WideGate: Beyond Directed Acyclic Graph Learning in Subcircuit Boundary Prediction

Jiawei Liu[1,2,†], Zhiyan Liu[1], Xun He[1], Jianwang Zhai[1,‡], Zhengyuan Shi[2,3,†], Qiang Xu[2,3], Bei Yu[3], Chuan Shi[1,‡]
[1]Beijing University of Posts and Telecommunications [2] The National Center of Technology Innovation for EDA
[3]The Chinese University of Hong Kong

***Abstract*—Subcircuit boundary prediction is an important application of machine learning in logical analysis, effectively supporting tasks such as functional verification and logic optimization. Existing methods often convert circuits into and-inverter graphs and then use directed acyclic graph neural networks to perform this task. However, two key characteristics of subcircuit boundary prediction do not align with the fundamental assumptions of DAG learning, which limits the model's expressiveness and generalization capabilities. To break these assumptions, we propose WideGate, which includes a receptive field generation module that extends beyond the fanin cone and fanout cone, as well as an adaptive aggregation module that focuses on boundaries. Extensive experiments show that WideGate significantly outperforms existing methods in terms of prediction accuracy and training efficiency for subcircuit boundary prediction. The code is available at https://github.com/BUPT-GAMMA/WideGate.**

I. Introduction

With the rise of deep learning in logical analysis and design, subcircuit boundary prediction has become a well-known basic task with extensive important applications in tasks such as functional verification, malicious logic detection, and macro-block optimization [1]. For example, combining subcircuit boundary prediction with network flow algorithms can be used to identify subcircuits [1], and when integrated with XOR and MAJ tasks, it can be utilized for extracting adder trees [2]. Compared to traditional structural [3] or functional methods [4], approaches that incorporate subcircuit boundary prediction have been proven to significantly enhance efficiency and accuracy in various downstream tasks [1] [2].

Since Boolean circuits can be viewed as directed acyclic graphs (DAGs), existing methods typically adopt DAG-based graph neural networks (GNNs) to handle related tasks, including the subcircuit boundary prediction task. The earliest method is ABGNN [1], which aggregates information within the fanin cone and fanout cone to predict the subcircuit boundaries in netlists, significantly outperforming traditional methods [5]–[7]. Since any Boolean circuit can be converted into an and-inverter graph (AIG), FGNN2 [8] predicts subcircuit boundaries on AIGs and improves ABGNN through unidirectional message passing and pre-training, thereby enhancing the model's functionality modeling capability.

However, are DAG-based GNNs really suitable for the subcircuit boundary prediction task? In Fig. 1(a), we can observe that the output boundaries of the same subcircuit often exhibit strong positional correlations, and capturing these correlations is crucial for accurately identifying subcircuit boundaries. Additionally, as shown in Fig. 1(b), most boundary nodes have non-boundary nodes as their direct neighbors, which is commonly referred to as neighbor heterophily [9]. However, most previous methods hold two prior assumptions of DAG learning: the reachability assumption [10] and the homophily assumption [9]. In the reachability assumption, DAG learning posits that valuable information is often located within the fanin cone [8] or fanout cone [1], thus existing methods overlook the intercorrelations between boundaries. In the homophily assumption, DAG learning assumes that neighboring nodes often belong to the same class, leading existing methods to make the representations of boundary nodes and non-boundary nodes overly similar. As shown in TABLE II, FGNN2 [8] performs poorly in situations with low label ratio, indicating that its expressiveness and generalization capabilities are still limited. Considering that the two important characteristics of boundary intercorrelation and neighbor heterophily contradict the fundamental assumptions of DAG learning, we believe this might be the fundamental reason limiting the predictive capability of existing methods.

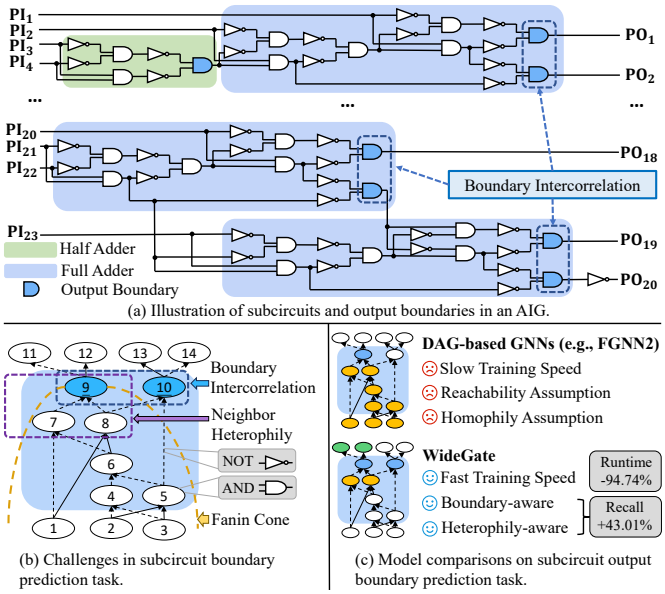


Fig. 1 Illustration, challenges and model comparisons on the subcircuit boundary prediction task.

