POLITECNICO DI TORINO
Repository ISTITUZIONALE

Fidelity-Based Quantum Device Selection Using Graph Neural Networks

Original
Fidelity-Based Quantum Device Selection Using Graph Neural Networks / Tudisco, A., Hopf, P., Schulte, L., Volpe, D., Turvani, G., Wille, R.. - (2026), pp. 129-140. (2026 IEEE International Conference on Quantum Software IEEE QSW 2026 Sydney (Aus) 13-18 July 2026) [10.1109/QSW72780.2026.00023].

Availability:
This version is available at: 11583/3015031 since: 2026-08-31T19:05:31Z

Publisher:
IEEE

Published
DOI:10.1109/QSW72780.2026.00023

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

25 October 2026

Fidelity-Based Quantum Device Selection Using Graph Neural Networks

Antonio Tudisco*, Patrick Hopf†‡, Linus Schulte†, Deborah Volpe§, Giovanna Turvani*, Robert Wille†‡

*Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy
†Chair for Design Automation, Technical University of Munich, Munich, Germany
§Istituto Nazionale di Geofisica e Vulcanologia, Rome, Italy
‡MQSC, Garching near Munich, Germany

{antonio.tudisco, giovanna.turvani}@polito.it;
{patrick.hopf, l.schulte, robert.wille}@tum.de;
deborah.volpe@ingv.it

***Abstract*—Modern High-Performance Computing (HPC) infrastructures are increasingly integrating multiple Quantum Processing Units (QPUs) of different types alongside classical resources. Unlike traditional accelerator scheduling, quantum device selection affects not only performance but also the quality of the computed result because current quantum hardware exhibits device-dependent noise characteristics. This makes efficient and fidelity-based device selection a key challenge in multi-QPU environments.**

In this work, we propose a Graph Neural Network (GNN) model that exploits the structural representation of quantum circuits to predict their execution fidelity across different QPUs. By operating on uncompiled circuits, the approach supports early-stage device selection and enables making informed scheduling decisions. We evaluate the model on a diverse set of benchmark circuits compiled for superconducting and trapped-ion devices. The results show that the proposed approach outperforms existing baseline models, achieving 30% Mean Squared Error (MSE) and 20% Mean Absolute Error (MAE) reduction, and provides an effective basis for fidelity-based scheduling in heterogeneous HPC-quantum systems.

***Index Terms*—quantum computing, graph neural networks, quantum circuits, fidelity estimation, quantum device selection, multi-QPU workflow, heterogeneous quantum computing, scheduling**

I. Introduction

High-Performance Computing (HPC) systems are fundamental to modern scientific research, providing the large-scale computational resources necessary to address increasingly complex and data-intensive problems [1]. These systems are inherently heterogeneous and typically combine conventional Central Processing Units (CPUs) with specialized accelerators such as Graphics Processing Units (GPUs) [2]. To coordinate access to such resources, large-scale HPC facilities rely on workload management systems and job schedulers, such as Slurm [3], Flux [4], the Load Sharing Facility (LSF) [5], and the Portable Batch System Professional (PBS Pro) [6]. These schedulers manage job queues, enforce allocation policies, and place workloads on heterogeneous nodes according to resource availability and user-specified constraints.

In this classical setting, resource selection is mainly driven by performance-oriented criteria such as execution time, throughput, memory bandwidth, and system utilization. Although different devices may exhibit different performance characteristics, they are generally expected to produce equivalent computational results up to numerical precision, such that device choice mainly affects efficiency rather than correctness.

This assumption no longer holds when Quantum Processing Units (QPUs) are integrated into HPC infrastructures. In recent years, major HPC centers have started to include QPUs alongside classical accelerators, thereby extending heterogeneous computing environments into the quantum domain. However, in the Noisy Intermediate-Scale Quantum (NISQ) era [7], available quantum processors provide only a limited number of qubits that remain highly susceptible to gate errors, calibration drift, environmental noise, and other sources of noise and control imperfections. As a result, different QPUs can yield substantially different execution fidelities for the same quantum program.

Consequently, in contrast to the classical case, device selection in hybrid HPC-quantum systems affects not only performance-related metrics but also the quality and reliability of the computed result itself. This shift makes resource selection for quantum workload scheduling [8] fundamentally different from traditional accelerator scheduling and calls for new approaches that explicitly account for hardware-dependent execution quality.

Leveraging Graph Neural Networks (GNNs), we address the challenge by proposing a regression model capable of estimating the Expected Success Probability (ESP) of quantum circuits across different quantum devices. This method enables informed device selection during scheduling and reduces the need for costly trial-and-error compilation and execution on multiple backends.

Evaluations show that the proposed GNN approach outperforms Random Forest-based models by 30% in Mean Squared Error (MSE) and 20% in Mean Absolute Error (MAE), demon-

strating its effectiveness for ESP-based device selection in heterogeneous multi-QPU environments.

The remainder of this work is structured as follows: Section II reviews the multi-QPU workflow, discusses how machine learning-based approaches can support device selection, and highlights the limitations of existing methods. Section III presents the proposed GNN-based regression workflow for ESP prediction. Section IV describes the methodological details of the model implementation. Section V then defines the evaluation setup used to test whether these predictions are accurate enough for device selection, and Section VI discusses the results. Finally, Section VII concludes the work.

II. Background and Motivation

Today, the quantum computing landscape is rapidly evolving, encompassing technologies such as superconducting qubits, trapped ions, neutral atoms, and photonics. As these platforms become increasingly integrated into existing HPC and cloud environments, comprehensive quantum software stacks have emerged to manage complex, end-to-end workflows. In the following, we outline a representative multi-QPU workflow and highlight where Machine Learning (ML) can meaningfully support decision-making, as well as where current models fall short of practical requirements.

A. Multi-QPU Workflow

Current quantum software stacks, such as those described in [9], are increasingly deployed within HPC environments. They orchestrate end-to-end workflows including device selection, compilation, scheduling, and execution across heterogeneous quantum resources. To manage this complexity, these stacks typically adopt a modular architecture in which device selection [10], compilation [11], and scheduling [12] are handled by distinct software components. This separation of concerns is essential for scalability, maintainability, and seamless integration with existing HPC workflows.

