

Prediction of Compilation Options for Quantum Circuits via Adaptive Deep Residual Neural Network

Shouli He ¹, Wen Liu ^{1,2,3,†,*}, Yangzhi Li ^{4,†,*}, Maoduo Li ¹, Kai Chen ¹ and Yaohua Lu ⁵

¹ School of Computer and Cyber Sciences, Communication University of China, Beijing 100024, China; hsl0108@qq.com (S.H.)

² State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China

³ Laboratory of Convergent Media and Intelligent Technology (Communication University of China), Ministry of Education, Beijing 100034, China

⁴ School of Artificial Intelligence, Changsha University of Science and Technology, Changsha 410114, Hunan, China

⁵ School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

[†] These authors contribution equally.

* Corresponding author: lw_8206@163.com (W.L.); yangzhilee@163.com (Y.L.)

Received date: 10 October 2025; Accepted date: 16 December 2025; Published online: 8 January 2026

Abstract: Quantum circuit compilation is an indispensable part of executing quantum circuits on quantum devices. It translates logical quantum circuits into physical quantum circuits that meet the device constraints. However, non-specialist users require extensive expertise to effectively select and evaluate quantum circuit compilation processes. Therefore, this paper proposes a quantum circuit compilation option prediction method based on Deep Residual Attention Neural Network. The method aims to automate the selection of the best compilation scheme and enhance the efficiency and accuracy of quantum circuit compilation. Firstly, a quantum circuit feature extraction algorithm based on Time-Weighted Interaction Graph is proposed. This algorithm can effectively represent quantum circuits using fixed-length vectors, making the model input scalable. Secondly, for existing one-dimensional features, a quantum circuit compilation option prediction model based on Deep Residual Attention Neural Network is designed and trained. This model is capable of quickly and accurately predicting the optimal compilation option combination. Extensive experimental evaluations and comparisons are carried out on the MQT (Munich Quantum Toolkit) quantum circuit compilation dataset. These evaluations verify the effectiveness and advancement of the proposed method. Compared with the current state-of-the-art method (Random Forest), the proposed method improves the accuracy by 5.44%, the Top-3 accuracy by 2.6%, and the F1 by 2.05%.

Keywords: quantum circuit compilation; deep residual neural network; attention mechanism; quantum computing; deep Learning

1. Introduction

With the continuous development of quantum science, quantum technology and theory are gradually maturing. This has led to the emergence of diverse quantum devices, each with varying technical approaches and associated software tools. In recent years, industry giants such as IBM, Quantinuum, Rigetti, and Xanadu have made significant strides. They have launched their own quantum devices and software development kits (SDKs), including Qiskit [1], TKET [2], Forest [3], and PennyLane [4]. These developments have accelerated the applications of quantum computing. They have showcased tremendous potential and advantages in addressing complex problems that are challenging for classical computing to solve. Examples include Grover's algorithm [5] for non-sequential database searching, Shor's algorithm [6] for prime factorization problems, and Harrow-Hassidim-Lloyd (HHL) algorithm [7] for linear equation problems. However, similar to classical computing, these quantum algorithms necessitate compilation. They must be compiled using appropriate software to align with the native gate sets, connectivity, and other constraints of the target quantum devices.

The compilation of quantum circuits involves multiple key steps, each presenting different choices—namely, compilation options. These include the selection of qubit technology, quantum device, quantum compiler, as well as specific compilation steps and optimization strategies (e.g., decomposition, mapping, and optimization). Faced with such a multitude of options, effectively selecting the optimal combination is undoubtedly a highly challenging task for ordinary end-users, especially those lacking background knowledge in quantum computing. The selection of the optimal compilation option combination has a crucial impact on the compilation efficiency and performance of quantum circuits (e.g., depth and fidelity). This reliance on expert knowledge and the high computational costs pose a significant barrier to the widespread adoption of quantum computing.

To address the optimal selection of compilation option combinations for quantum circuits, a series of researches have been carried out. In 2020, Salm et al. [8] proposed a rule-based method for selecting quantum circuit compilation options. This method can automate the selection of quantum devices for specific quantum algorithms, thereby improving the efficiency of quantum circuit execution. In 2022, Salm et al. [9] proposed a quantum circuit compilation option selection method based on multi-criteria decision analysis. This method enables the automated selection of quantum device and quantum compiler for given quantum circuits, achieving stable execution results. In 2023, Salm et al. [10] proposed an option selection method for quantum circuit compilation based on machine learning (ML) algorithm. This method can automatically select quantum device and quantum compiler for a given quantum circuit according to user requirements. It has the advantages of reducing resource consumption and enhancing method scalability. Also in 2023, Quetschlich et al. [11] proposed a selection method for quantum circuit compilation options based on machine learning technology. This method can automatically select qubit technology, quantum device, quantum compiler, as well as specific compilation steps and optimization strategies for a given quantum circuit, thereby reducing compilation time and enhancing compilation efficiency. In another 2023 study, Quetschlich et al. [12] proposed a method for selecting compilation options for quantum circuits based on reinforcement learning technology. This method can select appropriate compilation steps and optimization strategies for different quantum circuits. It enhances the quality of quantum circuit execution on quantum device. Furthermore, in 2023, Quetschlich et al. [13] proposed a quantum circuit compilation option selection method based on machine learning and reinforcement learning technologies. This method can automatically select the optimal qubit technology, quantum device, quantum compiler, as well as specific compilation steps and optimization strategies for different quantum algorithms. Not only does it significantly improve the compilation quality of quantum circuits, but it also provides the MQT Predictor framework as a practical tool for non-quantum computing experts to use quantum computing. Although these automated selection methods have alleviated the difficulty of manual selection to some extent, there is still room for improvement in the accuracy of predicting the best combination of options. Especially in the representation of quantum circuits and the use of quantum circuit compilation option prediction model, further research and improvement are needed to improve the accuracy and reliability of prediction.

Regarding the research on the representation of quantum circuits, in 2022, Tomesh et al. [14] proposed a composite feature representation method of quantum circuits based on application-level metrics. This method can better represent the interaction features within quantum circuits and comprehensively represent the hardware resource requirements of quantum circuits. In 2023, Bandic et al. [15] proposed a quantum circuit representation method based on interaction graphs, which can more effectively extract the edge features of quantum circuit interaction graphs to optimize the mapping of quantum circuits. Also in 2023, Qian et al. [16] proposed a quantum circuit representation method based on the priority of quantum gate timing weights. This method can synthesize the interaction graph and dependency graph to effectively extract the interaction frequency and importance of qubits in the quantum circuit, so as to optimize the mapping and conversion process of the quantum circuit. Currently, in the prediction of quantum circuit compilation options, the width (i.e., the number of qubits involved) and depth (including the number of single-qubit gates, two-qubit gates, and sequential executable gates) of quantum circuits, as well as the number of different types of gates and five application-level composite features (program communication, critical depth, entanglement ratio, parallelism, liveness, measurement) [14] have been used. However, the existing representation methods of quantum circuits have not fully considered other characteristic forms of quantum circuits, and lack sufficient attention to the arrangement and combination of features. This means that current representation methods fail to fully utilize all available information to represent quantum circuits. The representation of quantum circuits is crucial for the prediction model of quantum circuit compilation options, affecting the accuracy of prediction results. Therefore, to more accurately predict quantum circuit compilation options, it is necessary to further explore and develop new representation methods that can comprehensively capture quantum circuit information.

Regarding the research on the prediction model problem of quantum circuit compilation options, in 2023, Quetschlich et al. [11] applied a variety of traditional machine learning prediction models to predict quantum circuit compilation options. These include Random Forest [17], Gradient Boosting [18], Decision Tree [19], Nearest Neighbor [20], Multi-layer Perceptron [21], Support Vector Machine [22], and Naive Bayes [23]. Among them, Random Forest model has been proved to be the most effective choice due to its excellent performance [11]. However, with the continuous development of machine learning, its branch deep learning has made significant progress in various fields compared with traditional machine learning. For example, in 2016, He et al. [24] proposed an image recognition method based on deep residual network model (ResNet). This method employed skip connections to effectively solve the issues of gradient vanishing and exploding during model training. In 2017, Vaswani et al. [25] proposed a machine translation method completely based on the Transformer network model with an attention mechanism. This method greatly improves the model's perception and processing ability of important information. It also improved parallelism, thereby significantly reducing the time required for model training. Also in 2017, Wang et al. [26] proposed an image classification method based on the residual attention network model. This method can generate attention-aware features and adapt to the depth change of the model, so that the model can perform arbitrary depth transformation for specific problems. These studies indicate that deep learning has great application potential in the prediction of quantum circuit compilation options, further enhancing the accuracy and reliability of such predictions.

Therefore, we propose a new prediction method for quantum circuit compilation options based on Deep Residual Attention Neural Network (DRAN-Net). It not only enables common end-users to automatically select the optimal compilation options but also improves the efficiency and accuracy of quantum circuit compilation, thereby paving the way for building fully adaptive and scalable intelligent compilation systems in the future. Specifically, the main contributions of this paper are as follows:

(1) A feature extraction algorithm of quantum circuit based on Time-Weighted Interaction Graph (TWIG) is proposed. This algorithm transforms quantum circuit into TWIG and extracts the features of average weighted shortest path, maximum degree, minimum degree, average clustering coefficient, and standard deviation of the adjacency matrix. By combining these new features with traditional features, we construct a 36-dimensional vector that can more fully represent quantum circuit information and make the deep learning model scalable.

(2) A quantum circuit compiler option prediction model based on DRAN-Net is proposed. The model combines the residual connection mechanism and the attention mechanism. Specifically designed for existing features, the residual connections mechanism addresses the issue of gradient vanishing in deep networks, while the attention mechanism allows the model to focus on important features. By integrating the above technologies, a model with greater adaptability to the quantum circuit compilation option prediction task is constructed. This model not only effectively handles complex data patterns but also significantly improves generalization ability, thereby achieving fast and accurate prediction of quantum circuit compilation option sequences.

(3) Extensive experimental evaluations and comparisons are conducted on the MQT Bench [11] dataset, which contains 3,003 quantum circuits. It verifies the effectiveness and advancement of the proposed method. In the feature correlation experiment, it is verified that the proposed features have low correlation with traditional features. Their correlation with other features is less than 95%, thereby ensuring the inherent validity of the proposed features. In the feature importance experiment, it is verified that the importance of some proposed features in model prediction is greater than 0.1. This further proves the effectiveness of the proposed features for model training. In the ablation experiments, compared with traditional representation methods, the proposed quantum circuit representation method improves the performance of all or most metrics of all baselines, which proves the effectiveness and robustness of the proposed quantum circuit representation method. Compared with traditional machine learning methods, the proposed prediction model DRAN-Net significantly improves all performance metrics, which demonstrates the effectiveness and robustness of DRAN-Net. In the advancedness comparison experiment, compared with the current state-of-the-art method Random Forest [11], the accuracy of the proposed method is increased by 5.44%, Top-3 by 2.6%, and F1 by 2.05%. These results prove the advancement of the proposed method.

(4) The rest of the work in this paper is as follows: In Section 2, we review the fundamentals of quantum computing and the compilation option flow, as well as the principles of deep residual neural network and attention mechanism. On this basis, the specific details of the proposed method are introduced in Section 3, the experimental results are analyzed and evaluated in Section 4, and the work in this paper is summarized in Section 5.

2. Background Knowledge

This section provides a brief introduction to the fundamentals of quantum computing and compilation, as well as a technical introduction to deep residual neural networks and attention mechanisms.

2.1. Fundamentals of Quantum Computing and Compilation

2.1.1. Quantum Computing

1. Quantum Bit, Qubit: Qubits are the basic unit of quantum computing. Unlike bits in classical computing, qubits can be in a superposition of 0 and 1. Classical bits can only take on the value of 0 or 1, while qubits can be represented by a state vector $|\psi\rangle$ whose states are:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex numbers, representing the probability that qubits are measured as $|0\rangle$ and $|1\rangle$, respectively, and they satisfy the equation $|\alpha|^2 + |\beta|^2 = 1$. This superposition gives qubits great potential when dealing with complex problems.

2. Quantum Gate: Quantum gates are the basic operation units in quantum computing that are used to transform qubits. Different from logic gates in classical computing, quantum gates are linear transformations of quantum states, commonly represented by unitary matrices. Common single-qubit quantum gates include Pauli-X gate (analogous to classical NOT gate), Hadamard gate (H-gate, used to create superposition states), and Controlled NOT gate (CNOT-gate). These gate operations can be represented as the following matrix:

- X-gate (Pauli-X Gate):

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

It flips the state of the qubit, changing $|0\rangle$ to $|1\rangle$ and vice versa. Its symbol representation in a quantum circuit is shown in Figure 1a.

- H-gate (Hadamard Gate)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

It converts qubits from the ground state ($|0\rangle$ or $|1\rangle$) to a superposition state, whose symbol representation in a quantum circuit is shown in Figure 1b.

- CNOT-gate (Controlled NOT Gate)

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

This is a two-qubit quantum gate. Only when the control qubit is $|1\rangle$, the target qubit will be flipped. Its symbol representation in a quantum circuit is shown in Figure 1c.

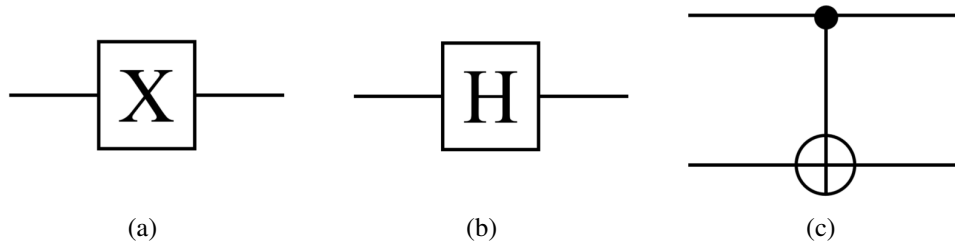


Figure 1. Circuit symbol representation of common quantum gates.

3. Quantum Circuit: A quantum circuit is a complex sequence of operations that implements quantum computing through (cascading) multiple quantum gates. It consists of qubits and quantum gates, where the qubits are used as input and are operated through a series of quantum gates to output the results. Each quantum gate acts on qubits from left to right, forming the process of quantum computing. Figure 2 shows a simple quantum circuit example, which includes an H-gate, two X-gates, and multiple CNOT-gates.

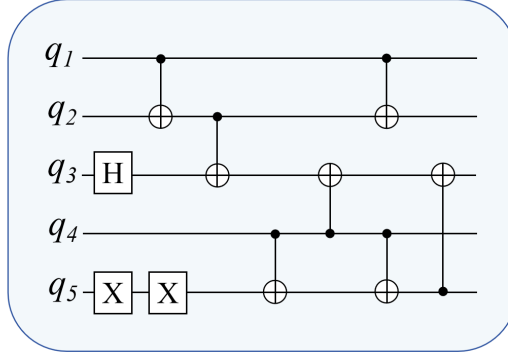


Figure 2. A simple quantum circuit example, including H-gate, X-gate, and CNOT-gate.

2.1.2. Compile Option

Quantum circuit compilation faces many choices (i.e., compilation options), which mainly include the selection of qubit technology, quantum device, quantum compiler, compilation steps, and optimization strategies. The combination of compilation options is crucial to the efficiency and performance of quantum circuit compilation.

1. Qubit Technology: Qubit technology determines the design and implementation of different quantum devices. Currently, the main qubit technology include superconducting qubits, ion traps, topological qubits, etc. Each technology has its unique advantages and limitations. For example, superconducting qubits are excellent in scalability, but they require an extremely low temperature environment; whereas ion trap qubits have long coherence times and high operational precision, yet they are constrained by slower gate operation speeds.

