ZXNet: ZX Calculus-Driven Graph Neural Network Framework for Quantum Circuit Equivalence Checking

Navnil Choudhury
Electrical and Computer Engineering
The University of Texas at Dallas
Richardson, USA
navnil.choudhury@utdallas.edu

Ameya S Bhave
Electrical and Computer Engineering
The University of Texas at Dallas
Richardson, USA
ameya.bhave@utdallas.edu

Kanad Basu
Electrical and Computer Engineering
The University of Texas at Dallas
Richardson, USA
kanad.basu@utdallas.edu

2025 62nd ACM/IEEE Design Automation Conference (DAC) | 979-8-3315-0304-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DAC63849.2025.11133226

***Abstract*—Quantum circuit execution often requires transpilation into hardware-compatible instructions, which can significantly alter the original design, making equivalence checking essential. However, existing approaches struggle with scalability and computational overhead. In this paper, we present ZXNet, a transformative framework for quantum circuit equivalence checking using ZX calculus-based graph abstractions. Leveraging graph neural networks, ZXNet captures complex equivalence patterns by integrating critical local and global circuit features. ZXNet achieves 99.4% validation accuracy, and up to 62× speedup over state-of-the-art methods, furnishing improvements of 45.83% in scalability, 42.22% in per-qubit verification time, and 5.94% in accuracy, outperforming state-of-the-art approaches.**

***Index Terms*—Quantum computing, equivalence checking, ZX calculus, graph neural networks.**

I. Introduction

Quantum computing represents a paradigm shift in computational science, offering significant speedups over classical computers by leveraging quantum mechanical principles such as superposition and entanglement, as demonstrated in Shor's and Grover's algorithms [1], [2]. Quantum computers operate on quantum bits (*qubits*), which are manipulated by quantum gates that apply unitary transformations to steer qubits toward desired states, with results determined probabilistically through measurement. The increasing demand for quantum computing, driven by applications in fields like cryptography, biochemistry, and machine learning [3]–[5], has led to the emergence of Quantum-as-a-Service (QaaS), allowing users to execute algorithms on cloud-based quantum hardware. However, given the constraints of current quantum hardware, such as limited connectivity and gate fidelity, the submitted algorithms undergo *quantum circuit transpilation* [6]. This process optimizes gate sequences, maps logical qubits to physical hardware, and decomposes complex gates into supported basis gates. While it ensures hardware compatibility, transpilation can significantly alter the original circuit's structure.

The modifications introduced to a quantum circuit during transpilation renders quantum circuit equivalence checking essential [7]. This process verifies that the transpiled circuit preserves the original quantum functionality by ensuring consistent output states and measurement probabilities across all possible inputs. Equivalence checking is particularly crucial for complex circuits, where even minor discrepancies can lead to significant computational errors [8]. As such, it acts as a safeguard to ensure that the executed quantum algorithm faithfully implements the user's intended operations.

To meet the growing demand for equivalence checking in increasingly complex quantum circuits, various frameworks have been developed, including Decision Diagrams, Matrix Product Operators, and simulation-based methods [9]–[11]. However, the existing approaches suffer from significant limitations in terms of time and memory overhead, which hinders their scalability and limits their applicability to complex quantum circuits, as explained in Section III. In addition to these methods, ZX calculus has emerged as a powerful framework for equivalence checking, utilizing its capability to represent quantum circuits as ZX graphs and systematically apply rewrite rules to transform these graphs into canonical forms that preserve circuit semantics [12].

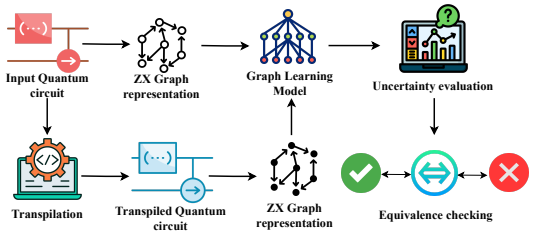


Fig. 1: Overview of our proposed equivalence checking framework using a novel graph learning method, ZXNet.

The graphical representation utilized by ZX calculus facilitates algebraic transformations and rule-based simplifications, enabling the identification of equivalence between circuits. Systematic transformations, such as full reduction, standardize equivalent circuits into near-canonical forms, enhancing the accuracy of equivalence detection. Additionally, ZX calculus effectively encodes both Clifford and non-Clifford operations, ensuring a comprehensive representation of diverse quantum transformations. While the standalone application of ZX calculus for equivalence checking poses computational challenges for large circuits due to extensive rule applications and complex pattern-matching processes, we observe its representation of quantum circuits is highly suitable for graph learning tasks. By encapsulating the circuit's structure and operations in a graphical format, ZX graphs provide a natural input for Graph Convolution Networks (GCNs), which excel at learning relational patterns and structural equivalence [13] [14] [15]. The ability of ZX graphs to represent both local gate operations and global entanglement patterns aligns well with the message-passing mechanisms of GCNs. This compatibility allows ZX calculus to act as a bridge between quantum circuit structures and machine learning, enabling scalable, efficient, and accurate equivalence checking for circuits with varying complexities.

