to find possible IR drop or congestion hot spots with a lower runtime. Figure 3 shows the motivation for our work. The conventional approach usually requires many iterations to converge, and if routing fails, the cost of rolling back to the previous stages is very high. On the other hand, our machine-learning-based optimization can serve as early forecasting in the placement stage. The optimization approach could help save more routing resources with negligible IR drop overhead before entering the routing stage. This approach could reduce the number of iterations and save much runtime. Therefore, it is desirable to develop a multiple-objective machine-learning model and a multiple-objective optimization scheme.

1.3 Our Contributions

The contributions of our work are summarized as follows.

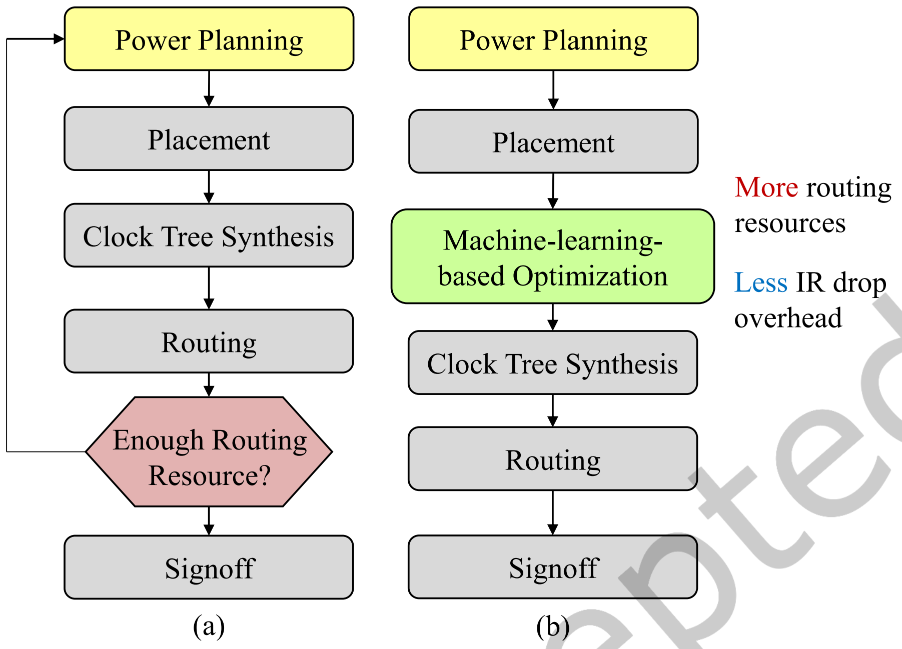


Fig. 3. Comparison between our proposed flow with the conventional design flow.

- We propose a machine-learning-based predictor by adopting multi-task learning. The predictor can simultaneously predict the worst dynamic IR drop and neighboring routing congestion of a cell. The model has been proven accurate, with a correlation coefficient of up to 0.99.
- We consider the spatial information of a cell instance by transforming its neighboring regions into images. To efficiently learn the features in neighboring images, we adopt transfer learning to reduce the CNN training time and handle the data scarcity.
- Our proposed flow can save more routing resources without much IR drop degradation. Experimental results show that our flow can reduce the number of overflows by about 12 % with less than 1% degradation on the worst dynamic IR drop value.
- Our proposed flow has good scalability. Compared to a state-of-the-art commercial IR drop tool analysis time, our method can achieve up to **29X** speedups.
- The benchmark circuits are all based on real industrial designs. Our proposed framework can easily be integrated into a state-of-the-art commercial tool to guide designers.

The remainder of the paper is organized as follows. Section 2 introduces the background knowledge of the dynamic IR drop and transfer learning and gives the problem formulation. Section 3 presents our proposed flow. Section 4 reports and analyzes the experimental results. Finally, Section 5 concludes this paper.

2 PRELIMINARIES

This section introduces dynamic IR drop and some background knowledge of machine learning and then gives the problem formulation of our multi-objective PG network optimization.

2.1 Dynamic IR Drop

IR drop results from the resistance of the interconnect in the power grid. It can be categorized into two types. Static IR drop is related to the power grid's robustness, which can be expressed as follows: $IR_{static} = I_{avg} \times R$, where I_{avg} is the average current, and R is the PG wire resistance. Static IR drop analysis is usually adopted in an early design stage when switching vectors are not available [28].

Dynamic IR drop, unlike static one, is dependent on the switching activity of a circuit [17]. The switching activity of a circuit would affect the peak current values. When a region of cells is simultaneously turned on, it would draw a large instantaneous current and cause a potential IR drop violation. Therefore, it is more difficult to handle than the static IR drop. Moreover, to get the worst dynamic IR drop, we need to consider the whole waveform of both the power and ground pins. The worst dynamic IR drop value does not equal the summation of the worst voltage at power and ground pins. See Figure 4 for an example. Summing the worst voltage at power and ground pins directly usually leads to an over-pessimistic estimation. Thus, we have to perform a simulation to consider every timestamp in a waveform, which makes analyzing dynamic IR drop more time-consuming.

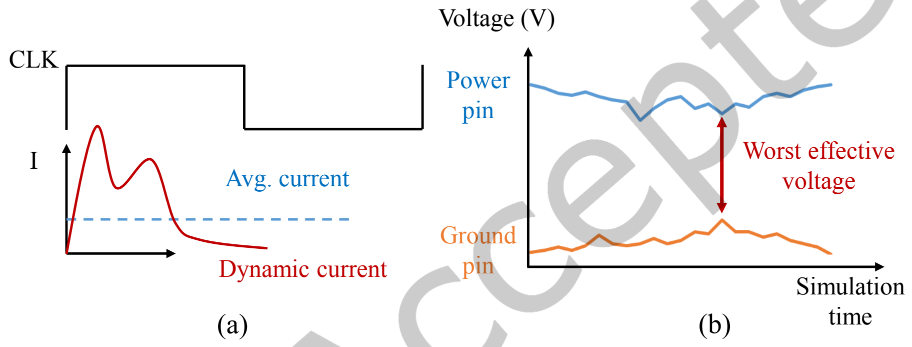


Fig. 4. An illustration of the dynamic IR drop. (a) Static IR drop vs. dynamic IR drop [17]. (b) Effective voltage of a given cell instance.

2.2 Transfer Learning

Transfer learning is a technique to apply knowledge learned from one related domain to another, which has great promise in various fields such as computer vision and speech recognition. Transfer learning aims to solve the problem of data scarcity or the task when collecting ground truth data is difficult. There are some ways to perform transfer learning, including instance-based, feature-based, and parameter-based transfer [20]. The parameter-based transfer is one of the popular approaches done by sharing parameters in pre-trained models. The pre-trained model could be trained in another related task but with more extensive data. Because the model performs well for extensive data, it should capture some general features well. We can then fine-tune the model parameters learned from the other tasks and prevent training from scratch. This technique can help save significant time and also avoid over-fitting.

2.3 Problem Formulation

Here, we divide the original multi-objective PG network optimization problem into two subproblems: (1) multi-objective prediction problem and (2) multi-objective PG network optimization problem. We formulate the two problems as follows:

Problem 1 (**The Multi-Objective Prediction Problem**). *Given a design at the placement stage and the IR drop analysis and routing congestion report, generate a machine learning model that can accurately predict the dynamic IR drop and routing congestion of each cell.*

Problem 2 (**The Multi-Objective PG Network Optimization Problem**). *Given a design at the placement stage, generate a set of PG vias/wires to be modified to minimize the number of routing congestion overflows, the number of IR drop violations, and the peak dynamic IR drop while satisfying the design rules and timing constraint.*

3 PROPOSED FLOW

This section gives an overview of our proposed flow and details. Figure 5 shows our proposed flow. Our flow consists of two main stages, data prepossessing for machine learning modeling and machine-learning-based optimization for dynamic IR drop and routing congestion minimization.

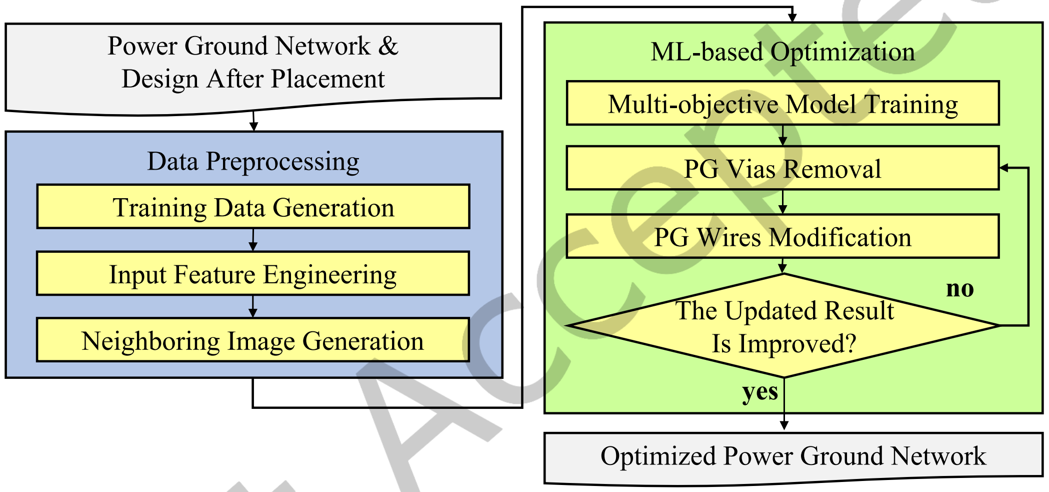


Fig. 5. The overview of our proposed flow.

3.1 Training Data Generation

To obtain an accurate model, we need to generate sufficient training data and extract related representative features. Given a design, a PG network, and a placed design, we collect data from this block and further perform training. The PG network has power rails in M1, and some larger blocks will also have power rails in M2. There are power meshes in the top two or three metal layers, and the width and spacing of the strips on each layer are set uniformly. The placement results are generated by *Synopsys IC Compiler II* [25], and the utilization rate is set between 0.6 and 0.7. After the placement results are obtained, we then use a commercial tool to perform rail and congestion analyses to get the golden values for IR drop and routing congestion. Then, we can obtain N_{cell} training data, where N_{cell} is the number of cell instances.

3.1.1 Golden IR drop. The golden dynamic IR drop on each cell instance is retrieved from the *ANSYS RedHawk* [2], an industrial golden IR drop analyzer. The IR drop simulation has two types, vector-based and vector-less, depending on whether the switching activity files are available. Our framework can support both types of IR drop simulation. In this work, we set the timestamp of simulation to 10 *ps* and the number of timestamps to 2000. For each design, we launch one dynamic IR drop analysis. After simulation, we can get the waveform at the power and ground pin to get the worst effective voltage drop on each cell instance. We can also get the static IR drop of PG via and wire after the simulation. We dump the IR drop value of PG vias and PG wires by using the commercial tool. These information will further be used in Section 3.5 and Section 3.6 for PG network optimization.