2. Quantum Device: A quantum device is a physical hardware that performs quantum computing, and its design is based on a specific qubit technology. Different quantum devices differ in qubit number, coupling topology, noise characteristics, and gate operation integration. Currently, the most widely adopted superconducting qubit-based devices in academic literature primarily include 27-qubits IBM Q Toronto, which adopts a two-dimensional planar coupling structure. The device coupling topology of IBM Q Toronto is shown in Figure 3a, where each qubit is connected through a coupling link. Currently, the most widely adopted ion trap-based devices in academic literature primarily include 11-qubits IonQ Harmony, whose device coupling topology is shown in Figure 3b. Unlike superconducting qubits, ion trap devices typically have a fully connected coupling graph, meaning that each qubit can operate directly with any other qubit. This fully connected topology greatly simplifies the compilation process of quantum circuits, but the technology is extremely demanding for control and prone to crosstalk.

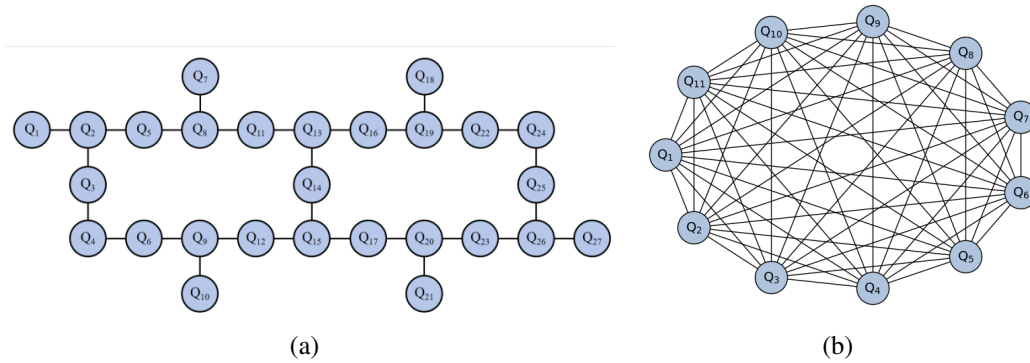


Figure 3. Common schematic diagram of quantum computing devices.

3. Quantum Compiler: Quantum compilers are tools that translate high-level quantum algorithms into quantum instructions that can be executed on specific quantum devices. Quantum circuits may be handled differently by different compilers, such as IBM's Qiskit, Google's Cirq, Quantinuum's TKET, etc. They each provide compilation steps and optimization strategies.

4. Compilation Steps and Optimization Strategies: Quantum circuit compilation mainly includes the following steps: quantum gate decomposition, qubit mapping, and optimization. The following examples explain the process of these main compilation steps.

Example 1. Take the quantum circuit shown in Figure 2 as an example. In order to make the circuit meet the restriction of the device's native gate set $\{Ry, Rz, X, CNOT\}$. The H gate is equivalently replaced by $Ry(\frac{\pi}{2})$ and $Rz(\pi)$ through the quantum gate decomposition process, as shown in Figure 4a. In order to compile the quantum circuit to the device that meets the restriction of the architecture constraint graph shown in Figure 4b, a SWAP gate is added in the mapping process to satisfy the operations between non-adjacent physical qubits Q_3 and Q_5 . When the SWAP gate is executed in the circuit, it is equivalently replaced by three CNOT gates, so the actual mapping results in the circuit shown in Figure 4c. The blue box represents the added SWAP gate. Finally, the gates in the green box in Figure 4d are optimized through the selected optimization strategy (i.e., removal of redundant gates). After the optimization is completed, the compilation circuit as shown in Figure 4e is finally obtained.

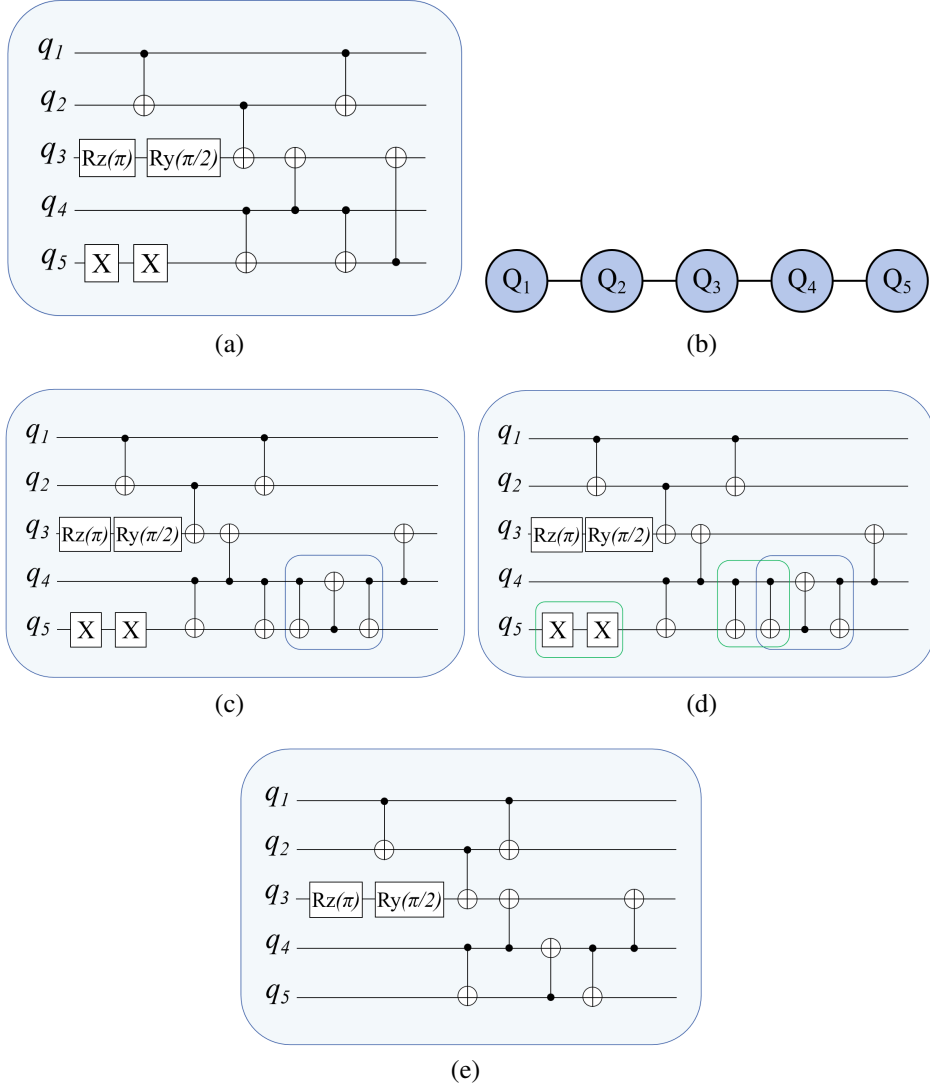


Figure 4. Quantum circuit compilation.

The compilation steps differ across compilers, and each step involves multiple choices. These choices directly impact the execution performance of the final quantum circuit. For example, Qiskit has four optimization levels (i.e., O0, O1, O2, and O3). O0 (No Optimization) does not do any optimization and is suitable for fast compilation and debugging. O1 (Level 1 Optimization) carries out basic gate-level optimizations by removing unnecessary gate operations to moderately optimize the circuit structure. O2 (Level 2 Optimization) includes all optimizations of O1. Additional circuit transformations and optimizations are performed to balance compilation time and circuit performance. O3 (Level 3 Optimization) uses all available optimization techniques in addition to containing all the optimizations of O2. It minimizes the gate count and depth of the circuit to improve circuit performance, but may significantly increase compilation time. Different optimization levels provide distinct circuit optimization strategies, leading to variations in quantum gate decomposition, mapping, circuit optimization results, and computational complexity. The choice of optimization level depends on the quantum circuit itself and the target device. TKET offers two different qubit mapping strategies: Line Placement and Graph Placement. These strategies are used to map logical qubits from quantum circuits to physical qubits from quantum devices. Line Placement maps logical qubits into a linear arrangement of physical qubits, suitable for simple device topologies. Graph Placement maps logical qubits to a graph structure of physical qubits. It optimizes operations across qubits taking into account the topological structure of the device and is suitable for complex device architectures. Therefore, the choice of mapping strategy depends on the structural constraints of the quantum device in use and the specific requirements of the circuit.

All available compilation options are defined as compilation option combinations. A complete set of compilation option combinations can accomplish the compilation of quantum circuit, and different compilation option combinations yield different compilation effects. The currently available compilation options are shown in Figure 5. It mainly includes two qubit technologies (i.e., superconducting and ion trap), five devices (i.e., 8, 27, 80, and 127-qubits devices based on superconducting and an 11-qubits device based on ion trap), two compilers (i.e., Qiskit and TKET), as well as compilation steps and optimization strategies (i.e., Qiskit's O0 to O3 optimization levels and TKET's Line Placement and Graph Placement mapping strategies). These compilation options can form a total of 30 combinations, and the sequence of options highlighted in red is one combination.

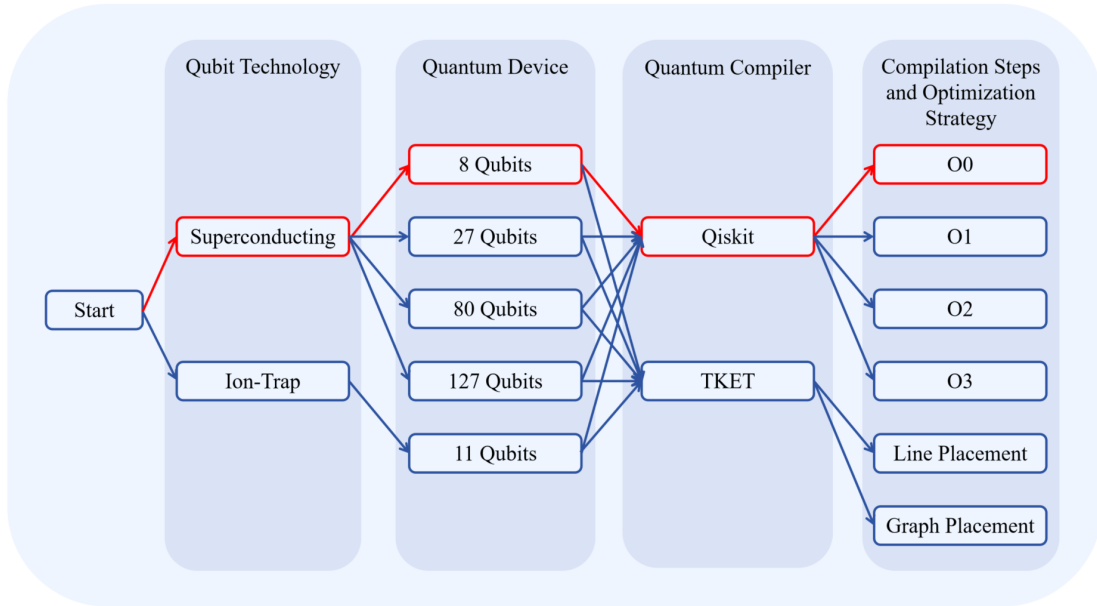


Figure 5. The currently available compilation options.

2.2. Deep Residual Neural Networks

Residual Neural Network(ResNet) is a deep neural network composed of multiple residual blocks. It is used to solve the problems of gradient vanishing and feature loss in deep network training, making the training of deep networks simpler and more effective. With the deepening of research on residual network, there have been many improved versions of deep residual neural network(ResNet). Such as deep residual shrinkage network, but its core idea is still the construction of residual block.

Residual Block is the basic unit of ResNet, and each residual block contains multiple convolutional layers (as well as other possible layers such as batch normalization and activation layers). However, the key lies in its introduction

of a skip connection, which allows the input to directly bypass certain network layers and be added to the output of subsequent layers. It enables the network layers to learn the changes in the input data directly rather than the input data itself. This design not only improves the trainability of the network but also solves the degradation problem of the deep network. A common residual block diagram is shown in Figure 6.

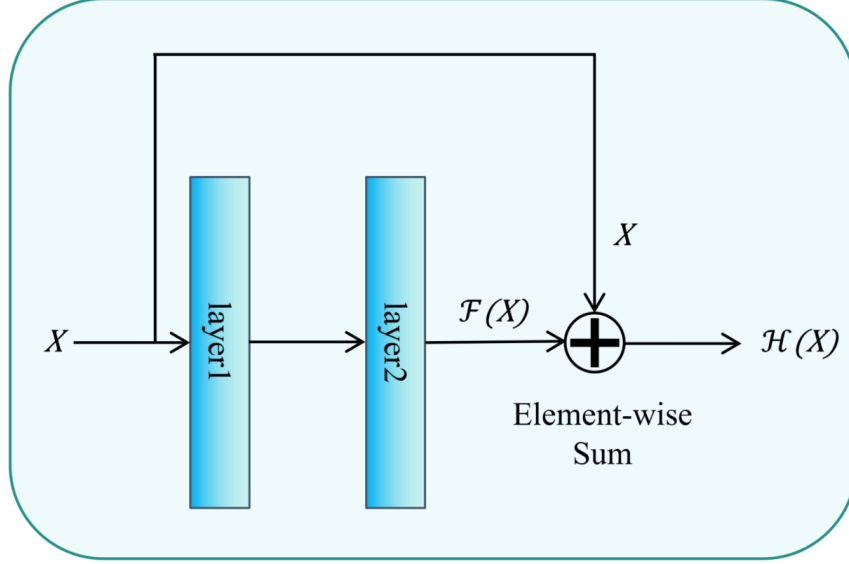


Figure 6. Residual block.

When the model input is X , the map that a certain layer in the network needs to learn is $\mathcal{H}(X)$. In the residual block, it only needs to learn the learning objective of a residual function $\mathcal{F}(X) = \mathcal{H}(X) - X$. Therefore, the output of the network can be expressed as:

$$\mathcal{H}(X) = \mathcal{F}(X) + X \quad (1)$$

With this design, the network no longer learns the complex mapping $\mathcal{H}(x)$ directly, but instead learns the relatively simple residuals $\mathcal{F}(x)$. This not only simplifies the learning task of the network but also effectively retains the input information during the training process and reduces the risk of gradient vanishing.

2.3. Attention Mechanism

Attention Mechanism is a technique that shows significant performance in serialized data processing tasks. By simulating the focusing characteristics of human attention, the model can dynamically focus on the key parts of the input data, thereby improving the processing quality and efficiency. It can generally be divided into two categories: hard attention and soft attention. The hard attention mechanism directly selects a small number of important information in the input for processing through discrete selection methods. However, its non-differentiability is typically solved with reinforcement learning methods. Soft attention mechanism assigns continuous weights to each input and performs a weighted average based on these weights. Because of their differentiability, they are easier to train through optimization methods such as gradient descent. In contrast, the soft attention mechanism not only allows the model to more precisely control the degree of attention to information but also facilitates end-to-end training through the backpropagation algorithm. Therefore, soft attention is used in this paper.

Attention Model is a neural network that incorporates an attention mechanism. The basic structure of the attention model is shown in Figure 7, which shows the entire process from input to output of the model. In the attention model, the vector s of the new hidden layer state is obtained by multiplying each initial hidden layer state h_i with the corresponding weight α_i , and then summing these products. The weight α_i reflects the weight of each initial hidden layer state in the new hidden layer state. Its calculation formula can be expressed in the following mathematical form:

$$s = \sum_{i=1}^t \alpha_i h_i; \alpha_i = \frac{e^{p_i}}{\sum_{j=1}^n p_j}; p_i = u_i \tanh(w_i h_i + b_i) \quad (2)$$

where p_i represents the energy value determined by the hidden layer state vector h_i at the i time, u_i and w_i are the weight matrix at the i time, and b_i represents the corresponding bias term at the i time. Through this design, the network model can dynamically focus on the important feature data in a period of time and amplify the influence of the important feature data, thus improving the prediction ability of the network.

