FuncGNN: Learning Functional Semantics of Logic Circuits with Graph Neural Networks

QIYUN ZHAO

As integrated circuit scale grows and design complexity rises, effective circuit representation helps support logic synthesis, formal verification, and other automated processes in electronic design automation. And-Inverter Graphs (AIGs), as a compact and canonical structure, are widely adopted for representing Boolean logic in these workflows. However, the increasing complexity and integration density of modern circuits introduce structural heterogeneity and global logic information loss in AIGs, posing significant challenges to accurate circuit modeling. To address these issues, we propose FuncGNN, which integrates hybrid feature aggregation to extract multi-granularity topological patterns, thereby mitigating structural heterogeneity and enhancing logic circuit representations. FuncGNN further introduces gate-aware normalization that adapts to circuit-specific gate distributions, improving robustness to structural heterogeneity. Finally, FuncGNN employs multi-layer integration to merge intermediate features across layers, effectively synthesizing local and global semantic information for comprehensive logic representations. Experimental results on two logic-level analysis tasks (i.e., signal probability prediction and truth-table distance prediction) demonstrate that FuncGNN outperforms existing state-of-the-art methods, achieving improvements of 2.06% and 18.71%, respectively, while reducing training time by approximately 50.6% and GPU memory usage by about 32.8%.

Additional Key Words and Phrases: Electronic Design Automation, Circuit Representation, And-Inverter Graphs, Graph Neural Network

ACM Reference Format:
Qiyun Zhao. 2025. FuncGNN: Learning Functional Semantics of Logic Circuits with Graph Neural Networks. *J. ACM* 1, 2, Article 3 (May 2025), 21 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

As the scale of integrated circuits continues to expand and the complexity of designs keeps increasing, researching how to efficiently and accurately model and analyze circuits can help the field of Electronic Design Automation (EDA) achieve circuit design optimization, verification, and the optimization of automated processes [34]. In the field of EDA, modeling complex logic circuits requires efficient and scalable representations, typically provided by Boolean networks. To address this need, AIGs simplify these networks into a compact, canonical form using two-input AND gates and inverters. By reducing the complexity of logic function processing, AIGs facilitate efficient logic synthesis, formal verification, and logic simulation, and thus serve as the standard intermediate representation in modern EDA workflows [19]. This canonical structure supports scalable processing by standardizing logic representations across diverse EDA tasks. By capturing both structural and functional information, AIGs enable advanced applications, including circuit optimization, equivalence checking, and logic representation learning in deep learning-based EDA frameworks [14, 22, 31].

Author's address: Qiyun Zhao, xiaozhao_666@dlmu.edu.cn.

0004-5411/2025/5-ART3 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Conventional AIG processing methods, including structural hashing [27], functional propagation, and functionally reduced AIGs (FRAIGs) [20], have enabled efficient logic synthesis and formal verification in EDA. However, these heuristic-based methods often struggle to capture the structural and functional complexities of large-scale logic circuits, leading to scalability and efficiency bottlenecks. This growing gap between the capabilities of traditional methods and the demands of modern circuit designs underscores the need for innovative AIG representation learning methods [10, 22].

To address these limitations, recent advances in deep learning, particularly Graph Neural Networks [23] (GNNs), have opened new avenues for circuit representation learning. GNNs excel at adaptively capturing complex dependencies between nodes in circuit graphs, enabling low-dimensional vector representations of logic gates. These representations have shown remarkable performance in various EDA tasks, including congestion prediction, power estimation, and testability analysis. In the context of AIG representation learning, GNN-based methods have gained traction by transforming circuit netlists into logic representations. For instance, DeepGate [17] uses AIG-based representation with signal probability supervision for functional modeling. Similarly, GAMORA [29] employs a multi-task framework to model circuit structure and logic function for gate-level recognition. Additionally, PolarGate [18] introduces polarity embeddings and logic operators to enhance message passing. Furthermore, DeepGate3 [25] combines GNNs and transformers with circuit-specific pretraining to enhance scalability.

Although existing methods have made notable progress, the growing integration density and functional complexity of modern integrated circuits have led to increasingly intelligent and large-scale logic designs. These trends introduce new demands on structural modeling and functional abstraction, giving rise to the following two challenges for existing methods:

Challenge 1: Facing structural heterogeneity in AIGs. Due to the wide variation in circuit complexity and gate arrangements in real-world scenarios, AIGs exhibit significant structural heterogeneity with diverse topologies and uneven gate distributions. Such heterogeneity complicates capturing consistent structural characteristics for AIG representation. Due to significant structural heterogeneity, existing methods struggle to capture consistent characteristics, making it difficult for existing GNN-based methods to generate stable and reliable representations [4, 15, 33]. Thus, achieving effective AIG representation to overcome structural heterogeneity remains a critical challenge.

Challenge 2: Lacking efficient capture of global structural information in AIGs. With the increasing integration density of modern Field Programmable Gate Arrays (FPGA) designs, accurate circuit analysis increasingly depends on understanding global logic interactions across the entire AIG [2, 32]. However, existing methods primarily rely on local message-passing mechanisms, which limits their ability to model long-range logic dependencies across large-scale AIGs. This often results in the loss of crucial global logic context, leading to reduced accuracy and limited scalability in downstream EDA tasks. While some methods introduce attention mechanisms to enhance global context modeling, they often suffer from significant computational overhead, making them inefficient for large-scale circuit analysis. Therefore, developing efficient and scalable representation methods that preserve and utilize global logic information remains a critical challenge.

To address these challenges, we propose FuncGNN, which combines local and global logic information to effectively address structural heterogeneity and global information loss, thereby enabling more efficient and accurate extraction of AIG logic semantics, which benefits EDA workflows such as design optimization and verification. Specifically, FuncGNN comprises three key components: the Hybrid Feature Aggregation Component, the Global Context Normalization Component and the Multi-Layer Integration Component.

The **Hybrid Feature Aggregation Component** integrates diverse topological patterns, enhancing circuit property estimation by addressing AIG structural diversity. This approach improves logic representation accuracy, enabling robust analysis of heterogeneous circuit structures. To further refine logic representations, the **Global Context Normalization Component** balances circuit-wide gate proportions, improving estimation accuracy. This mitigates uneven gate distributions, addressing structural heterogeneity (thus addressing Challenge 1). Building on this, the **Multi-Layer Integration Component** synthesizes comprehensive logic relationships across the circuit, enhancing estimation accuracy. This captures global logic information, addressing information loss (thus addressing Challenge 2).

To effectively evaluate the model performance, we conducted experiments on a large-scale dataset of 9,933 AIG samples from four circuit benchmark suites to assess FuncGNN's performance in circuit property estimation. The results show FuncGNN's Signal Probability Prediction (SPP) accuracy and Truth-Table Distance Prediction (TTDP) improve by 2.06% and 18.71%, respectively, over the state-of-the-art methods. Additionally, FuncGNN enhances computational efficiency, improving logic representation for accurate circuit property estimation.

