Graph Transformer for Quantum Circuit Reliability Prediction

Hanrui Wang[1], Pengyu Liu[2], Jinglei Cheng[3], Zhiding Liang[4], Jiaqi Gu[5], Zirui Li[6], Yongshan Ding[7], Weiwen Jiang[8], Yiyu Shi[4], Xuehai Qian[3], David Z. Pan[5], Frederic T. Chong[9], Song Han[1]
[1]MIT [2]Tsinghua Univ. [3] Purdue Univ. [4]Univ. of Notre Dame [5]Univ. of Taxes at Austin [6]Rutgers Univ. [7]Yale Univ.
[8]George Mason Univ. [9]Univ. of Chicago

Abstract

Quantum Computing has attracted much research attention because of its potential to achieve fundamental speed and efficiency improvements in various domains. Among different quantum algorithms, Parameterized Quantum Circuits (PQC) for Quantum Machine Learning (QML) show promises to realize quantum advantages on the current Noisy Intermediate-Scale Quantum (NISQ) Machines. Therefore, to facilitate the QML and PQC research, a recent python library called TorchQuantum has been released. It can construct, simulate, and train PQC for machine learning tasks with high speed and convenient debugging supports. Besides quantum for ML, we want to raise the community's attention on the reversed direction: ML for quantum. Specifically, the TorchQuantum library also supports using data-driven ML models to solve problems in quantum system research, such as predicting the impact of quantum noise on circuit fidelity and improving the quantum circuit compilation efficiency.

This paper presents a case study of the ML for quantum part in TorchQuantum. Since estimating the noise impact on circuit reliability is an essential step toward understanding and mitigating noise, we propose to leverage classical ML to predict noise impact on circuit fidelity. Inspired by the natural graph representation of quantum circuits, we propose to leverage a *graph transformer* model to predict the noisy circuit fidelity. We firstly collect a large dataset with a variety of quantum circuits and obtain their fidelity on noisy simulators and real machines. Then we embed each circuit into a graph with gate and noise properties as node features, and adopt a graph transformer to predict the fidelity. We can avoid exponential classical simulation cost and efficiently estimate fidelity with polynomial complexity.

Evaluated on 5 thousand random and algorithm circuits, the graph transformer predictor can provide accurate fidelity estimation with RMSE error **0.04** and outperform a simple neural network-based model by **0.02** on average. It can achieve **0.99** and **0.95** R^2 scores for random and algorithm circuits, respectively. Compared with circuit simulators, the predictor has over **200**× speedup for estimating the fidelity. The datasets and predictors can be accessed in the TorchQuantum library.

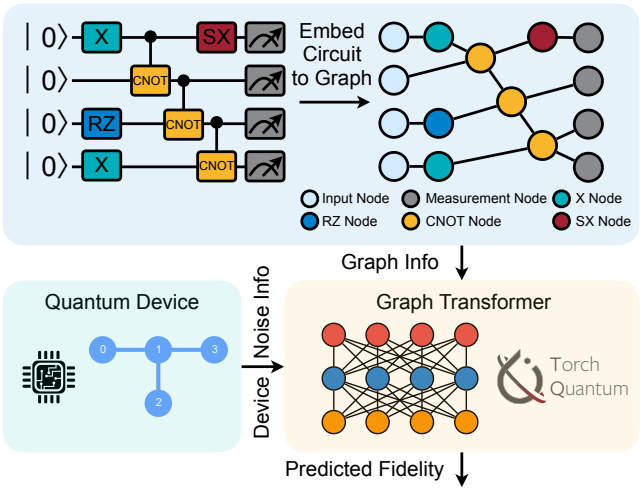

Figure 1: The proposed fidelity prediction framework. The quantum circuit is firstly embedded into a graph in which the nodes are gates and edges are execution orders. The feature vector on each node contains the device noise information, such as gate error rates. The graph is processed by a graph transformer in TorchQuantum to estimate circuit fidelity.

1 Introduction

Quantum Computing (QC) presents a new computational paradigm that has the potential to address classically intractable problems with much higher efficiency and speed. It has been shown to have an exponential or polynomial advantage in various domains such as combinatorial optimization [11], molecular dynamics [26, 34], and machine learning [3, 7, 15, 20, 27, 28, 37, 56], etc. By virtue of breakthroughs in physical implementation technologies, QC hardware has advanced quickly during the last two decades. Multiple QC systems with up to 127 qubits have been released recently [14, 18, 22, 38].