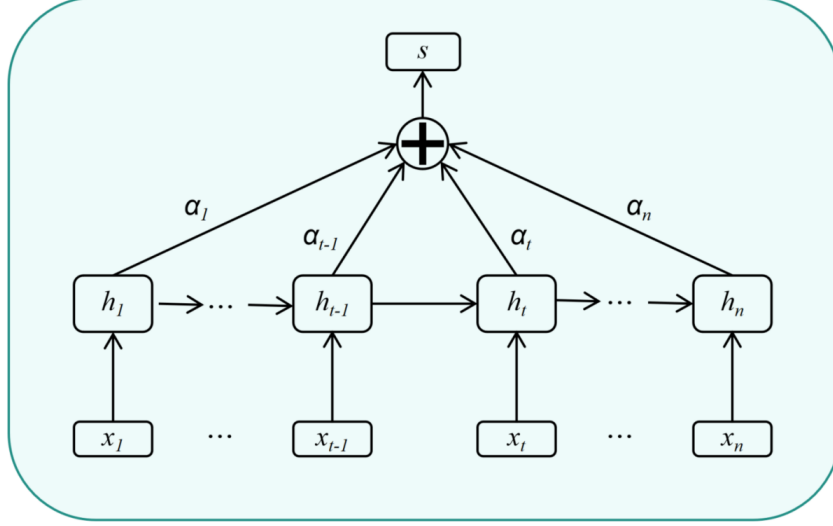


Figure 7. Basic structure of attention model.

3. Proposed Method

This section introduces a method for predicting quantum circuit compilation options based on DRAN-Net. Specifically, the proposed method is divided into the following three steps: (1) Firstly, the quantum circuit compilation option selection problem is modeled as a classification prediction task, then (2) the quantum circuit features are extracted to form a representation vector, and (3) a suitable deep neural network model is designed and trained to effectively solve the classification prediction task of selecting quantum circuit compilation options.

3.1. Problem Definition and Overview

Given a quantum circuit LC , it contains N_Q qubits $Q = \{q_1, q_2, \dots, q_{N_Q}\}$ and N_G quantum gates $G = \{g_1, g_2, \dots, g_{N_G}\}$, among which there are N_{TG} two-qubit gates $TG = \{tg_1, tg_2, \dots, tg_{N_{TG}}\}$. Given a quantum circuit compilation options set $O = \{A, B, C, D\}$, which is designed to cover the most frequently used and decision-uncertain compilation options in current practices. It should be noted that this set is extensible. This paper instantiates the abstract compilation option set by systematically selecting 30 combinations from two leading quantum computing compilers, Qiskit and TKET, serving as a representative instantiation of this abstract problem. The elements in O correspond, respectively, to qubit technology options set A , quantum device options set B , quantum compiler options set C , and compilation steps and optimization strategy options set D . Each compilation option has multiple corresponding choices, namely, $A = \{a_1, a_2\}$, $B = \{b_1, b_2, b_3, b_4, b_5\}$, $C = \{c_1, c_2\}$, $D = \{d_1, d_2, d_3, d_4, d_5, d_6\}$. Among them, a_1 and a_2 are superconducting qubit technology, and ion trap qubit technology, respectively. b_1, b_2, b_3 , and b_4 are 8-qubits, 27-qubits, 80-qubits, and 127-qubits quantum devices based on superconducting, respectively, and b_5 is an 11-qubits quantum device based on ion trap. c_1 and c_2 are Qiskit quantum compiler, and TKET quantum compiler, respectively. d_1, d_2, d_3 , and d_4 are the optimization levels of Qiskit's O0, O1, O2, and O3, respectively. d_5 and d_6 are the qubit mapping strategies of TKET's Line Placement, and Graph Placement, respectively. So b_1, b_2, b_3 , and b_4 have a conditional constraint (dependency) relationship with a_1 , while b_5 has a conditional constraint relationship with a_2 . Similarly, d_1, d_2, d_3 , and d_4 have a conditional constraint relationship with c_1 ; d_5 and d_6 have a conditional constraint relationship with c_2 . In other words, b_1, b_2, b_3 , and b_4 can be selected only after a_1 is selected; b_5 can be selected only after a_2 is selected; d_1, d_2, d_3 , and d_4 can be selected only after c_1 is selected; d_5 and d_6 can be selected only after c_2 is selected. For a quantum circuit LC compiled by any combination of quantum circuit

compilation options $o = \{a_i, b_j, c_k, d_l\}$ (where $i = 1, 2, j = 1, 2, 3, 4, 5, k = 1, 2, l = 1, 2, 3, 4, 5, 6, o \in O$), it is scored based on the following fidelity formula:

$$eval\ score = \prod_{i=1}^{N_G} gate\ fid(g_i) \prod_{j=1}^{N_Q} readout\ fid(q_j) \quad (3)$$

where $gate\ fid(g_i)$ represents the expected fidelity of gate g_i on its corresponding qubit, and $readout\ fid(q_j)$ represents the expected fidelity of the measurement operation on its corresponding qubit q_j . The fidelity score of the compilation result of any compilation option combination o of the quantum circuit is calculated through Equation (3). In order to obtain the compilation result with the highest fidelity for the quantum circuit, the quantum circuit compilation option selection problem can be defined as: Select a compilation option combination $o(o \in O)$ for the quantum circuit so that it can obtain the highest fidelity score ($eval\ score$) by compilation option combination o . It is noteworthy that fidelity is selected as the sole optimization objective in this work because, within the context of noisy intermediate-scale quantum (NISQ) computing, fidelity serves as the decisive metric for assessing computational reliability, directly determining the success or failure of quantum algorithms. Consequently, maximizing fidelity through Equation (3) most directly addresses the fundamental requirement of identifying the ‘optimal compilation option’. Furthermore, it should be noted that this objective can be readily extended to other optimization targets (such as circuit depth) by simply replacing the training labels.

In order to find the compilation option combination with the highest fidelity score for a specific quantum circuit, we can model the selection problem of quantum circuit compilation options as a classification prediction task. That is, for a quantum circuit LC , we aim to find a combination of compilation options $o(o \in O)$, which can be defined as a label y_i . All compilation options are combined to form a label set $Y = \{y_1, y_2, \dots, y_{30}\}$. Therefore, the prediction task of quantum circuit compilation options is mainly divided into the following two stages: (1) Propose a quantum circuit feature extraction method based on a TWIG. The extracted N_Z -dimensional graph feature vector Z is combined with the N_T -dimensional traditional feature T [11] to form an N_F -dimensional representation vector F , which fully represents the quantum circuit information. (2) Design and train a quantum circuit compilation option prediction model based on DRAN-Net. By mapping the quantum circuit representation vector F to label $Y(F \rightarrow Y)$, the label y corresponding to its highest fidelity compilation option o can be predicted quickly and accurately. The flowchart for the two phases is shown in Figure 8.

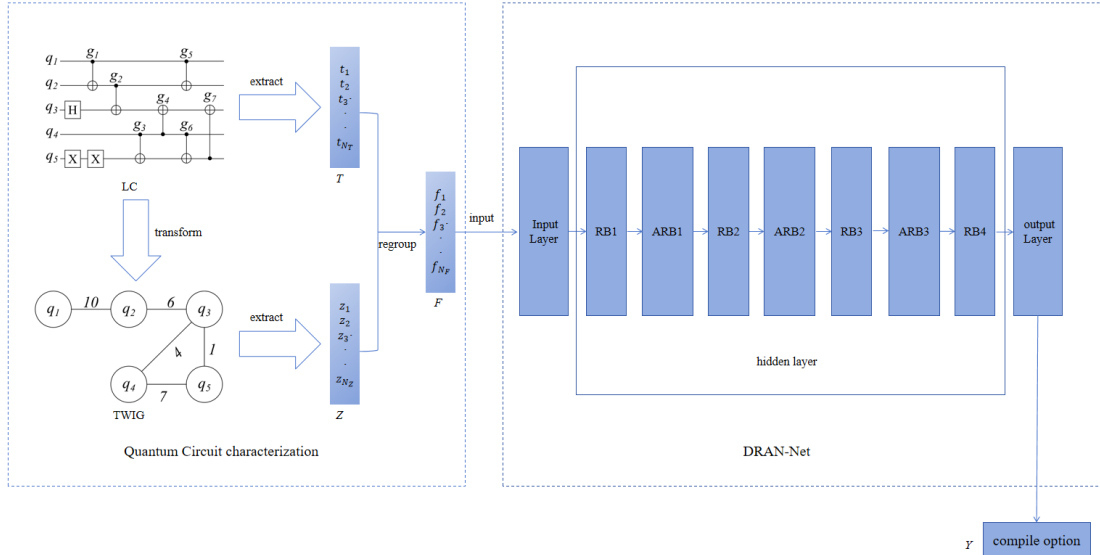


Figure 8. The flowchart for the two phases.

3.2. Feature Extraction of Quantum Circuits

The compilation optimization and mapping of quantum circuits are highly dependent on their structural characteristics. However, existing traditional features (e.g., circuit depth, entanglement ratio) can reflect global statistical information. But they fail to capture the complex, time-evolving interaction patterns between qubits. To address this

limitation, this paper proposes a new feature representation method that combines 31-dimensional traditional features [11] with graph features extracted from the TWIG to more comprehensively characterize quantum circuits.

As shown in Figure 8, a core innovation of this paper lies in introducing TWIG to encode the spatiotemporal structural information of the circuit and extracting graph features with clear physical meanings from it. Finally, we conduct correlation analysis and importance selection on all features. This forms a concise but effective feature vector. The vector serves as input for DRAN-Net, which is another core innovation of this paper. The following sections will first elaborate on the construction principle of TWIG, and then explain the extraction and selection rationale of graph features.

3.2.1. Construction of Time-Weighted Interaction Graph

Starting from the quantum circuit itself, converting the quantum circuit into a graph form allows for the extraction of new quantum circuit features from a different dimension. Therefore, the Time-Weighted Interaction Graph (TWIG) and graph features are introduced. Based on the Interaction Graph (IG), the TWIG incorporates the execution order of two-qubit gates in the circuit by weighting them and adding them to the IG, thereby introducing dependencies and constraints on the execution order between two-qubit gates. This enables the graph to contain more information about the quantum circuit, which is beneficial for the graph features to better represent the quantum circuit. TWIG is designed to convert the execution semantics of quantum circuits (e.g., interaction strength) into a graph structure. The core motivation for its construction lies in modeling interaction intensity: the greater the number of two-qubit gate operations that two qubits jointly participate in, and the ‘earlier’ these operations occur (i.e., the shallower their logical layer), the more important their interaction should be deemed, and the greater its influence on the selection of compilation strategies (e.g., mapping). Notably, multi-qubit gates (i.e., three-qubit gates) are all converted into two-qubit gates during final execution. Since the specific gates generated by this conversion can only be determined once the target device is selected, only two-qubit gates are considered when constructing the TWIG in this paper.

In the above given quantum circuit LC , we consider all qubits $Q = \{q_1, q_2, \dots, q_{N_Q}\}$ and two-qubit gates $TG = \{tg_1, tg_2, \dots, tg_{N_{TG}}\}$ is used to construct the TWIG. Firstly, we establish N_Q one-to-one corresponding graph nodes (i.e., $q_i \rightarrow q_i$) based on qubits Q . Then, to reflect the mutual influence between two-qubit gates that diminishes with logical topological distance—even if the gates are not optimized, their relative execution order can retain local ‘temporal proximity’ signals, which benefits the subsequent selection of mapping devices. To achieve this, a temporal level weighting is applied to each gate within the two-qubit gates. Specifically, the circuit undergoes topological levelization: gates that can be executed in parallel are grouped into the same level. Across different levels, weights decrease with level depth (gates within the same level have equal weights). It is assumed that the subscript of two-qubit gates $TG = \{tg_1, tg_2, \dots, tg_{N_{TG}}\}$ represents its execution level, we assign decreasing weights $\{N_{TG}, N_{TG} - 1, \dots, 1\}$ to each gate sequentially (i.e., $w_{g_1} = N_{TG}, w_{g_2} = N_{TG} - 1, \dots, w_{g_{N_{TG}}} = 1$). Finally, for any gate tg_i in the two-qubit gates TG that acts on two qubits q_i and q_j , we determine an edge e_{ij} and assign the weight of the gate as the weight of the edge (i.e., $w_{ij} = w_{tg}$). If an edge e_{ij} already exists between q_i and q_j , the weight of the quantum gate is added to the edge weight (i.e., $w_{ij} = w_{ij} + w_{tg}$). This process is performed iteratively until all two-qubit gates have been traversed. The final result is the TWIG, in which each node represents a qubit, each edge represents the interaction between two qubits, and the weight of the edge reflects the importance of its interaction. The TWIG is actually an undirected weighted graph, whose properties are described as follows:

- (1) Nodes: Each qubit in the quantum circuit corresponds to a node in the graph.
- (2) Edges: If there are any number of quantum gate operations between two qubits, there are edges between the corresponding two nodes in the graph.
- (3) Weights: All two-qubit gate weights acting between a certain pair of qubits are accumulated to their corresponding edge weights, thus highlighting the importance of the qubit pairs. The weights can represent the number of two-qubit gate operations between qubits and the timing information of execution.

The following is an example of the construction of a TWIG.

Example 2. Taking the quantum circuit shown in Figure 9 as an example, the circuit is executed in sequence from g_1 to g_7 sequentially. Among them, g_1 and g_3, g_4 and g_5 are parallel, so they are grouped into the same level and assigned weights $w_{g_1} = 5, w_{g_2} = 4, w_{g_3} = 5, w_{g_4} = 3, w_{g_5} = 3, w_{g_6} = 2, w_{g_7} = 1$, respectively. Since both g_1 and g_5 act on q_1 and q_2 , we create an edge(q_1, q_2) and assign it a weight value of $w_{12} = 5 + 3 = 8$. Similarly, we create an edge(q_2, q_3) with a weight value of $w_{23} = 4$, an edge(q_4, q_5) with a weight value of $w_{45} = 5 + 2 = 7$, an

edge(q_3, q_5) with a weight value of $w_{35} = 1$, an edge(q_3, q_4) with a weight value of $w_{34} = 3$. Finally, the quantum circuit is transformed into the TWIG shown on the right in Figure 9.

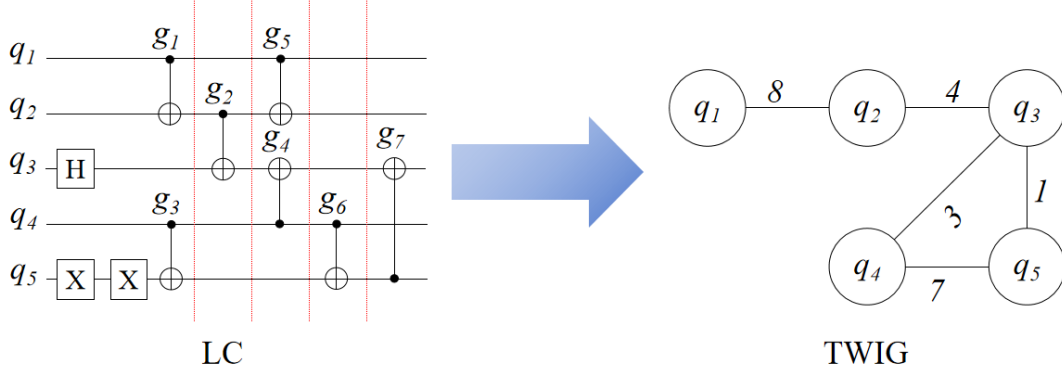


Figure 9. Transformation of TWIG.