† This work was partly done as an intern at National Center of Technology Innovation for EDA, P. R. China.
‡ Corresponding author: Jianwang Zhai, Chuan Shi

Therefore, to enhance the model's predictive capability, the key lies in breaking the two assumptions of DAG learning while maintaining structural and functional modeling.

To avoid being constrained by these two assumptions, this paper proposes a novel approach called WideGate. Firstly, to capture the boundary intercorrelation, we break the limitation that reachability can only be searched forward or backward, employing bidirectional search to ensure that the receptive fields of boundary nodes include other boundary nodes in the same subcircuit. During the bidirectional search process, we distinguish the direction information of each edge and incorporate this information into the subsequent GNN learning phase. Secondly, to alleviate the issue of neighbor heterophily, we weaken the influence of non-boundary nodes on boundary nodes during the GNN aggregation process through negative attention, and strengthen the self-influence of boundary nodes via self-aggregation. Finally, we adopt the synchronous message-passing to improve the training speed of the model.

The main contributions are summarized as follows:

- This work is the first to investigate the limitations of DAG-based GNNs in the subcircuit boundary prediction task and identifies two key challenges: boundary intercorrelation and neighbor heterophily.
- To capture the boundary intercorrelation, we propose a receptive field generation module that incorporates bidirectional search, thereby exceeding the scope of the fanin cone or fanout cone.
- To account for neighbor heterophily, we propose an adaptive aggregation module that can integrate more relevant information for boundary nodes.
- We conduct extensive experiments on subcircuit boundary prediction tasks. The experimental results show that WideGate significantly improves prediction accuracy and training efficiency compared to existing methods. For example, as shown in Fig. 1(c), WideGate improves prediction Recall by 43.01% over FGNN2, with a reduction in training time of 94.74%.

II. Preliminaries and Related Works

A. Boolean Networks and And-inverter Graphs

Boolean networks refer to networks composed of logical gates such as AND, OR, and NOT. Due to the ability to transform any Boolean network into AIGs using De Morgan's laws, AIGs are widely used during the logic synthesis and formal verification phases. The properties of AIGs can be characterized from both structural and functional perspectives, which has inspired two traditional approaches to AIG processing: structural hashing [3] [5] and functional propagation [4] [5]. However, these methods face challenges in scalability and are difficult to adapt to modern computing resources [2], which have driven the development of AIG learning [1] [2].

B. DAG Learning for AIG-related Tasks

Due to the DAG nature of AIGs, existing methods typically use DAG-based GNNs to learn node representations. One representative method is DAGNN [11], which defines the reachability scope of message passing based on partial ordering, corresponding to the fanin cone in AIGs. DeepGate [12] builds upon DAGNN by improving it to consider the reconvergent structure characteristics of circuits and applies it to the signal probability prediction task. DeepGate2 [13] further decouples structural representation from functional representation, developing a circuit pretraining model that supports Boolean satisfiability (SAT)-related downstream tasks. FGNN2 [8] integrates DAGNN with contrastive learning for circuit pretraining, significantly outperforming baseline methods like ABGNN [1] in subcircuit boundary prediction task.

However, DAG-based GNNs have some limitations, such as the time-consuming nature [10] of asynchronous message passing and strong assumptions about reachability [10] and homophily [14]. Therefore, recent methods have begun to explore learning architectures that surpass DAG-based GNNs. For example, Gamora [2] combines synchronous message passing with multi-task learning, effectively supporting the adder tree extraction task. HOGA [15] proposes a non-GNN model that transforms GNN message passing into data preprocessing and then uses a hop-wise attention mechanism to learn the intrinsic relationships between nodes at different distances. These methods significantly improve the training efficiency and scalability of GNNs. However, they are not customized for the subcircuit boundary prediction task and ignore its important characteristics.

C. Problem Definition

An AIG consists of AND gates, NOT gates, and the interconnects between them. In an AIG, a subcircuit refers to a functional block that performs simple arithmetic operations, such as full adders and half adders. Typically, our goal is to predict whether each gate is located on the boundary of a subcircuit. Following the setup of previous works [1] [8], we focus on predicting the boundaries of adders, which are the primary functional subcircuits.

Problem 1 (Subcircuit Boundary Prediction)**.** *Given an AIG, predict whether each gate is located on the boundaries of any adder.*

Based on the relationship between the gate's output wire and the subcircuit, this problem can be further divided into two tasks: subcircuit input boundary prediction (IBP) and subcircuit output boundary prediction (OBP).

