FanoutNet: A Neuralized PCB Fanout Automation Method Using Deep Reinforcement Learning

Haiyun Li[1], Jixin Zhang[2*], Ning Xu[1*], Mingyu Liu[3]

[1] School of Information Engineering, Wuhan University of Technology, Wuhan, China
[2] School of Computer Science, Hubei University of Technology, Wuhan, China
[3] Wuhan Research Institute, Huawei Device Co., Ltd., Wuhan, China
zhangjx@hbut.edu.cn, xuning@whut.edu.cn

Abstract

In modern electronic manufacturing processes, multi-layer Printed Circuit Board (PCB) routing requires connecting more than hundreds of nets with perplexing topology under complex routing constraints and highly limited resources, so that takes intense effort and time of human engineers. PCB fanout as a pre-design of PCB routing has been proved to be an ideal technique to reduce the complexity of PCB routing by pre-allocating resources and pre-routing. However, current PCB fanout design heavily relies on the experience of human engineers, and there is no existing solution for PCB fanout automation in industry, which limits the quality of PCB routing automation. To address the problem, we propose a neuralized PCB fanout method by deep reinforcement learning. To the best of our knowledge, we are the first in the literature to propose the automation method for PCB fanout. We combine with Convolution Neural Network (CNN) and attention-based network to train our fanout policy model and value model. The models learn representations of PCB layout and netlist to make decisions and evaluations in place of human engineers. We employ Proximal Policy Optimization (PPO) to update the parameters of the models. In addition, we apply our PCB fanout method to a PCB router to improve the quality of PCB routing. Extensive experimental results on real-world industrial PCB benchmarks demonstrate that our approach achieves 100% routability in all industrial cases and improves wire length by an average of 6.8%, which makes a significant improvement compared with the state-of-the-art methods.

Introduction

Modern PCB designs contain more than hundreds of nets, thousands of pins, and multiple layers (Yan and Wong 2010; Yan, Ma, and Wong 2012). Due to the complicated functionality and electronic constraints, PCB routing automation has become an extremely challenging problem. An industrial PCB layout takes about months to route manually, and PCB routing automation can tremendously improve the efficiency of industrial production.

In industry, the design of fanout is widely used for PCB routing. Basically, the PCB fanout includes assigning layers, determining the fanout locations, and routing from the pins to the fanout locations. As a pre-design of PCB routing, unlike detailed routing from pin to pin, PCB fanout focuses on connection topology, routing resource assignment, component placement, and layer segmentation. Human engineers pre-allocate routing resources, reduce topology crossings, and improve routability through PCB fanout. In modern high-speed PCB designs, the net connections have many topology crossings and the routing resources are highly limited, making direct pin-to-pin routing without PCB fanout difficult. Hence, PCB fanout is often a necessary stage in the modern PCB routing.

However, current PCB fanout design heavily relies on the experience of human engineers, and there is no existing solution for PCB fanout automation in the industry, which limits the quality of PCB routing automation. Although some related approaches to Ball Grid Array (BGA) package escape routing have been proposed (Liao and Dong 2020; Lin et al. 2021a), these approaches are only designed for BGA escape routing. The pin array in BGA is quite regular, while the pins in PCB are various, so that the BGA escape routing approaches cannot be used for solving PCB fanout problems. Although PCB fanout is very important for reducing the routing complexity and improving routability, which is especially effective in modern high-speed PCB, little consideration has been given to this field. In this paper, we aim to propose an intelligent fanout method to fill the vacancies of prior studies.

The main idea of our solution is to use Deep Reinforcement Learning (DRL) to learn and simulate the fanout designs of human engineers. We formulate the PCB fanout problem as a Markov Decision Process (MDP) and build a policy model to make optimal decisions on fanout actions and a value model to make accurate evaluations of fanout results. One of the biggest challenges is to transform the fanout optimization into gradient optimization so that we can efficiently optimize fanout results. To address the challenge, we propose to use deep neural networks to encode the representations of PCB layout and netlist so that the gradient information can be used to train the policy and value model. We also design a PPO algorithm to optimize the fanout results.

In this paper, we propose a neuralized PCB fanout method based on Reinforcement Learning (RL), named FanoutNet, to generate high-quality fanout results automatically. First, we extract the information of the netlist and partially gener-

*Corresponding author.

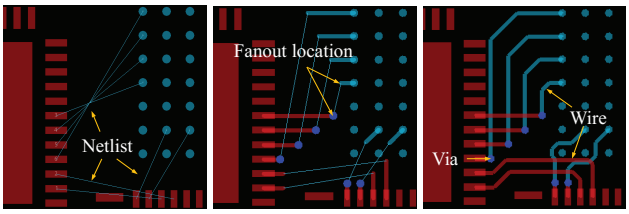
ated fanout result into a sequence, called structured feature, which includes pin locations, pin connections and current fanout locations for each net. We rasterize PCB layout as a multi-channel binary image, called spatial feature, and incorporate the spatial feature and the structured feature as the representations for RL training. Then, we propose a Convolutional Neural Network (CNN) and an attention-based network to encode the representations. Finally, we design a PPO algorithm to optimize the fanout results generated by FanoutNet iteratively. In addition, we develop a Multi-Constraint Router (MCR) that equips our FanoutNet to verify the quality of the fanout results and the final routing. MCR considers multiple routing constraints to ensure that it can be used for industrial PCB routing.