3.1.2 Bounding Box Anchoring. Because both dynamic IR drop and routing congestion are affected by neighboring cells, we further use the bounding box to perform feature engineering. Some features like density values are also evaluated based on this bounding box. The width and height of the bounding box are set to be integer multiples of a single-row height h_{cell}. We will experiment with various bounding box sizes in Section 4.3.2. In this work, we set the bounding box as square, and the bounding box's height h_{box} and width w_{box} are calculated as follows:

$$h_{box} = w_{box} = h_{cell} \times \lceil \sqrt{N_{row}} \rceil, \tag{1}$$

where N_{row} is the number of site rows in a design.

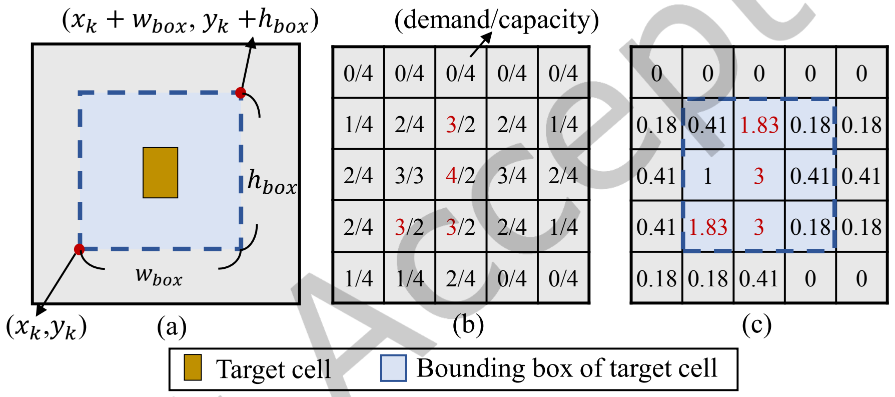


Fig. 6. A simple example of training data generation. (a) Select a cell as the target cell and use a larger bounding box to include neighboring features. (b) A congestion map generated by Synopsys' IC Compiler II. (c) Calculate the average congestion score within the bounding box.

3.1.3 Golden Congestion Score. We use the congestion map generated by IC Compiler II [25] as the golden indicator of the signal net routing congestion. Figure 6(b) gives an example congestion map, where a layout is divided into uniform grids, called *gcell*. Each gcell has its demand and capacity value. In this work, we set the size of a gcell equal to the standard cell height h_{cell}. To better express different congestion levels in each gcell, we define the *congestion score* $S_g(i, j)$ in the (i^{th}, j^{th}) grid $g_{i,j}$, calculated as follows:

$$S_g(i, j) = \sum_{k=1}^{L} \left(2^{\frac{dem(i,j,k)}{cap(i,j,k)}} - 1 \right), \tag{2}$$

where L is the number of layers and $dem(i, j, k)$ and $cap(i, j, k)$ are the demand and capacity of the (i^{th}, j^{th}) grid in the k^{th} layer, respectively. The *congestion score* $S_c(k)$ of the k_{th} cell instance can be further calculated as follows:

$$S_c(k) = \frac{1}{N_{gcell}} \sum_{i=x_k}^{x_k+w_{box}} \sum_{j=y_k}^{y_k+h_{box}} S_g(i,j), \tag{3}$$

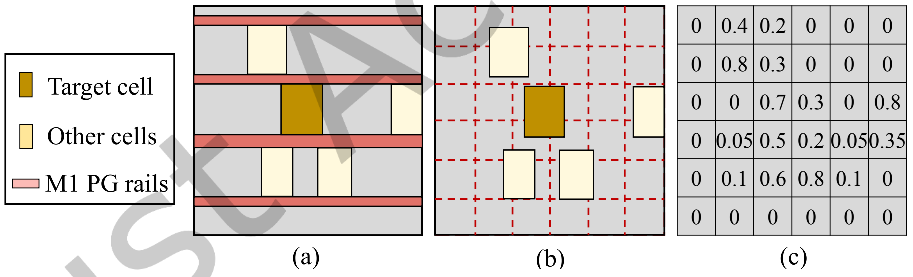
where N_{gcell} is the number of gcells within a bounding box. We normalize the congestion score of a cell by dividing $S_c(k)$ with N_{gcell}.

3.2 Input Feature Engineering

Feature extraction is to extract useful information from the raw data. Table 2 summarizes our input features. It is worth noting that, unlike most of the previous works, our input features do not include the effective resistance from the PG pins to the power pads. Our process can avoid additional calculation time in analyzing the effective resistance. Experimental results show that our model is still accurate even without the feature.

3.2.1 Cell-based & Pin-based Features. For the cell-based features, we consider the attribute of a given cell, such as cell type, location, area, and power. We also collect the number of pins N_{pin} and the number of fanout cells N_{fanout} since it would be helpful to predict routing congestion. For the pin-based features, we consider the behavior of the signal and PG pins in a cell. The transition time and capacity, such as t_{in}^{max}, t_{out}^{max}, C_{in}^{max}, and C_{out}^{max}, are extracted by a static timing analysis report. I_{peak} and P_{total} can be extracted from a *RedHawk* analysis report.

3.2.2 Local Features. To calculate local density, we first slice a layout into grids. We set the grid size to 1 μm and use the occupied ratio as the grid's value. When calculating the neighboring cell density of a target cell, we use the bounding box mentioned before and get the average density value within the bounding box. Figure 7 shows an example of neighboring cell density computation. The neighboring pin density and macro density can be calculated likewise.



0	0.4	0.2	0	0	0
0	0.8	0.3	0	0	0
0	0	0.7	0.3	0	0.8
0	0.05	0.5	0.2	0.05	0.35
0	0.1	0.6	0.8	0.1	0
0	0	0	0	0	0

Fig. 7. The flow of calculating neighboring cell density. (a) Select a cell as the target cell. (b) Slice a layout into grids and calculate the occupied ratio by each cell instance. (c) Get the density values in each grid.

3.2.3 Global Features. Since we do not use effective resistance or the minimum path resistance from power pads to cell instances as our feature, we use the distance to power pads instead. The distance of a cell to a power pad is calculated by Manhattan distance. We collect the average distance to power pads and maximum distance to power pads for each cell. In each design, there could be multiple power pads. We use (x_i, y_i) to denote the coordinate of the ith cell, and (x_k^p, y_k^p) to denote the coordinate of the kth power pad. The distance from the ith cell to the kth power pads can be calculated by the Manhattan distance as follows:

$$d(i,k) = |x_i - x_k^p| + |y_i - y_k^p|. \tag{4}$$

Table 2. The four types of features used for prediction

	Features	Description
Cell-based features	*Cell type*	Sequential or combinational
	x,y	Cell coordinates
	Area	Cell area
	P_{total}	Total power of a cell
	N_{pin}	Number of pins in a cell
	N_{fanout}	Number of fanout cells connected to the cell
Pin-based features	I_{peak}	Peak current at power/ground pin
	t_{in}^{max}	Maximum input transition time at signal pin
	t_{out}^{max}	Maximum output transition time at signal pin
	C_{in}^{max}	Maximum input capacitance at signal pin
	C_{out}^{max}	Maximum output capacitance at signal pin
Local features	D_{pin}	Neighboring pin density
	D_{cell}	Neighboring cell density
	D_{macro}	Neighboring macro density
Global features	Pad^{avg}	Average distance to power pad
	Pad^{max}	Maximum distance from the cell to power pad

The maximum distance from the i^{th} cell to the power pads can be calculated as follows:

$$pad_{max}^{i} = \max_{k} d(i,k). \tag{5}$$

The average distance from the i^{th} cell to the power pads can be calculated as follows:

$$pad_{avg}^{i} = \sum_{k} \frac{d(i,k)}{|N_{pad}|}, \tag{6}$$

where $|N_{pad}|$ is the number of power/ground pads. These features can roughly represent the resistance from the cell to the power pads.

3.3 Neighboring Image Generation

Besides the 17 scalar features extracted from raw data, we also generate the via & wire images in each cell's neighboring area. Because our objective is to perform incremental changes to PG vias & wires, we need to model this feature carefully. We consider using CNN to model this feature for its capability to recognize spatial information. Because many of the widely-adopted CNN models are designed for computer vision tasks, these models usually take an image as input, and the input layer is limited to three channels. Therefore, we need to transform a layout into a three-channel image to leverage transfer learning to perform feature extraction.

We first calculate the PG vias & wires density value according to the method used in Section 3.2.2. For all metal layers in the layout, we divide them into three channels. The lower channel is from M1 to M3, most of the signal net is routed here, and the other layers are divided into middle and upper channels. The grid value in each channel is calculated by averaging the density values from the corresponding layers. Then, we re-scale the value of each grid into $[0, 255]$ so these channels can be transformed into the three channels in an RGB image. These images are further adjusted to $64 \times 64 \times 3$ pixels and serve as feature maps to reflect the adjacent PG network topology near the target unit. The reason for resizing into a smaller picture dimension is that we want a trade-off between the extraction runtime and image precision. A larger dimension usually means more operation in the neural network and longer extraction runtime. From the experimental results, we believe the size of $64 \times 64 \times 3$ is sufficient to obtain an accurate prediction. Figure 8 shows the conversion from adjacent density values to images.

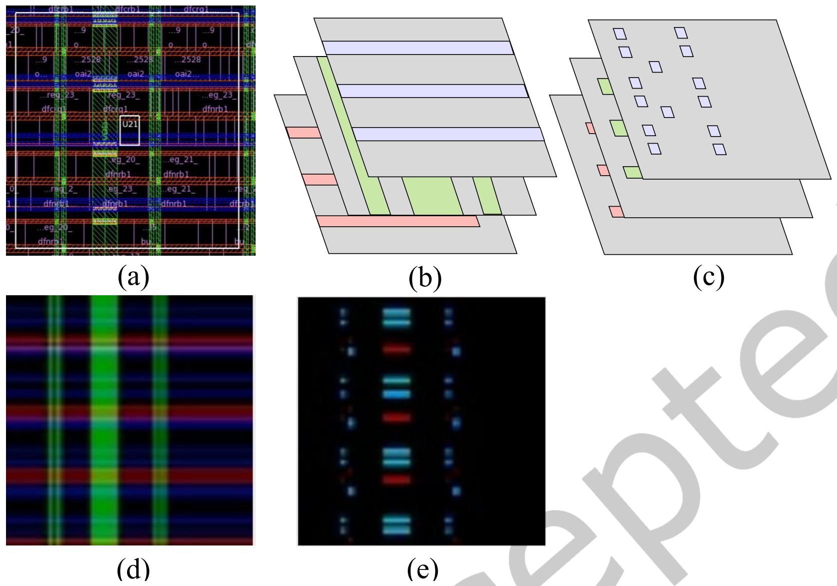


Fig. 8. A step-by-step example of neighboring image generation. (a) A snapshot of the target cell in ICC2. (b)(c) Distribute the PG vias & wires density values into three channels. (d)(e) The result of the generated neighboring images of a cell.