In this paper, we present ZXNet, a transformative equivalence checking framework that, for the first time, leverages graph abstractions derived from ZX calculus, followed by the usage of a novel Graph Convolution Network (GCN) model designed for scalable and efficient quantum circuit equivalence checking. An overview

This work is supported by the National Science Foundation (NSF award #2343652).

of the proposed framework is depicted in Figure 1. ZXNet combines a unique encoding of features extracted from simplified ZX graphs, with a novel GCN architecture designed to learn intricate equivalence patterns efficiently. Furthermore, ZXNet incorporates a novel uncertainty-driven output validation mechanism and a Jaccard-adjusted loss function, which together enhance the model's robustness and accuracy, especially for hard-to-detect errors. *By leveraging ZX graphs to capture essential quantum transformations, ZXNet significantly improves scalability, reduces computational overhead, and establishes a reliable, adaptable solution for equivalence checking across quantum circuits of varying sizes and complexities.* Our main contributions are summarized as follows:

- We present ZXNet, a novel framework for equivalence checking that leverages simplified ZX graph representations of quantum circuits. ZXNet employs an innovative GCN architecture designed to effectively utilize extracted features, enabling efficient and accurate equivalence checking.
- To enhance the robustness of ZXNet, we propose a novel uncertainty-driven validation mechanism utilizing an edge-based Jaccard index value that enables the detection of minor infringements during the transpilation process and significantly improves error detection capabilities.
- ZXNet addresses the challenges of scalability and complexity by being able to learn and subsequently generalize equivalence relations, thereby being able to handle large and highly entangled quantum circuits with up to 120 qubits.
- Our evaluations on extensive benchmarks demonstrate that ZXNet significantly outperforms state-of-the-art equivalence-checking methods, achieving speedups of up to $801\times$ and an accuracy of up to 99.4%, establishing its reliability and effectiveness. ZXNet also surpasses existing techniques, achieving a 45.83% improvement in scalability, a 42.22% reduction in per-qubit verification time, and a 5.94% increase in accuracy.

II. Background and Motivation

A. Quantum Computing Basics

1) Qubits: A qubit, or quantum bit, is the fundamental unit of quantum information, analogous to a classical bit [16]. Unlike classical bits, which exist in either 0 or 1, a qubit can exist in a superposition of both states:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

where $|0\rangle$ and $|1\rangle$ denote the basis states, and α and β are complex probability amplitudes such that $|\alpha|^2 + |\beta|^2 = 1$. This superposition property allows quantum computers to process a much larger state space than classical systems [17]. Qubits can also be entangled, enabling quantum circuits to perform powerful computations through correlated states. Measurement of a qubit collapses its state to either $|0\rangle$ or $|1\rangle$, with corresponding probabilities $|\alpha|^2$ and $|\beta|^2$.

2) Quantum Gates: Quantum gates are unitary operations that change the state of qubits. Mathematically, these gates can be expressed as matrices with dimensions of $2^n \times 2^n$, where n is the number of qubits. Depending on the number of qubits they are acting on, there are single-qubit gates (such as `X-gate`, `S-gate`, `H-gate`, *etc.*) and multi-qubit quantum gates (such as `CX-gate` and `CZ-gate`) [18].

B. ZX calculus

ZX calculus has emerged as a graphical language used to represent and reason about quantum circuits through structured diagrams

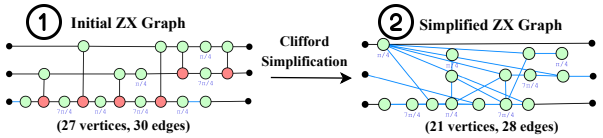


Fig. 2: Example of ZX graph representation and simplification.

instead of traditional gate sequences. It encodes quantum operations as graphs, where Z-spiders (green nodes) and X-spiders (red nodes) represent operations in different bases, and edges represent quantum entanglement or information flow [19] [20]. ZX calculus includes transformation rules—such as spider fusion, Hadamard edge transformation, and Clifford simplification—which simplify circuits while preserving equivalence. ZX calculus has been employed in various quantum computing applications. For instance, mixed quantum-classical optimization methods have utilized it to streamline computations, effectively bridging classical and quantum optimization techniques [21]. It has also been applied to optimize braided circuits on surface codes, demonstrating improvements in circuit efficiency [22]. In comparative studies, ZX calculus-based tools like PyZX have been benchmarked against Qiskit transpilation, where they have shown the potential to reduce gate counts significantly in certain scenarios [23]. An example of ZX graphs and their subsequent simplification is shown in Figure 2, where (1) is the initial ZX graph representation of a quantum circuit, which is subsequently simplified into (2) using Clifford simplification. These rules rooted in formalism render ZX calculus particularly valuable for tasks like circuit equivalence checking and quantum circuit simplification.

C. Graph Neural Networks for Quantum Circuit Equivalence

Our selection of Graph Neural Networks (GNN)s for quantum circuit equivalence checking with ZX graphs is motivated by their capability to directly and effectively process graph-structured data. [24]. Furthermore, unlike other ML models, GNNs natively handle variable-sized graph structures, preserving ZX graph topology and connectivity. Graph Convolution Networks are a type of GNNs, and they aggregate information from a node's neighbors, capturing local interactions (*e.g.*, entanglements) and propagating this information to model global structures [25] [26]. This propagation makes GCNs particularly effective for tasks like ZX graph equivalence checking, where small structural changes or variations in graph layouts (isomorphic graphs) must be handled robustly without being sensitive to the specific graph representation [27].

III. Related Work

Various algorithms have been developed to tackle the unique challenges of quantum circuit equivalence checking. However, they face several limitations, as discussed here. Decision Diagrams (DDs) are widely used for quantum circuit equivalence but struggle with exponential node growth and high memory usage for circuits exceeding 20 qubits [10]. They are also sensitive to variable ordering, which inflates diagram size and complicates verification. In contrast, the proposed ZXNet approach uses scalable, memory-efficient graph representations, effectively handling larger, highly entangled circuits without significant overhead.

Matrix Product Operators (MPO)-based equivalence checking approaches handle low-to-moderate entanglement efficiently using tensor networks, but face exponential scaling with high entanglement or depth, requiring costly tensor operations [9], [28], [29]. Furthermore, they struggle with non-local entanglement and error handling. ZXNet overcomes these challenges using efficient graph representations without intensive computations.