Example 1. *Consider the following scenario: a user submits a quantum program to the software stack. First, a hardware-agnostic compilation stage performs generic optimizations, such as gate cancellation or commutation-based simplifications—without any knowledge of the target device's specific characteristics. Next, a* suitable *QPU is selected from the pool of available devices. Then, a hardware-aware compilation stage performs gate synthesis, qubit mapping, routing, and optimization—taking into account the native gate set, the target device's coupling map and calibration data. Finally, the program is queued for execution on the device, considering factors like job priority.*

The example illustrates a sequential compilation pipeline in which each stage operates under well-defined assumptions. One key aspect of this pipeline, however, remains underspecified, namely what constitutes a *suitable* QPU for a given quantum program. Determining suitability is non-trivial and cannot be reliably captured using simple heuristics, for instance, qubit count or connectivity alone. Given the vast combinatorial space of quantum programs and the heterogeneous characteristics of available QPUs, this task naturally lends itself to data-driven and ML-based approaches, as the relationship between program/device features and the resulting execution outcomes is highly complex and not easily captured analytically.

B. ML-guided Workflow

In many application domains, ML-based techniques enable rapid performance predictions at the cost of (bounded) approximation error. In particular, ML applications have been developed in the formal verification field [13], for logic optimization [14], resource estimation [15], [16], placement [17], and routing [18]. Prior work has also explored the use of ML models for compilation-related tasks in quantum computing, including mapping [19], placement [20], and circuit synthesis [21].

In particular, GNNs have been exploited for reliability estimation. For instance, [22] integrates a GNN into a reinforcement learning loop to estimate fidelity during qubit mapping optimization. Similarly, [23] compares traditional regressors with GNN-based approaches for predicting output fidelity of compiled quantum circuits, demonstrating the advantages of graph-based representations. Other works leverage GNNs for physically aware compilation [24] and qubit allocation [25].

However, a common characteristic of these approaches is that they rely on information extracted from already hardware-compiled programs. As a result, the target quantum device must be selected in advance, and these methods can only estimate execution quality during or *after* hardware-aware compilation.

In practical quantum software stacks, however, the decision of which QPU is most suitable (and therefore should be targeted) must be made *before* hardware-dependent compilation, since this decision influences subsequent mapping, routing, scheduling, and optimization steps. This temporal mismatch between when predictions become available and when they are needed fundamentally limits the applicability of many existing ML-based approaches. This motivates the development of prediction models that operate at earlier, hardware-agnostic stages of the compilation pipeline.

Initial work in this direction, such as the MQT Predictor [10], addresses device selection with a Random Forest (RF) classifier-based approach. While effective at ranking devices, such random-forest-based classifiers provide only categorical decisions. They do not yield quantitative estimates of expected performance differences between devices for a given program.

Example 2. *Consider a user who wishes to execute a quantum circuit on one of two available noisy QPUs. Let the quality of the computation, i.e., the closeness to execution on a noise-less device, be measured for now by the execution fidelity* $\in [0, 1]$.

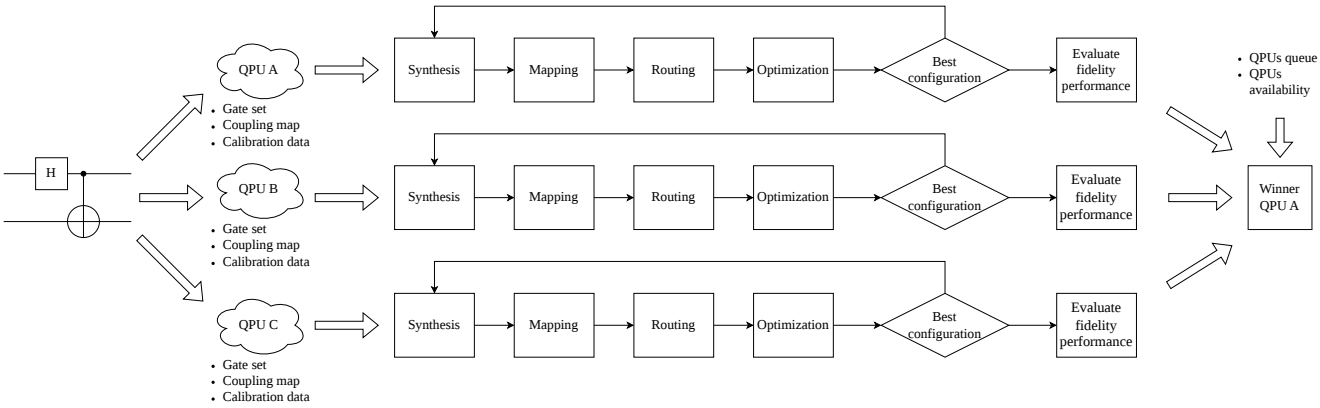


Fig. 1: Workflow for selecting the most suitable quantum device for a given program. The program is compiled for each available QPU, and the chosen fidelity/performance metric is evaluated using device-specific calibration data. The final device is then selected by maximizing the considered performance metric while accounting for additional factors such as queue length and availability.

The classifier predicts QPU-A as the most suitable device without providing a quantitative justification. After hardware-aware compilation, the actual achieved execution fidelity might be 0.11 *on QPU-A and* 0.10 *on QPU-B. While this prediction is technically correct, the performance difference is marginal and may not warrant prioritizing QPU-A.*

Without fine-grained predictions, secondary factors commonly available in HPC environments, such as queue length, device availability, or expected wait time, cannot be quantitatively weighed against the predicted execution outcome in the decision process. This limitation becomes particularly problematic when all candidate devices exhibit poor absolute performance. For instance, in the scenario of Example 2, both QPU-A and QPU-B yield low absolute fidelities. Without explicit performance estimates, the user may proceed with execution on QPU-A based solely on the classifier output, despite neither device being well suited for the task.

Purely classification-based approaches obscure low execution fidelities and may lead to unnecessary compilation and execution attempts. In contrast, access to quantitative performance estimates would allow users to discard unsuitable devices early, conserving both computational and experimental resources.

III. Proposed Solution

Motivated by the limited information gain of purely classification-based approaches, we reformulate the device-selection problem as a regression task. Instead of predicting a categorical device choice, the model directly estimates relevant performance metrics for a given quantum program across different quantum devices. In the following, we outline a suitable figure of merit for this task and discuss the advantages of using GNNs for its prediction.

A. Estimated Success Probability

Several figures of merit have been proposed to quantify execution performance without requiring costly execution on quantum hardware. One such metric is the *expected fidelity* f_{expected}, defined as the product of all gate and measurement fidelities appearing in a program. This metric captures the cumulative effect of imperfect operations but neglects decoherence effects arising from idle times [26].

Closely related, the *Estimated Success Probability* (ESP) [27] extends this notion by additionally accounting for qubit coherence properties and gate durations. It is defined as

$$f_{\text{ESP}} = f_{\text{expected}} \times \prod_{q \in \text{qubits}} \exp\left[-t^q_{\text{idle}} / \min\left(T^q_1, T^q_2\right)\right], \quad (1)$$

where t^q_{idle} denotes the total idle time of qubit q, and T^q_1 and T^q_2 denote its relaxation and dephasing times, respectively. Although ESP enables a more realistic assessment of noise effects on a compiled program, its manual evaluation requires device-specific calibration data, which is only available *after* device selection and compilation.