3.4 Multi-objective Model Training

Our training flow consists of two parts: (1) the pre-trained model to extract features and (2) the multi-task learning predictor training scheme.

3.4.1 Feature Extractor. Since our feature includes two-dimensional spatial images and other one-dimensional features, we need to preprocess these images to a size similar to one-dimensional ones. We adopt *EfficientNet-B0* [26] as our feature extractor. EfficientNet is a set of state-of-the-art models known for its efficiency in model size and high accuracy. We load the pre-trained model weights trained in the *ImageNet* [7], a large-scale image dataset. The EfficientNet-B0 originally takes an image of variable resolution in $(32 \times k, 32 \times k, 3)$, where k is in the range of $[1, 7]$. After passing many convolutional and pooling layers, the resulting feature map is in the shape of $(k, k, 1280)$, Finally, a Dense layer turns the feature map into the prediction of the 1000 *ImageNet* classes.

We remove the Dense layer and append a *GlobalMaxPooling* layer on the feature map for our transfer learning scheme. Therefore, the extracted output from the PG via and wire images will become the shape of 1280. The weights in the extractor are set to freeze throughout the training process. In this way, we can only allow updating weights in the backbone model, which can make the training process converge quickly. The whole model structure is shown in Figure 9. The neighboring images are passed through the feature extractor to obtain information and reduce the dimension. Afterward, the extracted output features can then be concatenated with other one-dimensional features. It should be noted that the feature extractor would only be called once rather than once per epoch of training. In this way, we can save much training time compared to training a customized CNN model from scratch.

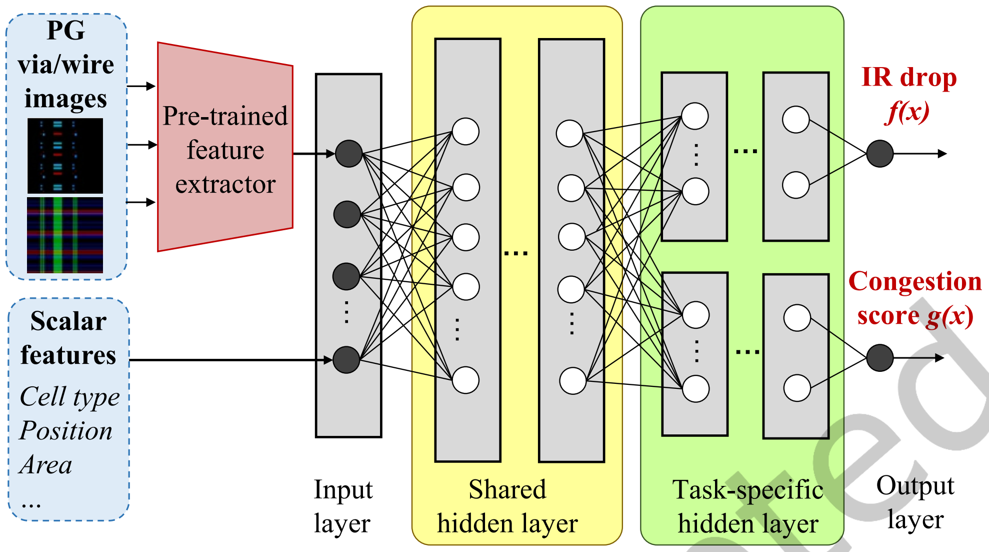


Fig. 9. The structure of our machine learning predictor.

3.4.2 Model Parameter. Because there are some similarities between our two objectives, we adopt the multi-task learning scheme. We perform this learning by sharing the hidden layers in the first few layers and predicting each objective with task-specific layers. In this way, we can save more training time and gain more generalization. There are five shared hidden layers and four task-specific layers in our model. We use the *Keras* [6] model plotting utilities to visualize the structure of our backbone model. The detail of the backbone model structure is shown in Figure 10. Each layer takes a specific dimension of input and outputs a specific shape tuple. The name *Dense* means the layer is a fully-connected layer. The *None* in dimension refers to the batch dimension, and it simply means that the layer can accept input of any size. This dimension does not affect the size of the network. It means the model can take an arbitrary number of samples as a batch during testing. The activation function we used is *Rectified Linear Unit (ReLU)*, and we adopt *mean square error* as our loss function. Suppose that the golden dynamic IR drop and golden congestion score of the i^{th} cell are $V_c(i)$ and $S_c(i)$, respectively. The goal of the model is to minimize errors between predicted values and golden values. Given a set of parameters θ in the model, the cost function $L(\theta)$ can be formulated as follows:

$$L(\theta) = \alpha \times \sum_{i=1}^{N_{cell}} (V_c(i) - f(x_i))^2 + \beta \times \sum_{i=1}^{N_{cell}} (S_c(i) - g(x_i))^2, \tag{7}$$

where x_i is the feature vector of the i^{th} cell instance, and $f(x_i)$ and $g(x_i)$ are the predicted values of dynamic IR drop and congestion score, respectively. In the multi-task learning scheme, we need to normalize the loss in IR drop and the loss in congestion score as they can be in a similar dimension. We need to balance these two kinds of losses so that the neural network will not preferably update only one objective during back-propagation. The parameters α and β are used to normalize between different costs, and we set $\alpha + \beta = 1$. which are given as follows:

$$\alpha = \frac{\hat{S}_c^2}{\hat{V}_c^2 + \hat{S}_c^2}, \beta = \frac{\hat{V}_c^2}{\hat{V}_c^2 + \hat{S}_c^2}, \tag{8}$$

$\hat{S}_c$ and $\hat{V}_c$ are average congestion scores and IR drop values calculated from the N_{cell} cell instances, respectively. The learning rate is set to 0.001, and we adopt *Adam* [12] as our optimizer. The maximum training epoch is set to 100, and we adopted early stopping with patience set to 5.

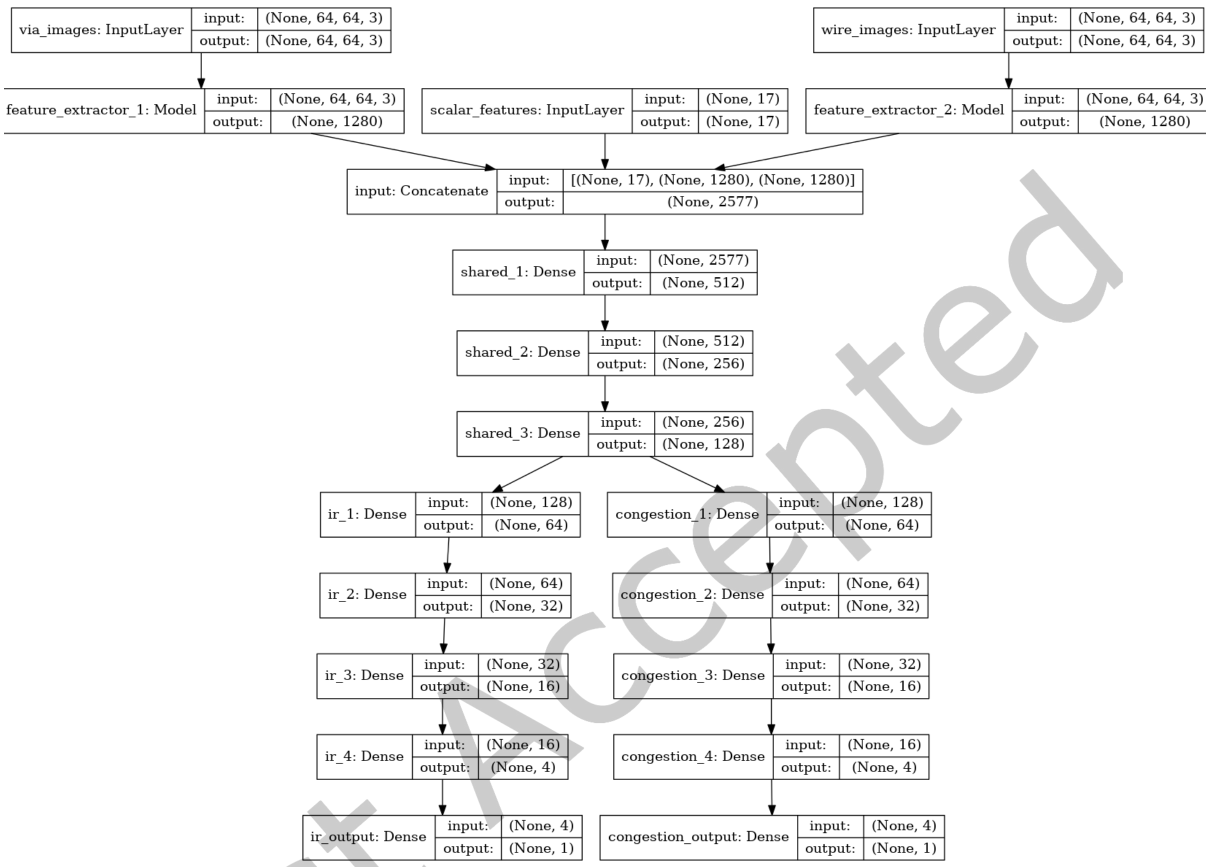

via_images: InputLayer | input: (None, 64, 64, 3) | output: (None, 64, 64, 3)
wire_images: InputLayer | input: (None, 64, 64, 3) | output: (None, 64, 64, 3)
feature_extractor_1: Model | input: (None, 64, 64, 3) | output: (None, 1280)
scalar_features: InputLayer | input: (None, 17) | output: (None, 17)
feature_extractor_2: Model | input: (None, 64, 64, 3) | output: (None, 1280)
input: Concatenate | input: [(None, 17), (None, 1280), (None, 1280)] | output: (None, 2577)
shared_1: Dense | input: (None, 2577) | output: (None, 512)
shared_2: Dense | input: (None, 512) | output: (None, 256)
shared_3: Dense | input: (None, 256) | output: (None, 128)
ir_1: Dense | input: (None, 128) | output: (None, 64)
congestion_1: Dense | input: (None, 128) | output: (None, 64)
ir_2: Dense | input: (None, 64) | output: (None, 32)
congestion_2: Dense | input: (None, 64) | output: (None, 32)
ir_3: Dense | input: (None, 32) | output: (None, 16)
congestion_3: Dense | input: (None, 32) | output: (None, 16)
ir_4: Dense | input: (None, 16) | output: (None, 4)
congestion_4: Dense | input: (None, 16) | output: (None, 4)
ir_output: Dense | input: (None, 4) | output: (None, 1)
congestion_output: Dense | input: (None, 4) | output: (None, 1)

Fig. 10. The detailed structure of our whole model.