III. The Proposed WideGate

A. Graph Construction and Feature Initialization Module

Before diving into the detailed algorithm, we introduce the graph construction, feature initialization, and corresponding notation used in this paper. Let $G = (\mathcal{V}, \mathcal{E})$ denote an AIG, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ represents N nodes, and $\mathcal{E}$ denotes the directed edges pointing from source nodes to target nodes. Additionally, all edges collectively form the adjacency matrix $\boldsymbol{A}$, where $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$, and $A_{ij} = 0$ otherwise. Following the setup of related works [2] [8] [15], we define the AIG as a graph consisting only of primary inputs (PIs), AND gates and primary outputs (POs) as nodes, while NOT

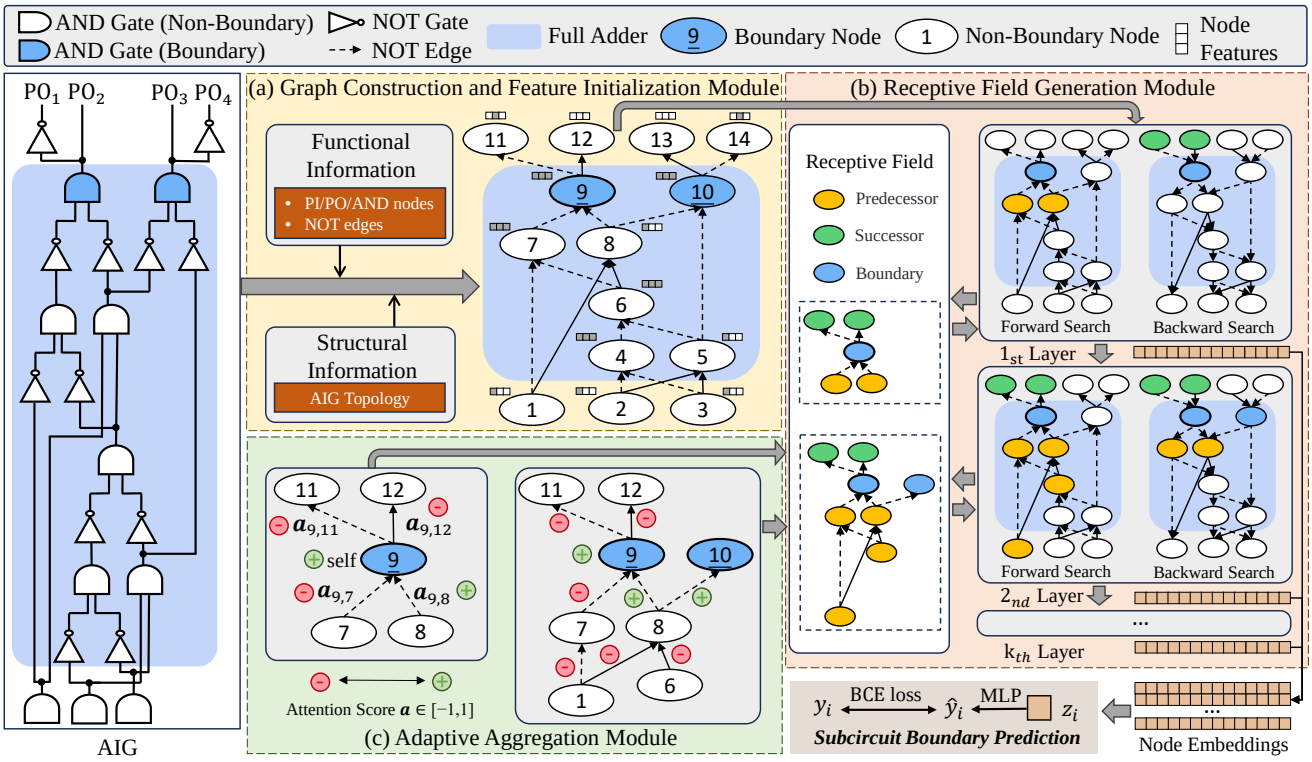


Fig. 2 Illustration of WideGate.

gates are defined as edges. Therefore, we use $A_{ij} = -1$ to indicate a NOT gate on the edge.

As for node features, we follow the approach used by Gamora [2] and HOGA [15] by considering the impact of both node types and edge types. The construction of the 3-dimensional feature vector is as follows: the first dimension records whether the node is an AND node, and the second and third dimensions record whether the two input edges have NOT edges. For example, in Fig. 2(a), node 8 and node 7 are internal nodes with zero and two NOT edge(s), so their feature vectors are [1, 0, 0] and [1, 1, 1], correspondingly. Then, for any node i, the feature vector $\boldsymbol{x}_i$ is mapped to obtain the initialized embedding $\boldsymbol{h}_i^0 = \sigma\left(\boldsymbol{x}_i \boldsymbol{W}^{(0)}\right)$, where σ is an activation function, and $\boldsymbol{W}^{(0)}$ is a parameter matrix.