Example 3. *Consider three available quantum devices, each with distinct native gate sets, coupling map, and calibration data, as illustrated in Figure 1. During compilation—encompassing gate synthesis, qubit mapping, routing, and optimization—the device-specific native gate set, coupling map, and calibration data such as gate error rates, coherence times, and gate durations must be considered.*

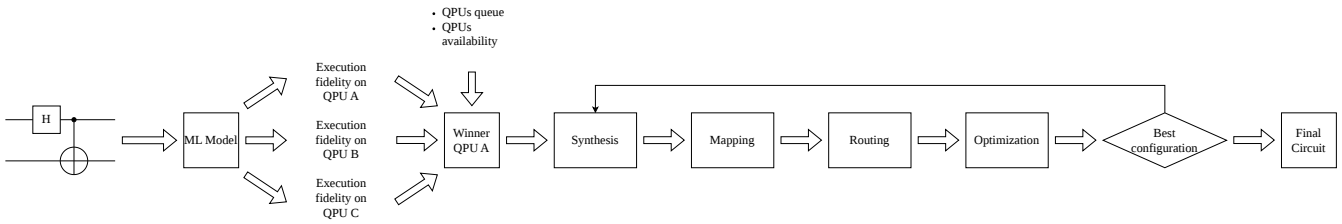


Fig. 2: Workflow for selecting the most suitable quantum device for a given program and compiling it accordingly. The device is chosen based on the fidelity predictions of a ML model, avoiding individual compilation for all candidate devices.

If multiple compiler options are available, the compilation is repeated several times to identify the best configuration. To find the most suitable QPU, a suitable performance metric (e.g., ESP) must be computed multiple times for all devices. The overall computational cost scales with both the number of compilation attempts and the number of devices.

Example 3 highlights repeated device-specific compilation makes exhaustive ESP-based hardware selection increasingly costly, motivating ML-based approaches. Using ML for performance prediction opens the possibility of estimating the ESP prior to device selection and compilation. In principle, such a model can be trained on ESP-labeled data obtained from previously compiled quantum programs and then used to approximate the ESP of candidate QPUs without explicitly compiling the program for each device. At a high level, such a model first learns program characteristics that matter across devices and then combine them with device-specific information, allowing a single model to produce ESP estimates for multiple QPUs.

Example 4. *Consider the same setup as in the previous example. Figure 2 illustrates an alternative workflow in which an ML model predicts the execution fidelity (e.g., ESP) for all available quantum devices prior to compilation and in parallel, rather than compiling each quantum circuit separately for every device. This reduces the compilation-and-evaluation loop to a single selected device while shifting the main computational burden to the offline training stage of the ML model. At inference time, the computational cost is then largely independent of the number of candidate devices.*

Given a suitable ML-model, this enables a data-driven device-selection stage prior to hardware-aware compilation.

B. Graph Neural Networks

Among recently popular ML models, GNNs are particularly well suited for *quantum circuit* related tasks [28]. Quantum circuits—the fundamental structure of a quantum program—admit a natural graph representation, typically as directed acyclic graphs encoding gate dependencies. Each node in the graph represents a quantum operation and can be encoded as a numerical feature vector. GNNs process such data through message passing, where each node iteratively aggregates numerical information from its neighbors to refine its representation. This is conceptually analogous to convolutional neural networks, which learn from local pixel neighborhoods in images. A key advantage of GNNs is their ability to operate on graphs of variable size without requiring a fixed input dimension, unlike many conventional ML models. This enables the prediction of circuit-level labels without imposing dimensional constraints while preserving the structural information of the circuit.

In this work, we employ SAGE convolution [29], which is well suited for inductive learning scenarios, when the model must generalize to new unseen graphs. This is particularly advantageous for quantum circuits, where circuit size and structure vary widely across workloads.

By jointly exploiting circuit topology and gate-level features, the GNN learns complex dependencies that are predictive of execution performance on different devices. Unlike device-specific models, this approach does not require training a separate model for each QPU. Instead, a single trained model can predict performance metrics for multiple devices in parallel. This significantly improves scalability and reduces both training and maintenance costs.

Summarizing these considerations, the main contributions of this work are:

- We define a GNN model to predict the ESP of quantum circuits on different target quantum devices *in parallel* and *before* compilation.
- We deliver a framework that, given a labeled dataset of quantum circuits, learns to identify the most suitable architecture; this framework is provided as an *open-source* implementation for public use.
- We evaluate the quality of the learned intermediate representations by a *Principal Component Analysis* (PCA), assessing whether the model can cluster circuits with similar performance labels.

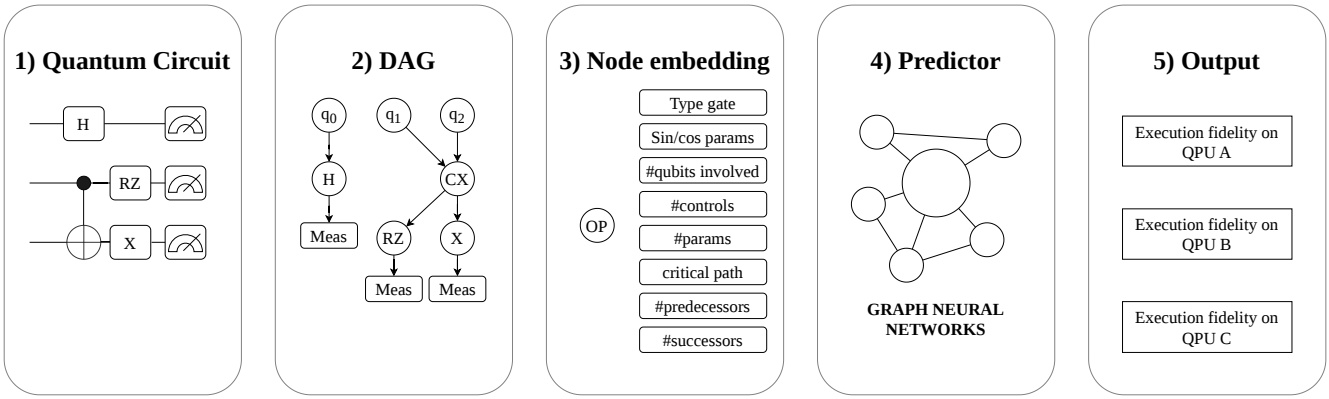


Fig. 3: Fidelity-based quantum device selection using a GNN. Starting from an input quantum circuit, the framework constructs a DAG, converts gate information into a numerical node representation, and feeds the resulting feature vector to a GNN. The model then estimates the execution fidelity for all considered target devices in parallel.