3.5 PG Vias Removal

After obtaining an accurate model, we need a strategy to select a set of candidate PG vias to be removed. We perform a profiling analysis about all kinds of PG vias before selecting vias to be removed. We observe that the PG vias in the upper layers, especially connecting PG meshes, have a substantial impact on the voltage and a small impact on routing congestion. It is because the upper-layer metal width is usually wider than that in the lower one, thus more sensitive to signal integrity. Therefore, in this work, we only select the PG vias below the PG meshes and within the core area in the design. We target the stacked vias to be removed as it may have a smaller impact on the voltage and can help improve routing congestion. The stacked via in Figure 1 shows that it would reduce routing resources by occupying routing tracks. We propose a two-stage approach: PG vias clustering followed by PG vias selection to remove PG vias. The updated design will be evaluated by the model trained in Section 3.4 to verify whether the change can reduce routing congestion.

Algorithm 1 Machine-learning-based PG Vias Removal Flow

```
1:  $fail\_times = 0, total\_num = 0, ir\_max = 0, cong\_avg = 0$;
2:  while $fail\_times < max\_fail$ and $total\_num < max\_num$
3:    for $k \leftarrow 1$ to $K$
4:      Select a not selected via $v_k$ with the highest congestion score from the cluster $C_k$;
5:      Set via $v_k$ as selected;
6:      Remove via $v_k$ and update features;
7:    for $i \leftarrow 1$ to $N_{cell}$
8:      Update the feature set in the $i^{th}$ cell
9:    Call model to get IR drop and congestion score of each cell;
10:   Calculate the top-10% average IR drop $m$ and average congestion score $n$;
11:   if $m \le ir\_max + \epsilon$ and $n \le cong\_avg - \delta$
12:     $ir\_max = m$, $cong\_avg = n$;
13:     $total\_num$ += N;
14:   else
15:     Undo changes from $v_i$ to $v_k$;
16:     $fail\_times$ ++;
```

3.5.1 PG Vias Clustering. Because the original solution space is too large, we need to develop an efficient way to select PG vias candidates. First, we combine all connected stacked PG vias into a group. The PG vias on different metal layers having the same x and y coordinate are the so-called stacked PG vias. For example, in Figure 1, there are stacked PG vias connecting from M1 to M5. We further group these PG vias together. If one via in a group is removed, so are other vias in this group. Supposed that the lower layer of the stacked vias is L_{lower} and the higher one is L_{higher}. We only select the PG vias with $L_{higher} - L_{lower} \geq 2$, i.e., the stacked vias must at least across two layers so that these vias occupy some routing tracks. Second, we avoid choosing PG via array as candidates because removing them may induce severe IR drop violation. Finally, we adopt and modify the well-known K-means algorithm for the rest of the PG vias to cluster them. The K-mean algorithm tries to find k centroid to cluster data points into k groups so that each data belongs to the cluster with the nearest centroid. Originally, the distance was measured by the squared Euclidean distance. In our work, we want to cluster vias with similar coordinates and with similar IR drop values together. Therefore we set the data of via in the vector of $[x_v^i, y_v^i, V_v(i)]$, where x_v^i, y_v^i are the coordinated and $V_v(i)$ is the worst dynamic IR drop value of the i^{th} via. Then, the modified distance of any two PG vias v_i, v_j are defined as follows:

$$d = (x_v^i - x_v^j)^2 + (y_v^i - y_v^j)^2 + (V_v(i) - V_v(j))^2, \tag{9}$$

By setting a predefined parameter k, we can cluster the PG vias into k groups. The PG vias in each cluster have a similar location and IR drop value. In this work, The number of clusters k is determined by dividing the chip core area by the bounding box area $h_{box} \times w_{box}$. After clustering PG vias, we calculate each group's average dynamic IR drop value and sort it in non-increasing order. The PG vias in groups with the top 10% IR drop value would be skipped in the following procedure.

3.5.2 PG Vias Candidate Selection for Removal. Our process of selecting PG candidates involves multiple iterations. In each iteration, we select a via with the highest congestion score from each cluster and then delete it from the PG density map. Then, the cell instance near this PG via needs to update its neighboring PG via image. Afterward, we use the model to query the dynamic IR drop values and congestion scores of all cell instances. We set a maximum tolerance threshold ϵ for IR drop and a minimum improvement threshold δ for congestion score. We record the average congestion score of all cells and the average IR drop of the cells with the top 10% IR drop and compare them with the original ones. If the degradation of the updated result exceeds the predefined tolerance threshold ϵ, or the improvement is less than the minimum improvement threshold δ, we undo the changes and mark the PG vias in the last iteration as *selected*. In this work, we set the maximum vias to be removed max_num as of the number of total PG vias $|V_{pg}|$, and the maximum failure times max_fail as 10. The whole process terminates when it reaches the maximum failure times or the removal count exceeds the limit. Algorithm 1 gives the pseudo-code of our PG vias removal process.

3.6 PG Wires Modification

In most designs, the power stripe metal width is uniform and much larger than the min-width design rule. This conservative design strategy may waste routing resources in regions where IR drop is not critical. We can reduce wire width according to the design rule and use the model to guide optimization. Among the PG wires in all layers, we exclude the PG rails and PG meshes as our candidates. The PG meshes are in higher layers, where routing congestion is usually not severe, while the PG wires in the intermediate layers have the most significant probability of competing for routing resources. Then, we calculate the area occupied by PG wires in each layer. If one layer has a total percentage over 15%, we will select PG wires from that layer.

The whole PG wire modification process is similar to PG via removal in Algorithm 1. First, we sort the PG wires according to their worst dynamic IR drop values in non-increasing order. In each iteration, we select a candidate wire with the highest congestion score from each intermediate layer. Then, the selected candidate wires are shrunk to min-width, and the PG wire density map is updated accordingly. Next, we use the model to query the dynamic IR drop value and congestion score value of all cell instances. For the updated design, we check whether the degradation in IR drop exceeds the maximum tolerance threshold ϵ and the improvement in congestion score is larger than the minimum improvement threshold δ. The whole process terminates when it reaches the maximum failure times or the removal count exceeds the limit.

4 EXPERIMENTAL RESULTS

In this section, we introduce the setting and the benchmarks. Then, we conduct several experiments to show that our trained model is accurate in predicting two different tasks. Then, we perform experiments to show that our optimization scheme is effective and has good scalability.

4.1 Settings & Benchmarks

We implemented our flow in the C++ and Python programming languages on an Intel Xeon CPU E5-2640 2.40GHz workstation with 256 GB RAM and 2 Tesla P100 GPU cards. The data generation, feature engineering, and image generation parts were implemented in C++ for better efficiency. The machine learning parts, such as model training and inference parts, were implemented in Python. We adopted *Keras* [6] and tensorflow [1] for the machine learning libraries. The learning process was done in an on-the-fly manner. We first retrieved golden data from a commercial tool and then performed our training and optimization process. We split 80% of the cell instances as training set and 20% as testing set in each design. The testing set is unseen throughout the training process and is used to evaluate the accuracy of our proposed model. Each design would have a separate model, and we used this model to predict the design after changing the PG network. We also conduct experiments on different placements of the same design to show the generality of our proposed model.

Table 3 gives the characteristics of the benchmark circuits in our experiment. We tested our algorithm on six industrial and four open-source designs (ibex [16], aes [18], jpeg [19], dynamic_node [22]). The industrial designs vary from different scales and technology nodes. Design5 and Design6 have more complex PG network structures. Besides the PG network in an ordinary design, these two designs also have extra PG augmentation wires. The open-source designs were synthesized using *Synopsys Design Compiler [24]* and used the *Synopsys* 32/28*nm Educational Design Kit* [9] as the standard cell library. Then, we used *Synopsys IC Compiler II* [25] for the physical design process. The utilization rate was set to 0.7, and we set the PG meshes in metal6 and metal9 with the width 2 µm and pitch 80 µm. In this table, "$|C|$" gives the total number of cells, "$|N|$" gives the number of signal nets, "$|V_{pg}|$" and "$|W_{pg}|$" give the number of PG vias and PG wires, respectively, "$|L|$" gives the number of routing layers, VDD gives the supply voltage of the design, and $Avg.IRdrop$ is the average worst dynamic IR drop calculated from all cell instances.

Table 3. B

enchmark statistics of the six industrial and four open-source designs.

Design	$\lvert C\rvert$	$\lvert N\rvert$	$\lvert V_{pg}\rvert$	$\lvert W_{pg}\rvert$	$\lvert L\rvert$	Tech node	VDD (V)	Avg. IR drop (mV)
Design1	2418	3027	6084	192	5	65nm	2.5	15.9
Design2	11688	16781	40837	10296	9	32nm	0.95	86.7
Design3	157194	190503	691687	110539	13	7nm	0.76	78.8
Design4	172188	224403	763706	107488	9	16nm	0.8	93.1
Design5	421921	828895	669876	222387	8	40nm	1	47.7
Design6	1579995	1592549	15923272	1724655	13	16nm	0.76	24.6
aes	13550	14384	7984	196	9	32 nm	0.95	30.2
dynamic_node	9665	21421	7752	192	9	32 nm	0.95	11.1
ibex	12781	21045	8796	210	9	32 nm	0.95	9.3
jpeg	52263	83069	26768	404	9	32 nm	0.95	22.2

4.2 Accuracy of the Proposed Model

The first experiment was to verify the accuracy of our proposed model. We analyzed the accuracy before changing the PG network. The evaluation metrics are listed as follows, where y_i is the ground truth value of the i^{th} data, x_i is the predicted value of the i^{th} data, and $\hat{y}$ is the average value on all the ground truth data:

- Mean absolute error (MAE)

$$MAE = \frac{1}{N_{cell}} \sum_{i=1}^{N_{cell}} |y_i - x_i| \tag{10}$$

- Mean absolute percentage error (MAPE)

$$MAPE = \frac{1}{N_{cell}} \sum_{i=1}^{N_{cell}} \frac{|y_i - x_i|}{y_i} \times 100\% \tag{11}$$

- Maximum absolute error (MaxE)

$$MaxE = \max_{1 \le i \le N_{cell}} |y_i - x_i| \tag{12}$$

- Root mean squared error (RMSE)

$$RMSE = \sqrt{\frac{1}{N_{cell}} \sum_{i=1}^{N_{cell}} (y_i - x_i)^2} \tag{13}$$

- Normalize root mean squared error (NRMSE)

$$NRMSE = \frac{RMSE}{\hat{y}} \tag{14}$$

- Correlation coefficient (CC)

$$CC = \frac{\sum_{i=1}^{N_{cell}} (x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{N_{cell}} (y_i - \hat{y})^2 \sum_{i=1}^{N_{cell}} (x_i - \hat{x})^2}} \tag{15}$$

4.2.1 Model Accuracy Before Changing PG network. Table 4 summarizes the accuracy of our proposed in the above evaluation metrics. The correlation coefficients (CC) of the worst IR drop and the congestion score achieve up to 0.99, indicating that our model accurately predicts the two objectives. The unit of mean absolute error (MAE) of IR drop prediction is mV. Because the magnitude of IR drop and congestion may differ in each design, we also show the mean absolute percentage error (MAPE) in Table 4 to better compare the prediction results between different designs. The MAPE of the IR drop and the congestion score is about 4% compared to the ground truth value. Figure 11 shows the scatter plots of our result where the x-axis gives the golden value, and the y-axis gives the predicted value. We show the plots for Design1, Design2,.., Design6, as these designs vary from different scales and technology nodes. The results of *aes*, *dynamic_node*, *ibex*, *jpeg* are similar to these results. The blue dots give the worst dynamic IR drop prediction results, and the green dots are for the congestion score ones. The red line is the ideal situation where all prediction results are equal to the ground truth, and our goal is to make more dots close to the red line as possible. Although there are some outliers, most predicted values are close to the golden one, and the predicted values and the ground truth are highly correlated.