Overall, the main contributions of this work are threefold:

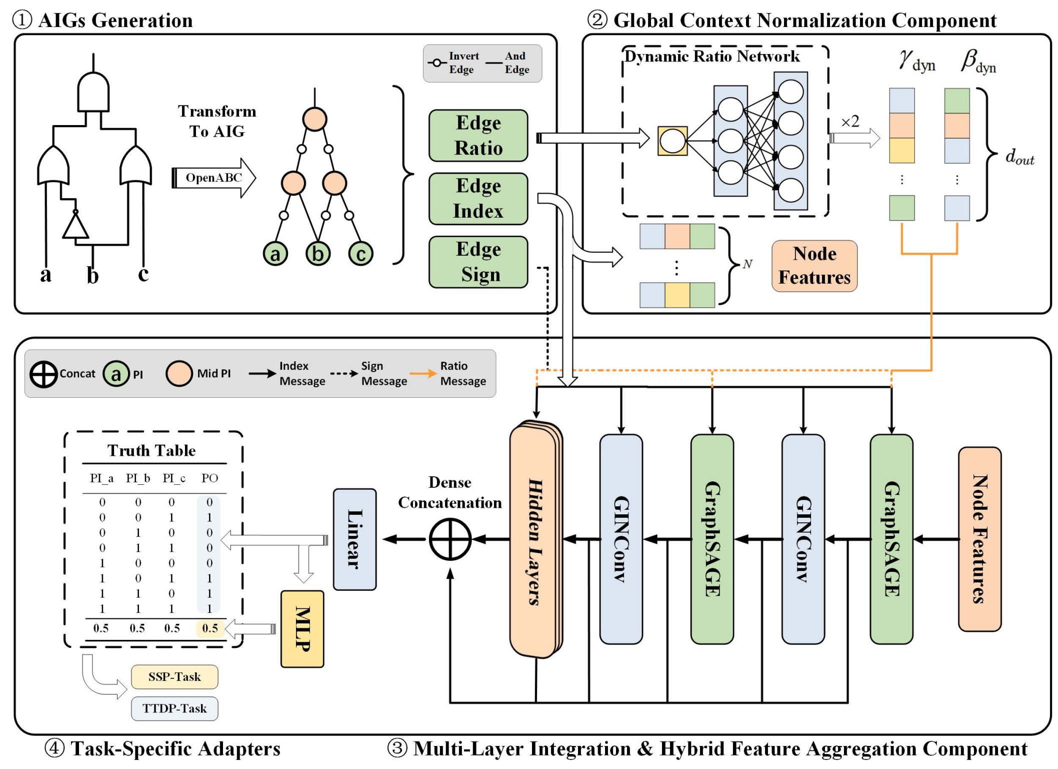
- We introduce an innovative architecture tailored for functional representation learning on AIGs. This method leverages structural and logical properties of AIGs to capture functional dependencies effectively, addressing the over-squashing problem commonly encountered in deep AIG processing. By incorporating task-specific feature propagation, our method ensures robust learning of complex circuit behaviors, improving performance in tasks like SPP and TTDP.
- Extensive experiments demonstrate that FuncGNN achieves superior performance on heterogeneous AIGs, reducing loss by 2.06% on the SPP task and 18.71% on the TTDP task compared to state-of-the-art methods. Compared to the best models with equivalent parameter counts, FuncGNN maintains slightly higher accuracy while reducing memory usage and runtime by 32.8% and 50.6%, effectively addressing learning difficulties in AIGs.
- We release the complete code of FuncGNN and the evaluation dataset to promote reproducibility and facilitate further investigation [11].

2 Related Work

As circuit scale and structural complexity continue to grow, circuit representation learning has become a key enabler in EDA. In this section, we categorize existing methods into two representative groups: attention-based approaches and message-passing-based approaches. We review their core designs, advantages, and limitations. This categorization clarifies the technical motivations underlying the design of FuncGNN.

2.1 Attention-Based Methods

These methods capture long-range dependencies and structural patterns by leveraging attention mechanisms or structure-aware graph networks. DeepGate [17] introduced AIG-based circuit representation with signal probability supervision, achieving functional-aware learning but with high computational cost. DeepGate2 [24] improved upon this by incorporating truth-table supervision and a more expressive single-round GNN, which enhanced both modeling precision and training efficiency. DeepGate3 [25] further extended the framework by combining GNNs and transformers with circuit-specific pretraining strategies, greatly improving scalability and generalization, albeit with increased resource requirements. HOGA [9] proposed hop-aware aggregation and gated attention mechanisms, offering better scalability for large circuits, though it still struggled with modeling global contextual dependencies.



PI_a	PI_b	PI_c	PO
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0.5	0.5	0.5	0.5

Fig. 1. The overall framework of FuncGNN

2.2 Message-Passing Methods

Message-passing models emphasize local propagation and structural coherence through lightweight architectures. PolarGate [18] introduced polarity embeddings and differentiable logic operators to enrich the message flow. While effective in certain tasks, it has difficulty generalizing across diverse AIG topologies. GAMORA [29] proposed a multi-task framework that jointly models circuit structure and logic function, improving gate-level recognition. However, its capacity to represent global logic information remains limited.

Despite their advances, current methods exhibit several limitations. Attention-based models often involve substantial computational overhead, limiting scalability to very large circuits. Message-passing approaches, though efficient, tend to lose global context and suffer from information degradation across deeper layers. Additionally, both categories face challenges in modeling the structural heterogeneity typical of AIG circuits. These issues reduce estimation accuracy and affect generalization across tasks.

To address these challenges, FuncGNN introduces a novel architecture that integrates Hybrid Feature Aggregation, Global Context Normalization, and Multi-Layer Integration. These components are designed to maintain computational efficiency while improving functional accuracy and structural adaptability. Together, they enable robust and scalable AIG representation learning, offering support for downstream EDA tasks such as logic synthesis and equivalence checking.

3 FuncGNN Model

3.1 Overview

The overall framework of FuncGNN is illustrated in Figure 1. FuncGNN is composed of three main components: the Hybrid Feature Aggregation Component, the Global Context Normalization Component and the Multi-Layer Integration Component.

The workflow begins with converting circuits into a unified AIG format using the open-source tool ABC [5], yielding a standardized Boolean logic representation. Structural and logic information is then extracted from the AIGs to construct the model input. FuncGNN operates on three primary inputs: a node feature matrix, an edge tensor encoding logical relations (AND as +1, NOT as −1), and a global prior capturing the circuit-specific ratio of AND-to-NOT edges. This input design enhances robustness against structural heterogeneity in AIGs by capturing diverse circuit topologies and gate distributions.

The node features are initially projected into a shared embedding space, while the edge tensor guides signal propagation and logic-aware aggregation during message passing.

To address structural heterogeneity, the **Hybrid Feature Aggregation Component** integrates two complementary feature extraction mechanisms with different granularities, alternating between them to enhance local and global information capture. This dual design captures both local circuit patterns and node-level logic differences across varying topologies, effectively mitigating instability caused by circuit-specific gate distributions.

Further, the **Global Context Normalization Component** introduces a gate-aware normalization mechanism that integrates the AND-to-NOT ratio as a global prior. This module stabilizes training across diverse AIGs by reducing variance in feature distributions, further improving adaptability to heterogeneous circuit structures.

To alleviate the loss of global structural information, the **Multi-Layer Integration Component** performs multi-granularity logic information concatenation, preserving intermediate outputs from all processing stages. These are fused through a fully connected layer enabling the integration of logic information at multi-granularity across different structural depths. Furthermore, since the concatenation operation is computationally lightweight, this method maintains high efficiency while enhancing sensitivity to global logic patterns.

By combining these components, FuncGNN enables accurate, robust, and scalable circuit representation learning,and demonstrates superior performance over state-of-the-art methods on logic-level analysis tasks such as SPP and TTDP.

3.2 Hybrid Feature Aggregation Component

This section presents the architecture and operational workflow of the Hybrid Feature Aggregation Component, detailing its design principles for addressing structural heterogeneity in AIGs and supporting robust logic representation.

AIGs exhibit significant structural heterogeneity, such as inconsistent AND-to-NOT gate ratios and diverse circuit topologies, which complicate the task of consistent feature extraction for logic synthesis and equivalence checking. To address this, the Hybrid Feature Aggregation Component combines GraphSAGE [12]-based neighborhood aggregation with GINConv [30]-based nonlinear enhancement; its overall architecture is shown in Figure 2.