B. Receptive Field Generation Module: Beyond Fanin Cones

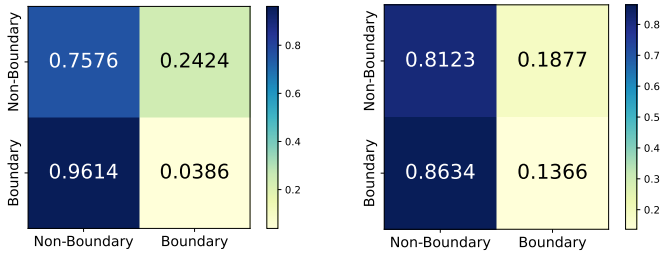
In Fig. 1(a), we observe that multiple boundary nodes of the same subcircuit are often not within each other's fanin cone or fanout cone but share a special positional correlation. To leverage this boundary intercorrelation, we design a specialized receptive field generation module as shown in Fig. 2(b).

The core of WideGate's receptive field generation method lies in bidirectional search. Existing GNN methods only perform forward or backward search [1] [8], which confines their receptive fields to the fanin cone or fanout cone. In contrast, WideGate conducts both forward and backward searches in each step, thereby efficiently placing multiple boundary nodes within each other's receptive fields. For example, in Fig. 2(b), node 9 first obtains information about node 8 through a forward search and then gets information about node 10 through a backward search. Similarly, node 10 needs only a forward followed by a backward search to obtain information about node 9. Note that bidirectional search is not the same as non-directional search. For instance, when using non-directional search, node 6 (non-boundary node) and node 10 (boundary node) would be equally considered second-order neighbors of node 9, thus making it impossible to distinguish between them during the GNN learning process. Bidirectional search, however, records that node 6 is reached through two forward searches, whereas node 10 is reached through one forward search and one backward search.

Additionally, bidirectional search can be efficiently integrated into the GNN learning framework through distinguishable matrix operations and parameter matrices. For clarity, we mark $\boldsymbol{A}$ as $\boldsymbol{A}_{\rightarrow}$, and its transposed matrix $\boldsymbol{A}_{\rightarrow}^{\top}$ as $\boldsymbol{A}_{\leftarrow}$. Without loss of generality, we use $\boldsymbol{h}_{i\rightarrow}^{(1)} = \boldsymbol{A}_{\rightarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\rightarrow}^{(1)}$ and $\boldsymbol{h}_{i\leftarrow}^{(1)} = \boldsymbol{A}_{\leftarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\leftarrow}^{(1)}$ [16] to denote information aggregated from the two directions, where $\boldsymbol{W}_{\rightarrow}^{(1)}$ and $\boldsymbol{W}_{\leftarrow}^{(1)}$ are different parameter matrices. Then, we sum them to serve as the input $\boldsymbol{h}_i^{(1)} = \boldsymbol{h}_{i\rightarrow}^{(1)} + \boldsymbol{h}_{i\leftarrow}^{(1)}$ for the next layer of the GNN. This update process can be seen as performing forward and backward searches for the target node i, and it can be extended to multi-layer GNNs. For example, after two layers of GNN, the node representations can be expanded as follows:

$$\begin{aligned}\boldsymbol{h}_i^{(2)} = {} & \boldsymbol{A}_{\rightarrow}\boldsymbol{A}_{\rightarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\rightarrow}^{(1)}\boldsymbol{W}_{\rightarrow}^{(2)} + \boldsymbol{A}_{\leftarrow}\boldsymbol{A}_{\leftarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\leftarrow}^{(1)}\boldsymbol{W}_{\leftarrow}^{(2)} \\ & + \boldsymbol{A}_{\leftarrow}\boldsymbol{A}_{\rightarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\rightarrow}^{(1)}\boldsymbol{W}_{\leftarrow}^{(2)} + \boldsymbol{A}_{\rightarrow}\boldsymbol{A}_{\leftarrow}\boldsymbol{h}_i^{(0)}\boldsymbol{W}_{\leftarrow}^{(1)}\boldsymbol{W}_{\rightarrow}^{(2)}.\end{aligned} \tag{1}$$

Note that four terms in Equation (1) correspond to the four combinations of performing forward or backward searches in two steps. In other words, the first two terms contain



(a) Input Boundary CM (b) Output Boundary CM

Fig. 3 The compatibility matrix (CM) and the neighbor heterophily in subcircuit boundary prediction task.

information from the fanin cone and fanout cone, while the latter two terms can capture information beyond the fanin cone and fanout cone. Additionally, since the bidirectional searches for each node do not interfere with each other, this process can be executed in parallel. This significantly improves training efficiency compared to FGNN2 [8], which requires each node to wait for updates from its predecessors.

C. Adaptive Aggregation Module: Focusing on Boundaries

As shown in Fig. 3, neighbor heterophily is another important characteristic of the subcircuit boundary prediction task. However, based on the homophily assumption [9], existing methods typically use averaging [1] or weighted averaging based on edge types [8], which makes the representations of boundary nodes too similar to those of their non-boundary neighbors. To make the representations of boundary nodes more distinguishable from non-boundary nodes, we break the homophily assumption of GNNs, thereby allowing boundary nodes to focus more on information from other boundary nodes, as shown in Fig. 2(c).