Table 4. Model accuracy before changing PG network

Benchmark	IR drop					Congestion score				
	MAE (mV)	MAPE (%)	MaxE (mV)	RMSE	CC	MAE	MAPE (%)	MaxE	RMSE	CC
Design1	0.75	4.56	6.80	1.07	0.992	3.17	3.87	21.80	4.03	0.985
Design2	5.72	6.74	96.83	10.67	0.994	3.34	7.60	32.37	4.38	0.997
Design3	1.06	1.51	14.09	1.43	0.999	0.22	1.21	1.92	0.28	0.999
Design4	1.39	2.01	18.80	2.03	0.997	0.46	1.20	3.74	0.59	0.999
Design5	0.37	0.76	54.83	0.70	0.981	1.20	1.35	43.63	1.59	0.998
Design6	0.69	4.53	21.36	0.98	0.998	1.84	1.82	23.94	2.34	0.996
aes	1.68	5.42	19.99	2.60	0.964	4.88	2.85	30.08	6.15	0.993
dynmaic node	0.70	6.30	4.87	1.00	0.971	4.70	4.69	22.54	6.07	0.990
ibex	0.90	8.93	14.61	1.52	0.952	8.63	6.24	55.51	11.46	0.982
jpeg	0.65	3.17	10.92	0.96	0.995	1.18	1.84	20.00	1.57	0.996
Avg.	**1.39**	**4.39**	**26.31**	**2.30**	**0.984**	**2.96**	**3.27**	**25.55**	**3.85**	**0.994**

Figure 12 shows the histogram of error distribution compared to the ground truth. The result shows that most predicted IR drop and congestion score values are close to the ground truth. Taking 5% error as a threshold, the predicted IR drop within the threshold in design4, design5, and design6 are 95.0%, 99.2%, and 92.8%, respectively. On the other hand, the predicted congestion score within the threshold in design4, design5, and design6 are 96.8%, 96.1%, and 98.7%, respectively. The error distribution between truth and prediction is close to a normal distribution, with the mean μ very close to 0. We also calculated the standard deviation σ of the error between ground truth and prediction. The σ of IR drop error distribution from design (1) to design (6) are 1.01, 10.42, 1.38, 2.03, 0.69, and 0.97, respectively. The σ of congestion scores error distribution from design (1) to design (6) are

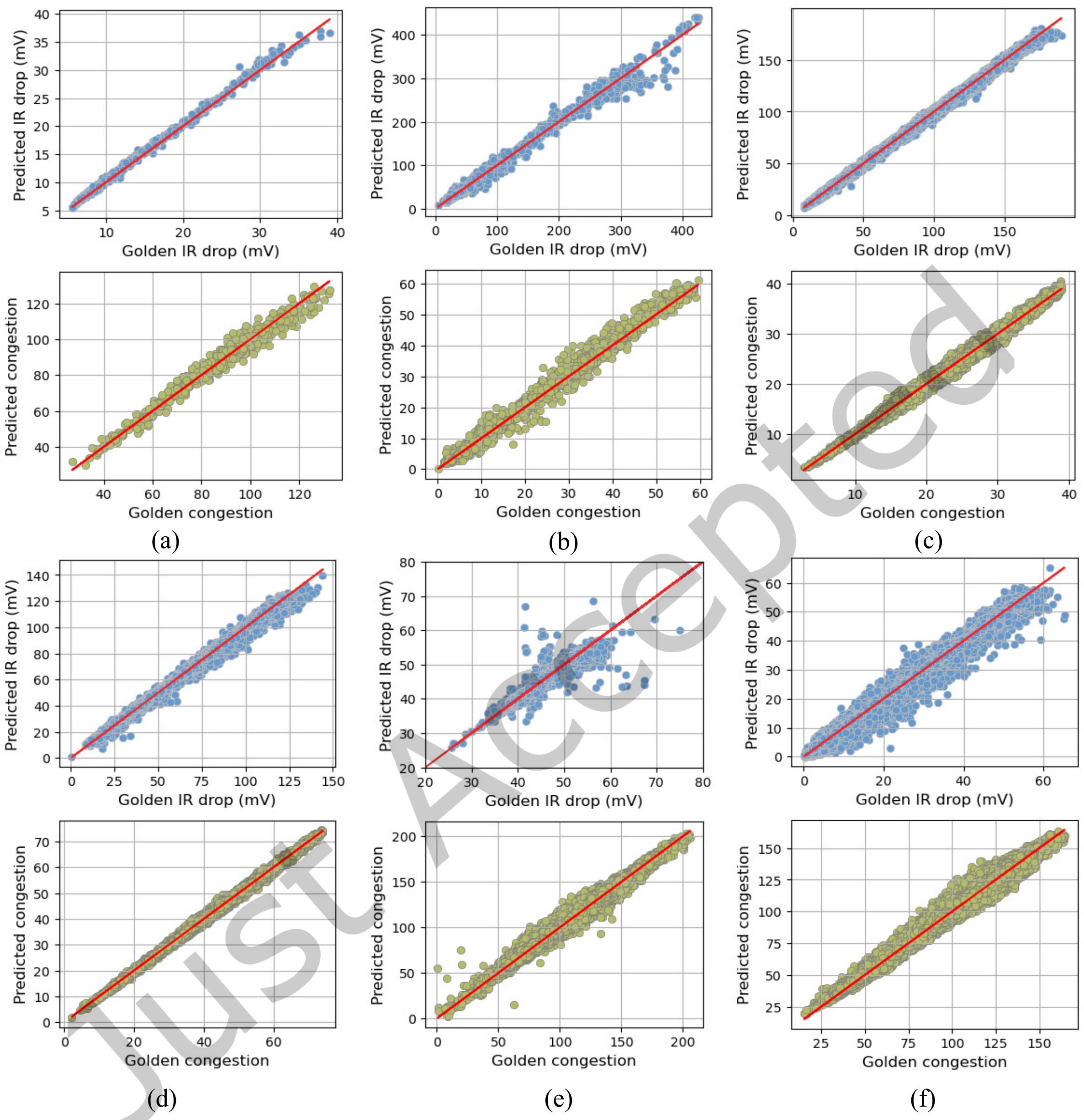


Fig. 11. The accuracy of the model before Changing the PG network. (a)–(f) The scatter plots from Design 1 to Design 6

3.82, 3.92, 0.28, 0.58, 1.55, and 2.03, respectively. We found that though the MaxE values are very large compared to the MAE in Table 4, these error predictions are at least 3 times σ away from μ. Therefore, we can conclude that these extreme outliers do not frequently happen in our prediction model. We then further analyzed the top 10 predictions with the most significant error. We found that the outliers of IR drop prediction are more likely to be underestimated, especially those close to the IR drop hotspot regions. On the other hand, the congestion score predictions tend to overestimate. To prevent these outliers from harming the optimization process, for each pass in the PG vias and wires removal flow, we took the cells with the top 10% IR drop or congestion for comparison.

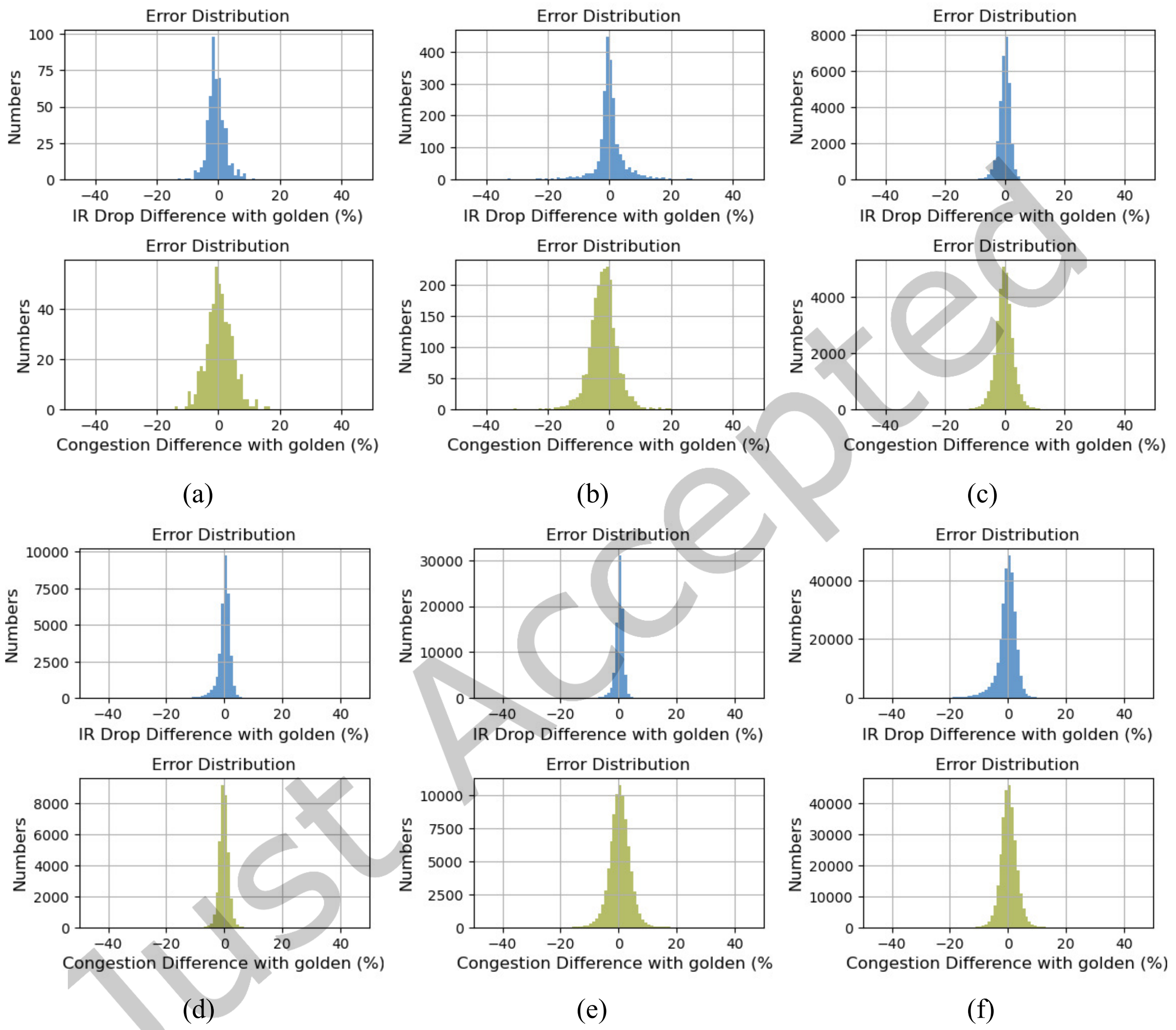


Fig. 12. The error distribution of the model. (a)–(f) The error distribution from Design 1 to Design 6