3.2.2. Extraction of Graph Features

After completing the construction of the TWIG, we need to further extract feature information from the graph that can fully represent the quantum circuit. There are many features of metric graphs, but not all features of metric graphs are relevant to our goals. After carefully studying Hernandez's classification of graph features [27] and Bandic's analysis of interactive graph features [15], we select some main graph features for extraction. The corresponding definition, meaning and calculation formula of each graph feature are as follows:

(1) Average Weighted Shortest Path (AP): refers to the average weight of the shortest paths between all node pairs in the TWIG. The size of the average weighted shortest path can indicate the average interaction strength and importance between qubits. The calculation formula is as follows:

$$AP = \frac{1}{N_Q(N_Q - 1)} \sum_{i \neq j} d_w(q_i, q_j) \quad (4)$$

where $d_w(q_i, q_j)$ represents the shortest weighted path length between node q_i and node q_j , and N_Q is the number of nodes.

(2) Max Degree (MX_d): refers to the maximum sum of connection weights of nodes in the graph. The maximum degree reveals the strength of the most important qubit interactions in a quantum circuit. The calculation formula is as follows:

$$MX_d = \max_{i \in N_Q} (deg(q_i)) \quad (5)$$

where $deg(q_i)$ is the weighted degree with node q_i , and $\max()$ is a function to find the maximum value.

(3) Min Degree (MI_d): refers to the minimum sum of connection weights of nodes in the graph. The minimum degree can reveal whether there are non-interacting qubits in a quantum circuit. The calculation formula is as follows:

$$MI_d = \min_{i \in N_Q} (deg(q_i)) \quad (6)$$

Its calculation method is similar to the maximum degree, but it takes the minimum value, and $\min()$ is a function to find the minimum value.

(4) Average Clustering Coefficient (AC): refers to the average value of the clustering coefficients of all nodes in a graph. A high clustering coefficient indicates close interactions between qubits and helps to understand the density of global interactions in quantum circuits. The calculation formula is as follows:

$$AC = \frac{1}{N_Q} \sum_{i \in N_Q} C(q_i); C(q_i) = \sum_{i \in N_Q} \frac{2 \sum_{\exists \Delta q_i q_j q_k} (w_{ij} w_{jk} w_{ki})^{\frac{1}{3}}}{eg(q_i)(eg(q_i) - 1) MX_d} \quad (7)$$

where $C(q_i)$ is the weighted clustering coefficient of node q_i based on the strength of triangle subgraphs [28], $eg(q_i)$ is the number of edges directly connected with node q_i , and $\triangle$ is the triangle symbol.

(5) Standard Deviation of Adjacency Matrix (SD): refers to the degree of dispersion in the distribution of connection strengths between nodes in the graph. A high standard deviation indicates diverse types of interactions between qubits and an uneven connection structure among qubits. The calculation formula is as follows:

$$SD = \sqrt{\frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (A_{ij} - \mu_A)^2}; \mu_A = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n A_{ij} \quad (8)$$

where A is the adjacency matrix of the graph, A_{ij} is an element of the adjacency matrix, and μ_A is the average value of all matrix elements.

By extracting these features to form the $N_Z = 5$ -dimensional feature vector $Z = [AP, MX_d, MI_d, AC, SD]$, we can better understand the structural properties of quantum circuits and optimize the representation. These features play an important role in the performance analysis of quantum circuits and the prediction of compilation options. Here is an example of extracting features.

Example 3. Taking the TWIG in Figure 9 as an example, its corresponding graph features can be calculated as $AP = 7.064$, $MX_d = 12$, $MI_d = 7.66$, $AC = 0.1169331126$, $SD = 2.86557$. The resulting $N_Z = 5$ -dimensional feature vector $Z = [7.064, 12, 7.66, 0.1169331126, 2.86557]$. Then, it is combined with the $N_T = 31$ -dimensional traditional feature $T = [u3, u2, u1, cx, u, p, x, h, t, tdg, rx, ry, rz, cz, swap, ccx, cswap, cry, cu1, cp, cu3, cu, rzz, rccx, num_qubits, depth, prog_comm, crit_depth, entang_ratio, parallelism, liveness]$ to form an $N_F = 36$ -dimensional representation vector F , which is used to represent the quantum circuit.

3.3. Prediction Based on DRAN-Net Model

After extracting the feature combination from the quantum circuit into 36-dimensional representation vector $F = [f_1, f_2, \dots, f_{36}]$, a prediction model is needed to map the quantum circuit representation vector F to the label set $Y = \{y_1, y_2, \dots, y_{30}\} (F \rightarrow Y)$ for prediction. To this end, this paper designs a prediction model DRAN-Net. DRAN-Net integrates attention mechanism into a deep residual neural network, enabling the model to gain a deeper understanding of the features of quantum circuits and thus provide more accurate prediction results. The design details of the prediction model are introduced in detail below.

3.3.1. DRAN-Net Model Design

The DRAN-Net network model is divided into three parts: input layer, hidden layer, and output layer. The model structure is shown in Figure 10. Each of the three parts is described below:

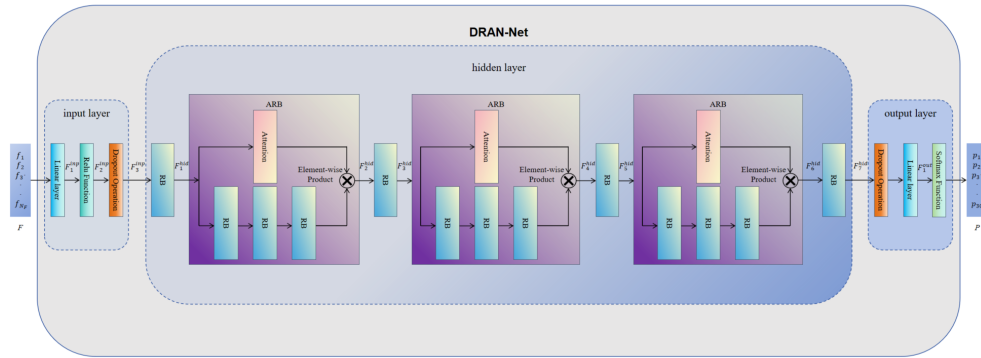


Figure 10. DRAN-Net.

(1) Input layer. The input layer is the part that receives the representation vector $F = [f_1, f_2, \dots, f_{36}]$ of the input quantum circuit and processes it into a high-dimensional vector. As shown in Figure 10, it first uses a linear layer to process the input representation vector, and maps the representation vector F from 36 dimensions to 256 dimensions F_1^{inp} for more complex processing. Because the relationships between features in the representation vector are complex and nonlinear, high dimensions help to capture these complex relationships. Later, using the ReLU

(Rectified Linear Unit) activation function that helps the model learn nonlinear features and Dropout that randomly sets some activation values to zero. The final output is a 256-dimensional high-dimensional feature vector F_3^{inp} .

Among them, the Liner Layer, also known as the fully connected Layer (FC Layer) or Dense layer, is a basic block in the neural network architecture. It is primarily used for linearly transforming input feature vector data to generate new feature representations. This transformation is realized through the learned weight matrix and bias term. As shown in the input layer in Figure 10, for input vector F , its mathematical expression is:

$$F_1^{inp} = W^{inp}F + b^{inp} \quad (9)$$

where W^{inp} is the weight matrix of the linear layer of the input layer, and b^{inp} is the bias term of the linear layer of the input layer.

(2) Hidden layer. The hidden layer is the part of the network model that further learns deeper features from the high-dimensional feature vector output by the input layer. It is also the most important part of the network model learning feature data. As shown in Figure 10, the hidden layer adopts a combination of 4 residual blocks (RBs) and 3 attention residual blocks (ARBs) stacked alternately in sequence. Firstly, the 256-dimensional feature vector F_3^{inp} is initially learned through the residual block (RB) to obtain F_1^{hid} , and then its output is used as the input of the attention residual block (ARB) for deeper learning to obtain F_2^{hid} . By alternating input and output through feature mapping, the 256-dimensional high-level feature F_7^{hid} is finally extracted.

Among them, the Residual Block (RB) makes the deep network easier to train and improves the performance of the model by introducing skip connections. The RB can prevent the loss of features in the network model, and a local residual network can be formed by stacking multiple RBs. It can solve the problems of gradient vanishing and gradient explosion in deep network training. Figure 11 illustrates the structure of the RB. To more effectively learn global feature information over a period of time, the input feature map $F_3^{inp} \in R^{1 \times 256}$ is first processed through a trunk branch consisting of a linear layer and a ReLU activation function. Then the skip connection branch transfers the original input F_3^{inp} to the trunk branch for Element-wise Product operation. Finally, the ReLU activation function is applied to introduce nonlinearity and output:

$$F_1^{hid} = S^{rb_1}(F_3^{inp}) = Relu((W_2^{rb_1} Relu(W_1^{rb_1} F_3^{inp} + b_1^{rb_1}) + b_2^{rb_1}) + F_3^{inp}) \quad (10)$$

where F_1^{hid} is the output of the first RB of the hidden layer, $S^{rb_1}()$ is its corresponding residual function. $W_1^{rb_1}$ and $W_2^{rb_1}$ are the weight matrices of the first and second linear layers in the first RB, respectively, and $b_1^{rb_1}$ and $b_2^{rb_1}$ are their corresponding bias terms, respectively.

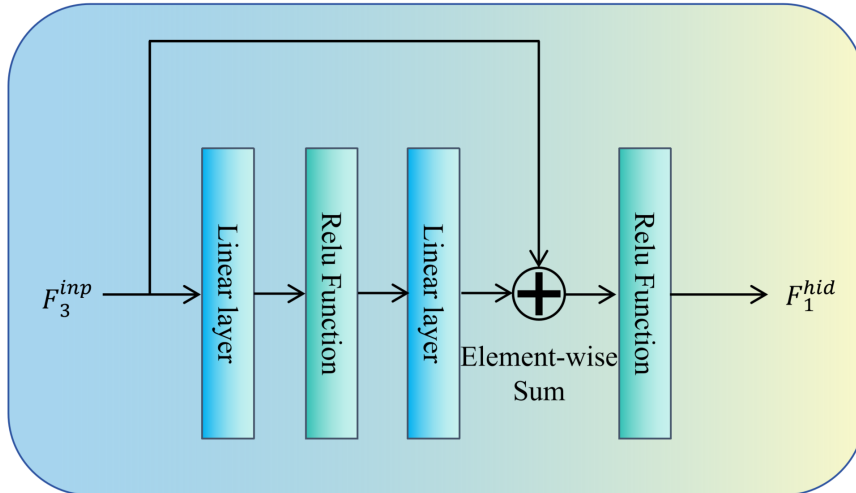


Figure 11. Residual block (RB).

Attention Residual Block (ARB) can make the deep prediction model pay more attention to important regions, concentrating on learning crucial features while weakening unimportant ones. As shown in Figure 12, the structural diagram of the ARB, it mainly consists of one attention block and three RBs. Three stacked RBs serve as the trunk branch, while the attention block acts as the mask branch. The output feature map F_1^{hid} of the previous RB is input

to the trunk branch and the mask branch at the same time. Through the trunk branch, more comprehensive feature information can be learned using the RBs, resulting in $S_3^{arb_1}(S_2^{arb_1}(S_1^{arb_1}(F_1^{hid})))$. Through the mask branch, a feature weight value $A^{arb_1}(F_1^{hid})$ is learned. Then, after adding 1 to each element of the feature weight value $A^{arb_1}(F_1^{hid})$, an Element-wise Product operation is performed with the output of the three RBs. Finally, the output F_2^{hid} of the ARB is:

$$F_2^{hid} = H(F_1^{hid}) = S_3^{arb_1}(S_2^{arb_1}(S_1^{arb_1}(F_1^{hid}))) \times (1 + A^{arb_1}(F_1^{hid})) \quad (11)$$

where $S_1^{arb_1}()$, $S_2^{arb_1}()$, and $S_3^{arb_1}()$ are the first, second, and third RB in the first ARB, respectively. Each element of $A^{arb_1}(F_1^{hid})$ has a value range of [0,1]. By applying $1 + A^{arb_1}(F_1^{hid})$, each element is assigned a weight within the range of [1,2], which makes the feature data become clearer as the depth increases. This can serve as a feature selector to enhance good features and suppress noise from the trunk branch features.

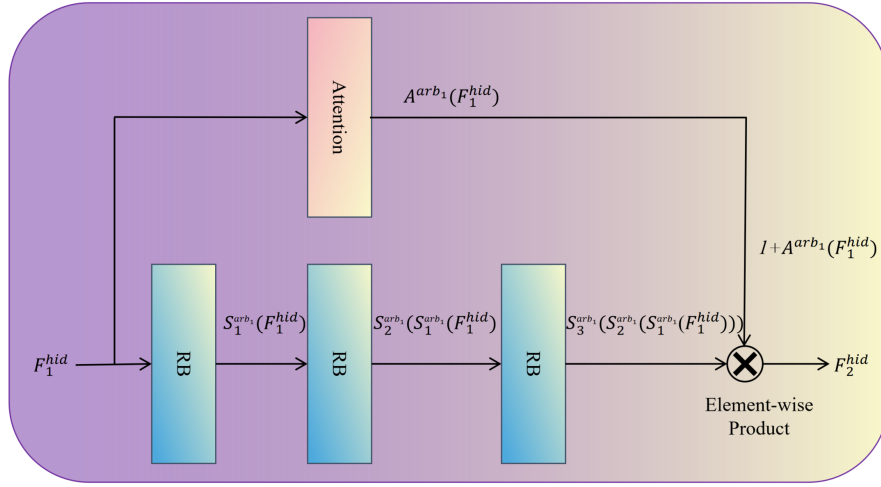


Figure 12. Attention residual block (ARB).