Firstly, since boundary nodes are certainly of the same category as itself, we incorporate self-information aggregation at each layer of the GNN, which can reduce the risk of the node being affected by its non-boundary neighbors. Secondly, because the categories of neighboring nodes are unknown, we employ a negative attention mechanism to adaptively adjust the influence of neighbor information, making it more likely for boundary node information to affect other boundary nodes. Unlike traditional attention mechanisms where scores range from 0 to 1, our negative attention scores range from -1 to 1, which has been proven effective for heterophilic graphs from a spectral graph theory perspective [14].

Considering both boundary intercorrelation and neighbor heterophily in the aggregation process of WideGate, we define the following three messages $\boldsymbol{m}$:

$$\begin{aligned}
\boldsymbol{m}_i^{(l)} &= \boldsymbol{h}_i^{(l-1)}, \\
\boldsymbol{m}_{\overrightarrow{\mathcal{N}_i}}^{(l)} &= \sum_{j \in \overrightarrow{\mathcal{N}_i}} \frac{\tanh\left(\boldsymbol{a}^\top \left[\boldsymbol{h}_i^{(l-1)}, \boldsymbol{h}_j^{(l-1)}\right]\right)}{\sqrt{d_i d_j}} \boldsymbol{h}_j^{(l-1)}, \\
\boldsymbol{m}_{\overleftarrow{\mathcal{N}_i}}^{(l)} &= \sum_{j \in \overleftarrow{\mathcal{N}_i}} \frac{\tanh\left(\boldsymbol{a}^\top \left[\boldsymbol{h}_i^{(l-1)}, \boldsymbol{h}_j^{(l-1)}\right]\right)}{\sqrt{d_i d_j}} \boldsymbol{h}_j^{(l-1)},
\end{aligned} \tag{2}$$

where d_i represents the degree of node i, $\boldsymbol{a}$ is a learnable attention score, and $\tanh(\cdot)$ maps the attention score to the range $[-1, 1]$. Then, the complete message-passing function of WideGate is as follows:

$$\boldsymbol{h}_i^{(l)} = \sigma\left(\boldsymbol{m}_i^{(l)} \boldsymbol{W}_{self}^{(l)} + \beta \boldsymbol{m}_{\overrightarrow{\mathcal{N}_i}}^{(l)} \boldsymbol{W}_{\rightarrow}^{(l)} + (1-\beta) \boldsymbol{m}_{\overleftarrow{\mathcal{N}_i}}^{(l)} \boldsymbol{W}_{\leftarrow}^{(l)}\right), \tag{3}$$

where β is a hyperparameter between 0 and 1, defaulting to 0.5. $\boldsymbol{W}_{self}^{(l)}$, $\boldsymbol{W}_{\rightarrow}^{(l)}$ and $\boldsymbol{W}_{\leftarrow}^{(l)}$ are three different parameter matrices in the l-th layer, and σ is an activation function.

After L layers of aggregation, the embeddings of all layers are used to generate the prediction value $\hat{y}_i$ for any node i:

$$\hat{y}_i = \sigma\left(\left[\boldsymbol{h}_i^{(0)}, \ldots, \boldsymbol{h}_i^{(L)}\right] \boldsymbol{W}_{pred}\right), \tag{4}$$

where σ is an activation function, $\boldsymbol{W}_{pred}$ is a parameter matrix.

Considering that the labels for boundary nodes are 1 and for non-boundary nodes are 0, we use the binary cross-entropy (BCE) loss $\mathcal{L}$ to train the model:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)]. \tag{5}$$

TABLE I The statistics of AIG datasets.

	Architectures		#
Adder	Brent-Kung Cond-Sum Hybrid Koggle-Stone Ling Sklansky	Sklansky Block Carry Look-head Carry Look-head Carry Select Carry-skip Ripple-Carry	1173
Subtractor	Hybrid Koggle-Stone Ling	Brent-Kung Cond-Sum Sklansky	645
Multiplier	Array Booth-Encoding Wallace Dadda	Overturned-stairs (4,2) compressor (7,3) counter Redundant binary addition	3788
Divider	Array		1180
Total	/		6786

IV. Experiments

In this section, we investigate the performance of WideGate. First, we introduce the evaluation tasks, metrics, datasets, and baselines. Then, we address the following questions:

- (Section IV-C) Can WideGate support various subcircuit boundary prediction tasks?
- (Section IV-D) Does WideGate perform well with different magnitudes of training data?
- (Section IV-E) Does WideGate perform well with different aggregation layers?
- (Section IV-F) Do each of the modules in WideGate contribute positively to the final model output?