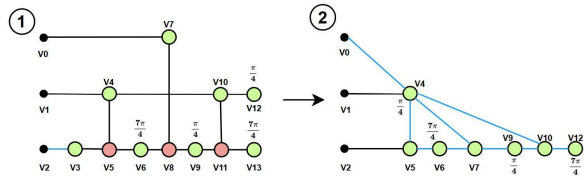


Fig. 3: ZX simplification reducing circuit complexity to enable critical feature extraction.

The existing ZX calculus-based equivalence checking approach struggles with deeply entangled or large circuits due to extensive transformations and inefficiencies [12]. They also lack flexibility for structural variations, limiting their applicability in complex circuits with high entanglement. These limitations primarily arise since this approach does not explicitly utilize any learning model to capture equivalence relations, and primarily relies on algebraic rule-based transformations [19]. In contrast to the existing approach, ZXNet embeds ZX graphs into a learnable framework using GCNs, eliminating exhaustive rule applications and enabling efficient, flexible verification for complex quantum circuits [30].

IV. Proposed ZXNet Framework

A. Feature extraction from ZX graph abstractions

Our approach begins by constructing ZX graph representations from quantum circuits, encoding the circuit's structure into a graphical form suitable for graph learning networks. We apply **full reduction on ZX graphs** to achieve a near-canonical representation, standardizing equivalent circuits and enhancing the model's learning capabilities. This approach preserves circuit equivalence while removing redundancies, thereby minimizing complexity. We have demonstrated an example of this simplification in Figure 3, where the initial ZX graph representation of a quantum circuit G, shown in (1) has 14 nodes, and is simplified to ZX graph (2) G' with 10 nodes, utilizing full reduction. Additionally, since full reduce supports both Clifford and non-Clifford gates, we ensure that the dataset effectively captures essential structural features for equivalence checking.

To design a graph-learning model capable of quantum circuit equivalence checking, we extract representative features from simplified ZX graphs G' that encode critical structural and functional information. This information is categorized into two types: *node features*, capturing critical properties of the vertices, and *edge features*, representing key relationships within the edges of the ZX graph. The extracted features from the simplified ZX graph (2) from Figure 3 are presented in Table I, with the corresponding features detailed subsequently.

a) Node features: These are denoted in columns one to six in Table I and their importance is detailed subsequently.

- **Vertex number**(v_i): The vertex number encodes positional and structural context within the ZX graph, enabling accurate node alignment and equivalence detection between quantum circuits.
- **Node type**(n_i): Node type in ZX graphs distinguishes Z and X-basis operations, ensuring consistent qubit mapping for equivalence checking and identifying mismatches in operation targets.
- **Row Index**(r_i): The row index captures operation sequence, ensuring temporal consistency critical for equivalence, as misalignment alters circuit behavior.
- **Degree**(d_i): The degree captures a node's connectivity and entanglement, aiding equivalence checking by identifying nodes with similar interaction patterns across circuits.

TABLE I: Feature Encoding of an Example Circuit.

Node Features						Edge Features	
Vertex Number (v_i)	Node Type (n_i)	Row Index (r_i)	Degree (d_i)	Phase (p_i)	Qubit Index (q_i)	Associated Edges (v_i, v_j)	Edge Type (e_i)
0	Boundary	0	1	0	0	(0, 7)	H
1	Boundary	0	1	0	1	(1, 4)	S
2	Boundary	0	1	0	2	(2, 5)	S
4	Z	2	3	0	2	(4, 5), (4, 7), (4, 10)	H, H, H
5	Z	2	3	1/4	1	(5, 2), (5, 4), (5, 6)	S, H, H
6	Z	3	2	7/4	2	(6, 5), (6, 7)	H, H
7	Z	4	3	0	2	(7, 6), (7, 9), (7, 4)	H, H, H
9	Z	5	2	1/4	2	(9, 7), (9, 10)	H, H
10	Z	6	3	0	2	(10, 9), (10, 12), (10, 14)	H, H, H
12	Z	7	1	7/4	2	(12, 10)	H

- **Phase**(p_i): The phase parameter encodes rotations or shifts, where differences signal distinct transformations and a lack of equivalence.
- **Qubit Index**(q_i): The qubit index specifies the target qubit of an operation, ensuring consistent qubit mapping and detecting mismatches during equivalence checking.

b) Edge features: These are detailed in columns seven and eight in Table I, with their importance mentioned as follows.

- **Associated Edges** (v_i, v_j): This feature defines connectivity and information flow, requiring identical source-target relationships for equivalence to preserve circuit functionality.
- **Edge type** (e_i): The edge type distinguishes regular connections from Hadamard transformations, and requires consistency across ZX graphs to avoid basis mismatches in equivalence checking.

B. GCN Model Design for Equivalence Checking

1) Model Construction: The ZX graph representation of quantum circuits, encoded into the node and edge features we extracted (Section IV-A), serves as the input for training our GCN model. The model processes pairs of ZX graphs $(\mathbf{G_1}, \mathbf{G_2})$ to produce a joint representation via graph convolutions and outputs a logit vector that indicates the likelihood of equivalence.

2) Model Training: The training process of our GCN model is outlined in Algorithm 1. The model parameters and the optimizer are initialized with a learning rate η to begin the training process (*line 1*). Following this, for each epoch, batches of graph pairs (G_1, G_2) and their labels (y) are extracted from the dataset. For each pair of graphs, the adjacency matrix $\mathbf{A_{N\times N}}$ and node feature matrix $\mathbf{X_{N\times F}}$ (where N represents the number of nodes in the graph and F refers to the number of features per node) are extracted, representing the structural and feature properties of the graphs (*lines 2–3*).