Attention block (Attention) can adaptively adjust the importance of various features, so that more important features receive higher weights in subsequent computations. As shown in Figure 13, which illustrates the structure of the attention block. In order to explore the location of important target features in the input feature map, the input feature map F_1^{hid} is simultaneously fed into three branches. Each branch first undergoes a linear transformation of the features through a linear layer, then introduces nonlinearity through the ReLU activation function to obtain preliminary feature relationships. Subsequently, a second linear layer further linearly transforms these features to obtain more expressive feature relationships. Immediately after, the elements from the three layers are multiplied together to obtain F^{a_1} , which assigns a score to each element. Finally, the Sigmoid activation function generates attention weights. This process assigns a larger weight to important features within the feature relationships, and the weight output is:

$$A^{arb_1}(F_1^{hid}) = Sigmoid(\prod_{i=1}^3 W_2^{a_1(i)} Relu(W_1^{a_1(i)}(F_1^{hid}) + b_1^{a_1(i)}) + b_2^{a_1(i)}) \quad (12)$$

Similarly, $W_1^{a_1(i)}$ and $W_2^{a_1(i)}$ are the weight matrices of the first and second linear layers of the i -th branch of the first attention block, respectively, $b_1^{a_1(i)}$ and $b_2^{a_1(i)}$ are their corresponding bias terms, respectively. For the input $F^{a_1} = [f_1^{(3)}, f_2^{(3)}, \dots, f_{256}^{(3)}]$, the mathematical expression of the Sigmoid activation function is:

$$A^{arb_1}(F_1^{hid}) = Sigmoid(F^{a_1}) = [\frac{1}{1 + e^{-f_1^{(3)}}}, \frac{1}{1 + e^{-f_2^{(3)}}}, \dots, \frac{1}{1 + e^{-f_{256}^{(3)}}}] \quad (13)$$

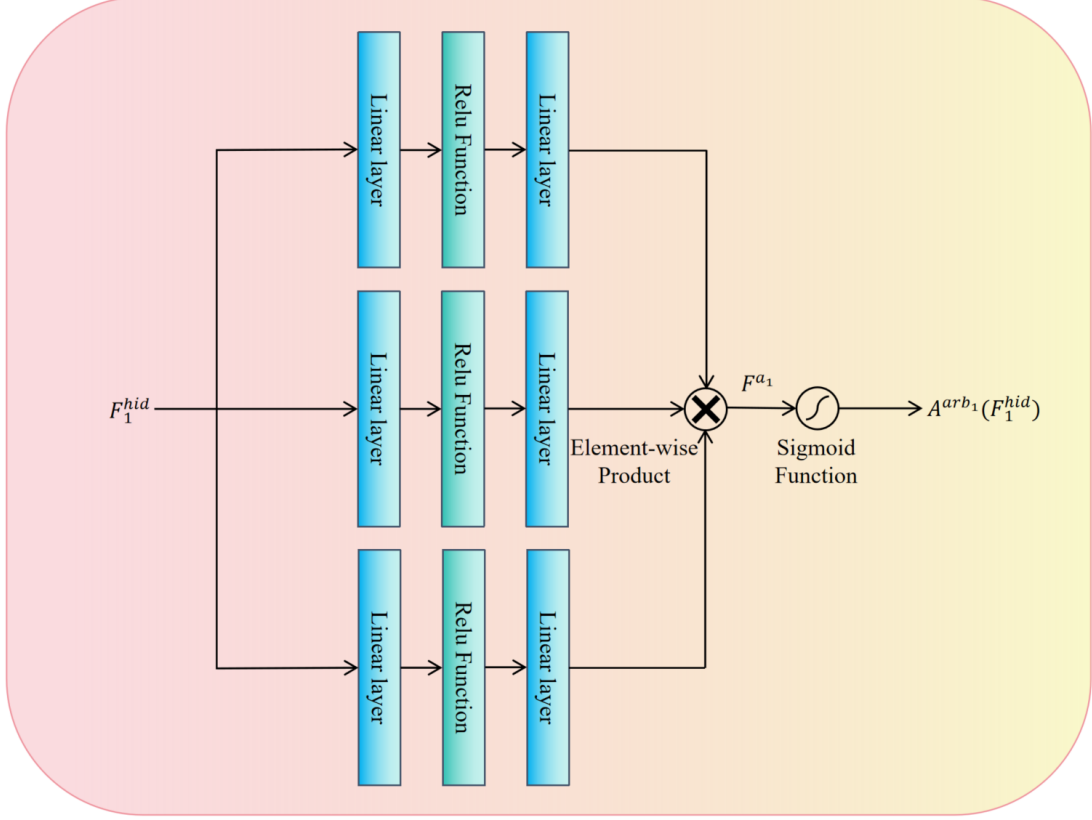


Figure 13. Attention block.

(3) Output layer. The output layer is the part that performs final aggregation processing on the 256-dimensional high-level mapping features F_7^{hid} outputted by the hidden layer. As shown in Figure 10, it first applies a Dropout to the high-level mapped features to prevent overfitting. Finally, a linear layer is used to map the dimensions to a 30-dimensional $F_1^{out} = [f_1^{(4)}, f_2^{(4)}, \dots, f_{30}^{(4)}]$ that matches the number of label categories, and the Softmax activation function is used to calculate the probability distribution $P = [p_1, p_2, \dots, p_{30}]$ corresponding to the prediction results of Y . The mathematical expression corresponding to the Softmax activation function is:

$$P = [p_1, p_2, \dots, p_{30}] = \text{Softmax}(F_1^{out}) = \left[\frac{e^{f_1^{(4)}}}{\sum_{i=1}^{30} e^{f_i^{(4)}}}, \frac{e^{f_2^{(4)}}}{\sum_{i=1}^{30} e^{f_i^{(4)}}}, \dots, \frac{e^{f_{30}^{(4)}}}{\sum_{i=1}^{30} e^{f_i^{(4)}}} \right] \quad (14)$$

The label with the highest probability $p_i (p_i \in P)$ serves as the predicted label $y_i (y_i \in Y)$ for the quantum circuit. Each label corresponds to one compilation option combination in Figure 5. All compilation option combinations form a label set $Y = \{y_1, y_2, \dots, y_{30}\}$. The core task of this paper is precisely to find the compilation option combination with the highest fidelity score for a given quantum circuit. Thus, the probabilities of these classification prediction results are used to calculate the cross-entropy loss, which serves to evaluate the model's performance.

3.3.2. Model Training

In this study, in order to train the proposed DRAN-Net, we adopt a series of standardized steps to ensure the model's effectiveness and generalization ability to different quantum circuits. The following is a detailed description of the model training process.

(1) Data preprocessing. To accelerate convergence during model training and enhance model performance, we preprocess all quantum circuit representation vectors by standardizing them before training. This step can expedite the convergence of the gradient descent method, improve model training efficiency, and consequently better enhance model performance. The standardization formula is as follows:

$$F = \left[\frac{F^1 - \mu^1}{\sigma^1}, \frac{F^2 - \mu^2}{\sigma^2}, \dots, \frac{F^{36} - \mu^{36}}{\sigma^{36}} \right] \quad (15)$$

where F^i is a vector composed of all feature values of the i -th feature in all quantum circuit representation vectors, and μ^i and σ^i are the mean and standard deviation of the corresponding feature, respectively. Through this formula, the mean of each feature is transformed to 0, and the standard deviation to 1. This process ensures that features of varying scales are converted to a common scale, thereby preventing any single feature with a larger scale from disproportionately influencing the model training process.

(2) Loss function. In forward propagation, the model uses the CrossEntropyLoss function to evaluate the discrepancy between the predicted results and the actual labels, thereby optimizing the model's parameters. The formula is as follows:

$$L^* = - \frac{\sum_{i=1}^{N_{bs}} \sum_{j=1}^{30} y_i \log p_{ij}}{N_{bs}} \quad (16)$$

where N_{bs} is the number of samples (batch_size), y_i is the real category label corresponding to the i -th sample, and p_{ij} is the predicted probability by the model that the i -th sample belongs to category j .

To prevent model overfitting and ensure that the model's parameters do not become excessively large, we introduce L1 and L2 regularization during the training process. This adds constraints on the model's parameters, avoiding them from becoming too large or overly complex, thereby preventing overfitting. L1 regularization achieves feature selection by adding the sum of the absolute values of the weights to the loss function, with the formula as follows:

$$L_1 = \lambda_1 \sum_{j=1}^{N_W} |w_j| \quad (17)$$

L2 regularization reduces the values of the weights by adding the sum of the squares of the weights to the loss function, with the formula as follows:

$$L_2 = \lambda_2 \sum_{j=1}^{N_W} w_j^2 \quad (18)$$

where λ_1 and λ_2 are hyperparameters representing the strength of the regularization, N_W is the total number of model weight parameters, $w_j (w_j \in W)$ is any parameter in any weight matrix. Therefore, the final loss function L of the model is:

$$L = L^* + L_1 + L_2 \quad (19)$$

4. Experimental

This section first elaborates on the experimental dataset and evaluation metrics, then gives detailed information on the parameter settings of the proposed model method, and finally explains and analyzes the experimental results of feature correlation and importance experiments, ablation experiments, and advancedness comparison experiments.

4.1. Dataset and Evaluation Metrics

The dataset used in this paper contains 3,003 quantum circuits, all sourced from the MQT Bench benchmark library (Version 0.2.2) [11]. We included all circuits in it that meet the following criteria: belonging to the 'target-independent' level, with the number of qubits ranging from 2 to 127, and the number of quantum gates ranging from 2 to approximately 14,000. This ensures the dataset has sufficient breadth and diversity. Since these circuits involve two qubit technologies (i.e., superconducting and ion trap), five devices (i.e., four superconducting-based devices with 8, 27, 80, and 127 qubits, and one ion trap-based device with 11 qubits), two compilers (i.e., Qiskit and TKET), and six compilation steps and optimization strategies (i.e., optimization levels O0 to O3 in Qiskit, and Line Placement and Graph Placement mapping strategies in TKET), the dataset is divided into 30 label categories.

To systematically evaluate the effectiveness of different feature representation methods, we define three input representation sets based on the 3003 quantum circuits:

(1) T-Pre (Traditional Representation Set) is based on the 31-dimensional traditional features from Reference [11], mainly used to verify the effectiveness of new features. It specifically includes counts of 24 types of gate operations described in accordance with the Open QASM 2.0 specification [29] (e.g., u3, u2, u1, cx, u, p, x, h, t, tdg, rx, ry, rz, cz, swap, ccx, cswap, cry, cu1, cp, cu3, cu, rzz, and rccx) and 7-dimensional global circuit attributes (num_qubits, depth, prog_comm, crit_dept, entang_ratio, parallelism, and liveness). This set only characterizes the statistical properties and macro-features of the circuit, without involving topological interactions between qubits; as a baseline input, its core value lies in isolating the variable of 'topological information' through comparison with new representation sets, thereby ensuring the interpretability of experimental results.

(2) N-Pre (Proposed Novel Representation Set) is the innovative hybrid representation set proposed in this paper. It serves as the default input for DRAN-Net. It has 36 dimensions in total. It is formed by fusing two parts: the 31-dimensional traditional features from T-Pre, and 5 new graph features extracted from the TWIG. Its design aims to fix T-Pre's weakness (i.e., T-Pre lacks topological information). Traditional features reflect the circuit's macro-attributes. Graph features describe how qubits interact. Together, they complement each other. This enables full-dimensional representation of quantum circuits. As a baseline input, its main purpose is twofold. It improves the model's prediction accuracy. It also verifies that feature synergy works in the new representation method.

(3) TWIG-Pre (Graph-Based Representation Set) is a graph-representation representation set for quantum circuits. It is constructed as described in Section 3.2.1. It converts the structured interactions of quantum circuits into computable structured features. These features support input for Graph Neural Networks (GNNs). Specifically, it first extracts four types of core node-level features from the TWIG. Node degree characterizes the frequency of qubit interactions; weighted node degree distinguishes the influence of gate operations; clustering coefficient reflects local topological compactness; and sum of gate counts measures the operation load of qubits. Then it applies per-graph zero-mean normalization. This step eliminates scale bias across samples. Finally, it adapts the features into a GNN-compatible data format. At the same time, it retains graph-level attributes. As a baseline input, it has two core purposes. One is to add baseline experiments. The other is to verify that T-Pre's hybrid representation method is superior to pure graph-based representation methods.

To comprehensively evaluate the model's prediction performance, this paper adopts the accuracy and Top-3 accuracy used in Reference [11] as the primary assessment metrics. Accuracy directly reflects a model's overall ability to classify all samples correctly. It serves as a fundamental metric for evaluating classification model performance. Top-3 Accuracy relaxes the prediction requirements: as long as the true label falls within the top three categories with the highest predicted probabilities by the model, the prediction is considered correct. This metric is more suitable for application scenarios with a large number of categories or fine distinctions between categories, and can assess the model's recognition ability in the sense of 'approximately correct'. In addition, to comprehensively evaluate the model's performance on class-imbalanced datasets, this paper also introduces the Weighted-Averaging F1-score. This metric integrates Precision and Recall, and assigns each category a weight proportional to its sample proportion in the dataset. This effectively alleviates the impact of uneven sample sizes across categories on evaluation results. Thus, the Weighted-Averaging F1-score can more fairly reflect the model's classification robustness across different categories, in particular those with small sample sizes, and acts as a valuable supplement to the aforementioned accuracy metrics. The formulas for calculating these metrics are as follows:

$$Accuracy = \frac{\sum_{i=1}^{N_{bs}} 1(\hat{y}_i = y_i)}{N_{bs}}; \hat{y}_i = \arg \max_j(p_{ij}) \quad (20)$$

where $\hat{y}_i$ is the prediction label corresponding to the i -th sample, and $\arg \max_j()$ can get the category parameter j for which the maximum value is.

$$Top-3 = \frac{\sum_{i=1}^{N_{bs}} 1(y_i \in \{\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \hat{y}_i^{(3)}\})}{N_{bs}} \quad (21)$$

where $\hat{y}_i^{(1)}$, $\hat{y}_i^{(2)}$ and $\hat{y}_i^{(3)}$ are the category labels corresponding to the top three predicted probabilities for the i -th sample.

$$F1 = \sum_{c=1}^{30} W_c \frac{2Precision_c \times Recall_c}{Precision_c + Recall_c}; Precision_c = \frac{TP_c}{TP_c + FP_c}; Recall_c = \frac{TP_c}{TP_c + FN_c} \quad (22)$$

where W_c is the weight of category c (the proportion of samples in that class within the dataset), $Precision_c$ and $Recall_c$ are precision and recall rates of category c , respectively, and TP_c , FP_c , and FN_c denote the true positives, false positives, and false negatives of category c , respectively.

4.2. Model Parameter Settings

In the quantum circuit compilation option prediction experiment, the hyperparameter settings required for training the model are as follows: the training and test sets are divided in an 80%:20% ratio, the batch size is set to 128, the Adam algorithm is used to optimize the loss function with an initial learning rate of 0.001. Weight decay is also added to the optimization function, which is a regularization method that can further prevent overfitting, and the weight decay parameter is set to 0.0001. Both L_1 and L_2 , which are hyperparameters representing the strength of regularization,

are set to 0.00005. For detailed settings of other parameters, refer to Table 1. The model in this paper is implemented using the PyTorch deep learning framework and programmed in Python 3.9.1. The version of PyTorch used is 1.7.1, the version of Qiskit is 0.39.3, the operating system is Ubuntu 20.04 LTS, the CPU is an Intel(R) Xeon(R) Silver 4114 @ 2.20GHz, and the GPU is a Tesla V100 graphics card with 32GB of memory.

Table 1. DRAN-Net parameter.

Layer name	Parameters
Liner layer	36×256
Dropout	0.01
Residual Block1	256×256
Attention Residual Block1	256×256
Residual Block2	256×256
Attention Residual Block2	256×256
Residual Block3	256×256
Attention Residual Block3	256×256
Residual Block4	256×256
Dropout	0.01
Liner layer	256×30

4.3. Experimental Results

To verify the effectiveness and advancement of the proposed method, we design three sets of experiments, namely, feature correlation and importance experiment, ablation experiment, and advancedness comparison experiment. The feature correlation and importance experiments aims to validate the effectiveness and robustness of the newly proposed features. The ablation experiment is conducted to separately verify the effectiveness of the proposed representation method and model method. The advancedness comparison experiment is intended to validate the overall advancement of the proposed method. The following sections provide detailed analyses of the different experimental results.

4.3.1. Feature Correlation and Importance Experiment

To build an efficient and interpretable representation set, we followed a systematic, data-driven feature engineering process. This subsection aims to achieve three goals: (1) verify the effectiveness of the newly proposed graph features; (2) identify and eliminate excessive redundancy between features; and (3) finally provide quantitative evidence for the effectiveness of our hybrid representation method.

Correlations between all features are unavoidable [27]. Multicollinearity (high correlation) between features not only increases model complexity and the risk of overfitting, but also indicates redundancy in the representation set. We calculated the Spearman’s rank correlation coefficients [30] between all 36 initial features (including 31-dimensional traditional features and 5 new graph features). We also plotted a heatmap of the correlation matrix, as shown in Figure 14 (-1/1 indicates perfect correlation, and 0 indicates no correlation).

Analysis of feature correlations reveals that many feature pairs are correlated, with some showing perfect correlation—meaning their metric values are identical. For example, the count of t gates is perfectly correlated with that of tdg gates, while other features also exhibit some degree of correlation. Notably, the correlation between the proposed new features and other features does not exceed 95%. This indicates that the new features successfully capture novel structural information not described by traditional statistical features, statistically validating their unique value as an effective supplement. Based on this analysis, we established strict screening rules: for any highly correlated feature pair (with similarity $\geq 95\%$), only one is retained. We prioritize retaining features that are more commonly used in domain knowledge or have clearer physical meanings (e.g., retaining the count of t gates while removing the perfectly correlated count of tdg gates). Through this step, we removed 3 redundant traditional features (tdg, parallelism, rzz), reducing the feature dimensionality from 36 to 33.