In particular, this component processes preprocessed graph data represented by a node feature matrix, an edge index matrix, and an edge sign vector. The edge sign vector distinguishes between AND and NOT relations, reflecting the logical structure of AIGs, and these elements are defined as follows:

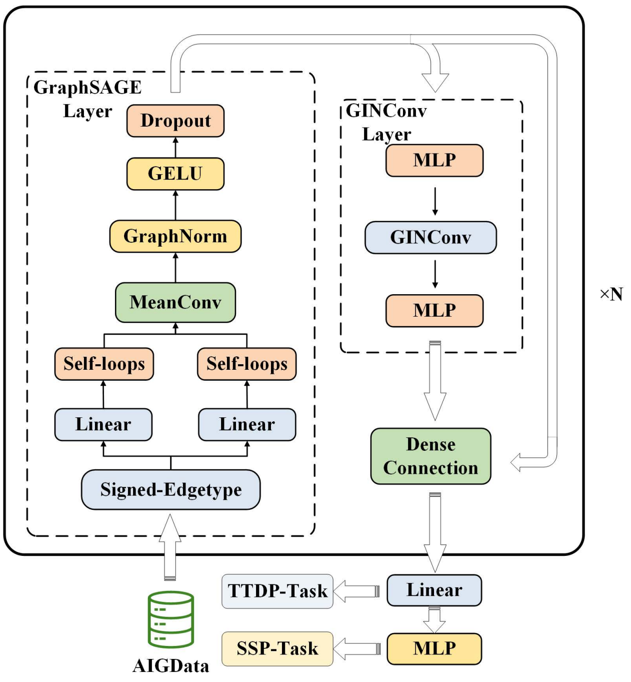


Fig. 2. The framework of Hybrid Feature Aggregation Component

$$X \in \mathbb{R}^{N \times d_{in}}, \quad E \in \mathbb{Z}^{2 \times E}, \quad \mathbf{s} \in \{-1, +1\}^E \tag{1}$$

where N denotes the number of nodes, d_{in} denotes the dimensionality of the input features, E denotes the number of edges; each column of the edge index matrix stores the source and target node indices, and each element of s indicates the sign of the edge weight (+1 denotes an 'AND' relation, -1 denotes a 'NOT' relation).

Differentiated initial features are introduced to capture node-level logical differences, embedding circuit-specific attributes to distinguish logical roles across varying AND-to-NOT gate ratios. These features undergo a linear transformation to map into a unified hidden space, defined as:

$$X_{\text{trans}} = WX \tag{2}$$

where W denotes the transformed feature matrix, d_{out} is the specified hidden-space dimensionality, and is the trainable weight matrix. This equation represents a linear mapping of the input features from d_{in} to d_{out}.

Then, a node retention mechanism incorporates self-loops to preserve each node's logical identity during propagation, defined as:

$$\textbf{edge\_index} \sim \textbf{edge\_index} \cup \{(v, v) \mid v \in V\}, \quad \textbf{edge\_sign} \sim \textbf{edge\_sign} \cup \{+1\} \tag{3}$$

where *edge_index* denotes the edge-index matrix and E the number of edges. FuncGNN add a self-loop edge for every node $v \in V$ (i.e., v connected to itself), introducing a new edge representing v's connection to itself.

Signal propagation weights each neighbor's contribution by its corresponding edge sign, distinguishing AND and NOT relations for accurate logical differentiation. For simplicity and efficiency, features for NOT edges are negated. This inter-node message passing is thus defined as:

$$m_j(v) = \mathrm{s}(e_{vj}) \cdot X_{\mathrm{trans},j}, \quad \forall j \in N(v) \tag{4}$$

where $m_j(v)$ is the message transmitted from neighbor j to node v; $\sigma(v, j) \in \{\breve{}1, 1\}$is the edge-sign weight for (v, j); is j's linearly transformed feature; and $N(v)$ is the set of v's neighbors.

Feature smoothing integrates neighbor information, enhancing efficiency while maintaining robustness across heterogeneous AIGs. A node retention mechanism further integrates the node's original features, mitigating information loss from over-smoothing, defined as:

$$h_v^{\mathrm{agg}} = \frac{1}{|N(v)|} \sum_{j \in N(v)} m_j(v) + h_v^{\mathrm{skip}} \tag{5}$$

where h_v^{agg} is the aggregated feature of node v, and h_v^{skip} denotes the skip-connection term using v's original features.

Stabilization techniques, such as GELU [13] activation and dropout [26], are employed to enhance numerical consistency and training robustness. GELU activation helps maintain stable feature distributions, while dropout mitigates overfitting by randomly deactivating neurons during training. By incorporating these techniques into the message aggregation framework, the overall node representation update can be formally expressed as follows:

$$h_v^{(k+1)} = \mathrm{dropout}\left(\sigma\left(\mathrm{Norm}\left(r, h_v^{(k)} + \sum_{j \in \mathcal{N}(v)} \mathrm{s}(e_{vj})\, h_j^{(k)}\right)\right)\right) \tag{6}$$

where $h_v^{(k+1)}$ denotes the updated feature of node v at layer $k+1$, and $h_v^{(k)}$ is its feature at the previous layer. $\mathrm{s}(e_{vj})$ represents the semantic weight of the edge from node j to node v, and $\mathcal{N}(v)$ is the set of neighbors of node v. Norm refers to the enhanced conditional normalization operation, which will be discussed in the following section.

The term σ refers to the GELU activation function, while dropout indicates a mechanism that randomly discards features to prevent overfitting. The process involves aggregating neighbor signals, combining them with the node's prior knowledge, and applying stabilization techniques. This yields stable and robust representations suitable for downstream processing across diverse circuit topologies.

To supplement stable feature extraction, a GINConv-based enhancement layer is attached after each aggregation stage, providing expressive logical enhancement to capture local circuit patterns and non-linear logical structures. A MLP is embedded within this enhancement, applying non-linear processing to each neighbor's features. The MLP consists of two linear transformations interleaved with ReLU activations, introducing non-linearity and boosting expressiveness. The MLP kernel is defined as:

$$\mathrm{MLP}(x) = W_4 \cdot \sigma(W_3 x + b_3) + b_4 \tag{7}$$

where W_3, W_4 are the trainable weight matrices of the two linear layers, b_3, b_4 are their bias vectors, σ denotes the Rectified Linear Unit activation(ReLU), introducing nonlinearity and boosting expressiveness.

The enhancement operation aggregates these transformed features into a rich node representation, balancing the node's own contribution. Combined with full-layer node retention, this design fuses multi-scale logical information across layers, heightening the capacity to represent complex logical structures inherent in AIGs. The GINConv operation is given by:

$$h_v^{(k+1)} = (1+\varepsilon)\, h_v^{(k)} \sum_{u \in \mathcal{N}(v)} h_u^{(k)} \tag{8}$$

where $h_v^{(k+1)}$ is the updated feature of node v, $h_v^{(k)}$ is its feature from the previous GraphSAGE layer, ε is a learnable scalar that balances the contribution of v's own feature, $\sum_{u \in \mathcal{N}(v)} h_u^{(k)}$ sums the features of all neighbors of v.

By integrating GraphSAGE-based aggregation for stable feature extraction and GINConv-based enhancement for expressive local pattern capture, FuncGNN addresses varying AND-to-NOT ratios and diverse circuit topologies in AIGs. Therefore, the challenge of structural heterogeneity is alleviated to a certain extent. These features, however, exhibit cross-AIG variances, necessitating normalization to stabilize subsequent processing.

3.3 Global Context Normalization Component

This section elaborates on the architecture and operational workflow of the Global Context Normalization Component. Highlights the internal design of the component, which incorporates global logic statistics with the AND-to-NOT ratio. This design helps address inter-graph distribution shifts and stabilizes representation learning across heterogeneous AIGs.

To better address cross-AIG variances and stabilize the training process, the Global Context Normalization Component leverages gate-ratio information for graph normalization. Specifically, since the AND-to-NOT edge ratio exhibits significant disparities across different AIG instances, the model uses this proportion as prior knowledge. Incorporating this statistical data into the normalization stage reduces inter-graph distribution shifts. Such conditional normalization mechanisms—akin to conditional BatchNorm (CBN) [8] or Feature-wise Linear Modulation (FiLM) [21]—have been shown to enhance the use of global structural attributes and to capture more complex logical relationships.

In circuit benchmarks where all AIGs in the training dataset are generated uniformly (see Section 4.2.1), functional differences between logic circuits often appear as variations in their AND-to-NOT gate-ratio. Injecting these graph-level functional attributes into the normalization step significantly improves functionality-aware representation learning. From a circuit-representation perspective, using AIG-level global features as input further strengthens the ability to model intricate logic structures.