- We compare the proposed approach with an RF regressor, demonstrating that the GNN improves prediction accuracy and provides more informative performance estimates.

IV. Methodology

As discussed above, selecting a QPU by brute force is computationally expensive because each circuit must be compiled and evaluated separately for every target backend. At the same time, most ML-based approaches for circuit performance prediction require information available only after QPU-specific compilation. To overcome these limitations, we employ a GNN-based approach that facilitates QPU selection by predicting ESP across different quantum devices *in parallel* and for *uncompiled* circuits.

An overview of the resulting automated workflow is shown in Fig. 3, highlighting the main steps from the input circuit to the predicted device. The framework takes 1) a quantum circuit as input, converts it into 2) a DAG, and transforms the node information into 3) a numerical representation suitable for further processing. The resulting feature vector is then passed to 4) a GNN model, which predicts 5) the execution fidelity (e.g., ESP) for all considered devices. This prediction is subsequently used to select the device on which the quantum circuit will be executed.

This section describes the main components necessary to realize that early-stage prediction workflow, including dataset generation and model definition.

A. Pre-processing and Dataset Generation

In contrast to existing approaches, our model is trained to make predictions at the stage where device selection is actually needed, namely, before hardware-aware compilation. The key idea is therefore to estimate circuit performance directly from a hardware-agnostic description.

To ensure consistent, uniform representation across all samples, input quantum circuits are expressed in OpenQASM 3 [30], which is widely adopted as a de facto standard for defining quantum circuits. Each uncompiled circuit is then converted into a DAG, where each node represents a quantum operation, and each edge encodes a dependency induced by the qubits acted on. Every node is assigned a 39-dimensional feature vector that describes the corresponding operation. The first 27 entries encode the operation type using a one-hot representation. Since an OpenQASM 3 quantum operation can have at most three parameters, these are represented by a 6-dimensional feature vector containing the sine and cosine of each parameter. This encoding captures the periodic structure of the parameters while ensuring that all resulting values remain in the interval $[-1, 1]$. The remaining 6 elements encode the following properties:

- the number of involved qubits,
- the number of control qubits,
- the number of angle parameters,
- a binary flag indicating whether the gate lies on the device-agnostic critical path, *i.e.*, the longest path in the DAG,
- the number of predecessors of the node,
- the number of successors of the node.

For training, each quantum circuit is associated with a vector of labels containing the values of the selected performance metric (in this work, the ESP) computed for that circuit on each target quantum device. This allows the model to learn relative device suitability without relying on device-specific compiled circuits as input.

B. Model Definition

Our approach employs a GNN composed of a stack of SAGE convolutional layers [29]. The overall architecture is defined by multiple hyperparameters, which are described below.

The proposed model includes N standard *SAGE* convolution layers, followed by M additional *SAGE* convolution layers arranged in a *residual configuration*. After each convolution, a module composed of a *Graph Normalization*, a *leaky relu* activation function, and a *dropout* layer is applied. The convolutional stack is followed by a feed-forward neural network, whose output dimensionality matches the number of devices considered in the target task, where each output corresponds to the predicted performance on an associated device.

To capture directional dependencies in circuit graphs, the model can optionally employ *bidirectional message passing*. This is implemented by applying a SAGE convolution both to the original graph and the edge-reversed graph, and then averaging the resulting node embeddings.

Optionally, a self-attention graph pooling layer (*SAGPooling*) can be applied after the convolutional stack. This layer learns node importance scores and retains only a fraction of the most informative nodes.

Overall, the model is designed so that most of the representation learning is carried out by the convolutional layers. This encourages the convolutional backbone to map circuits with similar structural and operational characteristics to nearby regions in the latent space, improving generalization to unseen circuits.

C. Implementation

The proposed method is available as part of the open-source MQT Predictor[1]. Given a range of target quantum devices, the framework compiles a set of quantum circuits for all devices during dataset creation, thereby generating the ESP-based labels required for supervised learning. It then converts the uncompiled circuit representations into DAGs and applies the preprocessing steps for feature generation as described above.

Once the dataset has been created, it is split into training and test sets, which are used for model training and final performance evaluation, respectively. The training set is further divided into a training subset and a validation subset, where the latter is used to determine the optimal hyperparameters.

The selection of the optimal hyperparameters of the model is performed using Optuna [31]. Optuna automates the hyperparameter optimization process, which would be impractical with grid search because of the large search space. In particular, Optuna uses a Bayesian optimization strategy to identify the model configuration that minimizes the validation loss. Once the best configuration has been found, the corresponding model is saved and subsequently evaluated on the test set.

[1]https://github.com/munich-quantum-toolkit/predictor

As a result, any model trained within the open-source framework can be used directly to estimate device-specific ESP values for previously unseen quantum circuits, thereby facilitating the device-selection step.

V. Evaluation Setup

The evaluation is designed to answer the practical question: can the proposed GNN model predict device-specific ESP accurately enough to enable quantitative device selection before hardware-aware compilation?

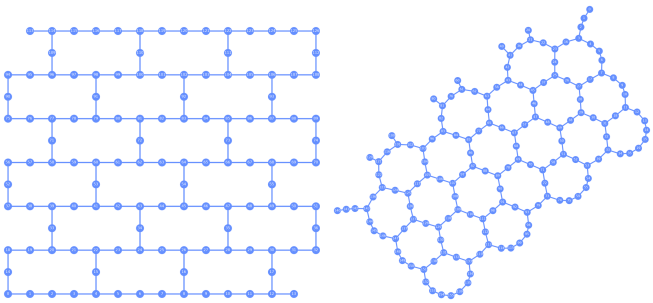
To answer this, we benchmark the proposed GNN-based approach against a state-of-the-art RF-based device selector. The following subsections describe the compared models, the heterogeneous target devices, and the training and evaluation datasets.

A. ML Models

To evaluate fidelity-based device selection fairly, the baseline must operate at the same point in the workflow, namely, before hardware-aware compilation. This substantially restricts the range of viable competitors. To this end, we consider the MQT Predictor [10], a state-of-the-art RF-based model for quantum device selection, as a baseline.

Note, the original MQT Predictor is formulated as an RF classifier that predicts the most suitable device for a given quantum circuit on the basis of a set of hand-crafted, and mostly gate-count-based circuit features. To enable a direct comparison with our GNN-based regression approach, we adapt the MQT Predictor to perform regression on the ESP labels instead of device classification, while retaining its original input features. Thanks to the open-source availability of the tool, this change is straight forward can be implemented by substituting scikit-learn's `RandomForestClassifier` class with its regressor counterpart [32].

Both the GNN and the RF regressor are amenable to hyperparameter optimization. The hyperparameters considered during model optimization, as well as the best-performing instances, are summarized in Appendix A.