The main contribution of this paper are summarized as follows:

- As far as we know, we are the first in the literature to solve the PCB fanout automation problem by using deep reinforcement learning.
- We propose a neuralized fanout method. We first formulate the PCB fanout problem as an MDP. Then we build a FanoutNet with a CNN and an attention-based network to encode the representations of PCB layout and netlist to make decisions and evaluations for fanout, transforming the fanout optimization into a gradient optimization.
- We propose an MCR algorithm for industrial PCB routing. MCR provides accurate and fast routing evaluation. FanoutNet takes the evaluation as reward feedback and uses PPO algorithm to optimize fanout results.
- Extensive experiments based on industrial PCB benchmarks are conducted to investigate the efficacy and quality of our proposed approach. The experimental results demonstrate that our approach achieves 100% routability in all industrial cases and improves wire length by an average of 6.8%, which makes a significant improvement compared with the state-of-the-art methods.

Related Work

Recent years, deep learning based EDA routing method has been widely concerned by the industrial and academic community. For integrated circuits (IC) routing, some deep reinforcement learning methods have been presented for global routing (Liao et al. 2020b), detail routing (Lin et al. 2021c), and track assignment (Liao et al. 2020a). Deep reinforcement learning has also been shown to solve some fundamental problems, including rectilinear Steiner minimum tree problem (Liu, Chen, and Young 2021) and travelling salesman problem (Kool, Van Hoof, and Welling 2018). In addition, combined with SAT optimization, deep learning predictions can be used to accelerate escape routing (Chen et al. 2021). These previous works are quite different from PCB routing, in other words, the problem and constraints are quite different from PCB routing, which makes the reuse of these previous works for PCB routing impossible. Deep learning methods can also be used to predict routing congestion, crosstalk and routability. The prediction results are used as feedback to optimize placement and routing (Liu et al. 2021; Su et al. 2022; Alawieh et al. 2020; Chen et al. 2020).



(a) (b) (c)

Figure 1: Example of (a) unrouted PCB layout; (b) PCB fanout result; (c) PCB routing result.

Our work is inspired by these previous works but has an entirely different focus on the studied problem. We propose to optimize PCB fanout by deep reinforcement learning to solving PCB routing problem.

Problem Formulation

PCB Fanout Problem Given a multi-layer PCB with a layout $\boldsymbol{L}$, a netlist $\boldsymbol{N}$, a set of pins $\boldsymbol{P}$, and routing constraints $\boldsymbol{C}$, $\boldsymbol{L}$ defines the routable area and obstacles in the entire PCB layout, and $\boldsymbol{N}$ contains a set of two-pin nets which need to be connected. Note that the net in this paper indicates two-pin net, as shown in Fig. 1(a). We decompose all multi-pin nets into multiple two-pin nets using a minimum spanning tree (MST). PCB fanout aims to find fanout location for each pin and connect the pin to the location by routing. The fanout locations of a net must be in the same layer, and the topology crossings $T_c(n)$ of nets must be minimized while satisfying the routing constraints $\boldsymbol{C}$. The objective function is defined as:

$$\min \sum_{n \in N} T_c(n)$$
$$\text{s. t. } \boldsymbol{C}$$

An example of fanout result is shown in Fig. 1(b), (c). The fanout result in Fig. 1(b) resolves the topology crossings of the nets, and the pins are connected to the corresponding fanout locations by routing and via assigning.

PCB Routing Problem Given a multi-layer PCB and a fanout result $\boldsymbol{F}$, all nets must be connected, and the via count and wire length must be minimized while satisfying routing constraints $\boldsymbol{C}$. The objective function is defined as:

$$\max \textit{Routability} - \textit{Viacount} - \textit{Wirelength}$$
$$\text{s. t. } \boldsymbol{C}$$

Fig. 1(c) presents an example of routing result. The connection of all nets is routed based on the fanout result.

Neuralized Fanout Method

The Overview of Our Method

We propose a deep reinforcement learning approach named FanoutNet by using CNN and attention-based network with the spatial feature and structured feature to capture the PCB information. In the FanoutNet, the RL agent produces an optimal fanout action at each timestep and estimates a value for