Specifically, enhanced GraphNorm adds two linear layers on top of standard GraphNorm [6]. These layers dynamically generate scaling and bias terms from the global ratio input, which are then added to the original weight and bias for adaptive normalization. During normalization, each node feature is first standardized along the feature dimension as:

$$h = \frac{h - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{9}$$

where μ and σ^2 are the mean and variance of all node features in the current graph, and ϵ is a stability constant to prevent division by zero. This operation eliminates differences in mean and scale across node features.

Then, given the global ratio input r, we pass it through two dynamic ratio linear layers to produce a set of per-feature scaling and bias parameters.

$$\gamma_{\text{dyn}} = \mathbf{W_1} \cdot \gamma \cdot r + \mathbf{b_1} \cdot \gamma, \quad \beta_{\text{dyn}} = \mathbf{W_2} \cdot \beta \cdot r + \mathbf{b_2} \cdot \beta \tag{10}$$

where W_i and b_i are the learnable weight matrix and bias vector of those linear layers, used to generate dynamic scaling and bias coefficients.

These two dynamically generated terms are added to the original weight and bias, yielding the final scaling and bias as:

$$\gamma = \gamma_0 + \gamma_{\text{dyn}}, \quad \beta = \beta_0 + \beta_{\text{dyn}} \tag{11}$$

where γ_0 and β_0 are the globally learned scaling and bias parameters shared across all samples. γ_{dyn} and β_{dyn} are dynamically generated based on the current graph's ratio r, allowing adaptation to each graph's global characteristics. Their sum yields the final scaling γ and bias β applied to the normalized features.

Then, the features are conditionally normalized, producing the final output as:

$$h_{\text{norm}} = h \odot \gamma + \beta \tag{12}$$

where $\odot$ denotes element-wise multiplication. By dynamically adjusting γ and β based on the AND-NOT ratio, the Global Context Normalization Component becomes sensitive to cross-AIG differences, thereby stabilizing training and enhancing the model's representation of diverse logic circuits.

In summary, this component enables dynamic, circuit-aware normalization by incorporating global structural priors into the feature scaling process, thereby mitigating numerical and structural disparities. As a result, it significantly enhances the model's stability and its capacity to generalize across diverse logic circuit topologies. The challenge of numerical and structural disparitie is effectively addressed.

3.4 Multi-Layer Integration Component

This section introduces the Multi-Layer Integration Component, detailing its design and workflow to effectively capture and fuse multi-granularity logic information from all layers of the model. The discussion emphasizes how this component mitigates the over-squashing problem and preserves complementary features from shallow to deep layers, thus enhancing global representation learning in large-scale AIGs.

For large-scale AIGs, efficiently capturing global logic information while avoiding information degradation across layers is critical for accurate circuit representation. Different layers contribute complementary information at varying levels of granularity. Shallow aggregation layers focus on fine-grained neighborhood features, while deeper layers encode abstract global patterns. However, over-squashing—where representations from distant nodes become indistinguishable—can hinder the expressive capacity of deep models.

To address this, the Multi-Layer Integration Component preserves intermediate representations from all layers via full-layer dense concatenation, allowing the model to retain multi-granularity logic information across structural depths. These concatenated features are then fused through a learnable linear projection that adaptively emphasizes informative layers. This design not only mitigates global information loss caused by progressive over-squashing but also enhances sensitivity to hierarchical circuit patterns.

The architecture is designed without relying on attention-based mechanisms, instead using lightweight operations such as mean aggregation and linear transformation. This design ensures computational efficiency and scalability, enabling fast and robust extraction of full-graph semantics from complex and heterogeneous AIG structures.

Next, the concatenated feature vectors pass through a linear mapping layer that re-weights and integrates multi-scale information, reduces dimensionality, and preserves key signals. The resulting node features are then used in the TTDP loss computation. The process is defined as:

$$h_v^{\text{dense}} = W_{\text{fuse}} \cdot \text{concat}\left(h_v^{(1)}, h_v^{(2)}, \ldots, h_v^{(L)}\right) + b_{\text{fuse}} \tag{13}$$

where h_v^{dense} denotes the node features after dense concatenation. The dense concatenation aggregates the output features from all L layers—alternating between GraphSAGE and GINConv—into a higher-dimensional feature vector. $h_v^{(l)}$ refers to the output node features at the l-th layer, and b denotes the bias term, which helps adjust the distribution of the fused features. The resulting output features are utilized for computations in the TTDP task or other similar downstream tasks.

Finally, these fused features enter an MLP-based read-out layer, which applies nonlinear transformations, LayerNorm, Dropout, and a Sigmoid activation to map outputs to the probability range (0,1), yielding the SPP results. The read-out MLP is formally defined as:

$$h_v^{(l+1)} = \text{dropout}\left(\sigma\left(\text{Norm}\left(W_l \cdot h_v^{(l)} + b_l\right)\right)\right), \quad \forall l \in \{0, \ldots, L\} \tag{14}$$

where $h^{(l+1)}$ is the hidden feature at the $(l+1)$-th layer of the MLP, dropout$(\cdot)$ is applied to prevent overfitting by randomly deactivating a subset of neurons, and $\sigma(\cdot)$ introduces non-linearity. Norm$(\cdot)$ denotes the Layernorm, which stabilizes the training of SPP task, while W_l and b_l are the learnable weight and bias of the l-th layer. L denotes the total number of layers in the MLP.

By retaining both stable local embeddings (from GraphSAGE) and fine-grained non-linear features (from GINConv), concatenating outputs from all layers, it combines early-stage local features with later-stage abstract representations, the dense connection layer captures comprehensive global structure. It leverages the complementary strengths of multi-layer outputs and alleviates representation insufficiency in large, structurally diverse AIGs, while keeping the computational cost within an acceptable range. Integrating global context in this way significantly enhances the model's learning and generalization capabilities for complex AIG graph prediction tasks. Therefore, the challenge of global structural deficiencies is effectively resolved.

In summary, the Multi-Layer Integration Component enables FuncGNN to combine stable local features with abstract global patterns by integrating outputs from every layer. This design alleviates global information loss caused by over-squashing, improves sensitivity to hierarchical circuit structures, and strengthens the model's ability to generalize across structurally diverse and complex AIG graphs, all while maintaining computational efficiency.

4 Experimental

This section outlines the experimental framework to evaluate FuncGNN on heterogeneous AIGs. Section 4.1 formulates the research questions guiding the experiments. Section 4.2 details the dataset, state-of-the-art methods, and evaluation metrics, while Section 4.3 elaborates on the experimental procedures and implementation specifics.

4.1 Research Questions

To systematically assess FuncGNN's performance and robustness in capturing logical structures across diverse circuit topologies, the following research questions (RQs) are investigated:

RQ1:Does FuncGNN improve SPP performance with lower resource consumption?

RQ2: Does FuncGNN enhance accuracy on the TTDP task?

RQ3: Does FuncGNN stabilize training for heterogeneous AIGs?

RQ4: Does FuncGNN mitigate feature convergence in deep layers?

RQ5: Are FuncGNN's components necessary and effective?