(a) IBM Brisbane (b) IBM Marrakesh

Fig. 4: Coupling maps of considered superconducting devices.

B. Target Devices

The target devices considered for ESP prediction comprise two superconducting processors and one trapped-ion processor. This mix allows us to assess whether the proposed ESP regressor remains informative across heterogeneous hardware families rather than within a single technology stack. In the following, we briefly outline the respective instances investigated and the associated hardware parameters.

a) Superconducting QPUs: We consider two different devices, IBM Brisbane and IBM Marrakesh. The former is a 127-qubit Eagle processor, belonging to IBM's previous-generation architecture, while the latter is a 156-qubit Heron r2 processor.

Their coupling maps are shown in Figure 4. The native gate set of IBM Brisbane consists of `[id, sx, rx, ecr, x]`, whereas IBM Marrakesh supports `[id, sx, rz, cz, x]`. Compilation for these architectures largely comprises gate synthesis as well as qubit mapping and routing. For the latter two, the objective is to minimize the number of `SWAP` operations, each of which decomposes into three native entangling gates.

The mean fidelities of the native gates for both devices required for ESP are reported in Table I, grouped into single-qubit and two-qubit gates. Since IBM Brisbane belongs to an older processor generation, its overall fidelities are lower than those of IBM Marrakesh. Nevertheless, it is included in our analysis because the device-selection problem is inherently heterogeneous and the model must remain informative across devices with different topologies and noise characteristics.

b) Trapped Ions QPU: We consider a *Quantum Charge-Coupled Device* (QCCD) [33] architecture, where qubits are distributed across multiple chains in a segmented trap and can be shuttled between chains to achieve full connectivity. This shuttling introduces idle windows beyond those visible in the general circuit picture, which should be accounted for when modeling the single-qubit idle time t^q_{idle} for ESP calculation. To this end, we perform shuttling compilation using the open-source tool *MQT IonShuttler* [34].

The considered device layout is depicted in Figure 5 and consists of a 4×4 grid of interconnected ion traps serving as a memory region, surrounded by four processing traps, that hold up to two ions each and support gates from the trapped-ion native gate set `[rx, ry, rz, rzz]`. Further details on the system model and compilation techniques are provided in Ref. [34]. The exact durations and relative timescales of different gate types and shuttling operations vary across hardware implementations and are subject to steady progress [35]–[38]. For the scope of this work, we assume a hyperfine clock-state qubit encoding with an effectively infinite relaxation time ($T_1 \to \infty$), a dephasing time of $T_2 = 30\,\text{s}$, and fidelities shown in Table I, consistent with current state-of-the-art trapped-ion hardware.

TABLE I: Average gate fidelities for all considered devices.

Device	Operation	Fidelity	Std(Fidelity)
IBM Brisbane	1-qubit gate	0.99288	0.02246
	2-qubit gate	0.98038	0.08313
IBM Marrakesh	1-qubit gate	0.99408	0.02666
	2-qubit gate	0.91944	0.26034
Trapped Ion	1-qubit gate	0.99980	0.00000
	2-qubit gate	0.98960	0.00000

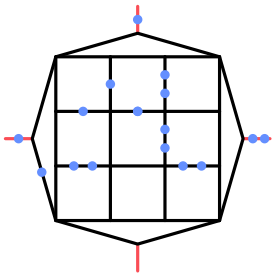


Fig. 5: Considered trapped-ion device layout. Ions (purple) can be shuttled across the 4×4 memory region (black). Gates can be executed only when all participating ions are co-located in the same processing zone (red).

C. Dataset

The dataset is constructed to mirror the intended deployment setting: the model always receives an uncompiled circuit as input, while the output labels encode the best ESP values achievable on each device after backend-specific compilation. Both the GNN and the RF regressor are trained and evaluated on this dataset for the three distinct target devices.

It includes the quantum circuits available in MQT Bench [39] (version 2.0), augmented with 432 variational circuits and 12,000 random circuits. The variational circuits are generated adopting an angle encoding implemented through `RX` and `RY` rotation gates. We employ the ansatz technique proposed in [40], limiting the number of ansatz layers to 10. The random circuits are generated using Qiskit (version 1.4.4) with a circuit depth limited to 100. For all generated circuits, the number of qubits is limited to 15. These limits reflect practical constraints associated with classical computational resources required for compilation and, more importantly, limits on the resulting execution fidelity, which degrades with noise in larger circuit instances.

All circuits are compiled using Qiskit's `transpile` [41], with a varying optimization level between 0 and 3 for each considered quantum device. For the trapped-ion device, an additional shuttling compilation step is applied to account for ion movement. For each circuit-device pair, the ESP is evaluated across all optimization levels, and the maximum value is selected as the target label. This choice is necessary because the highest optimization level does not always produce the highest ESP. This procedure allows us to determine the ESP under the best available compilation strategy, thus providing