Graph Embedding: Each node $v \in V(G_1) \cup V(G_2)$ (where $V(G)$ denotes the set of all vertices in graph G) is initialized with an embedding $\mathbf{e_v}$, which is a learnable vector representation which encodes the node attributes and relationships (*lines 4–7*).

Graph Convolutional Layers: The embeddings are iteratively refined across L number of graph convolutional layers. At each layer, the embedding of each node v is updated using the operation shown in Equation 2 (*line 8*).

$$e_v^{(l)} = \texttt{AggregateNeighbors}(e_v^{(l-1)}, A, W_l) \tag{2}$$

In Equation 2, the function, `AggregateNeighbors`, combines the embeddings of the node v's neighbors, weighted by the adjacency matrix A and a layer-specific weight matrix W_l [31]. Following this, the updated embeddings are passed through an activation function (*line 9*). For our GCN model, we utilize the `ReLU` (Rectified Linear Unit) activation [32], as shown in Equation 3.

$$e_v^{(l)} = \texttt{ReLU}\left(W_l \cdot e_v^{(l-1)} + b_l\right) \tag{3}$$

In Equation 3, W_l is the learnable weight matrix for layer l, $e_v^{(l-1)}$ is the embedding of node v from the previous layer and b_l

Algorithm 1: GraphMatching Network Training with Jaccard-Adjusted Loss.

Input: Dataset $\mathcal{D}$ with ZX graph pairs (G_1, G_2) and labels y; learning rate η; uncertainty threshold τ; Jaccard index threshold J_{thresh}; scaling factor β
Output: Trained GraphMatching Network model

1 Initialize model parameters and optimizer with learning rate η;
2 **foreach** epoch **do**
3 **foreach** batch $(G_1, G_2, y) \subset \mathcal{D}$ **do**
4 Extract adjacency matrix $A_{N\times N}$ and node feature matrix $X_{N\times F}$ from each graph pair (G_1, G_2);
5 **foreach** node $v \in G_1 \cup G_2$ **do**
6 Initialize embedding e_v for node v;
7 **end**
8 **for** layer $l \in L$ **do**
9 Update $e_v \leftarrow$ `AggregateNeighbours` $(e_v, A_{N\times N}, W_l)$;
10 Apply activation $e_v \leftarrow$ `ReLU` (e_v);
11 **end**
12 Compute logits $f(G_1, G_2)$ via forward pass;
13 Compute softmax probabilities: $P(y|G_1, G_2)$;
14 $\mathcal{L}_{\text{base}} \leftarrow$ `CrossEntropyLoss` $(P(y|G_1, G_2), y)$;
15 $(\hat{y}, \delta, \mathcal{L}_{\text{Jaccard}}) \leftarrow$ `JaccardCheck` $(f(G_1, G_2), E_1, E_2, \tau, J_{\text{thresh}})$;
16 **if** $\delta > \tau$ **then**
17 $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{base}} + \beta \cdot \mathcal{L}_{\text{Jaccard}}$;
18 **end**
19 **else**
20 $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{base}}$;
21 **end**
22 Backpropagate and update model parameters;
23 **end**
24 **end**
25 **return** Trained GraphMatching Network model;

Algorithm 2: Uncertainty Calculation with Jaccard Index Check.

Input: Model logits $f(G_1, G_2)$; edge sets E_1, E_2; uncertainty threshold τ; Jaccard index threshold J_{thresh}; scaling factor α
Output: Adjusted prediction $\hat{y}$, uncertainty δ, Jaccard penalty term $\mathcal{L}_{\text{Jaccard}}$

1 Compute: $P(y|G_1, G_2) = \text{softmax}(f(G_1, G_2))$;
2 Set confidence $\gamma \leftarrow \max(P(y|G_1, G_2))$;
3 Calculate sum of exponentials: $f_{\text{sum}} \leftarrow \sum_c e^{f_c(G_1,G_2)}$;
4 Compute uncertainty δ: $\delta \leftarrow -\log(f_{\text{sum}})$;
5 Initialize Jaccard penalty term: $\mathcal{L}_{\text{Jaccard}} \leftarrow 0$;
6 **if** $\delta > \tau$ **then**
7 Calculate edge intersection $E_\cap \leftarrow$ `Intersect` (E_1, E_2);
8 Calculate edge union $E_\cup \leftarrow$ `Union` (E_1, E_2);
9 Compute Jaccard index $J \leftarrow \frac{|E_\cap|}{|E_\cup|}$;
10 **if** $J > J_{\text{thresh}}$ **then**
11 Adjust prediction $\hat{y} \leftarrow$ equivalence class;
12 **end**
13 Update Jaccard penalty term: $\mathcal{L}_{\text{Jaccard}} \leftarrow \alpha \cdot |y - J|$;
14 **end**
15 **return** $\hat{y}$, δ, $\mathcal{L}_{\text{Jaccard}}$;

is the bias term that shifts the output of the linear transformation, allowing the model to represent features that are not centered at zero. The `ReLU` activation ensures that only positive values from the transformed embedding $W_l \cdot e_v^{(l-1)} + b_l$ contribute to the next layer's embedding $e_v^{(l)}$ (*line 10*). Taking only positive values through `ReLU` is important because it induces sparsity in the embeddings, reducing noise from less significant features (negative values) and focusing on meaningful contributions, which enhances model efficiency and robustness. After passing through all layers, logits $f(G_1, G_2)$ are computed via a forward pass, representing the model's predictions for equivalence classification (*line 11*). These logits are then converted into softmax probabilities for classification (*line 12*). The base cross-entropy loss, $\mathcal{L}_{\text{base}}$, is computed using the predicted probabilities and the true labels, quantifying the model's prediction error (*line 13*). However, there may be cases where the model's predictions are uncertain, indicating the model has low confidence associated with its prediction. In such cases, the **`JaccardCheck`** function is invoked (described in Section IV-C). This function computes the uncertainty metric δ and adjusts the loss term, if required, to include the Jaccard-adjusted loss $\mathcal{L}_{\text{Jaccard}}$. The adjustment is contingent on the Jaccard index threshold J_{thresh} and the uncertainty threshold τ (*line 14*).