A. Evaluation Tasks and Metrics

To evaluate the model's performance on the subcircuit boundary prediction task, we conduct experiments on both IBP and OBP tasks. We use two evaluation metrics from FGNN2 [8]: Recall and F1-score. These two metrics focus only on boundary nodes, which is most meaningful for our task because they are not affected by the accuracy of non-boundary nodes. Additionally, the average training time per epoch (i.e., Avg. Runtime) is used to evaluate the model's training speed.

TABLE II Results on the subcircuit output boundary prediction (OBP) task. (Label ratio=5%)

Metrics	GraphSAGE	HOGA	RelGCN	FGNN2	FGNN2+Pretrain	WideGate	Improvement (%)
Recall ↑	0.6298	0.3900	0.6448	0.5544	0.6806	**0.9733**	43.01%
F1 score ↑	0.7128	0.4925	0.7223	0.6624	0.7522	**0.9769**	29.87%
TNR ↑	0.9716	0.9599	0.9710	0.9753	0.9733	**0.9960**	2.12%
Accuracy ↑	0.9131	0.8623	0.9151	0.9032	0.9232	**0.9921**	7.46%
Avg. Runtime (s) ↓	1.2705	2.3797	1.8795	65.3075	63.9194	**3.1669**	94.74%

TABLE III Results on the subcircuit input boundary prediction (IBP) task. (Label ratio=5%)

Metrics	GraphSAGE	HOGA	RelGCN	FGNN2	FGNN2+Pretrain	WideGate	Improvement (%)
Recall ↑	0.8271	0.7040	0.8297	0.8047	0.8781	**0.9935**	13.14%
F1 score ↑	0.8115	0.7623	0.8137	0.8003	0.8412	**0.9938**	18.14%
TNR ↑	0.9502	0.9662	0.9506	0.9514	0.9506	**0.9986**	3.35%
Accuracy ↑	0.9267	0.9162	0.9276	0.9234	0.9367	**0.9976**	6.50%
Avg. Runtime (s) ↓	1.3063	2.4101	1.8993	68.5559	67.8344	**3.1541**	94.85%

Moreover, to provide a more comprehensive comparison, we introduce two additional evaluation metrics related to non-boundary nodes: True Negative Rate (TNR) and Accuracy. TNR calculates the proportion of non-boundary nodes that are correctly predicted out of all non-boundary nodes, while Accuracy measures the proportion of nodes that are correctly predicted out of all nodes. In other words, TNR considers only non-boundary nodes, whereas Accuracy considers both boundary and non-boundary nodes.

B. Datasets and Baselines

We use an extended version of the FGNN2 dataset [8], including adders, subtractors, multipliers, and dividers. These circuits are synthesized into gate-level netlists using Synopsys Design Compiler and then converted into AIG format using ABC [17]. The word lengths of circuits range from 8 bits to 32 bits, and the number of nodes varies from a few hundred to several thousand. The structures and functionalities of these circuits span a wide range, making them highly challenging. The detailed information is provided in TABLE I.

For the baseline models, we choose the current state-of-the-art model FGNN2 [8] for this task, as well as the strongest baseline model from the FGNN2 paper, RelGCN [18]. Additionally, we add the classic GraphSAGE [19] model and the latest generalizable circuit learning model HOGA [15] as baselines. For FGNN2, we compare two versions. FGNN2 refers to the model trained directly on the training data without considering pretraining. FGNN2+Pretrain indicates further fine-tuning on the training set using the open-sourced pretrained FGNN2 model [8]. FGNN2 uses asynchronous message passing and does not require setting the number of layers. For all other models, the number of layers defaults to 8, and the final node embedding dimension is 256. All models are trained using an early stopping strategy on a Linux server equipped with an NVIDIA 3090 GPU, with a maximum of 200 epochs and the patience is set to 50 epochs.

C. Main Results

TABLE II presents the experimental results of the OBP task. We have the following observations: Firstly, WideGate demonstrates the best results across two commonly used metrics (Recall, F1-score). Compared to the strongest baseline (FGNN2+Pretrain), WideGate improves Recall by 43.01% and F1-score by 29.87%, while reducing the average training time by 94.74%. This showcases the effectiveness and efficiency of WideGate. Secondly, almost all models perform well in terms of TNR and Accuracy, because predicting non-boundary nodes is much easier than predicting boundary nodes, and non-boundary nodes constitute a large proportion of all nodes. In some cases, the model's ability to predict non-boundary nodes and boundary nodes cannot be achieved simultaneously. For example, HOGA performs well on non-boundary nodes (TNR=0.9599) but poorly on boundary nodes (F1-score=0.4925). The pretraining step in FGNN2 improves the F1-score but at the cost of a slight decrease in TNR. In contrast, WideGate performs best in predicting both non-boundary and boundary nodes, reflecting the model's versatility.