Despite the promising developments, it is still anticipated that before we enter the fault-tolerant era, we will spend a number of years in the Noisy Intermediate Scale Quantum (NISQ) [35] stage. In this stage, the qubits and quantum gates suffer from significant error (around 10^{-3}), which is the bottleneck towards quantum advantages. Therefore, Parameterized Quantum Circuits (PQC) have attracted increasingly more attention thanks to their flexibility in the circuit architecture (ansatz) and parameters that provides vast space for noise mitigation and optimizations.

To facilitate the robust quantum circuits, especially parameterized quantum circuits for quantum machine learning, the TorchQuantum library is released, which supports easy construction, simulation, and fast parameter training of PQCs. Several noise mitigation techniques, such as noise-aware ansatz search [48], noise-aware parameter training [49], gradient pruning for robust on-chip training [50], are also supported in the library.

Although plenty of work has been focusing on quantum for machine learning with parameterized circuits, little research explores another direction – using machine learning to solve quantum system research problems. To fill this vacancy, the TorchQuantum library also provides multiple classical machine learning models to perform quantum compilation, reliability estimation tasks, etc.

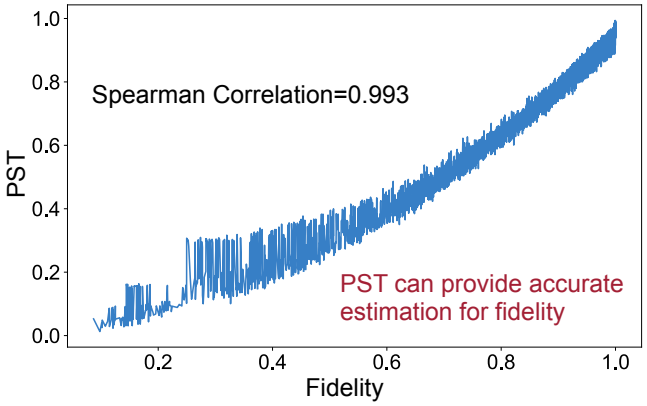

Figure 2: Relationship between fidelity and PST of random circuits. The PST of a circuit is obtained by appending the inverse circuit to the original one and executing. There is a strong positive correlation (Spearman = 0.993) between the two metrics, so it is sufficient for the predictor to output PST.

In this paper, we show one case study of machine learning for quantum – using graph transformer models to estimate the quantum circuit fidelity under noise impact, as shown in Figure 1. Due to the limited quantum resources, it is highly desirable to estimate the circuit performance before submitting it for execution. If the fidelity of a circuit is lower than a threshold, running it on real quantum machines will not generate any meaningful result. One straightforward method is to perform circuit simulation on noisy simulators, but the exponentially increasing cost is prohibitive for circuits with many qubits. Therefore, in this work, we propose a polynomial complexity method in which a data-driven graph transformer is trained to perform fidelity estimation. Intuitively, estimating the fidelity does not require precisely computing the complete density matrix. So there are opportunities that the data-driven method can provide accurate enough estimation with low computation costs. In fact, there have been works on predicting circuit reliability using simple machine learning models [30]. However, it considers neither any graph information of the circuit nor the noise information and thus has less accurate predictions in experimental results.

The first step of the framework is to collect a large dataset containing various randomly generated circuits and circuits from common quantum algorithms. We run the circuits on both noisy simulators and real quantum machines. On simulators, we change the properties of the qubits, such as T1 and T2, and the error rates of gates to diversify the data samples. The dataset contains over 20 thousand samples on simulators and 25 thousand samples on real quantum machines. In order to reduce the overhead of collecting a dataset, we use the "Probability of Successful Trials" (PST) [43] as the proxy for the fidelity following the setting in [30]. Specifically, for each circuit, we will concatenate the inverse of the circuit to the original one and execute. Since the original quantum state is all zero, the ground truth output of the concatenated circuit will still be all zero. Therefore, the PST will be the frequency of getting all zero bit-string. The dataset is embedded in the TorchQuantum library and can be easily accessed for future studies.

Secondly, motivated by the fact that *quantum circuits are graphs*, we propose to leverage a graph transformer to process the circuit information. The nodes of the graph are the quantum gates, input qubits, and measurements. The edges are determined by the sequence of gate executions. The feature vector on each node contains gate type, qubit index, qubit T1, T2 time, gate error rate, etc., to capture operation and noise information. In one layer of the graph transformer, the attention layer will capture the correlations between each node and its neighbors according to the graph and compute the updated feature vector. Several fully-connected layers are appended at the end to regress the circuit PST.