Most of the previous works focus solely on predicting IR drop, and few works try to predict as many objectives as we did. Therefore, we used the models adopted in the previous works as the baseline. These regression models usually take one-dimensional inputs, so we reshape the neighboring via and wires into one dimension. Then, these features are further concatenated with other scalar features and serve as input. We built the model of linear SVM by [21] and the XGBoost model by [5]. We also re-implemented the method in the state-of-the-art work [11] to compare the accuracy of dynamic IR drop predictions. We show the result of the six designs on average and use normalized root mean square error (NRMSE) as evaluation metrics. Table 5 shows the results of our model and the other baseline approaches, where [11]-impl. is the method in [11] re-implemented by us.

We observed that the linear SVM and XGBoost models predict well on the dynamic IR drop, but they do not predict congestion accurately. One possible reason is that our approach takes vision-based features to achieve a more generalized prediction. Furthermore, the experimental results of [11]-impl. are not as accurate as reported in their paper. One possible reason is that they assume designs with a regular mesh structure. However, our test cases vary between technology nodes and PG network topologies.

Table 5. Compare the accuracy of our model with other popular machine learning models.

	IR drop			Congestion score		
	MAPE	NRMSE	CC	MAPE	NRMSE	CC
Ours	**4.39%**	6.54%	**0.984**	**3.27%**	3.65%	**0.994**
SVM	6.40%	9.16%	0.972	14.02%	18.20%	0.793
Xgboost	4.69%	6.96%	0.985	5.21%	7.07%	0.931
[11]-impl.	14.80%	18.17%	0.951	-	-	-

4.2.2 Model Accuracy After Changing PG network. In our optimization process in Section 3.5 and Section 3.6, we will select a set of vias and wires to be changed. We performed experiments to check whether our model can still accurately predict after several incremental modifications. In our proposed flow, the number to be removed have a limitation at 5% and will be early stopping if the number of fail times exceeds. The total number of PG vias and PG wires modified are listed in the column *Modification* in Table 6. Table 6 summarizes the result of the six designs after incremental changes on the PG network. Although the model's accuracy has degraded, the mean absolute percentage error (MAPE) of the IR drop and congestion score are both still within. The correlated coefficients can still achieve up to 0.97. It shows that our model can learn the relationship between the feature and the golden IR drop and congestion score and give a relatively accurate evaluation in a much shorter runtime than a commercial tool. Designers can utilize this model as a reference to guide the PG network optimization without frequently rerunning the commercial tool.

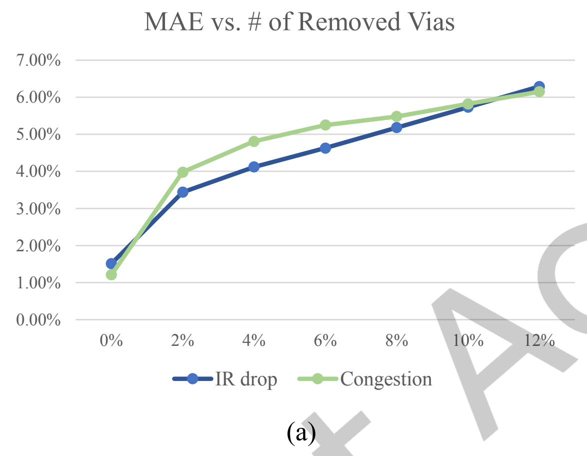
We performed stress testing to ensure our model is robust and does not require frequent re-training during removing PG vias and wires. We performed systematic PG vias removal by increasing the number by 2% each time to see the degradation of MAE and CC. We selected Design 3 because it has the smallest IR drop threshold margin, with the VDD at $0.76V$ and average IR drop at 78.8 mV. Figure 13 shows the degradation. We found that until reaching 10%, our model starts to have an MAE larger than 5%, while the correlation coefficients are still above 0.95, even removing such a large number of vias. The other designs show similar results.

4.2.3 Model Accuracy on Different Placement. The experiments in previous sections were all conducted on the same block. That is, we used one placed block to perform training and validate the result on the testing set. We then performed several permutations to the original block to check whether our model could still achieve generalization on different blocks. For all sequential cell instances, we randomly moved them ±3 site rows and randomly moved ±30 site width. After that, we used a commercial tool to perform legalization to ensure no cell instances overlap. Then, we perform IR drop and congestion analysis to obtain the ground truth. For each design, we generate 3 different blocks, and the results are the average of them.

Table 7 shows the result of predicting on different placement results. We found our model can still maintain an accurate precision, with the correlation coefficient achieving up to 0.98. We also found that compared to the congestion results, the IR drop ones are more accurate. A possible reason might be because we have PG vias

Table 6. Model accuracy after changing PG network

	IR drop					Congestion Score					Modification	
	MAE	MAPE	MaxE	RMSE	CC	MAE	MAPE	MaxE	RMSE	CC	# vias	# wires
Design1	0.99	6.25	7.05	1.36	0.989	6.70	8.21	23.33	8.30	0.952	304	0
Design2	5.55	6.35	88.60	10.41	0.994	3.43	7.39	22.40	4.45	0.997	2041	0
Design3	1.82	3.11	20.14	2.37	0.999	0.55	3.06	6.93	0.83	0.990	6218	2001
Design4	1.78	2.63	36.34	2.73	0.995	0.66	1.74	6.61	0.86	0.998	6262	0
Design5	0.38	0.78	54.83	0.77	0.978	1.40	1.57	40.74	1.95	0.997	11596	5569
Design6	0.74	6.16	22.25	1.10	0.997	3.24	3.42	72.03	6.66	0.955	1031859	18012
aes	2.23	7.11	22.59	3.28	0.944	6.69	3.83	33.56	8.45	0.986	399	0
dynmaic_node	0.78	7.01	5.95	1.10	0.965	5.33	5.37	23.17	6.63	0.987	388	0
ibex	0.98	9.50	19.96	1.73	0.943	9.44	6.53	51.41	12.53	0.982	440	0
jpeg	0.84	4.10	15.91	1.33	0.990	2.83	4.20	20.79	3.73	0.982	1338	0
Avg.	**1.61**	**5.30**	**29.36**	**2.62**	**0.979**	**4.03**	**4.53**	**30.10**	**5.44**	**0.983**	-	-



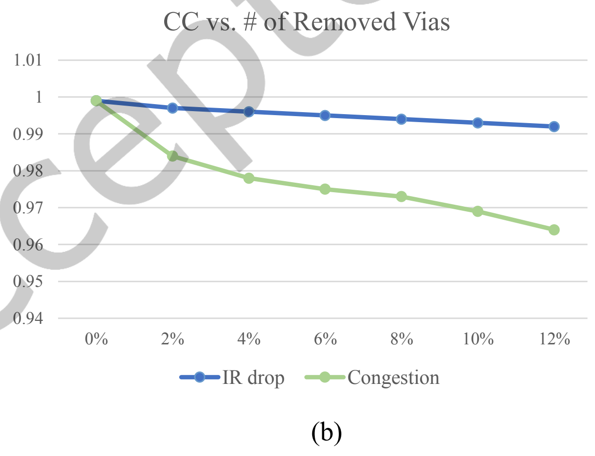


Fig. 13. The stress testing on the model by removing a different number of vias. (a) The MAE vs. the different number of removed vias. (b) The CC vs. the different number of removed vias.

and PG wires images as our input, but we only took a scalar input of cell density to reflect neighboring cells. Therefore, our model can have better generalization when predicting IR drop.

4.3 Experiments on the Model Parameters

In this section, we investigated the impact of our parameter settings. We performed experiments on the different settings of the machine learning training process to show our setting to be reasonable. The experiment includes the size of the selection of features and the size of the bounding box.

4.3.1 Experiments on the Feature selection. We use equation 15 to calculate the correlation between any two features. If any two features are highly correlated, we can pick one of them to use. Figure 14 shows the feature correlation analysis on design 4. We can find that most of the correlation coefficients fall within the $[-0.8, 0.8]$

Table 7. Model accuracy on different placement block

Benchmark	IR drop MAE (mV)	IR drop MAPE (%)	IR drop MaxE (mV)	IR drop RMSE	IR drop CC	Congestion score MAE	Congestion score MAPE (%)	Congestion score MaxE	Congestion score RMSE	Congestion score CC
Design1	0.78	4.78	6.24	1.13	0.991	10.01	10.16	31.50	11.60	0.971
Design2	5.51	7.04	101.45	10.14	0.995	3.58	7.60	26.87	4.86	0.996
Design3	1.17	1.77	14.33	1.57	0.999	1.24	6.41	4.05	1.45	0.990
Design4	3.29	4.26	26.54	4.03	0.996	2.29	5.56	6.68	2.51	0.997
Design5	2.52	4.95	46.87	2.61	0.983	9.04	8.63	40.13	9.61	0.991
Design6	0.73	4.63	61.17	1.04	0.997	4.28	4.11	42.72	5.12	0.988
aes	2.37	7.42	25.68	3.48	0.939	6.75	3.91	30.54	8.48	0.986
dynmaic_node	0.87	7.88	5.75	1.19	0.961	7.60	7.57	29.62	9.11	0.985
ibex	1.02	9.51	20.37	1.83	0.934	11.99	7.95	58.48	15.37	0.980
jpeg	0.89	4.24	16.21	1.40	0.990	3.85	5.54	24.90	4.87	0.982
Avg.	**1.92**	**5.65**	**32.46**	**2.84**	**0.979**	**6.06**	**6.74**	**29.55**	**7.30**	**0.986**

range, which means that most features do not heavily depend on other features. Thus, we think our selected feature set can contain different information and be helpful for training.