Similarly, TABLE III presents the results of the IBP task. Based on the results, we have the following observations. Firstly, similar to the OBP task, WideGate surpasses the baseline models across all metrics. Specifically, for the commonly used evaluation metrics Recall and F1-score, WideGate achieves improvements of 13.14% and 18.14% compared to FGNN2+Pretrain, with prediction performance approaching 100% (F1-score=99.38%). This demonstrates the high practicality of WideGate. Secondly, although input boundaries exhibit stronger heterophily compared to output boundaries as shown in Fig. 3, the proportion of input boundary nodes within the entire circuit (19.07%) is higher than that of output nodes (17.14%). This means that more positive class nodes are available for training, resulting in overall better performance of all models on the IBP task compared to the OBP task.

D. Impact of Label Ratio

In the EDA field, obtaining labeled data is often challenging, so the model's performance under low label ratios is crucial. In addition to the default ratio of 0.05, we set three label ratios for training data: 0.01, 0.02, and 0.10. The ratio of validation data is the same as training data, with the remaining data

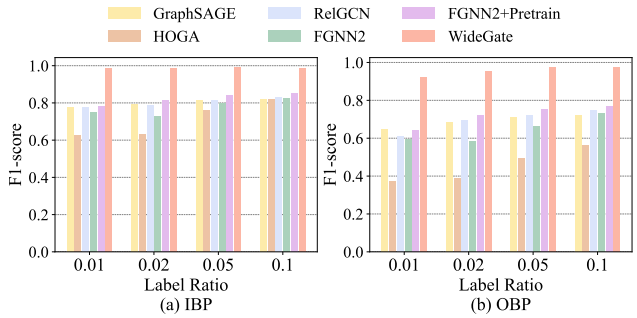


Fig. 4 Experimental results on different label ratios.

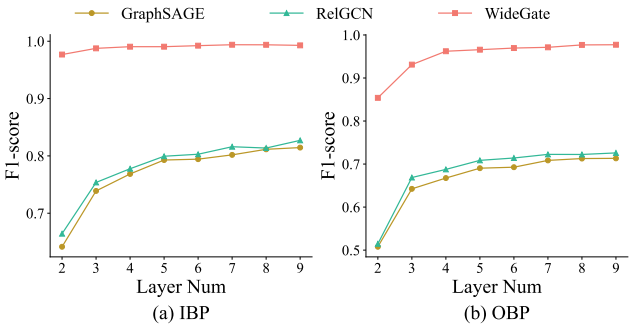


Fig. 5 Experimental results on different GNN layers.

serving as the test set.

Based on the experimental results in Fig. 4, we make the following observations. Firstly, WideGate exhibits stable and optimal performance, with an F1-score significantly higher than the baseline models. Secondly, WideGate's advantage is more pronounced at lower label ratios. For a label ratio of 0.01, WideGate outperforms the best baseline model by 25.78% on the IBP task and by 42.60% on the OBP task. Finally, WideGate surpasses the results of other models trained with 10% of the training data using only 1% of the training data, demonstrating the model's strong generalization ability.

E. Impact of Aggregation Layers

To capture information over a broader range, the number of layers in GNN models can be increased. However, increasing the number of GNN layers also increases the training time, and the choice of GNN layers often involves balancing efficiency and accuracy. As shown in TABLE III and TABLE II, despite WideGate achieving higher accuracy, its training time at the same number of layers (8 layers) is slightly longer than that of some GNN models (e.g., GraphSAGE and RelGCN), which could limit its applicability. Therefore, we examine the performance of GNN models at different numbers of layers to investigate whether WideGate can also achieve good performance with fewer layers.

We conduct experiments on both IBP and OBP tasks at a label rate of 0.05 and place the F1-score results in Fig. 5. Our observations are as follows: Firstly, WideGate achieves significantly higher F1-scores than GraphSAGE and RelGCN for various layer settings, demonstrating the superiority of WideGate's message-passing mechanism. Secondly, although

TABLE IV Ablation results on OBP and IBP tasks.

	OBP	IBP
WideGate (w/o forward search)	0.6960	0.8042
WideGate (w/o backward search)	0.7898	0.9842
WideGate (w/o self-aggregation)	0.9665	0.9910
WideGate	**0.9769**	**0.9938**

GraphSAGE is the most efficient model at the same number of layers, a 2-layer WideGate performs much better than a 9-layer GraphSAGE. Taking the OBP task as an example, the average runtime of a 2-layer WideGate is just 1.3401 seconds, even faster than GraphSAGE (1.3873 seconds), and the F1-score is much higher than that of GraphSAGE. This means that even in scenarios where training speed is critically important, WideGate is a better choice than GraphSAGE.

F. Ablation Study

To verify the role of each module, TABLE IV gives the results of the ablation study. Consistent with TABLE III and TABLE II, we continue to use a label rate of 0.05 and an 8-layer model setting. We propose three model variants, each removing one of the following: forward search (setting $\beta = 0$ in Equation (3)), backward search (setting $\beta = 1$ in Equation (3)) and self-aggregation (removing the first term $\boldsymbol{m}_i^{(l)}\boldsymbol{W}_{self}^{(l)}$ in Equation (3)).