RQ1 evaluates whether FuncGNN achieves higher accuracy in SPP with reduced model complexity, leveraging efficient feature integration. RQ2 assesses FuncGNN's capability to capture

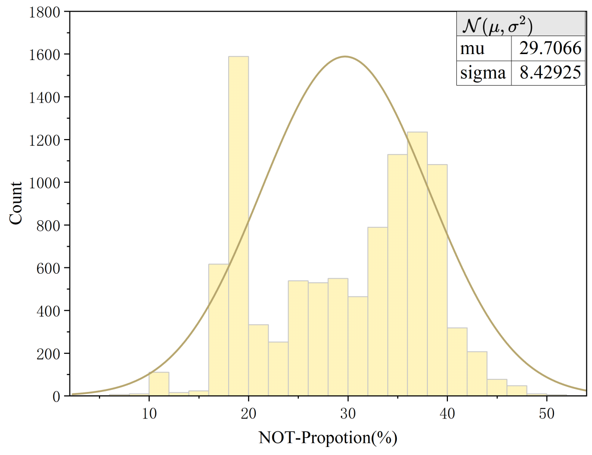


Fig. 3. NOT Proportion Distribution Histogram

Table 1. The Statistics of AIG Datasets [17]

Benchmark	#Subcircuits	#Node	#Level	#Gate Ratio (A/N)
EPFL	828	[52-341]	[4-17]	[1.27-30.4]
ITC99	7,560	[36-1,947]	[3-23]	[1.16-8.56]
IWLS	1,281	[41-2,268]	[5-24]	[0.90-22.2]
Opencores	1,155	[51-3,214]	[4-18]	[1.08-14.29]
Total	**10,824**	**[36-3,214]**	**[3-24]**	**[0.90-30.4]**

fine-grained functional similarities in TTDP, as required for precise logic equivalence checking. RQ3 investigates whether FuncGNN stabilizes training across AIGs with diverse circuit topologies, addressing the challenge of structural heterogeneity. RQ4 examines whether FuncGNN overcomes feature convergence, ensuring robust capture of global logical structures in deep architectures. RQ5 validates the necessity and contribution of each component, such as hybrid aggregation and multi-layer integration, to the model's effectiveness.

4.2 Experiment Settings

4.2.1 Datasets

For FuncGNN, the subcircuit dataset was constructed by extracting subcircuits from four benchmark suites—ITC'99 [7], IWLS'05 [1], EPFL [3], and OpenCore [28]—and converting them into a unified And-Inverter Graph (AIG) format using the ABC tool. Logic simulation with up to 100,000 random input vectors was performed to obtain accurate signal probabilities for each node. As presented in Table 1, the dataset comprises 9,933 valid subcircuits, covering circuit sizes from tens to thousands of nodes with varying logic levels and exhibiting diverse circuit topologies and AND-to-NOT gate ratios. The column labeled "#Subcircuits" specifies the number of subcircuits extracted from each benchmark. To visually illustrate the pronounced structural differences among AIGs in the dataset, a statistical analysis of the proportion of NOT nodes in the AIGs of all circuits was conducted. The histogram distribution is presented in Figure 3, revealing an irregular distribution of NOT gate proportions. Specifically, the mean NOT proportion is 29.7%, with a standard deviation of 8.42. Furthermore, Table 1 highlights substantial variability in subcircuit scale (node counts ranging from 36 to 3,214), logic levels (3 to 24 levels), and subcircuit counts (828 to 7,560), underscoring the dataset's high heterogeneity in topology, scale, and logical complexity. Notably, the "#Gate Ratio" column shows a wide range from 0.90 to 30.4, reflecting large disparities in the relative proportions of AND and NOT gates across subcircuits. This multi-dimensional structural variability poses significant challenges to model robustness. Training and evaluating on such a structurally heterogeneous dataset highlights FuncGNN's strong generalization capabilities and validates its effectiveness in handling real-world AIG variability.

4.2.2 State-of-the-Art Methods

To evaluate the effectiveness of FuncGNN, it is compared with four state-of-the-art methods.

DeepGate [17] employs a circuit-specific attention mechanism that mimics logical computation processes and a backpropagation layer that accounts for logical effects, capturing both topological and functional properties of AIGs to achieve robust logical structure capture.

DeepGate2 [24], an enhanced version of DeepGate, introduces a function-aware learning framework. It leverages pairwise truth-table hardship as supplementary supervision and adopts an efficient single-pass GNN design to simultaneously capture structural and functional information of logic gates.

DeepGate3 [25] builds on pretrained DeepGate2, utilizing a Refine Transformer to extract initial gate-level embeddings and further capture complex long-range dependencies among logic gates, excelling in handling large-scale circuit configurations.

HOGA [9] uses hop-wise feature aggregation with a gated self-attention module to integrate multi-hop features. This removes inter-node dependencies, simplifying training and enabling robust generalization across circuit topologies.

PolarGate [18] optimizes message passing mechanisms and model architecture to overcome functional representation bottlenecks, demonstrating notable improvements in training efficiency, prediction accuracy, and model flexibility compared to existing methods.

Additionally, classical graph neural network models **GCN** [16] and **GraphSAGE** [12] are included as comparative baselines for analysis. GCN [16] performs localized spectral convolutions that aggregate neighborhood features, while GraphSAGE [12] generates node embeddings by sampling and aggregating features from a node's local neighborhood to enable inductive learning on large graphs.

4.2.3 Evaluation Metrics

To assess the capability of FuncGNN in capturing logical structures within heterogeneous AIGs, two tasks are evaluated: SPP and TTDP. SPP, a fundamental task in AIG analysis [17], measures the accuracy of predicting node signal probabilities across diverse circuit topologies. TTDP, introduced by DeepGate2 [24], evaluates the ability to capture fine-grained functional similarities by predicting the normalized Hamming distance between truth tables of node pairs. The performance is quantified using the following metrics, systematically evaluating prediction accuracy, computational efficiency, model complexity, and hardware overhead.

For the SPP task, the Mean Absolute Error (MAE) quantifies the average absolute difference between predicted and simulated signal probabilities for all nodes in the subcircuit dataset, as defined in Equation 15:

$$Avg.Error_{SPP} = \frac{1}{N}\sum_{v\in\mathcal{V}} |y_v - \hat{y}_v| \tag{15}$$

where N is the total number of nodes, V denotes the set of all nodes in the AIG, y_v represents the simulated signal probability for node v, and $\hat{y}_v$ is the predicted signal probability.

For the TTDP task, the MAE measures the difference between the predicted and actual truth-table distances for sampled node pairs, capturing functional equivalence across AIGs with varying AND-to-NOT gate ratios. The computation involves calculating the normalized Hamming distance between truth tables and the distance between embedding vectors, followed by zero-normalization to align scales, as specified in Equations 16 to 19:

$$D^{T}_{(i,j)} = \frac{HammingDistance(T_i, T_j)}{length(T_i)}, \quad (i,j) \in \mathcal{N} \tag{16}$$

$$D^{Z}_{(i,j)} = 1 - CosineSimilarity(z_i, z_j) \tag{17}$$

$$D^{T'}_{(i,j)} = ZeroNorm(D^{T}_{(i,j)}), \quad D^{Z'}_{(i,j)} = ZeroNorm(D^{Z}_{(i,j)}) \tag{18}$$

where N is the set of sampled node pairs, T_i and T_j are the truth-table vectors for nodes i and j, $HammingDistance(T_i, T_j)$ computes the number of differing bits between T_i and T_j, and $length(T_i)$ is the length of the truth table. z_i and z_j denote the embedding vectors of nodes i and j, $CosineSimilarity(z_i, z_j)$ measures their similarity, and $D^{Z}_{(i,j)}$ represents the embedding distance. $ZeroNorm(\cdot)$ normalizes the distances to a zero-mean distribution, and $D^{T'}_{(i,j)}$ and $D^{Z'}_{(i,j)}$ are the normalized truth-table and embedding distances, respectively.

Finally, we can define the MAE of TTDP task as:

$$\text{Avg.Error}_{\text{TTDP}} = \frac{1}{N}\sum_{(i,j)\in\mathcal{N}} \left| D^{T'}_{(i,j)} - D^{Z'}_{(i,j)} \right| \tag{19}$$

where N is the number of node pairs, and $Avg.Error_{TTDP}$ is the average absolute difference between normalized distances.Where a lower MAE indicates higher accuracy and stronger capability in AIG representation learning.

Additional metrics include:

- **Parameter Count(Params)**: The total number of trainable parameters in FuncGNN, assessing model complexity and storage requirements.
- **GPU Memory Usage(Mem)**: The maximum GPU memory usage in GB, where a lower value signifies less resource consumption, indicating the model's feasibility in practical hardware environments.
- **Average Training Runtime (Avg.time)**: The average runtime per training epoch, where a shorter time reflects lower computational complexity on the same hardware, reflecting the computational efficiency and scalability of FuncGNN on large-scale AIG benchmarks.