Overall, we present a case study on using graph transformer models in the TorchQuantum library to estimate circuit fidelity under noise. The contributions are summarized as below:

- **A dataset for circuit fidelity** on various noisy simulators and real machines is presented and embedded in the TorchQuantum library to facilitate research on reliability estimations. It contains 20K simulation samples and 25K real machine samples.
- **A graph transformer model** is constructed and trained to process the quantum circuit graph and feature vectors on nodes to provide accurate fidelity prediction.
- **Extensive evaluations** on around 2 thousand circuits on noisy simulators and 3 thousand circuits on real machines demonstrate the high accuracy of the predictor. It achieves **0.04** RMSE and over **0.95** R^2 scores with **200**× speedup over circuit simulators.

2 Related Work

2.1 Quantum Basics

A quantum bit (qubit) can be in a linear combination of the two basis states 0 and 1, in contrast to a classical bit, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, for $\alpha, \beta \in \mathbb{C}$, where $|\alpha|^2 + |\beta|^2 = 1$. Only one of the 2^n states can be stored in a classical n-bit register. However, we can employ an n-qubit system to describe a linear combination of 2^n basis states due to the ability to build a superposition of basis states. To perform computation on a quantum system, we use a *quantum circuit* to manipulate the state of qubits. A given quantum system can be expressed as a Hamiltonian function and solved by Schrödinger's equation, and these operational steps can be performed by various *quantum gates*. Results of a quantum circuit are obtained by qubit readout operations called *measurements*, which collapse a qubit state $|\psi\rangle$ to either $|0\rangle$ or $|1\rangle$ probabilistically according to the amplitudes α and β.

2.2 Quantum Errors

Quantum errors are one of the most significant challenges that NISQ-era quantum computing experiences. On real quantum machines, errors occur because of the interactions between qubits and the environment, control errors, and interference from the environment [4, 24, 32]. Qubits undergo *decoherence error* over time, and quantum gates introduce *operation errors* (such as coherent/stochastic errors) into the system. These systems need to be characterized [32] and calibrated [19] frequently to mitigate the quantum noise impacts.

The errors seriously interfere with the function of quantum circuits and form obstacles to further optimization of quantum circuits.

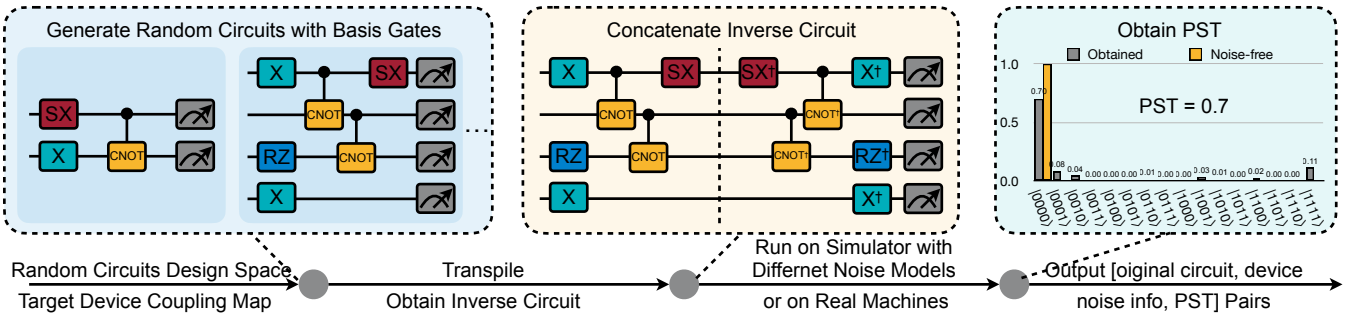

Figure 3: Overview of the dataset generation process. i) Prepare random circuits by mixing basis gate: `RZ`, `SX`, `X`, `CNOT`, namely constructing native circuits. ii) Inverse all the gates in the transpiled native circuit and then concatenate the inverse circuit to the original transpiled native circuit. iii) Calculate PST by dividing the number of trials (shots) with all zero state by the number of total trials (shots).

A number of noise mitigation techniques have been developed to attenuate negative effects [9, 16, 25, 28, 36, 48, 49]. [49] proposes a framework to improve the quantum circuits' robustness by making them aware of noise. It consists of three main techniques: injection of gate errors, regularization, and normalization of measurement outcomes. Another literature [28] integrates the gate error characteristics into the mapped quantum circuit to improve robustness.