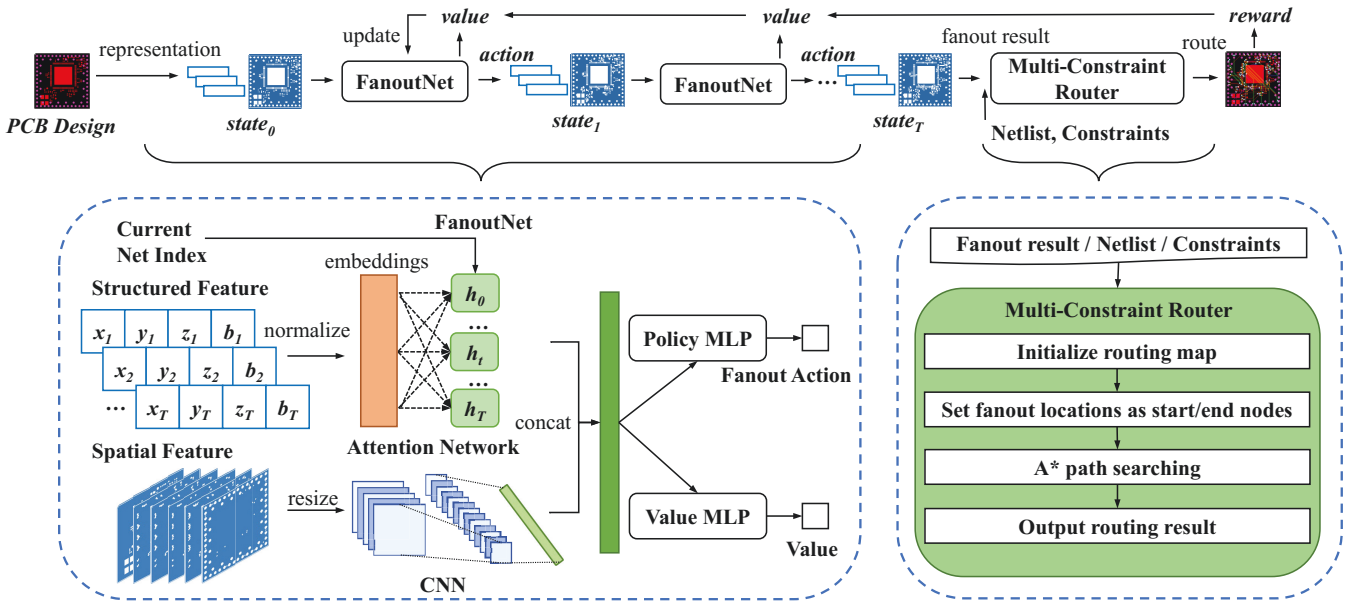


Figure 2: The overall architecture of the neuralized fanout method.

the current state after taking the action. The partially generated fanout result produced by the action will be added to the next state, and the estimation value will be used to update the parameters of the neural networks. MCR will be used to calculate the routability as a reward. Once the FanoutNet is trained, we can produce a fanout result in a few seconds to improve the final routing quality. We present the overall architecture of our method in Fig. 2.

MDP for Fanout

We formulate the PCB fanout as a Markov decision process. A fanout action will be taken in every timestep. The episode length T is equal to the total number of nets.

The state in timestep t is a tuple $s_t = (\boldsymbol{S}_t, \boldsymbol{M}_t)$ of spatial feature $\boldsymbol{S}_t$ and structured feature $\boldsymbol{M}_t$.

The action in PCB fanout problem refers to two fanout locations of the two-pin net. The fanout location (l, o, d) of a pin can be defined as a combination of the fanout layer l, the fanout orientation o, and fanout distance d by searching around the center of the pin. The action for each timestep can be defined as a two-pin combination $a = (l, o^1, d^1, l, o^2, d^2)$, where two pins have the same fanout layer and cooperative orientations and distances. The orientation can rotate 45° along the z-axis, resulting in a number of candidate orientations $O = 8$. The number of candidate layers L depends on the number of layers in the PCB layout. The candidate distances are n_d times the pin size ($n_d = 0, 2, 4, 8$). So the number of candidate distances is $D = 4$. The total number of the fanout action for a net is $d_a = L \times (O \times D)^2$.

We assign a probability for each fanout action and represent all actions as a vector of probabilities. At each timestep, an action is sampled according to the probability distribution produced by the policy model. After an action has been selected, the state transition is deterministic for the next state.

After taking actions for all the nets, the reward is given by calculating the routability of MCR, according to Eq. (1).

$$R = \frac{N_{succ}}{N_{total}} \quad , \tag{1}$$

where N_{succ} is the number of successfully routed nets and N_{total} is the total number of nets.

Representation

We propose the spatial feature and the structured feature as PCB representations. PCB layout includes irregular routing obstacles and boundaries, a number of pins, wires and vias. These are usually shaped by different data types, such as polygon, circle and line. We represent them in a uniform format by a multi-channel binary image as the spatial feature. We represent the information of netlist and the partially generated fanout results by a vectorized sequence as structured feature. The following is a detailed description of the representations:

Spatial Feature The spatial feature is a multi-channel binary image $\boldsymbol{S} \in \mathbb{R}^{L \times H \times W}$ of the multi-layer PCB layout, where L is the number of layers, H and W are the height and width of the PCB layout respectively. Initially, the initial spatial feature includes the obstacles of pin and boundary. New routed wires and vias of each timestep will be rendered to the spatial feature.