From the experimental results, we make the following observations: Firstly, the complete model achieves the best performance, confirming the value of each module. Secondly, forward search is essential for both tasks, but backward search plays a significantly larger role in the OBP task than the IBP task. Finally, even when backward search is removed, the model still surpasses the baseline models, which means the adaptive aggregation module enhances the performance.

V. Conclusion

Predicting subcircuit boundaries is crucial for several EDA tasks in logical analysis and design, e.g., logic detection and logic optimization. This paper identifies two fundamental shortcomings in existing GNN models when handling this task: the difficulty in considering boundary intercorrelation and neighbor heterophily, which severely limits the models' performance. To address these challenges, we propose a novel GNN model, WideGate, incorporating a receptive field generation module and an adaptive aggregation module. We conduct extensive experiments on subcircuit boundary prediction tasks, and the results validate WideGate's effectiveness, efficiency, versatility, flexibility, and generalization capabilities.

Acknowledgment

This work was supported by the National Key R&D Program of China (2022YFB2901100), the National Natural Science Foundation of China (No. U20B2045, U1936220, 62192784, 62172052, 62002029, 61772082), the Beijing Natural Science Foundation (No. 4244107, QY24204), the General Research Fund of the Hong Kong Research Grants Council (No. 14212422), and the Research Grants Council of Hong Kong SAR (No. CUHK14210723, CUHK14211824).

REFERENCES

[1] Z. He, Z. Wang, C. Bai, H. Yang, and B. Yu, "Graph learning-based arithmetic block identification," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–8.

[2] N. Wu, Y. Li, C. Hao, S. Dai, C. Yu, and Y. Xie, "Gamora: Graph learning based symbolic reasoning for large-scale boolean networks," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[3] W. Li, A. Gascon, P. Subramanyan, W. Y. Tan, A. Tiwari, S. Malik, N. Shankar, and S. A. Seshia, "Wordrev: Finding word-level structures in a sea of bit-level gates," in *2013 IEEE international symposium on hardware-oriented security and trust (HOST)*. IEEE, 2013, pp. 67–74.

[4] A. Gascón, P. Subramanyan, B. Dutertre, A. Tiwari, D. Jovanović, and S. Malik, "Template-based circuit understanding," in *2014 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 2014, pp. 83–90.

[5] P. Subramanyan, N. Tsiskaridze, W. Li, A. Gascón, W. Y. Tan, A. Tiwari, N. Shankar, S. A. Seshia, and S. Malik, "Reverse engineering digital circuits using structural and functional analyses," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 1, pp. 63–80, 2013.

[6] X. Wei, Y. Diao, T.-K. Lam, and Y.-L. Wu, "A universal macro block mapping scheme for arithmetic circuits," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1629–1634.

[7] A. Fayyazi, S. Shababi, P. Nuzzo, S. Nazarian, and M. Pedram, "Deep learning-based circuit recognition using sparse mapping and level-dependent decaying sum circuit representations," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 638–641.

[8] Z. Wang, C. Bai, Z. He, G. Zhang, Q. Xu, T.-Y. Ho, Y. Huang, and B. Yu, "Fgnn2: A powerful pre-training framework for learning the logic functionality of circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[9] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in neural information processing systems*, vol. 33, pp. 7793–7804, 2020.

[10] Y. Luo, V. Thost, and L. Shi, "Transformers over directed acyclic graphs," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[11] V. Thost and J. Chen, "Directed acyclic graph neural networks," in *International Conference on Learning Representations (ICLR)*, 2021.

[12] M. Li, S. Khan, Z. Shi, N. Wang, H. Yu, and Q. Xu, "DeepGate: Learning neural representations of logic gates," in *ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 667–672.

[13] Z. Shi, H. Pan, S. Khan, M. Li, Y. Liu, J. Huang, H.-L. Zhen, M. Yuan, Z. Chu, and Q. Xu, "DeepGate2: Functionality-aware circuit representation learning," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[14] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 3950–3957.

[15] C. Deng, Z. Yue, C. Yu, G. Sarar, R. Carey, R. Jain, and Z. Zhang, "Less is more: Hop-wise graph attention for scalable and generalizable learning on circuits," in *DAC*, 2024.

[16] E. Rossi, B. Charpentier, F. Di Giovanni, F. Frasca, S. Günnemann, and M. M. Bronstein, "Edge directionality improves learning on heterophilic graphs," in *Learning on Graphs Conference*. PMLR, 2024, pp. 25–1.

[17] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *Computer Aided Verification: 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*. Springer, 2010, pp. 24–40.

[18] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer, 2018, pp. 593–607.

[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.