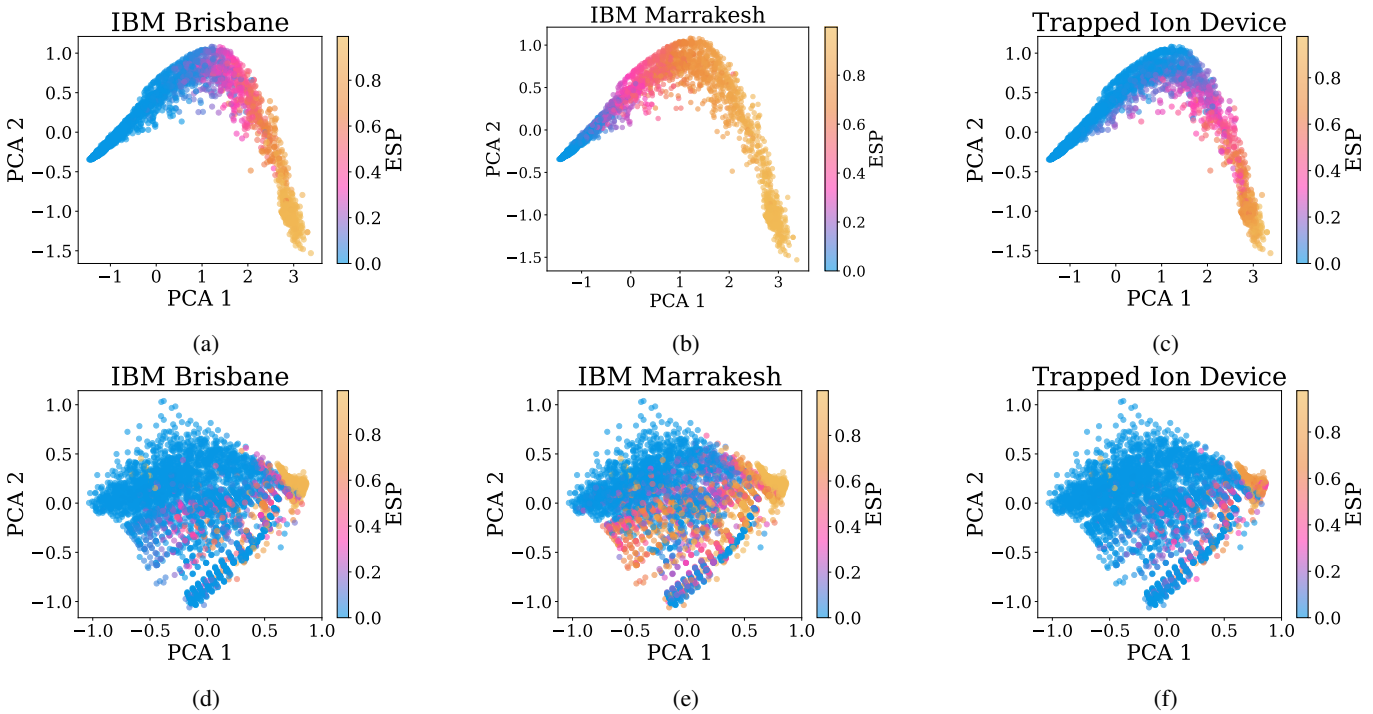


Fig. 6: PCA results for the three considered devices. The top row, panels (a)–(c), shows the latent representations produced by the convolutional component of the GNN model projected onto the first two principal components. The bottom row, panels (d)–(f), shows the feature representations used by the MQT Predictor projected onto the first two principal components.

a measure of the maximum achievable performance on each device once a target QPU has been chosen.

In summary, the resulting dataset consists of numerical representations of uncompiled quantum circuits together with label vectors. Each sample, therefore, pairs a hardware-agnostic circuit description with the optimal ESP values achievable on the target devices. Both approaches are trained on the same dataset (modulo their distinct feature representations) and use identical training, validation, and test split ratios.

VI. Results and Discussion

We investigate in two stages whether the proposed GNN model predicts device-specific ESP accurately enough to enable quantitative device selection before hardware-aware compilation. First, we examine whether the convolutional backbone organizes circuits according to their device-specific ESP values. Second, we quantify how accurately the resulting model predicts these values using standard regression metrics, and compare the results with the RF-based baseline.

A. Principal Component Analysis:

An expressive ML-model for uncompiled circuits, should organize circuits with similar achievable ESP values into nearby regions of the latent space. Accordingly, we assess the quality of the learned convolutional embeddings for both the GNN-based and the RF-based regression models. To this end, we extract the feature vectors generated by the convolutional layers, as well as the RF-features, and apply a Principal Component Analysis (PCA) to reduce their dimensionality to the first two principal components.

The resulting two-dimensional representations are visualized for IBM Marrakesh, IBM Brisbane, and the trapped-ion device in Figure 6, respectively. Each point corresponds to a circuit instance and is color-coded according to its true ESP value. The more closely instances with similar colors are positioned in the projection space, the more effectively the model clusters (and thereby represents) circuits with similar labels. In short, *clusters with similar colors indicate better model performance.*

The GNN learned representations (Figure 6, top) exhibit clear and smooth color gradients, indicating a meaningful organization of circuits according to their ESP values. This behavior indicates that samples with similar ESP values are organized coherently in the embedding space, suggesting that the convolutional component captures meaningful structure for the regression task and preserves the information needed for quantitative device comparison.

In contrast, the PCA projection of the manually engineered RF features (Figure 6, bottom) does not exhibit similarly smooth color gradients or a clear organization with respect to ESP values. This indicates that the relationship between the mostly gate-count-based features and the target metric is less structured in the projected space, making the regression task more challenging.

TABLE II: Regression results of the best performing RF and GNN model.

	Random Forest			Our approach			Improvement		
	MAE $[10^{-2}]$	**MSE** $[10^{-3}]$	R^2	**MAE** $[10^{-2}]$	**MSE** $[10^{-3}]$	R^2	**MAE** [%]	**MSE** [%]	R^2 [%]
IBM Marrakesh	2.394	2.548	0.981667	1.781	1.383	0.990048	25.6	45.7	0.9
IBM Brisbane	1.492	1.852	0.980202	1.255	1.379	0.985263	15.9	25.6	0.5
Trapped Ion Device	1.231	1.236	0.978736	1.059	1.085	0.981336	14.0	12.2	0.3
Average	**1.706**	**1.879**	**0.980201**	**1.365**	**1.282**	**0.985549**	**20.0**	**31.8**	**0.5**

B. Regression Performance:

To assess whether the observed clustering capabilities also translate into quantitative ESP prediction accuracy, we report the Mean Absolute Error (MAE), the Mean Squared Error (MSE), and the coefficient of determination (R^2). Table II summarizes these metrics for the proposed GNN-based approach and for an RF regressor derived from the MQT Predictor.

We begin with the MAE, as it directly reflects the average magnitude of the prediction error and is therefore the most intuitive metric for comparing both approaches. According to this measure, the GNN model clearly outperforms the RF baseline and reduces the average absolute prediction error by 20%. This result indicates that the improved clustering structure of the learned GNN embeddings, observed in the previous PCA, indeed does translate into more accurate quantitative ESP estimates.

A consistent picture emerges from the MSE, which is more sensitive to large deviations because it squares the residuals. Here, the GNN-based approach reduces the error by almost 32% overall, indicating not only lower average error but also a substantially reduced impact of severe mispredictions. This advantage is particularly pronounced for the IBM Marrakesh device, where the MSE decreases by approximately 45%.

The coefficient of determination R^2 remains high, confirming that the GNN model continues to explain a large fraction of the variance in the target values.

Overall, the proposed model predicts ESP more accurately and with fewer severe outliers than the RF-based approach. For the intended multi-QPU workflow, this is particularly important because accurate quantitative ESP estimates make it possible to weigh expected execution quality against secondary system-level factors such as queue length or device availability.

VII. Conclusion

This work introduced a fidelity-based approach for quantum device selection that estimates the execution quality of a given circuit before hardware-aware compilation. In this way, it closes the gap between the point where the scheduling decision is needed and the point where fidelity information traditionally becomes available. Instead of formulating device selection as a purely categorical decision, the proposed method predicts quantitative ESP values for multiple target devices and thereby enables a more informed trade-off between expected execution quality and system-level constraints such as availability or queue length.

Methodologically, the approach is based on a GNN regression model that exploits the natural graph representation of quantum circuits. By operating directly on the circuit DAG, the model learns structural and operational patterns relevant to fidelity prediction without relying on manually engineered features. This makes the framework suitable for early-stage device selection in heterogeneous multi-QPU environments.