2.3 Fidelity Estimation and Prediction

In order to validate and characterize the states generated by a quantum computer, it is crucial to estimate the fidelity of quantum states [12, 55]. However, calculating fidelities is already quite computationally expensive. Numerous efforts have been made to address this problem in the past few years. Variational quantum algorithms have been adopted by recent works to perform fidelity estimation [5, 6, 41]. Machine learning-based and statistical methods are also proposed to estimate the fidelity [30, 57, 59]. In addition, "classical shadow" is proposed for more efficient tomography [17], which can also benefit fidelity estimation. The works mentioned above present various methods for estimating fidelity. Fewer works, however, have focused on predicting fidelity given a quantum circuit and a noisy backend. [30] derives a fidelity prediction model using polynomial fitting and a shallow neural network. The noisy backend is considered as a black box in that work. [33, 42] calculate fidelity with a simple equation and use it as a metric to optimize the compilation workflow. These methods are inaccurate and do not account for the structure of quantum circuits or noisy backends.

2.4 Randomized Benchmarking

Plenty of techniques have been developed to estimate the fidelity of quantum circuits and identify errors in NISQ computers, and they can provide indicators of the quality of quantum circuits and directions for further improvement of quantum hardware. Among them, randomized benchmarking is the most prominent [23, 31, 32] one. Randomized benchmarking can estimate the fidelity of certain gates or circuits and further characterize noises to very high accuracy in the presence of state preparation and measurement errors. However, randomized benchmarking has several limitations. For example, it usually requires strong assumptions about the error pattern, such as assuming the errors are gate-independent, and the benchmarked gate set must have group structures.

2.5 Transformers

The attention [1, 45] based Transformer models [47, 54] have prevailed in sequence modeling. Recently, it is also widely applied in other domains such as vision transformer [10] for computer vision and graph transformer (graph attention networks) [46, 51, 53, 58] for graph learning. The graph transformer leverages the attention mechanism to generate the updated features of the next layer for each node. The Query vectors come from the center node, while the Key and Value vectors are calculated from the neighboring nodes. Recently, several variants of traditional transformers have been proposed, including AGNN, which removes all the FC linear layers in the model [44], Modified-GAT [39], which proposes gate-augmented attention for better feature extraction, Linear Attention [40], which reduces the complexity of attention to linear cost, and Hardware-Aware Transformer [52] that adjusts the architecture according to the hardware latency feedback.

3 Circuit Fidelity Dataset

In classical computing, training datasets must be fed into the machine learning algorithms before validation datasets (or testing datasets) can be employed to validate the model's interpretation of the input data. However, when dealing with the fidelity prediction problem, we do not have an off-the-shelf dataset that can be used to train and evaluate different methods. To address this problem, we present a scheme for generating datasets and incorporating the gathered datasets into TorchQuantum in order to provide relevant researchers with appropriate starting points.

3.1 Metrics

In order to accurately estimate the "success rate" of quantum circuits on noisy devices, the conception of fidelity is introduced, which is a measure of the "closeness" of two quantum states. In noisy quantum computing, fidelity is adopted to illustrate the difference between the quantum states generated by noisy devices and those generated by noiseless classical simulations. Obtaining the fidelity of quantum circuits is, however, computationally costly – exponential to the qubit number. Intricate tomography would be required to "restore"