4.3.2 Experiments on the Bounding Box Size. In this work, the size of the bounding box is set to be integer multiple of standard cell height h_{cell}, i.e., $h_{box} = w_{box} = k \times h_{cell}$. We considered using different functions related to the number of site rows N_{row} to decide the parameter k. Five different functions are listed in Table 8, while the size of $\sqrt{N_{row}}$ gives the best result in both IR drop and congestion score prediction The bounding box with an overly large or over small size may lead to inaccurate congestion prediction.

Table 8. Model accuracy on different bounding box size

	IR drop MAPE	IR drop NRMSE	IR drop CC	Congestion score MAPE	Congestion score NRMSE	Congestion score CC
Ours	**4.39%**	6.54%	**0.984**	**3.27%**	3.65%	**0.994**
$\log(N_{row})$	8.00%	9.49%	0.959	23.76%	27.87%	0.797
$\sqrt[3]{N_{row}}$	6.89%	8.54%	0.964	21.60%	25.80%	0.850
$0.5N_{row}$	6.48%	7.73%	0.981	52.88%	66.44%	0.664
$0.2N_{row}$	5.08%	6.31%	0.979	17.68%	22.68%	0.867

4.4 Effectiveness of the Proposed Algorithm

We also performed experiments to show that our framework can save routing resources without degrading IR drop. We recorded the original worst dynamic IR drop and the commercial tool's total overflow (TO). Then, we performed our proposed flow and followed by global routing to evaluate the results. Table 9 summarizes the results of congestion improvements in different designs. Our flow can achieve about an 13% reduction in overflows, and the dynamic IR drop degradation was negligible. Figure 15 gives the congestion map and IR drop map before and after changing the PG network. The black line in the congestion map indicates routing overflow in design, and the red color in the IR drop map denotes the region with a higher IR drop. We found that the IR drop hot spot region does not significantly change after changing the PG network, while some congested areas have more routing resources now.

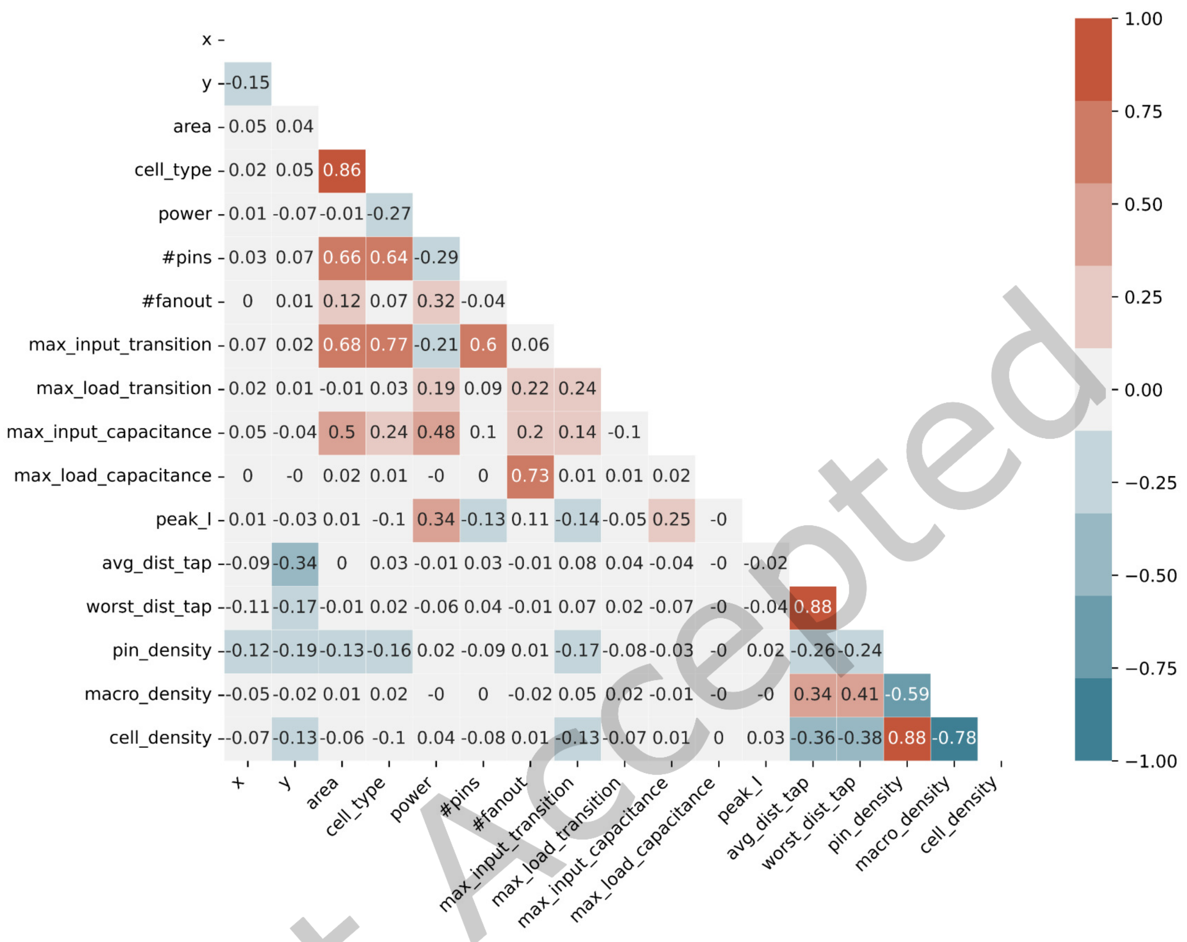


Fig. 14. The feature correlation analysis on design 4.

We then completed the designs to the detail routing stage to show that the overflow reduction in the global routing stage can also give more routing resources to detail routing. We used *Synopsys IC Compiler II* [25] to perform routing and set the maximum iteration number in the detail routing to 30. After the routing was finished, we performed a timing analysis to ensure that the modifications would not impact the timing. Table 10 shows the total negative slack (TNS), the worst negative slack (WNS), the total wirelength of signal nets, and the number of DRC violations. We observed that our changes have negligible impacts on the timing, with the TNS and WNS values still close to the original ones. On the other hand, the saved routing resources can further help to reduce the number of DRC violations. Four designs results in reductions in DRC violations, with the largest one having about an 11% reduction in DRC violations. Thus, we justify that our algorithm can further optimize the routability with minimal design changes.

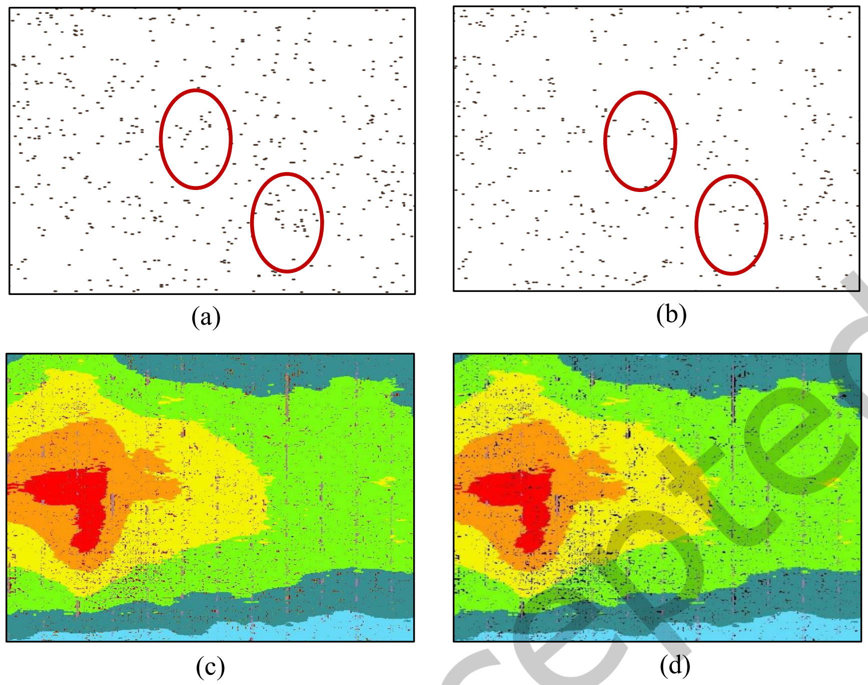


Fig. 15. The routing congestion map and IR drop map in Design3. (a) The original congestion map. (b) The updated congestion map. (c) The original IR drop map. (d) The updated IR drop map.

Table 9. Congestion improvements after changing PG networks.

	Original		**w/ our flow**				Modification	
	worst IR	# TO	worst IR	Percentage	# TO	Percentage	# of vias	# of wires
Design1	-45.34	89	-45.55	**0.46%**	57	**35.96%**	66	0
Design2	-426.35	235	-428.46	**0.49%**	206	**12.34%**	2675	0
Design3	-191.62	906	-193.69	**1.08%**	734	**18.98%**	3039	2001
Design4	-151.06	1350	-151.19	**0.09%**	1183	**12.37%**	4596	0
Design5	-124.56	1145	-124.57	**0.00%**	1010	**11.79%**	9499	5569
Design6	-73.09	95861	-73.11	**0.02%**	94410	**1.51%**	1031859	18012
aes	-72.94	398	-72.61	**-0.45%**	326	**18.09%**	550	0
dynamic_node	-26.27	332	-26.29	**0.09%**	294	**11.45%**	560	0
ibex	-54.99	510	-55.02	**1.11%**	467	**8.43%**	560	0
jpeg	-54.36	579	-54.37	**0.00%**	542	**5.35%**	2050	0
Avg.	-	-	-	**0.29%**	-	**13.63%**	-	-

Table 10. Timing and DRC analysis after completing detail routing.