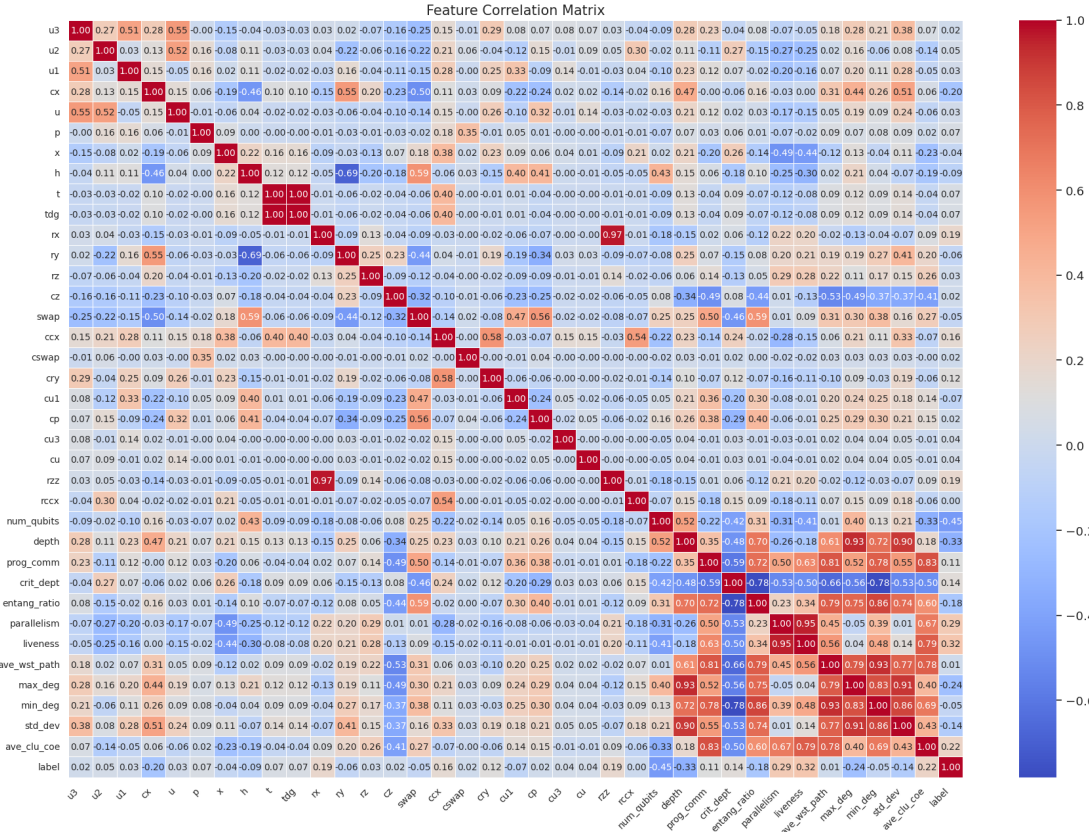


Figure 14. Feature correlation.

To further quantify the actual contribution of each feature to model predictions, this paper employs the permutation feature importance method. The trained DRAN-Net model is used to generate a feature importance ranking chart based on the permutation feature importance in descending order, as shown in Figure 15. The results indicate significant differences in the influence of various features. Among them, the number of qubits has an importance greater than 0.6 in the model's predictions, holding the highest significance. This strongly demonstrates the successful training of the model, as a quantum circuit cannot be executed if the device's number of qubits is less than what the circuit requires. Among the new features proposed in this paper, the Average Clustering Coefficient has an importance greater than 0.3 in the model's predictions, which convincingly proves the effectiveness of the proposed features. Moreover, features with a permutation importance of zero do not necessarily imply they are useless. Because redundancy among features and the model's independence can result in a permutation importance of zero. Through this analysis, not only is the model's decision-making basis revealed, but it also provides a clearer understanding of the role of each feature in the prediction task, further validating the effectiveness of the proposed new features.

4.3.2. Ablation Experiment

To deeply verify the effectiveness of the proposed method, we design a series of ablation experiments to evaluate their impact on the overall performance. In the experimental results section, we detail the model performance metrics for each experimental configuration, including accuracy, Top-3 accuracy, and F1. All experiments are conducted under the same evaluation criteria, and results are reported as the mean $\pm$ standard deviation from five independent runs.

First, we apply the proposed quantum circuit representation method to part of traditional models for experimentation and analyze its performance results. The purpose of this step is to observe the impact of the proposed quantum circuit representation method on prediction performance, as well as its effectiveness and robustness, and obtain the specified evaluation results as shown in Table 2.

Table 2. Prediction results between the new representation and the traditional model combination.

Model+Representation Set	Accuracy(%)	Top-3(%)	F1(%)
Random Forest+T-Pre	77.39 $\pm$ 0.05	95.54 $\pm$ 0.12	80.75 $\pm$ 0.12
Random Forest+N-Pre	78.88 $\pm$ 0.07	95.87 $\pm$ 0.16	81.90 $\pm$ 0.14
Gradient Boosting+T-Pre	76.63 $\pm$ 0.21	94.58 $\pm$ 0.17	80.47 $\pm$ 0.76
Gradient Boosting+N-Pre	77.31 $\pm$ 0.05	95.24 $\pm$ 0.61	79.95 $\pm$ 0.27
Decision Tree+T-Pre	72.93 $\pm$ 0.10	95.17 $\pm$ 0.00	78.52 $\pm$ 0.20
Decision Tree+N-Pre	73.47 $\pm$ 0.34	95.17 $\pm$ 0.64	74.35 $\pm$ 1.15
Nearest Neighbor+T-Pre	72.48 $\pm$ 0.00	93.68 $\pm$ 0.00	77.13 $\pm$ 0.00
Nearest Neighbor+N-Pre	73.73 $\pm$ 0.00	93.34 $\pm$ 0.00	78.56 $\pm$ 0.00

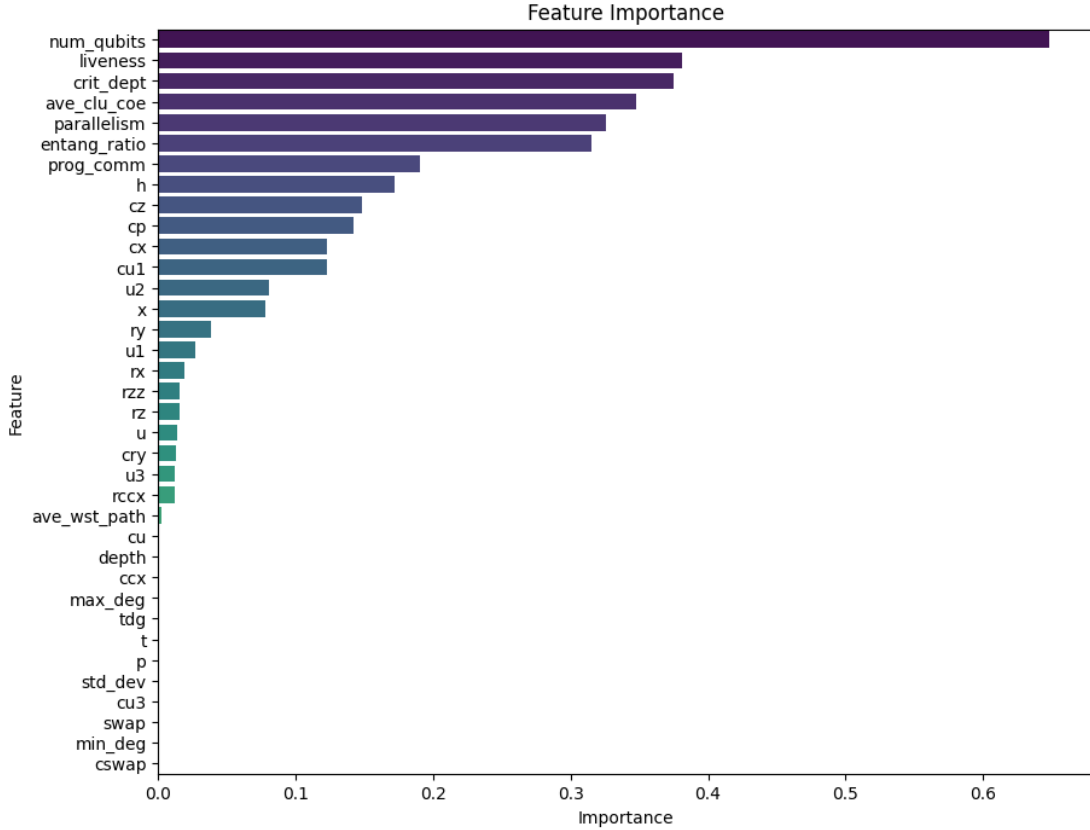


Figure 15. Feature importance.

Figure 16 shows that the proposed new quantum circuit representation method (N-Pre) has improved the accuracy, Top3 and F1 of Random Forest, which performs best among traditional models. Compared with the traditional representation method (T-Pre) of quantum circuits, the proposed quantum circuit representation method improves the Random Forest accuracy by 1.49%, the Top-3 accuracy by 0.33%, and the F1 by 1.15%, proving the effectiveness of the proposed quantum circuit representation method. However, since the N-Pre method introduces more features than the T-Pre method, and the Gradient Boosting model, Decision Tree model, and Nearest Neighbor model are sensitive to feature scales, the computational complexity of these three models increases. This leads to a decrease in the F1 of Gradient Boosting, the F1 of Decision Tree, and the Top-3 of Nearest Neighbor. In addition, the N-Pre method improves the performance of other models, which demonstrates the robustness of the proposed method. In summary, although TWIG features demonstrate certain correlations with conventional features (Figure 14), ablation studies conclusively verify their ability to provide incremental predictive information. This indicates that the correlations primarily stem from these features describing the same underlying circuit complexity from different perspectives, while the unique temporal and structural dynamic information captured by TWIG features serves as the key driver for performance enhancement. Thus, the value of these novel features lies in providing complementary perspectives for a more comprehensive understanding of quantum circuits.

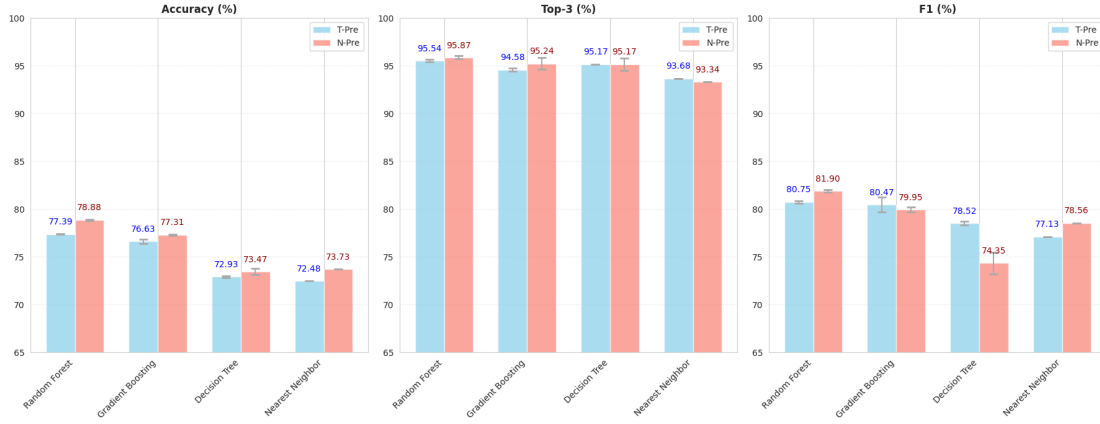


Figure 16. Performance Comparison of T-Pre and N-Pre.

Notably, in the permutation importance analysis, only the new feature Average Clustering Coefficient demonstrated high importance, while the other four TWIG features ranked lower. To investigate the actual contribution of these features, we conducted an ablation experiment: we removed these four lower-importance features to form a combination of traditional features with the single Average_Clustering_Coefficient (T-Pre+AC). This feature set was subsequently evaluated using traditional models, with the designated assessment results presented in Table 3.

Table 3. Prediction results after removal of four lower-importance features.

Model+Representation Set	Accuracy(%)	Top-3(%)	F1(%)
Random Forest+T-Pre+AC	78.13 $\pm$ 0.08	95.84 $\pm$ 0.11	80.93 $\pm$ 0.22
Random Forest+N-Pre	78.88 $\pm$ 0.07	95.87 $\pm$ 0.16	81.90 $\pm$ 0.14
Gradient Boosting+T-Pre+AC	76.84 $\pm$ 0.19	94.91 $\pm$ 0.17	79.82 $\pm$ 0.20
Gradient Boosting+N-Pre	77.31 $\pm$ 0.05	95.24 $\pm$ 0.61	79.95 $\pm$ 0.27
Decision Tree+T-Pre+AC	73.05 $\pm$ 0.12	94.64 $\pm$ 0.22	76.14 $\pm$ 1.46
Decision Tree+N-Pre	73.47 $\pm$ 0.34	95.17 $\pm$ 0.64	74.35 $\pm$ 1.15
Nearest Neighbor+T-Pre+AC	72.48 $\pm$ 0.00	93.68 $\pm$ 0.00	77.13 $\pm$ 0.00
Nearest Neighbor+N-Pre	73.73 $\pm$ 0.00	93.34 $\pm$ 0.00	78.56 $\pm$ 0.00

The experimental results in Figure 17 demonstrate that models using the condensed representation set (T-Pre+AC) achieve comprehensively lower accuracy than those utilizing the complete representation set (N-Pre). Apart from fluctuations observed in Decision Tree and Nearest Neighbor models due to their sensitivity to feature scaling, the same pattern holds for the other two metrics. This finding indicates that although these features individually exhibit limited impact on model output when perturbed in isolation, they collectively contribute complementary information that enhances the model's overall representational capacity and generalization performance when integrated into the complete representation set. Consequently, we ultimately retained all TWIG features to achieve optimal performance.

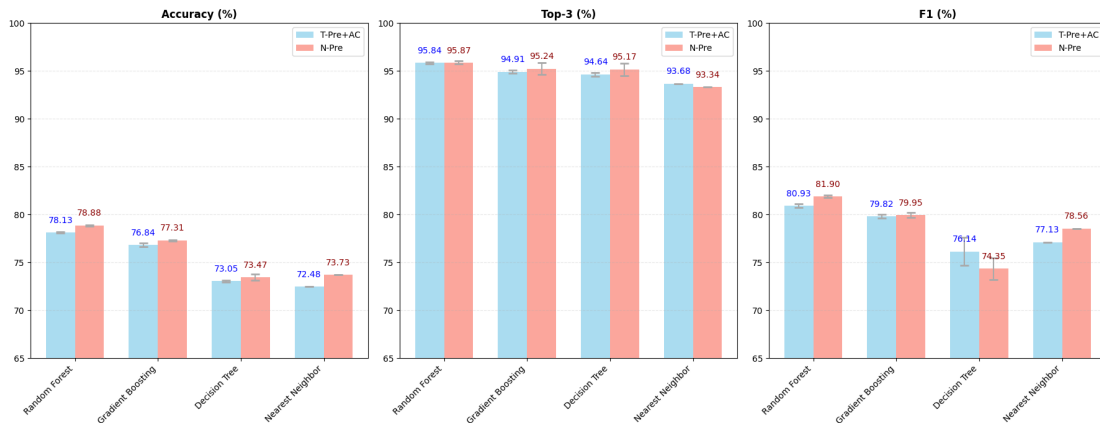


Figure 17. Performance Comparison of T-Pre+AC and N-Pre.

Next, we apply the proposed model DRAN-Net to traditional quantum circuit representation method and analyze its result performance. The core of this experimental step is to evaluate the effectiveness and robustness of DRAN-Net with traditional circuit representations. The specified evaluation results are shown in Table 4. By comparing and analyzing the performance differences between traditional models and DRAN-Net, we can gain insights into the advantages of DRAN-Net.