The experimental results confirm the effectiveness of this design. Compared with the RF-based baseline derived from the MQT Predictor, the proposed model yields more informative latent representations in the PCA evaluation and achieves consistently better regression accuracy across all considered devices. Overall, it reduces the MSE by about 30% and the MAE by about 20%, demonstrating that graph-based learning provides a clear advantage for pre-compilation fidelity estimation.

Future work will focus on broadening the applicability of the framework. Promising directions include expanding and diversifying the training dataset, incorporating mid-circuit measurements and other dynamic circuit features into the prediction model, and extending the framework beyond fidelity estimation to additional scheduling-relevant metrics such as execution time.

VIII. Acknowledgements

The authors acknowledge funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program grant agreement No. 101001318 and No. 101114305, and the Munich Quantum Valley (MQV K5 + K7), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. Furthermore, this work was supported by the BMFTR under grant numbers 13N17298 and 01MQ25001I, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number 563436708, and the Austrian Research Promotion Agency (FFG) together with the states of Upper Austria and Tyrol under grant number 923923.

While preparing the manuscript, Claude Sonnet 4.6, Claude Opus 4.7, Gemini 3 Pro, and GPT 5.4 were used to improve readability, spelling, grammar, and clarity throughout the manuscript. Each LLM output was reviewed by the authors and edited manually as needed. The authors take full responsibility for the final content.

TABLE III: Hyperparameters and search space used during Optuna-based model selection.

Hyperparameter	Type	Search Space / Value	Optimal value
Hidden dimension	Integer	$[16, 256]$	114
Number of conv. layers	Integer	$[1, 3]$	3
Number of ResNet layers	Integer	$[1, 9]$	4
Dropout rate	Categorical	$\{0.0, 0.1, 0.2\}$	0.0
SAG pooling	Boolean	$\{\texttt{True}, \texttt{False}\}$	`False`
Bidirectional	Boolean	$\{\texttt{True}, \texttt{False}\}$	`True`
Learning rate	Categorical	$\{10^{-4}, 10^{-3}, 10^{-2}\}$	10^{-2}
MLP architecture	Categorical	$(), (32), (64), (128), (256), (512), (64, 32), (128, 32), (128, 64), (256, 32), (256, 64), (256, 128), (512, 256), (512, 128), (512, 64), (512, 32), (128, 64, 32), (256, 128, 64), (512, 256, 128)$	(64,)

Appendix
Hyperparameter Optimization

Both ML models evaluated in this work depend strongly on the choice of hyperparameters. To ensure a fair comparison, we perform automated hyperparameter optimization for both approaches in order to identify their best-performing configurations.

For the GNN-based approach, the tuning procedure is carried out using Optuna [31] over 50 trials. Each candidate configuration is trained with an early-stopping strategy using a patience of 50 epochs and a maximum training budget of 1000 epochs. The optimal hyperparameters obtained during training are reported in the last column of Table III.

The RF regressor derived from the MQT Predictor, originally formulated as a classifier, is optimized using a grid-search procedure, consistent with the methodology adopted in the original work [10] and with common practice for tuning tree-based models. The hyperparameters considered for this baseline are summarized in Table IV.

TABLE IV: RF model parameterization.

Hyperparameter	Type	Search Space	Optimal
Number estimators	Categorical	100, 200, 500	500
Max depth	Integer	[8, 20]	14
Min Samples Split	Integer	[2, 20]	2
Min Samples Leaf	Integer	[2, 20]	2
Bootstrap	Boolean	$\{\texttt{True}, \texttt{False}\}$	True

Correlation Analysis

The GNN model proposed in this work is a single multi-output model that simultaneously predicts ESP scores for all three target devices. This design choice was motivated by the high correlation among the scores, as shown in Table V, which reports the correlation matrix of the target values. Despite the technological differences between the considered QPUs (in particular, the trapped-ion device relies on a fundamentally different physical implementation compared to IBM superconducting processors), the predicted performance metrics remain strongly correlated.

This strong correlation suggests that the ESP is largely driven by intrinsic circuit characteristics, such as the specific ordering of gates and the overall circuit structure, rather than being exclusively determined by hardware-specific features. It further indicates that the same convolutional backbone can be reused to estimate circuit performance on additional quantum devices.

TABLE V: Correlation of ESP scores on evaluated devices.

	IBM Brisbane	IBM Marrakesh	Trapped Ion
IBM Brisbane	1.000000	0.835874	0.952921
IBM Marrakesh	0.835874	1.000000	0.752126
Trapped Ion	0.952921	0.752126	1.000000

References

[1] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *High Performance Computing for Computational Science – VECPAR 2010*, J. M. L. M. Palma, M. Daydé, O. Marques, and J. C. Lopes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–25.

[2] W. J. Dally, S. W. Keckler, and D. B. Kirk, " Evolution of the Graphics Processing Unit (GPU) ," *IEEE Micro*, vol. 41, no. 06, pp. 42–51, Nov. 2021. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MM.2021.3113475

[3] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[4] D. H. Ahn, J. Garlick, M. Grondona, D. Lipari, B. Springmeyer, and M. Schulz, "Flux: A next-generation resource management framework for large hpc centers," in *2014 43rd International Conference on Parallel Processing Workshops*, 2014, pp. 9–17.

[5] S. Zhou, "Lsf: Load sharing in large heterogeneous distributed systems," in *I Workshop on cluster computing*, vol. 136, 1992.

[6] B. Nitzberg, J. M. Schopf, and J. P. Jones, *PBS Pro: Grid computing and scheduling attributes*. USA: Kluwer Academic Publishers, 2004, p. 183–190.

[7] J. Preskill, "Quantum computing in the nisq era and beyond," *Bulletin of the American Physical Society*, vol. 64, no. 9, 2019.

[8] A. Esposito and U.-U. Haus, "Slurm heterogeneous jobs for hybrid classical-quantum workflows," *arXiv preprint arXiv:2506.03846*, 2025.

[9] L. Burgholzer, J. Echavarria, P. Hopf, Y. Stade, D. Rovara, L. Schmid, E. Kaya, B. Mete, M. N. Farooqi, M. Chung, M. De Pascale, L. Schulz, M. Schulz, and R. Wille, "The Munich Quantum Software Stack: Connecting End Users, Integrating Diverse Quantum Technologies, Accelerating HPC," 2025.

[10] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Predictor: Automatic Device Selection with Device-Specific Circuit Compilation for Quantum Computing," *ACM Transactions on Quantum Computing (TQC)*, 2025.

[11] G. Lancellotti, G. Agosta, A. Barenghi, and G. Pelosi, "Improving quantum compilation via circuit partitioning and floorplanning," in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2025, pp. 742–748.