Structured Feature The structured feature is a vectorized sequence $\boldsymbol{M} \in \mathbb{R}^{T \times 6}$. Each item consists of a vector corresponding to a net $m_t = (x_t^1, y_t^1, z_t^1, x_t^2, y_t^2, z_t^2, b)$, where $t \in \{1,...,T\}$ and T is the episode length. The vector includes a boolean b and two locations. b denotes whether the corresponding net completes fanout. The locations are the

pin locations or fanout locations when $b = 0$ or $b = 1$. In the first timestep, all b are set to zero.

FanoutNet

FanoutNet takes the representations as inputs and outputs the fanout action and value. The network consists of a CNN and an attention-based network. The policy model and value model output action and value by a Multi-Layer Perceptron (MLP) attached to the end of the backbone. The two model use the same network architecture but have independent parameters.

Modeling the Spatial Feature with CNN The spatial feature involves the information of the unroutable area formed by the pins, boundaries, wires and vias. We propose to use CNN to encode the spatial information. Given a spatial feature $\boldsymbol{S}$, CNN computes the spatial feature into a hidden feature vector that includes high-level information:

$$h_l = \mathrm{AvgPool}(\mathrm{CNN}(\boldsymbol{S})) \quad , \tag{2}$$

where $h_l \in \mathbb{R}^d$ is the hidden vector produced by the CNN, and AvgPool is the global average pool operator on the two-dimensional plane. The dimension of the vector is d = 128. We adopt ResNet18 (He et al. 2016) as the CNN architecture for a good balance between capacity and efficiency.

Modeling the Structured Feature with Attention The structured feature involves the pin locations and net connections. The context of structured feature provides a global view of net topology. Following the Transformer architecture (Vaswani et al. 2017), which is widely used for processing sequences, we adopt the attention layer to encode the structured feature. Each attention layer consists of a Multi-Head Attention (MHA) layer and a Feed-Forward (FF) layer. The encoder first maps the structured feature $\boldsymbol{M}$ into an embedding vector $\boldsymbol{h}^{(0)} \in \mathbb{R}^{T \times d}$ ($d = 128$) by a linear layer:

$$\boldsymbol{h}^{(0)} = \mathrm{Linear}(\boldsymbol{M}) \quad . \tag{3}$$

Then the embedding vector are computed recursively by N attention layers:

$$\hat{\boldsymbol{h}} = \boldsymbol{h}^{(i-1)} + \mathrm{MHA}^{(i)}(\boldsymbol{h}^{(i-1)}) \quad , \tag{4}$$

$$\boldsymbol{h}^{(i)} = \hat{\boldsymbol{h}} + \mathrm{FF}^{(i)}(\hat{\boldsymbol{h}}) \quad , \tag{5}$$

where $\boldsymbol{h}^{(i)}$ is the output of layer $i \in \{1, ..., N\}$ ($N = 4$). MHA uses 8 heads with 16-dimension, and the FF layer has one hidden layer with 256-dimension and ReLU activation. $\boldsymbol{h}^{(N)}$ is the output of the last attention layer. In t timestep, we select the *t-th* vector $h_m \in \mathbb{R}^d$ of $\boldsymbol{h}^{(N)}$ as the output of attention-based network.

Policy and Value Model We use the MLP with two hidden layers to implement policy and value model. We apply a concat function to concatenate the hidden vectors of spatial feature and structured feature into a vector $h_c \in \mathbb{R}^{2d}$:

$$h_c = \mathrm{concat}(h_l, h_m) \quad . \tag{6}$$

For the policy model, we apply MLP to map the h_c into a probability distribution for actions:

$$P_a = \mathrm{softmax}(\mathrm{MLP}(h_c)) \quad , \tag{7}$$

where $P_a \in \mathbb{R}^{d_a}$ is the probability distribution. Then the action can be selected by sampling from the distribution P_a.

For the value model, it outputs a single value computed as follows:

$$V = \mathrm{MLP}(h_c) \quad , \tag{8}$$

where the value $V \in \mathbb{R}^1$ is the estimated expectation of discounted cumulative reward drawn from the policy model based on the current state. It will be used to update the parameters of policy model.

Parameters Update We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al. 2017) to update the parameters of the policy and value model. The rewards can be estimated by computing discounted cumulative reward (Browne et al. 2012):

$$R_t = \gamma^{T-t} R_T \quad , \tag{9}$$

where t is the timestep in [0, T], R_T is the final reward at the end of the episode, gamma is a discount factor ($\gamma = 0.99$).

Then we can define the loss functions of the policy and value model. For the policy model, we use the clipping loss function:

$$L^{PF}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \min(r_t(\theta)A_t, r_t^{clip}(\theta)A_t) \quad , \tag{10}$$

where $r_t^{clip}(\theta) = clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$, $A_t = R_t - V_t$, θ is the parameter of the policy model, $r_t(\theta)$ is the probability ratio between old policy and current policy, ϵ is set to 0.2 (Schulman et al. 2015, 2017). The loss function of the value model is defined by the mean square error:

$$L^{VF}(\phi) = \frac{1}{T}\sum_{t=1}^{T} \|V_t - R_t\|_2^2 \quad , \tag{11}$$

where ϕ is the parameter of the value model. We update the parameters of the policy and value model by minimizing these two loss functions with the Adam optimizer (Kingma and Ba 2014).