When $\delta > \tau$, the total loss is updated, as shown in Equation 4 (*lines 15–17*):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{base}} + \beta \cdot \mathcal{L}_{\text{Jaccard}} \tag{4}$$

In Equation 4, β is a scaling factor controlling the contribution of the adjusted loss, and can be treated as a hyperparameter of the GCN model we develop.

If the uncertainty δ is below τ, the total loss is set to the base cross-entropy loss $\mathcal{L}_{\text{base}}$ (*lines 18–20*). The Jaccard-based adjustment ensures that structural discrepancies in the graphs are penalized, thereby improving the robustness of the training process. Moreover, this adaptive loss adjustment enables the model to dynamically focus on hard-to-classify instances where structural similarity is a factor. The model parameters are updated via backpropagation using the total loss $\mathcal{L}_{\text{total}}$, gradually minimizing it over the course of training (*line 21*). After all the epochs are complete, the final trained model is returned, ready for validation and testing (*lines 22–24*).

C. Uncertainty-Driven Loss Adjustment

A central element of our methodology is a novel *uncertainty-driven Jaccard index validation* with loss adjustment, which enhances both the robustness and accuracy of equivalence checking. When our GCN model's prediction confidence drops below a predefined threshold, we incorporate the Jaccard index to serve as a secondary validation measure. The Jaccard index is utilized for its efficiency and effectiveness in capturing the overlap between the edge sets of graphs [33]. In the context of equivalence checking, it serves to identify discrepancies arising from subtle changes in the ZX graph representations, often caused by erroneous gate insertions in the original quantum circuit. We integrated the Jaccard index into a specially designed loss function during training, ensuring the model prioritizes learning in high-uncertainty scenarios. This process, (summarized as the `JaccardCheck` function in Section IV-B) is detailed in Algorithm 2. This algorithm can be broken down into *three* distinct phases, which are explained subsequently.

1) Uncertainty Threshold and Jaccard Index Validation: The GCN model outputs a probability distribution over the two classes (equivalent and non-equivalent) for each pair of graphs, represented in Equation 5 as follows:

$$P(y|G_1, G_2) = \text{softmax}(f(G_1, G_2)) \tag{5}$$

In Equation 5, $f(G_1, G_2)$ denotes the logits for the classes, and $P(y|G_1, G_2)$ is the probability that graphs G_1 and G_2 are equivalent. The *confidence score* and the *uncertainty* is computed are given by Equations 6 and 7, respectively, as follows:

$$\gamma = \max(P(y|G_1, G_2)) \tag{6}$$

$$\delta = -\log\left(\sum_c e^{f_c(G_1,G_2)}\right) \tag{7}$$

In Equation 6, the softmax probabilities $P(y|G_1, G_2)$ are computed to determine the model's predicted probabilities (*line 1*) in Algorithm 2. The model's confidence score γ is set to the maximum of these probabilities, representing the model's confidence in its prediction (*line 2*). Following this, the uncertainty score δ is calculated, as shown in Equation 7, using the sum of exponentials of the logits $f_c(G_1, G_2)$ for each class c, capturing the model's uncertainty in its prediction (*lines 3-4*) of Algorithm 2.

2) Jaccard Index Calculation and Threshold Adjustment: Let E_1 and E_2 be the edge sets of graphs G_1 and G_2. The Jaccard index $J(E_1, E_2)$ is defined in Equation 8:

$$J(E_1, E_2) = \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|} \tag{8}$$

If the uncertainty score δ is above a set threshold τ, we proceed with calculating the Jaccard index J on the edge sets of the two graphs. The intersection $E_\cap$ and union $E_\cup$ of the edge sets E_1 and E_2 are calculated to compute J, (*lines 6-9*) of Algorithm 2. A threshold, J_{thresh} is selected empirically to maximize the separation between equivalent and non-equivalent pairs. If $J(E_1, E_2) > J_{\text{thresh}}$, we adjust the prediction $\hat{y}$ to indicate equivalence (*lines 10-12*).

3) Loss Function Modification with Jaccard Index: To incorporate the Jaccard index into training, we adjust the loss function based on both the uncertainty score and the Jaccard index value. Let $\mathcal{L}_{\text{base}}$ denote the standard cross-entropy loss for the GCN model, as shown in Equation 9:

$$\mathcal{L}_{\text{base}} = -\sum_{i=1}^{N} y_i \log P(y_i|G_1, G_2) \tag{9}$$

In Equation 9, y_i is the true label for the i-th example in the batch, and N is the batch size. A Jaccard-based penalty term $\mathcal{L}_{\text{Jaccard}}$ is defined to reinforce correct predictions for high-uncertainty cases. This term is based on the absolute difference between the model's prediction and the Jaccard index decision, shown in Equation 10:

$$\mathcal{L}_{\text{Jaccard}} = \alpha \cdot |\hat{y} - J(E_1, E_2)| \tag{10}$$

In Equation 10 α is a scaling factor that controls the impact of the Jaccard-based adjustment, and $\hat{y}$ is the model's predicted equivalence probability. This penalty term is updated (*line 13*). The total loss $\mathcal{L}_{\text{total}}$ is computed as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{base}} + \beta \cdot \mathcal{L}_{\text{Jaccard}} \tag{11}$$

where the uncertainty-based weighting factor β is defined by:

$$\beta = \begin{cases} 1 & \text{if} \quad \delta > \tau, \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Our proposed loss function incorporates a penalty term for uncertain predictions, aligning the model's output with the Jaccard index for structurally similar graphs. This adjustment improves the GCN's accuracy and robustness by reinforcing learning in high-uncertainty cases where structural similarity is high.