[12] P. Seitz, M. Geiger, C. Ufrecht, A. Plinge, C. Mutschler, D. D. Scherer, and C. B. Mendl, " SCIM MILQ: An HPC Quantum Scheduler ," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Los Alamitos, CA, USA: IEEE Computer Society, Sep. 2024, pp. 292–298. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/QCE60285.2024.10294

[13] E. M. Elmandouh and A. G. Wassal, "Guiding formal verification orchestration using machine learning methods," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 5, Aug. 2018. [Online]. Available: https://doi.org/10.1145/3224206

[14] W. Haaswijk, E. Collins, B. Seguin, M. Soeken, F. Kaplan, S. Süsstrunk, and G. De Micheli, "Deep learning for logic optimization algorithms," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–4.

[15] L. Servadei, E. Mosca, E. Zennaro, K. Devarajegowda, M. Werner, W. Ecker, and R. Wille, " Accurate Cost Estimation of Memory Systems Utilizing Machine Learning and Solutions from Computer Vision for Design Automation ," *IEEE Transactions on Computers*, vol. 69, no. 06, pp. 856–867, Jun. 2020. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TC.2020.2968888

[16] M. U. Jamal, Z. Li, M. T. Lazarescu, and L. Lavagno, "A graph neural network model for fast and accurate quality of result estimation for high-level synthesis," *IEEE Access*, vol. 11, pp. 85 785–85 798, 2023.

[17] A. Agnesina, K. Chang, and S. K. Lim, "Vlsi placement parameter optimization using deep reinforcement learning," in *Proceedings of the 39th International Conference on Computer-Aided Design*, ser. ICCAD '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3400302.3415690

[18] K. Baek, H. Park, S. Kim, K. Choi, and T. Kim, "Pin accessibility and routing congestion aware drc hotspot prediction using graph neural network and u-net," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3508352.3549346

[19] A. Paler, L. Sasu, A.-C. Florea, and R. Andonie, "Machine learning optimization of quantum circuit layouts," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, Feb. 2023. [Online]. Available: https://doi.org/10.1145/3565271

[20] H. Fan, C. Guo, and W. Luk, "Optimizing quantum circuit placement via machine learning," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 19–24. [Online]. Available: https://doi.org/10.1145/3489517.3530403

[21] M. Weiden, E. Younis, J. Kalloor, J. Kubiatowicz, and C. Iancu, "Improving quantum circuit synthesis with machine learning," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 1–11.

[22] V. Saravanan and S. M. Saeed, "Noise adaptive quantum circuit mapping using reinforcement learning and graph neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 5, pp. 1374–1386, 2024.

[23] ——, "Data-driven reliability models of quantum circuit: From traditional ml to graph neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 5, pp. 1477–1489, 2023.

[24] G. Kizhuvettil, G. Ioannou, S. Niu, and S. M. Saeed, "Physically-aware qubit reuse for optimized quantum circuit compilation using machine learning," in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2025, pp. 720–730.

[25] T. LeCompte, F. Qi, X. Yuan, N.-F. Tzeng, M. H. Najafi, and L. Peng, "Graph neural network assisted quantum compilation for qubit allocation," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 415–419. [Online]. Available: https://doi.org/10.1145/3583781.3590300

[26] P. Hopf, N. Quetschlich, L. Schulz, and R. Wille, "Improving figures of merit for quantum circuit compilation," in *2025 Design, Automation and Test in Europe Conference (DATE)*. IEEE, Mar. 2025, p. 1–7. [Online]. Available: http://dx.doi.org/10.23919/DATE64628.2025.10992761

[27] ——, "Improving figures of merit for quantum circuit compilation," in *2025 Design, Automation & Test in Europe Conference (DATE)*, 2025, pp. 1–7.

[28] A. Tudisco, D. Volpe, G. Orlandi, and G. Turvani, "Graph neural network-based predictor for optimal quantum hardware selection," in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 02, 2025, pp. 296–301.

[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1025–1035.

[30] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. M. Gambetta, and B. R. Johnson, "Openqasm3: A broader and deeper quantum assembly language," *ACM Transactions on Quantum Computing*, vol. 3, no. 3, Sep. 2022. [Online]. Available: https://doi.org/10.1145/3505636

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[32] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[33] D. Kielpinski, C. Monroe, and D. J. Wineland, "Architecture for a large-scale ion-trap quantum computer," *Nature*, vol. 417, no. 6890, pp. 709–711, Jun. 2002.

[34] D. Schoenberger, S. Hillmich, M. Brandl, and R. Wille, "Shuttling for Scalable Trapped-Ion Quantum Computers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 6, pp. 2144–2155, Jun. 2025.

[35] T. P. Harty, D. T. C. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, "High-Fidelity Preparation, Gates, Memory, and Readout of a Trapped-Ion Quantum Bit," *Physical Review Letters*, vol. 113, no. 22, p. 220501, Nov. 2014.

[36] A. Kreuter, C. Becher, G. P. T. Lancaster, A. B. Mundt, C. Russo, H. Häffner, C. Roos, W. Hänsel, F. Schmidt-Kaler, R. Blatt, and M. S. Safronova, "Experimental and theoretical study of the $3d\phantom{\rule{0.3em}{0ex}}^{2}D$–level lifetimes of $^{40}\mathrm{Ca}^{+}$," *Physical Review A*, vol. 71, no. 3, p. 032504, Mar. 2005.

[37] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. G. Bohnet, N. C. Brown, N. Q. Burdick, W. C. Burton, S. L. Campbell, J. P. Campora, C. Carron, J. Chambers, J. W. Chan, Y. H. Chen, A. Chernoguzov,

E. Chertkov, J. Colina, J. P. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. M. Dreiling, C. T. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Foltz, M. Foss-Feig, D. Francois, J. P. Gaebler, T. M. Gatterman, C. N. Gilbreth, J. Giles, E. Glynn, A. Hall, A. M. Hankin, A. Hansen, D. Hayes, B. Higashi, I. M. Hoffman, B. Horning, J. J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. T. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. B. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. N. Price, Z. Price, M. Pugh, A. Ransford, A. P. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. T. Sprenkle, R. P. Stutz, M. Swallows, R. I. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. M. Zolot, and J. M. Pino, "A Race-Track Trapped-Ion Quantum Processor," *Physical Review X*, vol. 13, no. 4, p. 041052, Dec. 2023.

[38] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, Aug. 2016.

[39] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing," *Quantum*, vol. 7, p. 1062, 2023, MQT Bench is available at https://mqt-bench.app/.

[40] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019. [Online]. Available: https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900070

[41] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. [Online]. Available: https://arxiv.org/abs/2405.08810