Table 4. Prediction results between the new model and the traditional representation combination.

Model+Representation Set	Accuracy (%)	Top-3 (%)	F1 (%)
DRAN-Net+T-Pre	82.32 ± 1.16	97.87 ± 0.48	82.18 ± 1.08
Random Forest+T-Pre	77.39 ± 0.05	95.54 ± 0.12	80.75 ± 0.12
Gradient Boosting+T-Pre	76.63 ± 0.21	94.58 ± 0.17	80.47 ± 0.76
Decision Tree+T-Pre	72.93 ± 0.10	95.17 ± 0.00	78.52 ± 0.20
Nearest Neighbor+T-Pre	72.48 ± 0.00	93.68 ± 0.00	77.13 ± 0.00
Multilayer Perceptron+T-Pre	65.35 ± 0.54	82.06 ± 1.18	58.49 ± 1.57
Support Vector Machine+T-Pre	62.49 ± 0.00	78.70 ± 0.00	69.51 ± 0.00
Naive Bayes+T-Pre	33.60 ± 0.00	54.08 ± 0.00	29.48 ± 0.00

As shown in Figure 18, the proposed DRAN-Net model improves all metrics of the traditional model. Compared with the best performing Random Forest in the traditional model, DRAN-Net improves the accuracy by 4.93%, the Top-3 accuracy by 2.33%, and the F1 by 1.43%, which proves the effectiveness of DRAN-Net. Furthermore, the prediction results obtained by DRAN-Net using traditional representation are close to the best prediction performance currently achievable, thereby proving the robustness of DRAN-Net.

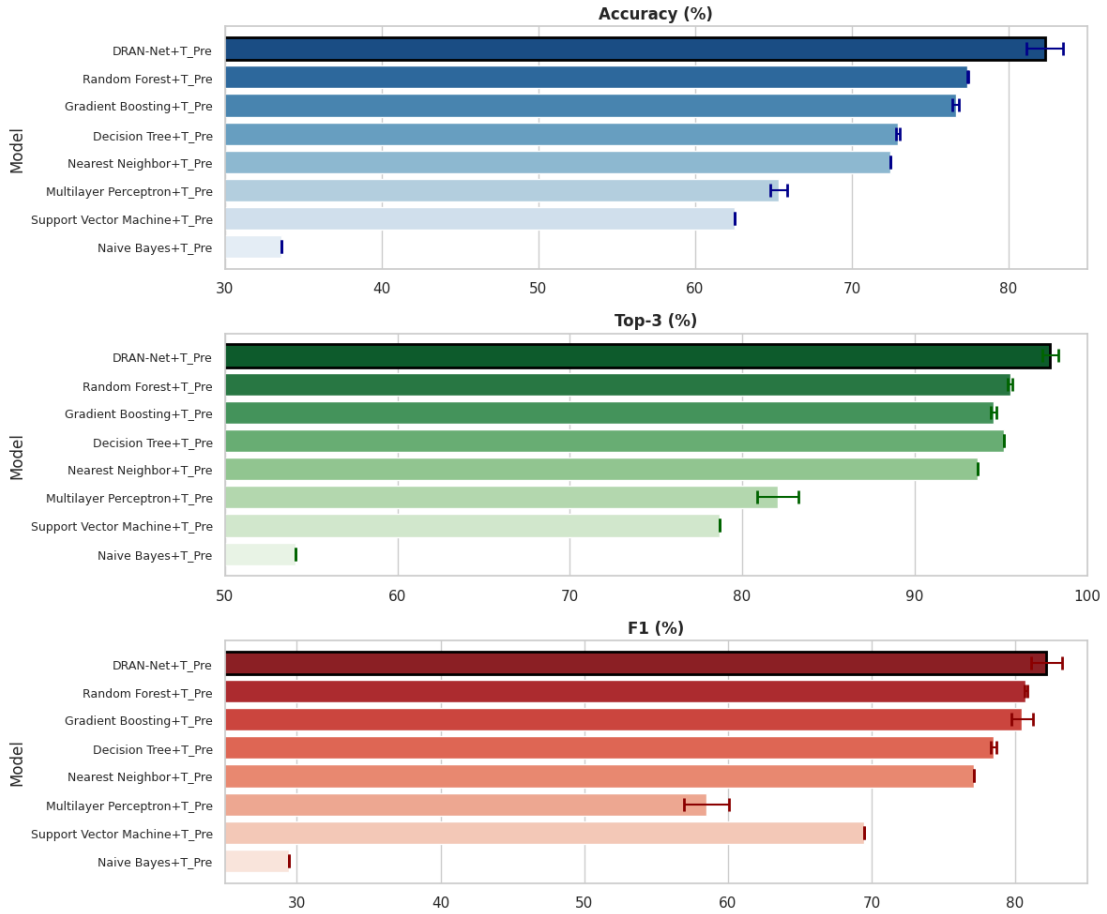


Figure 18. Performance Comparison of model using T-Pre.

Through these ablation experiments, we can more clearly understand how the various components of the method work together and their specific impact on improving prediction accuracy. These supplementary experiments further

demonstrate the contribution of these improvements to the overall performance improvement and verify the effectiveness, robustness and reliability of the proposed representation method and model method.

4.3.3. Advancedness Comparison Experiment

To comprehensively verify the superiority of the method proposed in this paper, we systematically compare it with two types of baseline methods. One type includes traditional machine learning models used in [11]: Random Forest, Gradient Boosting, Decision Tree, Nearest Neighbor, Multilayer Perceptron, Support Vector Machine, and Naive Bayes. The other type consists of more advanced graph neural network models: GIN (Graph Isomorphism Network), GCN (Graph Convolutional Network), and GraphSAGE. All comparative experiments are conducted under the same dataset and evaluation criteria. Results are reported as the mean $\pm$ standard deviation from five independent runs, as shown in Table 5.

Table 5. Prediction results of various model methods.

Model+Representation Set	Accuracy (%)	Top-3 (%)	F1 (%)
Random Forest+T-Pre	77.39 $\pm$ 0.05	95.54 $\pm$ 0.12	80.75 $\pm$ 0.12
Gradient Boosting+T-Pre	76.63 $\pm$ 0.21	94.58 $\pm$ 0.17	80.47 $\pm$ 0.76
Decision Tree+T-Pre	72.93 $\pm$ 0.10	95.17 $\pm$ 0.00	78.52 $\pm$ 0.20
Nearest Neighbor+T-Pre	72.48 $\pm$ 0.00	93.68 $\pm$ 0.00	77.13 $\pm$ 0.00
Multilayer Perceptron+T-Pre	65.35 $\pm$ 0.54	82.06 $\pm$ 1.18	58.49 $\pm$ 1.57
Support Vector Machine+T-Pre	62.49 $\pm$ 0.00	78.70 $\pm$ 0.00	69.51 $\pm$ 0.00
Naive Bayes+T-Pre	33.60 $\pm$ 0.00	54.08 $\pm$ 0.00	29.48 $\pm$ 0.00
GraphSAGE+TWIG-Pre	71.10 $\pm$ 1.02	95.77 $\pm$ 0.88	60.41 $\pm$ 2.49
GIN+TWIG-Pre	48.28 $\pm$ 0.64	84.52 $\pm$ 1.57	21.82 $\pm$ 1.50
GCN+TWIG-Pre	30.97 $\pm$ 0.25	56.54 $\pm$ 3.23	3.83 $\pm$ 0.49
Ours (DRAN-Net+T-Pre)	82.83 $\pm$ 1.63	98.14 $\pm$ 0.32	82.80 $\pm$ 1.52

Figure 19 demonstrates that Ours (DRAN-Net+T-Pre) achieves the best performance in terms of accuracy, Top-3 accuracy and F1. The accuracy has reached 82.83%, the Top-3 accuracy has reached 98.14%, and the F1 has reached 82.80%. In particular, the Top-3 accuracy is close to error-free prediction. This can avoid a large number of invalid attempts for end users, which is of great significance. Compared with the current best Random Forest prediction results, the method proposed in this paper has been greatly improved in all metrics, including an increase of 5.44% in accuracy, 2.6% in Top-3 accuracy, and 2.05% in F1. The analysis of the experimental results verifies the advancement of our proposed method. Moreover, it shows significant advantages over graph neural network (GNN) baseline methods. Compared with the best-performing GNN model (GraphSAGE), it achieves an 11.73% improvement in accuracy, a 2.37% improvement in Top-3 accuracy, and a 22.39% improvement in F1. This significantly enhances the balance of classification. Even when compared with advanced GNNs designed specifically for graph structure processing, DRAM-Net+N-Pre still demonstrates clear advantages. Analysis of the experimental results fully verifies the superiority of the method proposed in this paper.

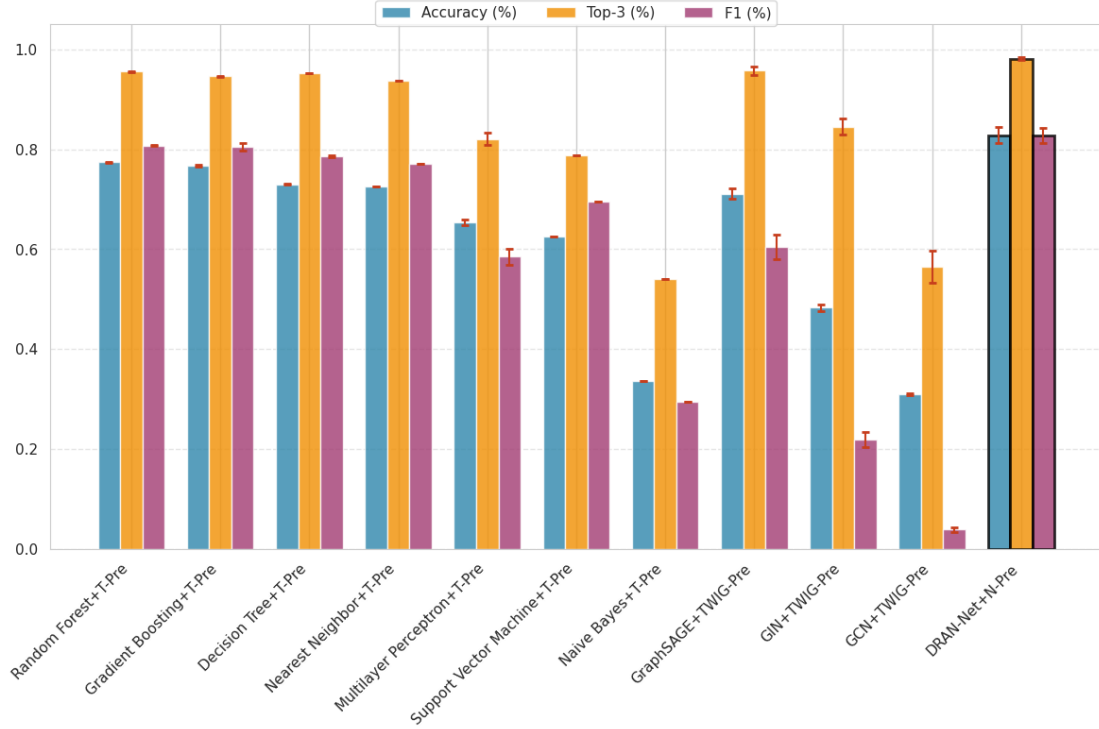


Figure 19. Performance Comparison of various model methods.

To evaluate the applicability and scalability of the proposed method in broader quantum compilation decision-making scenarios, we migrate our approach to the device selection task defined in [13] and conduct direct comparisons with the baseline method (Random Forest) adopted in that study, with experimental results summarized in Table 6. It is worth noting that this task aims to select the optimal execution device for a given quantum circuit and has been formulated as a classification problem, with the original dataset comprising 600 circuits.

Table 6. Prediction results on the device selection task from [13].

Model	Accuracy (%)	Top-3 (%)	F1 (%)
DRAN-Net	98.82 ± 0.41	100 ± 0.00	98.71 ± 0.50
Random Forest	98.15 ± 0.63	100 ± 0.00	97.01 ± 1.00

As shown in Figure 20, our DRAN-Net model comprehensively outperformed the original baseline method on this task introduced by [13]. Specifically, our model achieved statistically significant improvements in both accuracy (0.67% increase) and F1-score (1.7% increase). Particularly noteworthy is that while maintaining a perfect Top-3 accuracy of 100%, our model demonstrated more reliable performance on the more stringent accuracy. These results sufficiently indicate that the proposed method can effectively capture features applicable to various compilation decision scenarios, exhibiting excellent transferability and generalization potential. This provides empirical evidence for its integration as an intelligent component into more complex compilation frameworks such as [13]

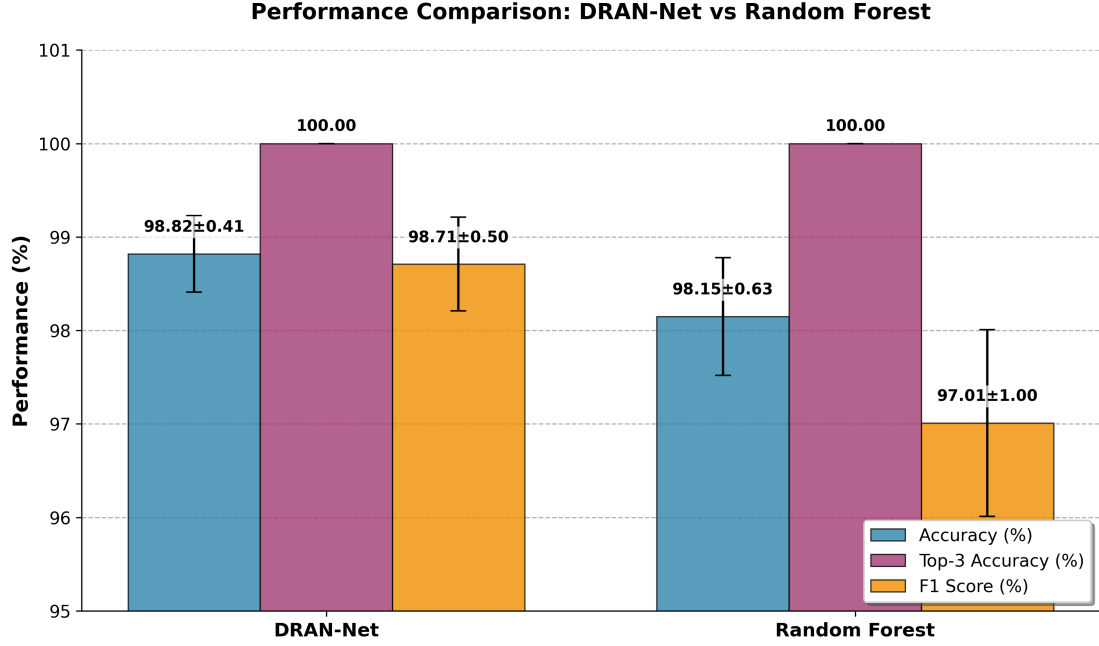


Figure 20. Performance comparison on the [13] device selection task.

In addition, when compared with the top-performing DRAN-Net+T-Pre in Table 4, DRAN-Net+N-Pre further delivers performance gains, with accuracy improved by 0.51%, Top-3 accuracy improved by 0.27%, and F1 improved by 0.62%. Since both configurations employ the same model method, this result further validates the advancedness of the representation method proposed in this paper. Separately, in comparison with the classification results of the top-performing Random Forest+N-Pre in Table 2, DRAN-Net+N-Pre exhibits performance improvements, including a 3.95% increase in accuracy, a 2.27% increase in Top-3 accuracy, and a 0.9% increase in F1. As both approaches utilize the same input representation set, this finding further confirms the advancedness of the DRAN-Net model.