Multi-Constraint Router

In this paper, we propose a Multi-Constraint Router (MCR) to verify the improvement of our FanoutNet. We consider the connection between pins and fanout locations as a two-terminal path searching problem which subjects to routing constraints. We take the 3-D A* algorithm as the base path searching algorithm and improve it for PCB routing. Our MCR calculates a cost function $\hat{f}(\boldsymbol{n})$ for each grid node $\boldsymbol{n}$, which aims to find a path with minimum cost. The cost function is written as the sum of two parts, according to Eq. (12):

$$\hat{f}(\boldsymbol{n}) = \hat{g}(\boldsymbol{n}) + \hat{h}(\boldsymbol{n}) \quad , \tag{12}$$

We design specific cost functions for the PCB routing with complex industrial constraints.

For $\hat{g}(\boldsymbol{n})$, we design several sub-cost functions for the routing constraints, including wire length and via count, which are given by the following Eq. (13)-(14), respectively.

$$w(\boldsymbol{n}) = \|\boldsymbol{n}_c - \boldsymbol{n}\|_2 \quad , \tag{13}$$

where $\boldsymbol{n}_c$ is the current node and $\boldsymbol{n}$ is its successor.

$$v(\boldsymbol{n}) = |z(\boldsymbol{n}_c) - z(\boldsymbol{n})| \quad . \tag{14}$$

where $z(\cdot)$ is the z-axis coordinate, i.e., the layer index. We use the z-axis distance as the cost to limit frequent layer changes so that the via count can be minimized.

In addition, we also formulate differential pairs, reference layer shielding, and spacing constraints as sub-cost functions $d(\boldsymbol{n})$, $r(\boldsymbol{n})$, and $s(\boldsymbol{n})$. Combining all sub-cost functions, $\hat{g}(\boldsymbol{n})$ can be obtained as follows:

$$\hat{g}(\boldsymbol{n}) = c_1 w(\boldsymbol{n}) + c_2 v(\boldsymbol{n}) + c_3 d(\boldsymbol{n}) + c_4 r(\boldsymbol{n}) + c_5 s(\boldsymbol{n}), \tag{15}$$

where c_1, c_2, c_3, c_4, c_5 are coefficients, which are the weights of different constraints.

For $\hat{h}(\boldsymbol{n})$, we use the shortest possible wire length constrained by 135° routing angle:

$$\begin{aligned}\hat{h}(\boldsymbol{n}) = &\max\left(|x(\boldsymbol{n}_g) - x(\boldsymbol{n})|, |y(\boldsymbol{n}_g) - y(\boldsymbol{n})|\right) \\ &+ \left(\sqrt{2} - 1\right) \cdot \min\left(|x(\boldsymbol{n}_g) - x(\boldsymbol{n})|, |y(\boldsymbol{n}_g) - y(\boldsymbol{n})|\right),\end{aligned} \tag{16}$$

Combining $\hat{g}(\boldsymbol{n})$ and $\hat{h}(\boldsymbol{n})$, 3-D A* algorithm can produce routing results which satisfy industrial constraints by adaptively searching for the path with minimum cost. For better routing quality, we adopt the congestion negotiation rerouting algorithm (McMurchie and Ebeling 2008) to produce final routing result. We initialize the coefficients c_5 to a negligible value and increase it as the number of iterations increases.

Experiments

Experimental Setup

Our method is implemented in Python with Pytorch for reinforcement learning, and C++ with the mingw64 compiler for PCB router. The experiments are performed on a 64-bit Windows workstation with AMD Ryzen 7 5800X 8-Core Processor 3.79 GHz, and 64 GB RAM. Our PPO hyperparameters, *K* epochs and the learning rate of policy and value model, are 20, 3e-4 and 1e-3, respectively.

Dataset We build two benchmark datasets consisting of eleven open-source PCB cases[1] and five industrial PCB cases. The open-source PCB cases are 2-4 layer layout with various components including chips, capacitors, and through-hole pin arrays. The industrial PCB cases are high-speed six-layer HDI layout with hundreds of nets and perplexing connection topology. The two datasets will be used for verifying the effectiveness of our proposed fanout method (FanoutNet). The details of the benchmark datasets are listed in tables 1 and 2, respectively.

Evaluation Metrics We adopt a variety of evaluation metrics widely used in previous works (Lin et al. 2021), including Routability (RT), Via Count (#Via), Wire Length (WL), and Runtime (T). RT computes the proportion of successfully routed nets to the total nets in the final routing results. Besides, we use Pre-Routability (Pre-RT) to compute approximate routability for RL training.