Benchamrk	w/o our flow				w/ our flow			
	TNS (ns)	WNS (ns)	wirelength (μm)	DRC	TNS (ns)	WNS (ns)	wirelength (μm)	DRC
Design1	0	0	200253.8	0	0	0	199238.7	0
Design2	0	0	688564.5	0	0	0	688107.8	0
Design3	-412.2	-0.12	1582517.1	0	-419.7	-0.12	1592188.0	0
Design4	-1765.5	-0.95	2870026.3	80	-1755.3	-0.94	2866945.4	72
Design5	0	0	9028827.5	10	0	0	9028033.5	5
Design6	-61.5	-0.48	40097047.9	1903	-66.5	-0.43	40127071.9	1684
aes	-332.3	-1.38	245512.8	0	-335.3	-1.40	244868.9	0
dynmaic node	-18.9	-0.52	172207.1	0	-18.4	-0.50	172116.7	0
ibex	-6611.4	-4.06	309185.4	0	-6567.5	-4.04	308750.7	0
jpeg	-954.0	-1.99	631774.1	1	-953.8	-2.00	631313.5	0
Avg.	-1015.6	-1.0	5582591.6	199.4	-1011.6	-0.9	5585863.5	176.1
comparison	1.004	1.007	0.999	1.132	1	1	1	1

4.5 Scalability of the Proposed Algorithm

In this section, we compared the runtime time between a commercial tool and our proposed flow. Table 11 shows the detailed runtime breakdown in both the commercial tool and our proposed flow. In the commercial tool runtime, the column *IR drop* shows the runtime of dynamic IR drop analysis, and the column *congestion* shows the runtime of running global routing to obtain a congestion report. This runtime generates a global route congestion map without actually creating route segments. The column *feature* is the runtime in Section 3.2 to collect scalar feature and in Section 3.3 to collect neighboring images. The column *training* is the step in Section 3.4 to train the backbone model. The column *testing* is the runtime for calling a machine learning model to predict IR drop and congestion score on all cell instances. The golden result generation, feature engineering, and the training part only need to execute once, and the model can be used for analysis repeatedly after the training stage. The results in Sections 4.2.2 and 4.2.3 justify that our model can still provide accurate predictions for designs with minor changes. Therefore, we can use the model to analyze IR drop and congestion in multiple iterations with the first-round golden results. Compared with the runtime of a commercial tool, the testing time of our proposed model can achieve up to 29X speedup and 20X speedup on average. Even if we include the training and feature engineering runtime, our proposed flow can still have a 3.5X speedup on average. Figure 16 shows the runtime between our flow and a commercial tool. We find that as the design scale increases, a commercial tool's analysis time grows exponentially. On the other hand, our proposed flow has good scalability for training and testing.

5 CONCLUSIONS

This paper has addressed multiple objectives of a PG network design. We have proposed a multi-task learning model to simultaneously consider the worst dynamic IR drop and routing congestion. To consider the spatial information, we have transformed the neighboring regions of a cell instance into images and adopted transfer learning to reduce the training time. Furthermore, we have proposed a scheme to perform incremental PG network modifications without significant IR drop degradation and reduce about 13% overflows. The experimental results have shown that our framework can effectively reduce routing congestion and achieve up to 29X faster analysis time than a leading commercial tool.

Table 11. Runtime comparison with the commercial tool.

Benchmark	Commercial tool			Ours				Comparison	
	IR drop (s)	Congestion (s)	Total (s) (i)	Feature (s)	Training (s)	Testing (s) (ii)	Total (s) (iii)	Speedup (i/ii)	Speedup (i/iii)
Design1	50.7	1.2	51.9	0.7	50.6	14.7	66.0	3.54	0.79
Design2	155.7	5.6	161.3	5.0	31.3	19.1	55.3	8.46	2.92
Design3	472.9	235.1	708.0	15.8	219.3	40.1	275.2	17.64	2.57
Design4	339.4	426.5	766.0	19.6	259.6	44.4	323.6	17.27	2.37
Design5	1205.5	689.0	1894.6	66.8	264.4	82.3	413.5	23.03	4.58
Design6	4190.2	2246.2	6436.3	550.2	692.7	219.0	1462.0	29.39	4.40
aes	91.6	9.3	100.9	3.1	77.5	20.6	101.2	4.90	1.00
dynmaic_node	91.5	4.3	95.8	3.2	34.7	18.4	56.3	5.21	1.70
ibex	94.1	7.9	102.0	3.3	43.1	17.8	64.2	5.73	1.59
jpeg	121.0	15.7	136.7	5.7	111.9	24.9	142.5	5.49	0.96
Avg.	681.3	364.1	1045.4	67.3	178.5	50.1	296.0	20.86	3.53

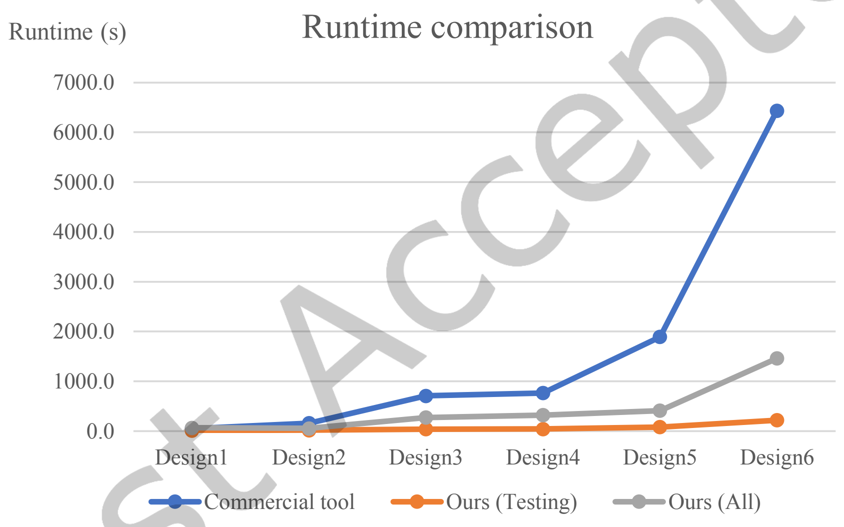


Fig. 16. The runtime comparison of the six industrial designs.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.
[2] ANSYS. 2022. *RedHawk User Manual.* https://www.ansys.com/products/semiconductors/ansys-redhawk-sc Accessed: 2022-11-24.
[3] Wen-Hsiang Chang, Mango C-T Chao, and Shi-Hao Chen. 2013. Practical routability-driven design flow for multilayer power networks using aluminum-pad layer. *IEEE Trans. Very Large Scale Integration Systems (TVLSI)* 22, 5 (June 2013), 1069–1081.

[4] Wen-Hsiang Chang, Chien-Hsueh Lin, Szu-Pang Mu, Li-De Chen, Cheng-Hong Tsai, Yen-Chih Chiu, and Mango C-T Chao. 2017. Generating routing-driven power distribution networks with machine-learning technique. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 36, 8 (January 2017), 1237–1250.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the international conference on knowledge discovery and data mining*. San Francisco, CA, 785–794.

[6] Francois Chollet et al. 2015. *Keras*. https://github.com/fchollet/keras

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Miami, FL, 248–255.

[8] Yen-Chun Fang, Heng-Yi Lin, Min-Yan Su, Chien-Mo Li, and Eric Jia-Wei Fang. 2018. Machine-Learning-Based Dynamic IR Drop Prediction for ECO. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. San Diego, CA, 1–7.

[9] R. Goldman, K. Bartleson, T. Wood, K. Kranen, V. Melikyan, and E. Babayan. 2013. 32/28nm Educational Design Kit: Capabilities, deployment and future. In *2013 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*. Visakhapatnam, India, 284–288.

[10] Chia-Tung Ho and Andrew B. Kahng. 2019. IncPIRD: Fast Learning-Based Prediction of Incremental IR Drop. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. Westminster, CO, 1–8.

[11] Xuan-Xue Huang, Hsien-Chia Chen, Sheng-Wei Wang, Iris Hui-Ru Jiang, Yih-Chih Chou, and Cheng-Hong Tsai. 2020. Dynamic IR-Drop ECO optimization by cell movement with current waveform staggering and machine learning guidance. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. San Diego, CA, 1–8.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] S. Köse and E. G. Friedman. 2011. Fast algorithms for IR voltage drop analysis exploiting locality. In *Proc. ACM/IEEE Design Automation Conf. (DAC)*. San Diego, CA, 996–1001.

[14] J. Lin, J. Syu, and I. Chen. 2018. Macro-Aware Row-Style Power Delivery Network Design for Better Routability. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. San Diego, CA, 1–8.

[15] Shen Lin and Norman Chang. 2001. Challenges in power-ground integrity. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. San Jose, CA, 651–654.

[16] lowRISC org. 2022. *Ibex RISC-V Core*. https://github.com/lowRISC/ibex/ Accessed: 2022-11-24.

[17] S. K. Nithin, G. Shanmugam, and S. Chandrasekar. 2010. Dynamic voltage IR drop analysis and design closure: Issues and challenges. In *Proc. of IEEE International Symposium on Quality Electronic Design (ISQED)*. San Jose, CA, 611–617.

[18] OpenCores. 2018. *AES (Rijndael) IP Core*. https://opencores.org/projects/aes_core/ Accessed: 2022-11-24.

[19] OpenCores. 2018. *JPEG encoder*. https://opencores.org/projects/video_systems/ Accessed: 2022-11-24.

[20] S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (November 2011), 2825–2830.

[22] Princeton University. 2021. *PrincetonUniversity OPDB*. https://github.com/PrincetonUniversity/OPDB/ Accessed: 2022-11-24.

[23] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (August 2017).

[24] Synopsys. 2022. *Design Compiler*. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html Accessed: 2022-11-24.

[25] Synopsys. 2022. *IC Compiler II User Manual*. https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/ic-compiler-ii-ds.pdf Accessed: 2022-11-24.

[26] Mingxing Tan and Quoc V Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946* (September 2019).

[27] S. Wang, G. Liou, Y. Su, and M. P. Lin. 2019. IR-aware Power Net Routing for Multi-Voltage Mixed-Signal Design. In *Proc. of IEEE/ACM Design, Automation and Test in Europe (DATE)*. Florence, Italy, 72–77.

[28] Zhiyao Xie, Hai Li, Xiaoqing Xu, Jiang Hu, and Yiran Chen. 2020. Fast IR Drop Estimation with Machine Learning. In *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*. San Diego, CA, 1–8.

[29] Zhiyao Xie, Haoxing Ren, Brucek Khailany, Ye Sheng, Santosh Santosh, Jiang Hu, and Yiran Chen. 2020. PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network. In *Proc. of IEEE/ACM Asia and South Pacific Design Automation Conf. (ASPDAC)*. Beijing, China, 13–18.

[30] Y. Yamato, T. Yoneda, K. Hatayama, and M. Inoue. 2012. A fast and accurate per-cell dynamic IR-drop estimation method for at-speed scan test pattern validation. In *Proc. of IEEE International Test Conference (ITC)*. Anaheim, CA, 1–8.

[31] F. Ye, F. Firouzi, Y. Yang, K. Chakrabarty, and M. B. Tahoori. 2014. On-chip voltage-droop prediction using support-vector machines. In *Proc. of IEEE VLSI Test Symposium (VTS)*. Napa, CA, 1–6.