V. Evaluation of the Proposed Framework

A. Experimental Setup

Benchmarks for training: For training our proposed GCN model, we utilized 16 distinct benchmarks, with eight different transpilation configurations at different stages of compilation, obtained from MQTBench [34]. Furthermore, each benchmark circuit was considered for a range of four to sixteen qubits, to furnish a total of *1792* quantum circuits to allow our GCN model to adequately capture traits pertaining to equivalence of quantum circuits.

Benchmarks for evaluation: To evaluate our verification framework, we consider a total of 25 distinct benchmark circuits (*all scalable benchmarks*), with 4-120 qubits, available from MQTBench and their transpiled configurations, conducted on IBM Qiskit [35]. The circuits are converted into ZX graph representations using PyZX [36].

Model evaluation metrics: We assess our GCN model's performance using accuracy, precision, and recall: *accuracy* measures overall correctness in classifying circuit equivalence, *precision* evaluates the reliability of positive (equivalent) classifications, and *recall* gauges the model's ability to identify all actual equivalent circuits.

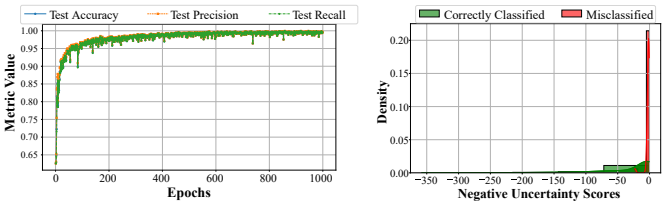
Comparison metrics: To compare the performance of our equivalence checking framework with existing models, we utilize *scalability* of the approach, signifying the number of qubits up to which we can verify quantum circuit equivalence, followed by *time consumption*, signifying the time taken to verify per qubit. Finally, we also compare the *accuracy* for verification of ZXNet with state-of-the-art.

B. Evaluation

For quantum circuit equivalence checking, we utilize our novel GCN model, which consists of *two graph convolutional layers with 64 hidden channels each*, a *single global mean pooling layer*, and *two fully connected layers* with ReLU activations. The architecture is designed to process paired ZX graph representations and output logits for equivalence classification. *It should be noted that our model conducts binary classification, where we feed the model with two quantum circuits to subsequently determine their equivalence.*

Figure 4a depicts the precision, recall, and accuracy of our model across epochs. The y-axis indicates the values each metric achieves across epochs, represented in the x-axis. From this figure, it can be observed that ZXNet furnishes accuracy, precision, and recall values of up to 99.4%, 99.2%, and 99.8%, respectively. Furthermore, the figure depicts a strong overlap among these metrics, indicating balanced performance across classes. This alignment suggests that our model effectively generalizes new data [37]. Additionally, the close values of precision and recall demonstrate that our model neither over-predicts equivalences nor misses true ones. This balance highlights the reliability of our approach in accurately identifying circuit equivalences.

Figure 4b illustrates the distribution of negative uncertainty scores for correctly classified and misclassified instances in our GCN model. The x-axis represents negative uncertainty scores, where lower (more negative) values denote higher model confidence, while values closer to zero indicate lower confidence. The y-axis shows the density of data points at specific uncertainty levels. The correctly classified instances, denoted by the green curve, span a wide range of scores from -350 to 0 with a modest peak density of 0.015, reflecting the model's ability to confidently classify across diverse uncertainty



(a) Accuracy, Precision, and Recall values during GCN model validation.

(b) Uncertainty distribution following GCN model training.

Fig. 4: GCN model evaluation of our proposed ZXNet framework.

TABLE II: Error detection accuracy using ZXNet for different benchmarks and induced errors.

Benchmark	Detection of Induced Error (%)					
	Add 10 CX gates	Remove 10 CX gates	Add 5 CX gates	Remove 5 CX gates	Add 1 CX gate	Remove 1 CX gate
Grover	98.21	97.46	97.1	97.3	96.6	96.18
Random walk	99.16	98.75	97.68	98.24	96.36	97.3
QPE-exact	98.88	98.24	97.25	97.25	96.2	96.01

levels. In contrast, the red curve, representing misclassified instances, is sharply concentrated near -0.1 with a peak density of 0.23, indicating that misclassifications primarily occur at low confidence levels. This separation highlights our model's robustness in distinguishing between confident predictions and uncertain misclassification.

C. Error detection using ZXNet

Accurate equivalence-checking frameworks must be sensitive enough to detect and distinguish even minor discrepancies that lead to non-equivalence between quantum circuits. To evaluate this capability, we assessed error detection using our ZXNet framework. Specifically, we introduced variations in quantum circuits by arbitrarily adding or removing `CX` gates and tested whether ZXNet could identify these changes. The results, summarized in Table II, highlight the framework's accuracy in detecting non-equivalence across various benchmarks. The first column lists the benchmark circuits (these were selected since they are commonly used algorithms with real-world applications), while the subsequent columns show detection accuracy for different types of errors. It should be noted that the errors of addition and removal of `CX` gates were conducted randomly throughout the quantum circuit, and the accuracy of detection was averaged over 10 executions to observe the general performance of ZXNet. `CX` gates were chosen for this study due to their significant impact on functional equivalence, making any modifications to their numbers during transpilation particularly impactful [38]. From the table, it can be observed that **ZXNet can detect errors with an accuracy of up to 99.16%** across the tested benchmarks. Moreover, ZXNet demonstrates remarkable sensitivity, **being able to detect discrepancies caused by the addition or removal of a single `CX` gate with up to 97.3% accuracy**, further underscoring the robustness of our framework.