[1]PCB dataset was acquired at https://github.com/aspdac-submission-pcb-layout/PCBBenchmarks/

Design	Nets	Components	Pins	Layers
bm1	99	60	319	2
bm2	34	19	77	2
bm3	80	58	229	2
bm4	54	48	163	2
bm5	38	34	138	2
bm6	52	28	140	2
bm7	15	8	40	2
bm8	70	36	188	2
bm9	63	61	314	4
bm10	35	58	233	4
bm11	69	46	207	2

Table 1: The open-source dataset used in the experiments

Design	Nets	Components	Pins	Layers
case1	129	61	277	6
case2	94	252	937	6
case3	358	267	711	6
case4	368	269	772	6
case5	290	206	858	6

Table 2: The industrial dataset used in the experiments

Methods for Comparison We consider the following methods for comparison:

- Freerouting[2] v1.4.5.1: It is an open-source PCB routing automation tool used in several commercial PCB design tools, such as KiCad[3]. It provides a grid-based automatic router with a rerouting algorithm to optimize via count and wire length.
- ELECTRA[4] v5.91: It is a commercial PCB auto-routing tool based on a shape-based algorithm with high PCB routing efficiency.
- NB-A* (Lin et al. 2021b): It presents a negotiation-based router for PCB routing by adjusting the A* cost function to different routing constraints.
- GA (Whitley 1994): It is the conventional genetic optimization algorithm. We use the algorithm as a comparison to search for optimal fanout results.
- ARL (Kool, Van Hoof, and Welling 2018): It proposes attention-based reinforcement learning to solve the travelling salesman problem. We also use this method to search fanout results as a comparison.

Among these methods, Freerouting, ELECTRA, and NB-A* can complete full-board PCB routing, so we directly use their results for comparison. Since ARL and GA are optimization algorithms, they are only used to generate fanout results, and their routing results are done by our MCR.

[2]FreeRouting was acquired at https://freerouting.org/

[3]KiCad was acquired at https://www.kicad.org/

[4]ELECTRA was acquired at https://konekt.com/

Metric	Method	case1	case2	case3	case4	case5
RT (%)	MCR	100	85	98	99	97
	FanoutNet	**100**	**100**	**100**	**100**	**100**
#Via	MCR	133	164	307	262	337
	FanoutNet	**113**	185	393	302	359
WL (10^2mil)	MCR	174	199	480	740	464
	FanoutNet	**170**	228	520	772	477

Table 3: Impact of fanout

Metric	Repr.	case1	case2	case3	case4	case5
Pre-RT (%)	Spatial	83	76	82	77	78
	Structured	83	75	83	79	79
	Both	**84**	**78**	**85**	**83**	**80**
RT (%)	Spatial	100	99	100	100	100
	Structured	100	96	100	100	100
	Both	**100**	**100**	**100**	**100**	**100**
#Via	Spatial	126	193	413	425	452
	Structured	144	168	402	334	426
	Both	**113**	185	**393**	**302**	**359**
WL (10^2mil)	Spatial	180	228	531	771	481
	Structured	182	215	525	795	488
	Both	**170**	228	**520**	772	**477**

Table 4: Importance of representations

The Impact of Fanout

To evaluate the performance of our proposed fanout method, we take MCR as baseline to observe the improvements of FanoutNet. As shown in Table 3:

- FanoutNet obtains 100% routability in all PCB cases while baseline fails to complete the routing for four cases. It reflects that our FanoutNet facilitates resource allocation to improve routability.
- Note that the via count and wire length increase with the increasing of routability because the via count and wire length of more nets are counted, as we can see in case2-5. For case1, both of our FanoutNet and the baseline method achieve 100% routability. Our FanoutNet produce less via count and wire length.

The Importance of Representations

We perform an ablation experiment to investigate the importance of different representations used in our approach. We compare the results from single spatial feature, single structured feature and the fusion of these two features. As shown in Table 4:

- When both spatial feature and structured feature are incorporated, FanoutNet can achieve a better pre-routability. This indicates that fusing the two features provides more information for training a better model.
- The final routing are completed by re-routing version MCR. The results show that our FanoutNet trained with the two features achieves 100% routability in all PCB cases and make less via count and wire length in most cases.

PCB Routing Performance Evaluation

We first verify our FanoutNet in open-source dataset. The commercial PCB routers, i.e., Freerouting and ELECTRA, are taken as baselines. As shown in Table 5:

- Our FanoutNet achieves consistently better routing quality than all other methods in all cases. In most cases has a comparable runtime to ELECTRA. It provides a robust performance even on bm11, which is a high-density PCB case, while other methods fail to achieve 100% routability.
- Both Freerouting and ELECTRA perform well with an average routability of 99%. That means the two baseline methods can achieve good performance in PCB cases with fewer nets, layers and simple constraints. FreeRouting minimizes wire length and via count and produces better results than ELECTRA. However, it also leads to a very long runtime.
- As a comparison, ELECTRA completes the routing in seconds for all test cases but produces higher wire length and via count.