These four metrics—MAE, Avg.time, Parameter Count, and GPU Memory Usage—systematically evaluate FuncGNN's performance from the perspectives of prediction accuracy, computational efficiency, model complexity, and hardware overhead, addressing the challenges of structural heterogeneity in large-scale AIGs.

4.3 Training Details

The experiments were conducted on a high-performance server equipped with an NVIDIA A800 GPU, running Ubuntu 22.04.5 LTS. For state-of-the-art methods, training hyperparameters were consistent with those reported in their respective papers, employing single-task training for up to 500 epochs. A default batch size of 128 was used, and the dataset was split into training, validation, and test sets with proportions of 0.05, 0.05, and 0.9, respectively. This extreme data split simulates real-world scenarios where labeled data is scarce, placing significant demands on a model's generalization and learning ability.

For FuncGNN, the configuration of the Hybrid Feature Aggregation Component varies the number of layers L across experiments, with a default of $L = 3$. The hidden dimension is set to 256, and the dropout rate is 0.1. In the Global Context Normalization Component, the output dimension of the mapped sum is set to 256. For the Multi-Layer Integration Component, the concatenated feature output dimension is 256, and the prediction MLP for the SPP task adopts a hidden dimension of 256 with a dropout rate of 0.1.

During training, initial feature vectors are generated as one-hot encodings based on node indices. The Adam optimizer is employed with an initial learning rate of 0.001 and a weight decay of 1×10^{-4}. An early stopping mechanism is applied, terminating training if no performance improvement is observed after 100 epochs, with the best-performing model selected as the final model.

5 Results

In this section, five experiments were conducted to address the five research questions outlined in Section 4.

5.1 Answer to RQ1: Performance on the SPP Task

Motivation. The objective of this research question is to evaluate the performance of the proposed method on the SPP task and compare it with state-of-the-art methods.

Methodology. To assess the effectiveness of the method on the SPP task, all state-of-the-art methods were trained on the same dataset using default data splitting ratios and predefined hyperparameters. Performance comparisons were conducted based on four evaluation metrics: MAE, Params, Mem, and Avg.time. The layer count (L=n) denotes model complexity, with higher values indicating increased complexity and parameter counts. For DeepGate2, the two-stage training process described in its original paper was preserved.

Results. Table 2 reports the experimental results on the AIG benchmark dataset. Overall, FuncGNN consistently outperforms all baselines in both predictive accuracy and resource efficiency, validating its design for scalable and functionality-aware AIG representation learning.

Compared to PolarGate (L=9), which uses a comparable parameter size, FuncGNN achieves a 2.06% lower MAE, while also reducing memory consumption by 32.83% and average training time by 50.65%. This performance gain can be attributed to the Hybrid Feature Aggregation Component, which replaces resource-intensive attention modules with lightweight aggregation, enabling more efficient learning of logic patterns without sacrificing accuracy.

When compared to DeepGate2, a strong attention-based baseline, FuncGNN shows even more significant improvements. It reduces the MAE by 57.01%, with a dramatically smaller parameter count and lower memory usage. This is largely due to the introduction of the Global Context Normalization Component, which explicitly incorporates global logic statistics (e.g., AND-to-NOT ratio) into the learning process, allowing the model to generalize across structurally diverse circuits with fewer resources.

Table 2. The performance on the SPP Task

Model	Params($\times 10^6$)	Mem(GB) ↓	Avg.time(s) ↓	MAE ↓
GCN [16]	0.06	0.37	0.66	0.0623
GraphSAGE [12]	0.09	0.40	0.89	0.0895
DeepGate [17]	0.07	0.31	56.11	0.1241
DeepGate2 [24]	1.34	0.89	47.61	0.0221
DeepGate3 [25]	8.56	63.46	13.79	0.0346
HOGA [9]	1.18	3.85	5.27	0.1344
PolarGate(L=3)	0.41	0.54	3.52	0.0437
PolarGate(L=9) [18]	0.98	0.67	6.12	0.0097
FuncGNN	0.96	**0.45**	**3.02**	**0.0095**

Against simpler models such as GCN and GraphSAGE, FuncGNN achieves 84.11% and 88.94% lower MAE respectively. These results highlight the benefits of FuncGNN's Multi-Layer Integration Component, which retains and fuses multi-granularity information across layers. This allows the model to better capture both local structures and high-level circuit logic, addressing the over-squashing problem commonly seen in shallow message-passing networks.

In summary, FuncGNN not only achieves state-of-the-art accuracy on AIG property estimation tasks but also significantly improves training efficiency and scalability. These advantages stem directly from its targeted architectural innovations designed to address the structural heterogeneity and global information challenges inherent to AIG-based circuits.

5.2 Answer to RQ2: Performance on the TTDP Task

Motivation. The objective of this research question is to evaluate the performance of the proposed method on the TTDP task and compare it with state-of-the-art methods.

Methodology. Consistent with RQ1, the performance of the method on the TTDP task was evaluated using four metrics: MAE, Params, Mem, and Avg.time.

Results. Table 3 summarizes the performance of FuncGNN on the TTDP task. Across all evaluated metrics, FuncGNN consistently outperforms prior methods in both predictive accuracy and computational efficiency, demonstrating its effectiveness for functionality-aware AIG representation learning.

Compared to the DeepGate series, which rely heavily on attention mechanisms, FuncGNN achieves up to a 56.06% reduction in MAE, while also exhibiting significantly faster training speeds. This performance gain is largely due to the Hybrid Feature Aggregation Component, which uses lightweight message-passing rather than attention, greatly reducing computational overhead while maintaining functional sensitivity. Compared to PolarGate (L=9), which serves as a strong message-passing baseline with a comparable parameter count, FuncGNN achieves notable improvements. It reduces memory usage by 34.85%, lowers average training time by 50.55%, and enhances accuracy with an 18.71% reduction in MAE. These gains are primarily attributed to the Global Context Normalization Component, which dynamically adjusts feature scaling based on logic-specific statistics. This mechanism enables better generalization across heterogeneous circuits while maintaining computational efficiency. When compared with simpler graph models such as GCN and GraphSAGE, FuncGNN reduces MAE by 59.68% and 44.08% respectively. These results emphasize the effectiveness of the Multi-Layer Integration Component, which preserves and fuses

Table 3. The performance on the TTDP Task

Model	Params($\times 10^6$)	Mem(GB) ↓	Avg.time(s) ↓	MAE ↓
GCN [16]	0.06	0.39	0.79	0.4581
GraphSAGE [12]	0.09	0.41	1.21	0.3303
DeepGate [17]	0.17	0.33	58.23	0.4203
DeepGate2 [24]	1.34	0.84	49.95	0.3912
DeepGate3 [25]	8.56	63.49	14.23	0.4367
HOGA [9]	1.18	3.97	5.44	0.3244
PolarGate(L=3)	0.41	0.54	3.66	0.2513
PolarGate(L=9) [18]	0.98	0.66	6.39	0.2272
FuncGNN	0.96	**0.43**	**3.16**	**0.1847**

multi-depth representations, mitigating over-squashing and enabling richer functional abstraction across circuit depths.

Together, these results affirm that FuncGNN not only delivers state-of-the-art predictive performance for the TTDP task but also scales efficiently to large and complex AIGs. It strikes an optimal balance between accuracy and resource consumption through its carefully crafted architectural components.

5.3 Answer to RQ3: Performance in Stability

Motivation. This research question investigates whether FuncGNN can maintain stable performance under the inherent structural variability of AIGs, especially when trained on a limited dataset. In real-world EDA scenarios, acquiring large-scale annotated data is often costly or impractical. Under such constraints, models with low robustness may exhibit significant performance fluctuations across different training sets. Therefore, we assess whether FuncGNN can mitigate training instability and consistently generalize across diverse circuit structures.