Finally, to ensure the reliability of evidence, we conducted paired t-tests on all metrics of the proposed method and baseline methods on the test set. The goal was to evaluate the statistical significance of performance differences. For all comparisons, Welch’s corrected t-test was used when variances were unequal, while the independent samples t-test was applied when variances were equal. Additionally, the sample size was set to 5, which ensured the reliability of the statistical results.

Table 7 shows that for all performance metrics, the performance differences between Ours (DRAN-Net+N-Pre) and all baseline methods are statistically significant ($p < 0.05$). This indicates that this advantage does not only come from random errors, but more from the fusion advantages of the N-Pre representation and DRAN-Net’s modeling capability for feature fusion. These results fully validate the superiority and effectiveness of the overall method proposed in this paper for the quantum circuit compilation option prediction task.

Table 7. Statistical significance tests.

Ours (DRAN-Net+N-Pre) vs.	Metric	t	p-Value ($\alpha = 0.05$)
Ours (DRAN-Net+N-Pre) vs. Random Forest+T-Pre	Accuracy	8.4356	0.0009
	Top-3	17.0113	0.0000
	F1	3.0064	0.0391
Ours vs. Gradient Boosting+T-Pre	Accuracy	7.4592	0.0017
	Top-3	21.9686	0.0000
	F1	3.0658	0.0154
Ours vs. Decision Tree+T-Pre	Accuracy	13.5555	0.0002
	Top-3	20.7535	0.0000
	F1	6.2425	0.0030
Ours vs. Nearest Neighbor+T-Pre	Accuracy	14.1983	0.0001
	Top-3	31.1652	0.0000
	F1	8.3411	0.0011
Ours vs. Multilayer Perceptron+T-Pre	Accuracy	22.7628	0.0000
	Top-3	29.4089	0.0000
	F1	24.8754	0.0000
Ours vs. Support Vector Machine+T-Pre	Accuracy	27.9028	0.0000
	Top-3	135.8411	0.0000
	F1	19.5509	0.0000
Ours vs. Naive Bayes+T-Pre	Accuracy	67.5347	0.0000
	Top-3	307.8786	0.0000
	F1	78.4389	0.0000
Ours vs. GraphSAGE+TWIG-Pre	Accuracy	13.6408	0.0000
	Top-3	5.6596	0.0005
	F1	17.1617	0.0000
Ours vs. GIN+TWIG-Pre	Accuracy	44.1176	0.0000
	Top-3	19.0074	0.0000
	F1	63.8515	0.0000
Ours vs. GCN+TWIG-Pre	Accuracy	70.3203	0.0000
	Top-3	28.6586	0.0000
	F1	110.5693	0.0000

To evaluate the practical application value of the proposed method, this paper randomly selected 100 representative circuits from MQT Bench (due to limited computational resources of the device, the selected circuits have qubit counts ranging from 5 to 50 and gate counts ranging from 50 to 5000). Exhaustive search (enumeration) of all 30 compilation options and DRAN-Net model prediction experiments were conducted separately to assess the execution efficiency (time) and effectiveness (quality) of exhaustive search versus the DRAN-Net model in single-circuit compilation decision-making. The experimental results are presented in Table 8.

Table 8. Efficiency and effectiveness of different methods.

Method	Average Time(s)	Time Ranges(s)	Accuracy (%)	Top-3 (%)
Exhaustive Search	118	5-243	100 $\pm$ 0.00	100 $\pm$ 0.00
DRAN-Net Prediction	0.27	0.19-0.51	90.40 $\pm$ 0.80	99.40 $\pm$ 0.49

As shown in Figure 21, the DRAN-Net model significantly reduces the compilation decision time from an average of 118 seconds to 0.27 seconds, achieving a 437-fold performance improvement. This indicates that our model accomplishes a leap from minute-level to sub-second response times, providing crucial technical support for interactive quantum program development. Furthermore, while achieving this dramatic speed improvement, the DRAN-Net model maintains competitive decision-making quality. The model's Top-3 accuracy reaches 99.4%, meaning that for nearly all circuits, the optimal compilation option is included among the top three recommendations provided by the model. This ensures that in practical scenarios, users only need to try a very limited number of options to obtain near-optimal compilation results. The model's accuracy is 90.4%, meaning that in approximately nine out of ten cases, its

top recommendation is the globally optimal solution. Although there is a gap compared to the theoretical optimum achieved by exhaustive search, this level of accuracy is attained alongside a 437-fold acceleration. For the vast majority of application scenarios, the minimal suboptimal risk accepted in exchange for such a substantial efficiency gain is acceptable. In summary, the DRAN-Net model achieves an effective trade-off in the compilation decision-making problem, further demonstrating the validity of the DRAN-Net model.

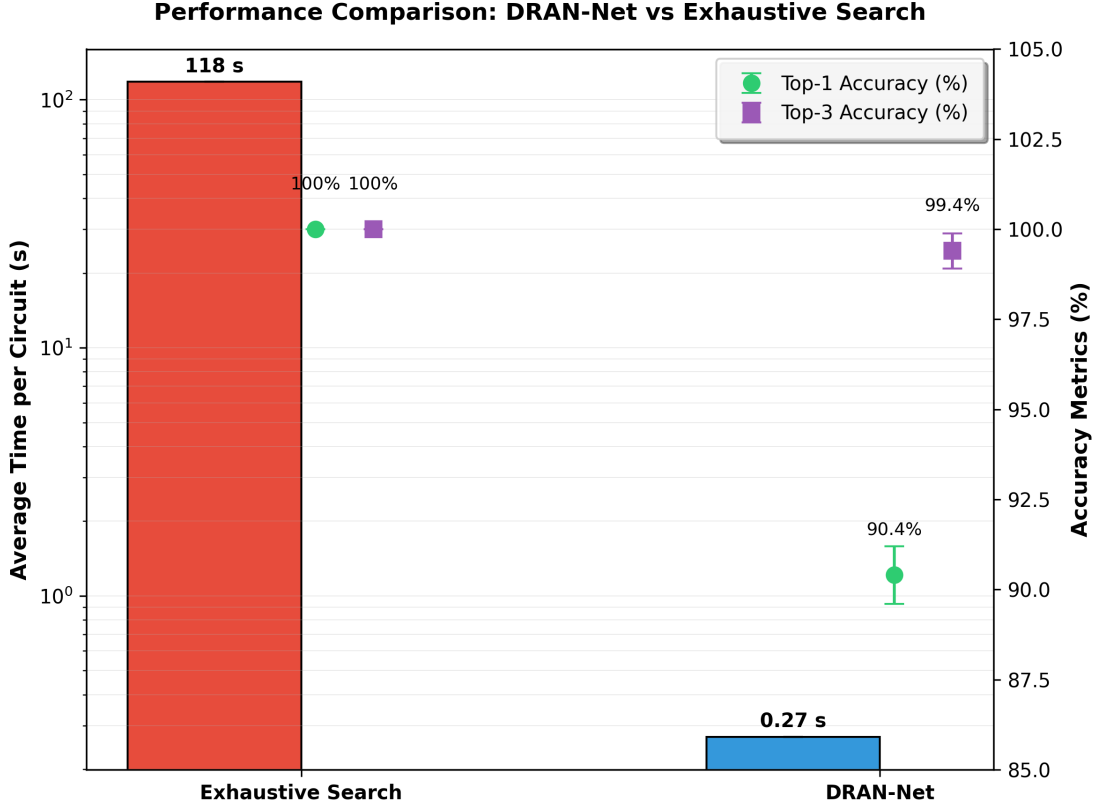


Figure 21. Exhaustive search vs. DRAN-Net prediction.

5. Limitations and Future Work

This paper presents an efficient quantum compilation option prediction method. However, with the rapid development of quantum computing technology, the current study still has some limitations. The following directions warrant further exploration:

Evolution of Model Architecture. Although DRAN-Net performs excellently in compilation option prediction tasks, the current model still faces challenges when processing extremely large-scale quantum circuits. In the future, Graph Neural Networks (GNN) could be introduced for end-to-end circuit representation, combined with Reinforcement Learning (RL) to achieve dynamic compilation strategy sequence generation as explored in [13], thereby enhancing the model’s adaptability to complex quantum circuits.

Dataset Expansion and Benchmark Establishment. The current research is based on the MQT Bench benchmark dataset, which primarily contains arithmetic and logical quantum circuits. To improve the model’s generalization capability, we will incorporate more representative quantum algorithms such as VQE [31], QAOA [32], and QFT [33] to build a more diverse training dataset and conduct systematic comparative analysis with advanced models like Transformer [25].

Exploration of Fundamental Compilation Issues. This paper focuses on the high-level decision-making problem of compilation option prediction and has not yet deeply explored the dependencies between compilation options or the impact of hardware noise. In the future, we will commit to explicit dependency modeling between compilation options and develop noise-aware adaptive compilation strategies to address the noise challenges in real quantum hardware [34].

6. Conclusions

In this paper, we propose a quantum circuit compilation option prediction method based on DRAN-Net, which can automatically select the optimal compilation scheme. The efficiency and accuracy of quantum circuit compilation are improved, allowing end users from non-quantum computing fields to implement their applications more efficiently on actual hardware. Firstly, we propose a new quantum circuit representation method, introduce graph features based on TWIG, and combine them with traditional features into a fixed 36-dimensional representation vector, which more fully extracts quantum circuit information. Secondly, we propose a new quantum circuit compilation option prediction model DRAN-Net. The introduction of residual and attention mechanism methods prevents the gradient vanishing problem in model training, which greatly improves the prediction performance of the model. Finally, in order to verify the effectiveness and advancement of the proposed method, we design three groups of experiments. In the feature correlation and importance experiments, we verify the effectiveness of the proposed features and obtain a new feature arrangement form. In the ablation experiment, we verify the effectiveness of the proposed quantum circuit representation method and prediction model method, respectively. In the advancedness comparison experiment, the proposed method improves the accuracy by 5.44%, the Top-3 accuracy by 2.6%, and the F1 by 2.05% compared with the current advanced methods.

Author Contributions

Investigation: Shouli He and Yangzhi Li; Methodology: Wen Liu and Shouli He; Software: Shouli He and Yangzhi Li; Validation: Shouli He, Yangzhi Li, and Wen Liu; Formal analysis: Wen Liu and Shouli He; Writing-original draft preparation: Shouli He, Yangzhi Li, and Wen Liu; Writing-review and editing: Shouli He and Yangzhi Li; Visualization: Kai Chen and Maoduo Li; Supervision: Wen Liu and Yaohua Lu; Project administration: Wen Liu. All authors read and approved the final manuscript.

Funding

This research was funded by the National Key Research and Development Program in China (grant 2022YFC3302103-01), the Strategic Research Program of Science and Technology Commission of the Ministry of Education of China (grant JYB2022-01), and the Fundamental Research Funds for the Central Universities.

Conflicts of Interest Statement

There is no conflict of interest.

Data Availability Statement

The data presented in this study are available upon reasonable request from the corresponding author.

References

1. A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop and A. W. Cross (2024). “Quantum computing with Qiskit”. *arXiv preprint*.
2. S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington and R. Duncan (2020). *tket* : A retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6, 014003.
3. R. S. Smith, M. J. Curtis and W. J. Zeng (2016). A practical quantum instruction set architecture. *arXiv preprint*.
4. V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. Akash-Narayanan and A. Asadi (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint*.
5. L. K. Grover (1996). “A fast quantum mechanical algorithm for database search”, in: *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pp. 212–219.
6. P. W. Shor (1999). “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. *SIAM Review*, 41, 303–332.
7. A. W. Harrow, A. Hassidim and S. Lloyd (2009). “Quantum algorithm for linear systems of equations”. *Physical Review Letters*, 103, 150502.
8. M. Salm, J. Barzen, U. Breitenbücher, F. Leymann, B. Weder and K. Wild (2020). “The NISQ analyzer: automating the selection of quantum computers for quantum algorithms”, in *Symposium and Summer School on Service-Oriented Computing*, pp. 66–85. Springer
9. M. Salm, J. Barzen, F. Leymann and B. Weder (2022). “Prioritization of compiled quantum circuits for different quantum computers”, in: *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 1258–1265. IEEE.

10. M. Salm, J. Barzen, F. Leymann and P. Wundrack (2023). “How to select quantum compilers and quantum computers before compilation”, in: *CLOSER*, pp. 172–183.
11. N. Quetschlich, L. Burgholzer and R. Wille, “Predicting good quantum circuit compilation options”, in: *2023 IEEE International Conference on Quantum Software (QSW)*, pp. 43–53 (2023). IEEE.
12. N. Quetschlich, L. Burgholzer and R. Wille (2023). “Compiler optimization for quantum computing using reinforcement learning”, in: *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE.
13. N. Quetschlich, L. Burgholzer and R. Wille (2023). “MQT predictor: Automatic device selection with device-specific circuit compilation”. *ACM Transactions on Quantum Computing*.
14. T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi and F. T. Chong (2022). “Supermarq: A scalable quantum benchmark suite”, in: *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 587–603. IEEE.
15. M. Bandic, C. G. Almudever and S. Feld (2022). “Interaction graph-based profiling of quantum benchmarks for improving quantum circuit mapping techniques”. *arXiv preprint*.
16. Y. Qian, Z. Guan, S. Zheng and S. Feng (2023). “A method based on timing weight priority and distance optimization for quantum circuit transformation”. *Entropy* 25(3), 465.
17. L. Breiman (2001). “Random forests”. *Machine Learning* 45, 5–32.
18. J. H. Friedman (2001). “Greedy function approximation: A gradient boosting machine”. *Annals of Statistics* 1189–1232.
19. W.-Y. Loh (2011). “Classification and regression trees”. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1(1), 14–23.
20. T. Cover and P. Hart (1967). “Nearest neighbor pattern classification”. *IEEE Transactions on Information Theory* 13(1), 21–27.
21. M. Riedmiller and A. Lernen (2014). “Multi-layer perceptron”. *Machine Learning Lab Special Lecture, University of Freiburg* 24.
22. C. Cortes (1995). “Support-vector networks”. *Machine Learning*.
23. A. P. Dempster, N. M. Laird and D. B. Rubin (1977). “Maximum likelihood from incomplete data via the EM algorithm”. *Journal of the Royal Statistical Society, Series B*, 39:1, 1–22.
24. K. He, X. Zhang, S. Ren and J. Sun (2016). “Deep residual learning for image recognition”, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.
25. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin (2017). “Attention is all you need”. *Advances in Neural Information Processing Systems*.
26. F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang and X. Tang (2017). “Residual attention network for image classification”, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164.
27. J. M. Hernández and P. Van Mieghem, “Classification of graph metrics”, University of Technology: Mekelweg, The Netherlands 1.
28. L. d. F. Costa, F. A. Rodrigues, G. Travieso and P. R. Villas Boas (2007). “Characterization of complex networks: A survey of measurements”. *Advances in Physics*, 56(1), 167–242.
29. A. W. Cross, L. S. Bishop, J. A. Smolin and J. M. Gambetta (2017). “Open quantum assembly language”. *arXiv preprint*.
30. P. Sedgwick (2014). “Spearman’s rank correlation coefficient”. *BMJ*, 349.
31. M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan and L. Cincio (2021). “Variational quantum algorithms”. *Nature Reviews Physics*, 3(9), 625–644.
32. J. Choi and J. Kim (2019). “A tutorial on quantum approximate optimization algorithm (QAOA): Fundamentals and applications”, in: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 138–142. IEEE.
33. F. Mandl and G. Shaw (2013). *Quantum Field Theory*. John Wiley & Sons.
34. Y.-Z. Li, W. Liu, G.-S. Xu, M.-D. Li, K. Chen and S.-L. He (2025). “Quantum circuit mapping based on discrete particle swarm optimization and deep reinforcement learning”. *Swarm and Evolutionary Computation*, 95, 101923.