Table 6 presents the results of all the comparison methods in industrial PCB dataset. From the results:

- FanoutNet achieves 100% in all test cases, outperforming all other methods. Although wire length and via count increase with routability increases, our method still achieves less wire length and via count than others in most cases. The training time of our FanoutNet is much longer than most methods, but once the FanoutNet is trained, the fanout results can be generated in seconds.
- Commercial PCB routers fail to complete routing in most cases, especially in case2 and case5, which have the highest pin density. A possible reason is that they do not well utilize the resource of different layers to reduce topology crossings. They resolve crossings by assigning vias or detouring wires when wire conflict happens, which results in a waste of resources.
- NB-A* proposes to minimize the total resources cost of routing. However, it cannot balance the user-defined weights of different objectives. We set routability as the first objective as in the original paper. When routability is difficult to reach 100% in complex PCB cases, the iteration count increases significantly, and all other metrics degrade.
- We use optimization methods, GA and ARL, to optimize fanout results instead of our FanoutNet. However, GA takes a very long runtime to find the optimal solution randomly, and it still makes higher wire length and via count than ARL and NB-A*. ARL as a one-stage method encodes the initial spatial and structured features and decodes a complete fanout result at once. Although it efficiently improves routing quality, our method achieves better performance in terms of routability, wire length and via count.

In general, FanoutNet significantly improves the routability, wire length and via count in most cases, which outperforms commercial PCB routers and state-of-the-art methods.

Metric	Method	bm1	bm2	bm3	bm4	bm5	bm6	bm7	bm8	bm9	bm10	bm11	avg.
RL (%)	FR	100	100	93	99	100	100	100	100	100	100	94	99
	EL	100	100	98	100	100	100	100	100	100	100	92	99
	FanoutNet	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
#Via	FR	45	2	30	22	10	4	0	1	76	27	53	26
	EL	170	6	57	37	19	28	5	1	111	55	63	50
	FanoutNet	86	**2**	**17**	**15**	**8**	6	**0**	**0**	**73**	**18**	**52**	**25**
WL (mm)	FR	4,517	354	1,366	1,055	588	850	93	1,127	3,154	1,543	1,033	1,456
	EL	5,583	327	1,474	1,123	569	926	118	1,191	3,527	1,728	983	1,595
	FanoutNet	**4,096**	**203**	**887**	**739**	**331**	**614**	95	**879**	**2,885**	**1,146**	**875**	**1,160**
T (s)	FR	1740	23	n/a	n/a	133	53	2	12	1800	285	n/a	n/a
	EL	11	1	2	2	1	1	1	1	9	5	7	4
	FanoutNet	77	4	19	21	8	6	2	6	353	40	19	50

Table 5: Experimental results on open-source dataset ("n/a": over 24 hours)

Metric	Method	case1	case2	case3	case4	case5	avg.
RT(%)	FR	100	79	96	91	40	81
	EL	100	79	98	99	79	91
	NB-A*	100	88	98	100	89	95
	GA	100	91	99	100	100	98
	ARL	100	92	99	100	100	98
	FanoutNet	**100**	**100**	**100**	**100**	**100**	**100**
#Via	FR	135	101	125	228	74	133
	EL	146	121	183	294	264	202
	NB-A*	183	142	480	449	468	344
	GA	223	182	590	382	385	352
	ARL	166	141	489	394	355	309
	FanoutNet	**113**	185	393	302	359	270
WL (10^2mil)	FR	303	257	563	951	533	521
	EL	258	191	579	919	373	464
	NB-A*	175	210	545	803	478	442
	GA	188	217	609	797	488	460
	ARL	172	223	541	779	480	439
	FanoutNet	**170**	228	**520**	**772**	477	**433**
T(min)	FR	1	n/a	n/a	n/a	n/a	n/a
	EL	<1	1	<1	<1	5	1
	NB-A*	9	65	75	108	89	69
	GA	57	244	953	747	693	539
	ARL	19	136	297	266	214	187
	FanoutNet	31	109	394	352	340	245

Table 6: Experimental results on industrial dataset

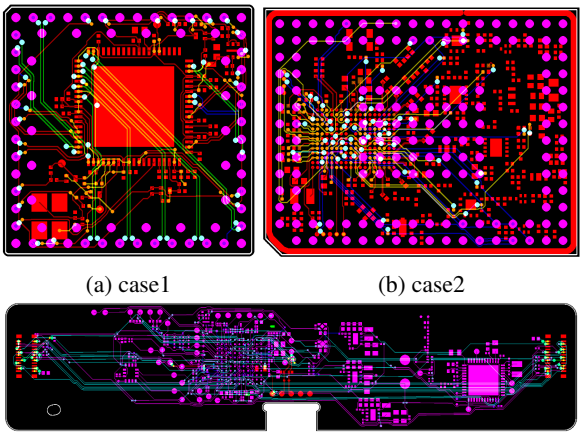
Case Studies

Fig. 3 shows the visualization results of the industrial cases from Huawei Device Co., Ltd. In case1 and case4, our method satisfies the complex routing constraints of many differential pairs. In case2, which has high pin-density regions and numerous topology crossings, our method adaptively finds the fanout locations, resolves the crossings and achieves 100% routability. The results demonstrate that our method performs robustly under varying constraints and limited routing resources.