Methodology. To rigorously evaluate the robustness of each model under limited-data conditions, all hyperparameters and training configurations were fixed, with the random seed serving as the sole variable. Each random seed determines not only the data shuffling order but also the partitioning of training and test sets as well as the composition of mini-batches, thereby inducing diverse training subsets. Ten distinct seeds (42, 42×2, 42×3, ..., 42×10) were employed for training, with no early stopping applied. For PolarGate, L=9.

This experimental design emulates realistic scenarios characterized by scarce and structurally diverse data. A model demonstrating strong robustness is expected to yield minimal variation in MAE across different seeds, whereas significant fluctuations would indicate limited adaptability to the structural heterogeneity of AIGs and a higher susceptibility to data shifts. Furthermore, the training setup was deliberately designed to be highly sparse, with only 496 AIGs allocated to training and 8941 AIGs to testing, resulting in an approximate train-to-test ratio of 1:18.

Results. Figure 4 reports the MAE variance of FuncGNN and state-of-the-art methods across ten different random seeds. On the SPP task, FuncGNN achieves the lowest MAE interquartile range (IQR) of 0.000175 and a whisker span of 0.0004, representing a dramatic reduction of 96.92% in IQR and 97.14% in span compared to the best DeepGate variant (IQR = 0.0057, span = 0.012), and a striking advantage over HOGA(IQR = 0.012, span = 0.0331) and PolarGate (IQR = 0.00155, span = 0.0036). These results demonstrate FuncGNN's exceptional training stability and minimal sensitivity to initialization, particularly in the SPP task. On the TTDP task, FuncGNN also achieves

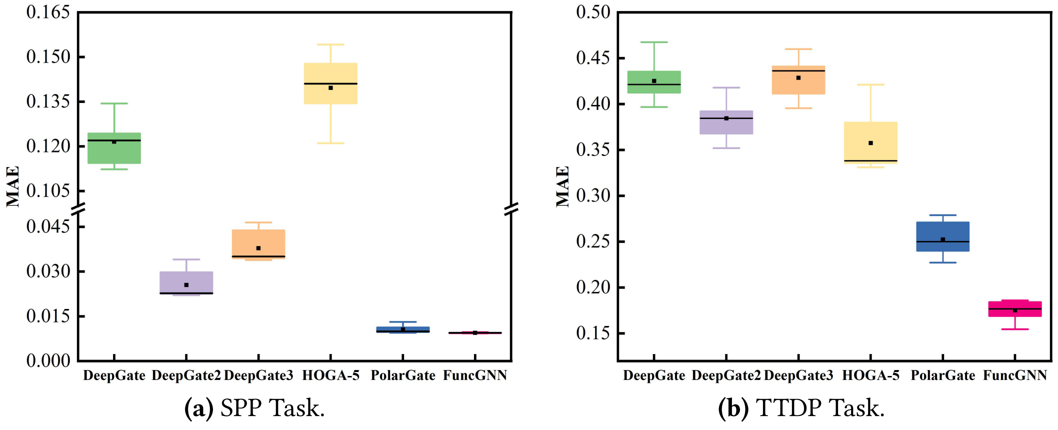


Fig. 4. Stability analysis on SPP and TTDP tasks.

a significant stability improvement, with an IQR of 0.0126 and a span of 0.0314. This marks a 35.38% reduction in IQR and 51.63% reduction in span compared to the best DeepGate result (IQR = 0.0195, span = 0.0643), and a noticeable advantage over HOGA(IQR = 0.0344, span = 0.0901) and PolarGate (IQR = 0.0268, span = 0.0516).

These improvements stem from the joint effect of Hybrid Feature Aggregation and Global Context Normalization, which together stabilize learning by preserving multi-scale structural signals and dynamically adapting to inter-graph logic variation. FuncGNN consistently generalizes well under this setup, highlighting its robustness, high data efficiency, and ability to abstract functional patterns from limited supervision.

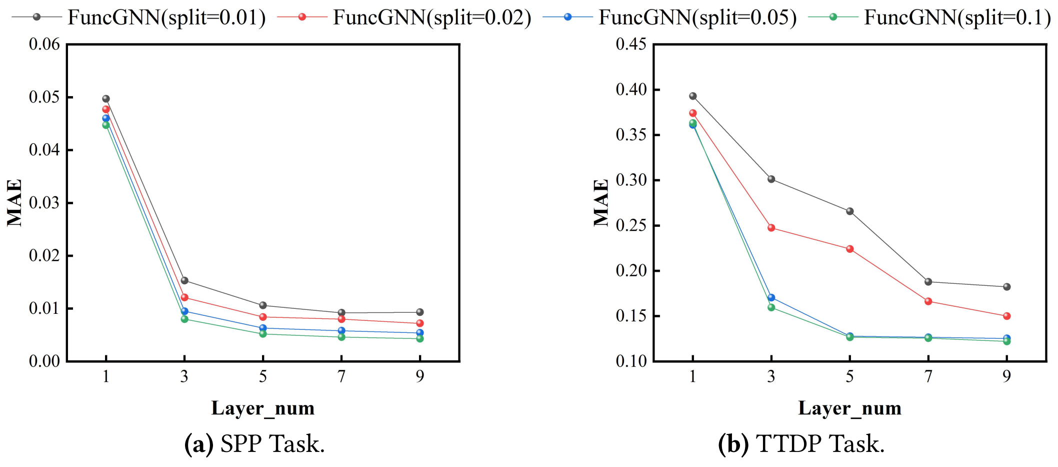


Fig. 5. Comparison of different layers on SPP and TTDP tasks.

5.4 Answer to RQ4: Model Performance Across Different Layer Counts

Motivation. GNNs inherently suffer from the over-squashing problem, where information from distant nodes is compressed into fixed-size embeddings as messages propagate through multiple

Table 4. Performance On Ablation Study

Strategy	SPP	TTDP
FuncGNN (w/o Component1 + GCN)	0.0178	0.3361
FuncGNN (w/o Component2)	0.0113	0.2142
FuncGNN (w/o Component3)	0.0231	0.3491
FuncGNN (with simple GraphNorm)	0.0107	0.2241
FuncGNN (Full)	**0.0095**	**0.1847**

layers. This compression limits the effective receptive field and leads to the loss of crucial global structural and functional information in large-scale circuit graphs such as AIGs. As a result, existing GNN-based methods struggle to accurately capture long-range dependencies critical for circuit property estimation. This research question aims to experimentally validate the ability of FuncGNN to mitigate the loss of global graph information due to over-squashing. Attention-based methods are not included in this comparison, as they are generally more resilient to over-squashing and rely on fundamentally different global aggregation mechanisms.

Methodology. To assess the performance of the method under varying model complexities and data volumes, all hyperparameters except layer count were fixed, with model layer count and data splitting ratios as variables. Increasing the layer count typically elevates model complexity; however, deep models often suffer from over-squashing, leading to degraded accuracy despite higher complexity. To thoroughly evaluate whether FuncGNN overcomes over-squashing, experiments were conducted with odd layer counts (L=1, 3, 5, 7, 9) and multiple data splitting ratios (training and validation set proportions of 0.01, 0.02, 0.05, and 0.1). By monitoring MAE trends across these configurations, we evaluated FuncGNN's robustness to over-squashing and its scalability with increasing depth.

Results. Figure 5 presents the performance of FuncGNN across different layer counts and data splits. As the number of layers increases, FuncGNN exhibits a steady decline in MAE for both the SPP and TTDP tasks, indicating consistent performance improvement. This trend is particularly evident under the 0.05 data split ratio, where MAE shows a noticeable reduction across different layer counts. Specifically, from L=1 to L=3, MAE decreases by approximately 79.3% and 52.8% for SPP and TTDP, respectively; from L=3 to L=5, it drops a further 33.7% and 25.0%; from L=5 to L=7, an additional 7.9% and 0.9%; and from L=7 to L=9, a total reduction of 6.9% and 0.9% is achieved. These results demonstrate that FuncGNN effectively leverages deeper architectures to integrate richer circuit features, thereby enhancing representational capacity and mitigating the challenge of over-squashing. Moreover, even under smaller data split ratios (such as 0.01 and 0.02), FuncGNN maintains stable and accurate performance, highlighting its robustness and sample efficiency in low-data regimes.