D. Comparative analysis against state-of-the-art approaches

We compare ZXNet, our learning-based equivalence checking framework, against three state-of-the-art methods: MPO-based, DD-based, and rule-based ZX calculus verification [9], [10], [12]. The comparison evaluates two critical metrics: *scalability* and *runtime efficiency*. Parameterized two-local random circuits with linear entanglement are utilized as benchmarks to maintain consistency, as existing results for MPO-based methods are available exclusively for this circuit class. The results, illustrated in Figure 5, present a scalability analysis, with the x-axis representing the number of qubits and the y-axis denoting runtime. The figure demonstrates that ZXNet is consistently faster compared to state-of-the-art methods, **achieving speedups of up to 62×.** *Please note that since Parameterized Two-local random was the only available benchmark circuit that has been evaluated using all existing approaches*, we utilized it in this section for comparison.

TABLE III: Comparison vs state-of-the-art methods for verification.

Name	Gate Depth	Equivalent			Speedup (%)	Non-equivalent			Speedup (%)
		t_{zx} (s)	t_{dd} (s)	t_{zxnet} (s)		t_{zx} (s)	t_{dd} (s)	t_{zxnet} (s)	
Grover_n9	12482	12.15	0.14	0.1	**28.57**	3.16	8.26	0.31	**90.19**
Random walk_n8	14084	1289.13	0.57	0.3	**47.37**	78.91	62.2	2	**96.78**
QPE-Exact_n39	3823	3.19	>3600	0.32	**89.97**	2.89	>3600	0.3	**86.16**

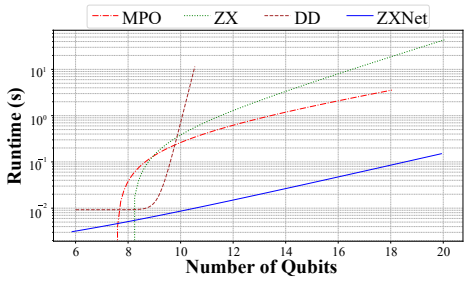


Fig. 5: Scalability comparison of ZXNet with state-of-the-art.

Next, we conduct a more detailed evaluation of ZXNet's performance and compare the results against rule-based ZX calculus and decision diagrams (DD) [10], [12]. The MPO-based method is excluded from this comparison because its results are limited to QFT circuits [9]. For brevity, the evaluation results are limited to three widely used benchmark circuits, previously analyzed by state-of-the-art methods, as depicted in Table III. In this table, columns one to three depict the name, number of qubits, and number of gates of the benchmark circuit, respectively. Columns four through six indicate the time taken by each approach to verify the functional equivalence of the transpiled benchmark with the original in cases when they are equivalent. Column seven showcases the speedup provided by our ZXNet framework over the state-of-the-art methods. Similarly, columns eight to eleven replicate the contents of columns four to seven, but for scenarios when the transpiled benchmark is not equivalent to the existing benchmark. From the table, it is evident that as gate depth increases, ZXNet, consistently outperforms the rule-based ZX verification and DD-based verification, **achieving speedups of up to 96.78%.** These findings demonstrate that ZXNet improves verification efficiency and scales effectively to handle larger and more complex quantum circuits compared to existing methods.

Furthermore, we conducted a comprehensive evaluation of our ZXNet framework across all the scalable benchmarks obtained from MQTBench, with qubit counts ranging from 4 to 120 [34]. Upon evaluation, ZXNet demonstrated its superiority over state-of-the-art methods by **successfully verifying quantum circuits with up to 120 qubits, achieving a 45.83% increase in circuit size, a 5.94% improvement in verification accuracy, and a 42.22% reduction in verification time**, compared to existing approaches. These results establish ZXNet as a more scalable, efficient, and accurate framework for quantum circuit equivalence checking.

VI. Conclusion

In this paper, we introduced ZXNet, a novel equivalence-checking framework for quantum circuits that leverages ZX graph representations in an encoded format for training a GCN model to verify the functional equivalence of quantum circuits. Our experimental results demonstrated that ZXNet achieves notable **accuracy, precision, and recall values of up to 99.4%, 99.2%, and 99.8%**, respectively. Compared to traditional equivalence-checking methods, ZXNet significantly improves computational efficiency, achieving a speedup of up to 62× over existing methods. Moreover, **our model furnishes improvements of up to 45.83%, 42.22%, and 5.94% in terms of scalability, time consumption, and accuracy**, respectively, over state-of-the-art methods. This establishes a robust framework for reliable quantum circuit verification by reducing computational overhead and enhancing the scalability and adaptability to varying circuit complexities.

References

[1] T. Monz et al., "Realization of a Scalable Shor Algorithm," Science, vol. 351, no. 6277, pp. 1068–1070, 2016.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in ACM TC, 1996, pp. 212–219.

[3] J. P. Lowe and K. Peterson, Quantum chemistry. Elsevier, 2011.

[4] Y. Li, M. Tian, G. Liu, C. Peng, and L. Jiao, "Quantum optimization and quantum learning: A survey," Ieee Access, vol. 8, pp. 23 568–23 593, 2020.

[5] C. Lu, S. Kundu, A. Arunachalam, and K. Basu, "Survey on quantum noise-aware machine learning," in 2022 IEEE 15th Dallas Circuit And System Conference (DCAS). IEEE, 2022, pp. 1–2.

[6] E. Wilson, S. Singh, and F. Mueller, "Just-in-time quantum circuit transpilation reduces noise," in 2020 IEEE international conference on quantum computing and engineering (QCE). IEEE, 2020, pp. 345–355.