Conclusion

In this paper, we propose a neuralized PCB fanout method based on reinforcement learning named FanoutNet. FanoutNet encodes the spatial and structured features and optimizes fanout results by the PPO algorithm to generate high-quality routing. In addition, we develop a multi-constraint router that equips our FanoutNet to verify the quality of the fanout results and the final routing. We construct extensive experiments on real-world industrial PCB benchmarks. The results demonstrate that our approach significantly improves routability, wire length, and via count compared with state-of-the-art methods.

In future work, we will extend our fanout method to various PCB layout and system in package field.



(a) case1 (b) case2

(c) case4

Figure 3: Case studies

Appendices

Explanation of Terms

- **Topology crossing** between different nets on the same layer, as shown Fig. 1(a), makes wires unable to directly connect these nets.
- **Differential pair** consists of two wires of similar length and routed side-by-side, each of them carries a signal of equal magnitude and opposite polarity.
- **PCB pin** is placed on PCB and connected by wires. Pins have various shapes such as polygons, circles and rectangles.

- **PCB layout** represents the entire designable area on PCB, on which various electronic elements can be placed, such as pins, wires, vias, etc.
- **PCB netlist** is a list of all two-pin nets, each of which has a connection relationship, a signal type, etc.
- **MST for multi-pin net** Given a multi-pin net with a set of pins. We construct an undirected complete graph in which the vertex represents the pin and the edge represents the net. We take the Euclidean distance between every two vertexes as the edge weight. Minimum spanning tree algorithm is applied to decompose the multi-pin net to several two-pin nets, and the edges in the minimum spanning tree connect the decomposed two-pin nets.

Acknowledgments

This work is partially supported by the National Natural Science foundation of China under Grant No. 62002106, the National Key R&D Program of China under Grant No. 2021ZD0114600.

References

Alawieh, M. B.; Li, W.; Lin, Y.; Singhal, L.; Iyer, M. A.; and Pan, D. Z. 2020. High-definition routing congestion prediction for large-scale FPGAs. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 26–31. IEEE.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Chen, J.; Kuang, J.; Zhao, G.; Huang, D. J.-H.; and Young, E. F. 2020. PROS: A plug-in for routability optimization applied in the state-of-the-art commercial EDA tool using deep learning. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 1–8. IEEE.

Chen, Z.; Ji, W.; Peng, Y.; Chen, D.; Liu, M.; and Yao, H. 2021. Machine Learning Based Acceleration Method for Ordered Escape Routing. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 365–370.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kool, W.; Van Hoof, H.; and Welling, M. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Liao, H.; Dong, Q.; Qi, W.; Fallon, E.; and Kara, L. B. 2020a. Track-assignment detailed routing using attention-based policy model with supervision. In *2020 ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD)*, 105–110. IEEE.

Liao, H.; Zhang, W.; Dong, X.; Poczos, B.; Shimada, K.; and Burak Kara, L. 2020b. A deep reinforcement learning approach for global routing. *Journal of Mechanical Design*, 142(6).

Liao, Z.; and Dong, S. 2020. A constraint-driven compact model with partition strategy for ordered escape routing. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 393–398.

Lin, S.-T.; Wang, H.-H.; Kuo, C.-Y.; Chen, Y.; and Li, Y.-L. 2021a. A complete PCB routing methodology with concurrent hierarchical routing. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 1141–1146. IEEE.

Lin, T.-C.; Merrill, D.; Wu, Y.-Y.; Holtz, C.; and Cheng, C.-K. 2021b. A unified printed circuit board routing algorithm with complicated constraints and differential pairs. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 170–175. IEEE.

Lin, Y.; Qu, T.; Lu, Z.; Su, Y.; and Wei, Y. 2021c. Asynchronous Reinforcement Learning Framework and Knowledge Transfer for Net Order Exploration in Detailed Routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Liu, J.; Chen, G.; and Young, E. F. 2021. REST: Constructing Rectilinear Steiner Minimum Tree via Reinforcement Learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 1135–1140. IEEE.

Liu, S.; Sun, Q.; Liao, P.; Lin, Y.; and Yu, B. 2021. Global placement with deep learning-enabled explicit routability optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1821–1824. IEEE.

McMurchie, L.; and Ebeling, C. 2008. PathFinder: A negotiation-based performance-driven router for FPGAs. In *Reconfigurable Computing*, 365–381. Elsevier.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Su, M.; Ding, H.; Weng, S.; Zou, C.; Zhou, Z.; Chen, Y.; Chen, J.; and Chang, Y.-W. 2022. High-Correlation 3D Routability Estimation for Congestion-guided Global Routing. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 580–585. IEEE.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Whitley, D. 1994. A genetic algorithm tutorial. *Statistics and computing*, 4(2): 65–85.

Yan, T.; Ma, Q.; and Wong, M. D. 2012. Advances in PCB routing. *IPSJ Transactions on System LSI Design Methodology*, 5: 14–22.

Yan, T.; and Wong, M. D. 2010. Recent research development in PCB layout. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 398–403. IEEE.