This consistency is primarily attributed to the proposed Multi-Layer Integration Component, which retains and adaptively fuses intermediate representations from all layers. By preserving rich hierarchical semantics across the network depth, this component enables FuncGNN to effectively aggregate both local and global information, ensuring high-quality circuit representations and strong generalization performance even under limited supervision.

5.5 Answer to RQ5: Performance in Ablation Studies

Motivation. This research question aims to validate the necessity of each component in FuncGNN and quantitatively assess their individual contributions to model performance.

Methodology. Ablation studies were conducted on FuncGNN to evaluate the impact of its key components. Individual components were either removed or replaced, and the resulting performance was assessed on the SPP and TTDP tasks using MAE as the primary metric. The experiments maintained consistent hyperparameters and a data splitting ratio of 0.05-0.05-0.9, as used in prior sections.

Results. The results of the ablation studies are presented in Table 4, demonstrating that each component significantly influences FuncGNN's performance. Specifically:

- Replacing Component 1 with GCN led to MAE increases of 87.37% and 81.97% on the SPP and TTDP tasks, respectively, indicating severe performance degradation and underscoring the critical role of Component 1.
- Removing Component 2 resulted in MAE increases of 18.95% and 15.97% on the SPP and TTDP tasks, respectively, confirming its indispensability to model performance.
- Eliminating Component 3 caused MAE increases of 142.11% and 89.01% on the SPP and TTDP tasks, respectively, highlighting its importance in capturing global information.
- Substituting Component 2 with an unoptimized GraphNorm increased MAE by 12.63% and 21.33% on the SPP and TTDP tasks, respectively, demonstrating the performance benefits of incorporating global gate proportions into conditional normalization, particularly pronounced in the TTDP task.

In summary, the ablation studies confirm the essential role of each component in FuncGNN, collectively enhancing its robust and efficient AIG representation learning across diverse tasks.

6 Conclusions and Future Work

In this paper, we propose FuncGNN, a lightweight and robust framework for general circuit representation learning. It is designed to support a wide range of EDA tasks by effectively modeling AIG structures. FuncGNN combines three key components: Hybrid Feature Aggregation, Global Context Normalization, and Multi-Layer Integration. These components work together to improve learning efficiency, reduce computational costs, and enhance generalization, especially under conditions of structural diversity and limited training data.

Extensive experiments show that FuncGNN achieves superior performance compared to state-of-the-art baselines. It consistently delivers higher accuracy, better training stability, and greater scalability. This is particularly evident in scenarios with deep architectures or extreme train/test splits, where efficient and reliable circuit representation is critical for downstream EDA tasks such as logic synthesis and property prediction.

Future work includes validating FuncGNN on structurally diverse circuit datasets and broadening its application to a wider range of EDA tasks. Additionally, there will be attempts to theoretically demonstrate how incorporating AIG ratio information as a global prior may contribute to constructing globally stable feature representations.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.62472062, No.62202079), the Dalian Science and technology Innovation Fund project (No.2024JJ12GX022), the key research and development project of Liaoning Province (No.2024JH2/102400059).

References

[1] Christoph Albrecht. 2005. IWLS 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS)*, Vol. 9.

[2] Uri Alon and Eran Yahav. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205* (2020).

[3] Luca Amarú, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. 2015. The EPFL combinational benchmark suite. In *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*.

[4] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. 2021. Size-invariant graph representations for graph classification extrapolations. In *International Conference on Machine Learning*. PMLR, 837–851.

[5] Robert Brayton and Alan Mishchenko. 2010. 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, 24–40.

[6] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. 2021. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*. PMLR, 1204–1215.

[7] Scott Davidson. 1999. Characteristics of the ITC'99 benchmark circuits. In *IEEE International Test Synthesis Workshop (ITSW)*. 87.

[8] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. 2017. Modulating early visual processing by language. *Advances in neural information processing systems* 30 (2017).

[9] Chenhui Deng, Zichao Yue, Cunxi Yu, Gokce Sarar, Ryan Carey, Rajeev Jain, and Zhiru Zhang. 2024. Less is more: Hop-wise graph attention for scalable and generalizable learning on circuits. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*. 1–6.

[10] Wenji Fang, Jing Wang, Yao Lu, Shang Liu, Yuchao Wu, Yuzhe Ma, and Zhiyao Xie. 2025. A survey of circuit foundation model: Foundation ai models for vlsi circuit design and eda. *arXiv preprint arXiv:2504.03711* (2025).

[11] FuncGNN. 2025. *FuncGNN Project Repository*. https://github.com/Vandbs/FuncGNN

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[13] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[14] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, et al. 2021. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 26, 5 (2021), 1–46.

[15] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. 2022. Learning the travelling salesperson problem requires rethinking generalization. *Constraints* 27, 1 (2022), 70–98.

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[17] Min Li, Sadaf Khan, Zhengyuan Shi, Naixing Wang, Huang Yu, and Qiang Xu. 2022. Deepgate: Learning neural representations of logic gates. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 667–672.

[18] Jiawei Liu, Jianwang Zhai, Mingyu Zhao, Zhe Lin, Bei Yu, and Chuan Shi. 2024. Polargate: Breaking the functionality representation bottleneck of and-inverter graph neural network. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[19] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. 2006. DAG-aware AIG rewriting a fresh look at combinational logic synthesis. In *Proceedings of the 43rd annual Design Automation Conference*. 532–535.

[20] Alan Mishchenko, Satrajit Chatterjee, Roland Jiang, and Robert K Brayton. 2005. *FRAIGs: A unifying representation for logic synthesis and verification*. Technical Report. ERL Technical Report.

[21] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[22] Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z Pan, Marilyn Wolf, and Jörg Henkel. 2021. MLCAD: A survey of research in machine learning for CAD keynote paper. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 10 (2021), 3162–3181.

[23] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[24] Zhengyuan Shi, Hongyang Pan, Sadaf Khan, Min Li, Yi Liu, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, Zhufei Chu, and Qiang Xu. 2023. Deepgate2: Functionality-aware circuit representation learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 1–9.

[25] Zhengyuan Shi, Ziyang Zheng, Sadaf Khan, Jianyuan Zhong, Min Li, and Qiang Xu. 2024. Deepgate3: Towards scalable circuit representation learning. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[27] Pramod Subramanyan, Nestan Tsiskaridze, Wenchao Li, Adria Gascón, Wei Yang Tan, Ashish Tiwari, Natarajan Shankar, Sanjit A Seshia, and Sharad Malik. 2013. Reverse engineering digital circuits using structural and functional analyses. *IEEE Transactions on Emerging Topics in Computing* 2, 1 (2013), 63–80.

[28] O. Team. Accessed: 2025. Opencores. https://opencores.org/. Accessed: May 2025.

[29] Nan Wu, Yingjie Li, Cong Hao, Steve Dai, Cunxi Yu, and Yuan Xie. 2023. Gamora: Graph learning based symbolic reasoning for large-scale boolean networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[31] Zhihao Xu, Shikai Guo, Xiaochen Li, Zun Wang, and He Jiang. 2025. SIMTAM: Generation Diversity Test Programs for FPGA Simulation Tools Testing Via Timing Area Mutation. *ACM Transactions on Design Automation of Electronic Systems* 30, 2 (2025), 1–25.

[32] Zhihao Xu, Shikai Guo, Guilin Zhao, Peiyu Zou, Xiaochen Li, and He Jiang. 2025. A novel HDL code generator for effectively testing FPGA logic synthesis compilers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[33] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. 2021. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*. PMLR, 11975–11986.

[34] He-Teng Zhang, Jie-Hong R Jiang, and Alan Mishchenko. 2021. A circuit-based SAT solver for logic synthesis. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–6.