[7] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 9, pp. 1810–1824, 2020.

[8] X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, "Approximate equivalence checking of noisy quantum circuits," in 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 637–642.

[9] A. Sander, L. Burgholzer, and R. Wille, "Equivalence checking of quantum circuits via intermediary matrix product operator," 2024. [Online]. Available: https://arxiv.org/abs/2410.10946

[10] C. Lu, N. Choudhury, U. Banerjee, A. A. Saki, and K. Basu, "Qubec: Boosting equivalence checking for quantum circuits with qec embedding," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 7, pp. 2037–2042, 2024.

[11] L. Burgholzer and R. Wille, "The power of simulation for equivalence checking in quantum computing," in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.

[12] T. Peham, L. Burgholzer, and R. Wille, "Equivalence checking of quantum circuits with the zx-calculus," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 12, no. 3, pp. 662–675, 2022.

[13] S. Das, S. Kundu, P. Madhusoodhanan, P. Viswanathan Pillai, R. Parekhji, A. Raha, S. Banerjee, S. Natarajan, and K. Basu, "Graph learning-based fault criticality analysis for enhancing functional safety of e/e systems," in Proceedings of the 61st ACM/IEEE Design Automation Conference, ser. DAC '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3649329.3656498

[14] M. Balcilar, P. Heroux, B. Gauzere, P. Vasseur, S. Adam, and P. Honeine, "Breaking the limits of message passing graph neural networks," in Proceedings of the 38th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 599–608. [Online]. Available: https://proceedings.mlr.press/v139/balcilar21a.html

[15] H. Cardot and O. Lezoray, "Graph of neural networks for pattern recognition," in 2002 International Conference on Pattern Recognition, vol. 2, 2002, pp. 873–876 vol.2.

[16] T. Wong, Introduction to Classical and Quantum Computing. Rooted Grove, 2022. [Online]. Available: https://books.google.com/books?id=M3jqzgEACAAJ

[17] N. S. Yanofsky and M. A. Mannucci, Quantum Computing for Computer Scientists. Cambridge University Press, 2008.

[18] M. A. Nielsen et al., Quantum Computation and Quantum Information. Cambridge University Press, 2010.

[19] B. Coecke and R. Duncan, "Tutorial: Graphical calculus for quantum circuits," in Reversible Computation, R. Glück and T. Yokoyama, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–13.

[20] B. Coecke, "Quantum picturalism," Contemporary Physics, vol. 51, pp. 59–83, 2009.

[21] A. Borgna, S. Perdrix, and B. Valiron, "Hybrid quantum-classical circuit simplification with the ZX-calculus," in Programming Languages and Systems, H. Oh, Ed. Cham: Springer International Publishing, 2021, pp. 121–139.

[22] M. Hanks, M. P. Estarellas, W. J. Munro, and K. Nemoto, "Effective compression of quantum braided circuits aided by zx-calculus," Phys. Rev. X, vol. 10, p. 041030, Nov 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.10.041030

[23] L. Yeh, E. Dasgupta, and E. Winston, "Benchmarking ZX-Calculus Circuit Optimization Against Qiskit Transpilation," ACM SRC Grand Finals Candidates, 2020.

[24] A. Said, M. Shabbir, B. Broll, W. Abbas, P. Völgyesi, and X. Koutsoukos, "Circuit design completion using graph neural networks," Neural Computing and Applications, vol. 35, no. 16, pp. 12 145–12 157, 06/01 2023. [Online]. Available: https://doi.org/10.1007/s00521-023-08346-x

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017. [Online]. Available: https://arxiv.org/abs/1609.02907

[26] X. Hong, M. Ying, Y. Feng, X. Zhou, and S. Li, "Approximate equivalence checking of noisy quantum circuits," in 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 637–642.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2019. [Online]. Available: https://arxiv.org/abs/1810.00826

[28] U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states," Annals of Physics, vol. 326, no. 1, p. 96–192, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.aop.2010.09.012

[29] X. Hong, X. Zhou, S. Li, Y. Feng, and M. Ying, "A tensor network based decision diagram for representation of quantum circuits," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 27, no. 6, pp. 1–30, 2022.

[30] M. Nägele and F. Marquardt, "Optimizing zx-diagrams with deep reinforcement learning," Machine Learning: Science and Technology, vol. 5, no. 3, p. 035077, Sep. 2024. [Online]. Available: http://dx.doi.org/10.1088/2632-2153/ad76f7

[31] H. Chen, Z. Huang, Y. Xu, Z. Deng, F. Huang, P. He, and Z. Li, "Neighbor enhanced graph convolutional networks for node classification and recommendation," Knowledge-Based Systems, vol. 246, p. 108594, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122002660

[32] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An empirical study on generalizations of the relu activation function," in Proceedings of the 2019 ACM Southeast Conference, ser. ACMSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 164–167. [Online]. Available: https://doi.org/10.1145/3299815.3314450

[33] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," Bulletin de la Société Vaudoise des Sciences Naturelles, vol. 37, pp. 547–579, 1901.

[34] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Bench: Benchmarking software and design automation tools for quantum computing," Quantum, 2023, MQT Bench is available at https://www.cda.cit.tum.de/mqtbench/.

[35] IBM Research, "Qiskit: Open-Source Quantum Development," Accessed: March 2022. [Online]. Available: https://qiskit.org.

[36] A. Kissinger and J. van de Wetering, "Pyzx: Large scale automated diagrammatic reasoning," arXiv preprint arXiv:1904.04735, 2019.

[37] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," BMC genomics, vol. 21, pp. 1–13, 2020.

[38] M. Bataille, "Quantum circuits of cnot gates: optimization and entanglement," Quantum Information Processing, vol. 21, no. 7, p. 269, 2022.