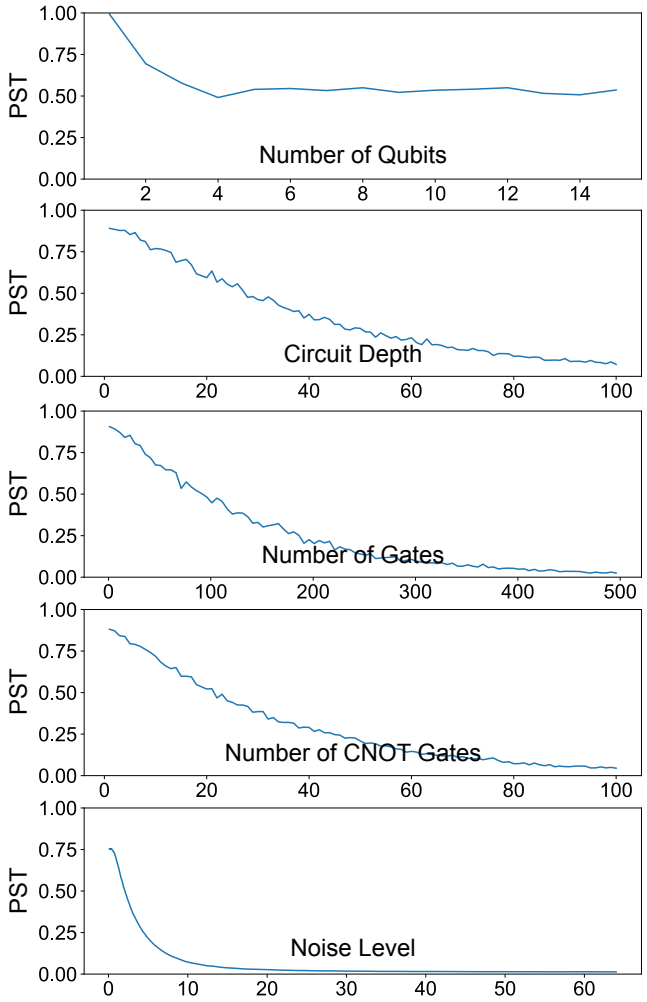

Figure 4: Dataset property profiling.

or "describe" quantum states[17]. To solve such a problem, we adopt the idea of "Probability of Successful Trials" (PST) [43] as the proxy of fidelity.

$$PST = \frac{\#Trials\ with\ output\ same\ as\ initial\ state}{\#Total\ trials}$$

Instead of measuring the fidelity of quantum circuits, we count the proportion of unchanged qubits (all zeros) after concatenating the circuits with their inverse. For concatenated circuits, the proportion will be one if we conduct simulations on a noise-free simulator. We compare the PST with fidelity for 1400 quantum circuits on simulators. As shown in Figure 2, they exhibit a strong correlation with a Spearman correlation coefficient of 0.993. Therefore, we can conclude that PST can provide accurate fidelity estimations.

3.2 Dataset Generation

As shown in Figure 3, the generation of random datasets can be broken down into three major steps: initial random circuit generation, concatenation with inverse circuits, and PST calculation.

Native Circuit Construction. In the first step, random gates are generated from the basis gate set {RZ, SX, X, CNOT} and assigned to quantum circuits to create an initial version of random circuits. Single-qubit gates are assigned to all possible qubits, and two-qubit gates are assigned to all available connections in the quantum device. After finishing the assignments, the circuits will be compiled to eliminate duplicated gates. As a result, we consider the number of qubits and gates, the coupling map of quantum devices, and the number of random circuits as parameters during the random circuits generation process.

Concatenation of Inverse Circuit. Furthermore, the obtained random circuits will be concatenated with their inverse. The inverse circuit is obtained by reversing the gate sequence of the original circuit and replacing each gate with its inverse gate, as shown in Figure 3 middle. The purpose of concatenation is to use PST rather than fidelity as our metrics, thereby allowing us to avoid the computationally expensive state tomography. The detailed reasons are elaborated on in the Section 3.1. For example, assuming a circuit consisting of a CNOT gate and an X gate, after concatenation, the circuit will be "CNOT + X + barrier + X + CNOT". A barrier is placed to prevent gate cancellation. The concatenated circuits will be sent to the backends to obtain PSTs. Note that the dataset only contains the original circuits *without* concatenation. As a result, if we need to evaluate a new quantum circuit, we will feed it to the ML predictor. Then the predicted PST for the concatenated circuit will be returned by the ML model, which is highly correlated with the circuit's fidelity.

PST Calculation. The concatenated circuits will then be passed to noisy backends to calculate the PST. We begin with the default initial state $|00....0\rangle$, and the PST represents the proportion of $|00....0\rangle$ in the output distribution. Our prediction model takes into account the information from both quantum circuits and noisy backends. As a result, the quantum circuits are simulated on backends with differing noise levels to create our datasets. The backends' noise configurations are derived from real NISQ machines, with random constants to change the noise levels.

3.3 Dataset Properties

Figure 4 depicts the relationships between PST and various circuit and backend properties. We can anticipate a lower PST as the number of gates increases. The PST numbers are also influenced by the number of CNOT gates, circuit depth, and noise level. To cover these dimensions, we create random datasets with varying numbers of qubits, gates, and backends of different noise levels. The PSTs of these circuits are simulated on backends with five different noise levels. As a result, the random circuits datasets contain 10000 data points on noisy simulators. We also measure the PSTs of these circuits from five different real NISQ machines. The dataset on real machines contains around 25000 data points. The performance of our graph transformer model on random circuits is demonstrated in Figure 7. In addition, our datasets include circuits used in quantum algorithms such as quantum error correction [29], variational quantum eigensolver [21], Grover search [13], quantum fourier transform [8], quantum approximate optimization algorithm [11] and quantum teleportation [2]. We select a total of 30 circuits derived from quantum algorithms. The simulations are also carried out on noisy simulators with varying noise levels to collect data

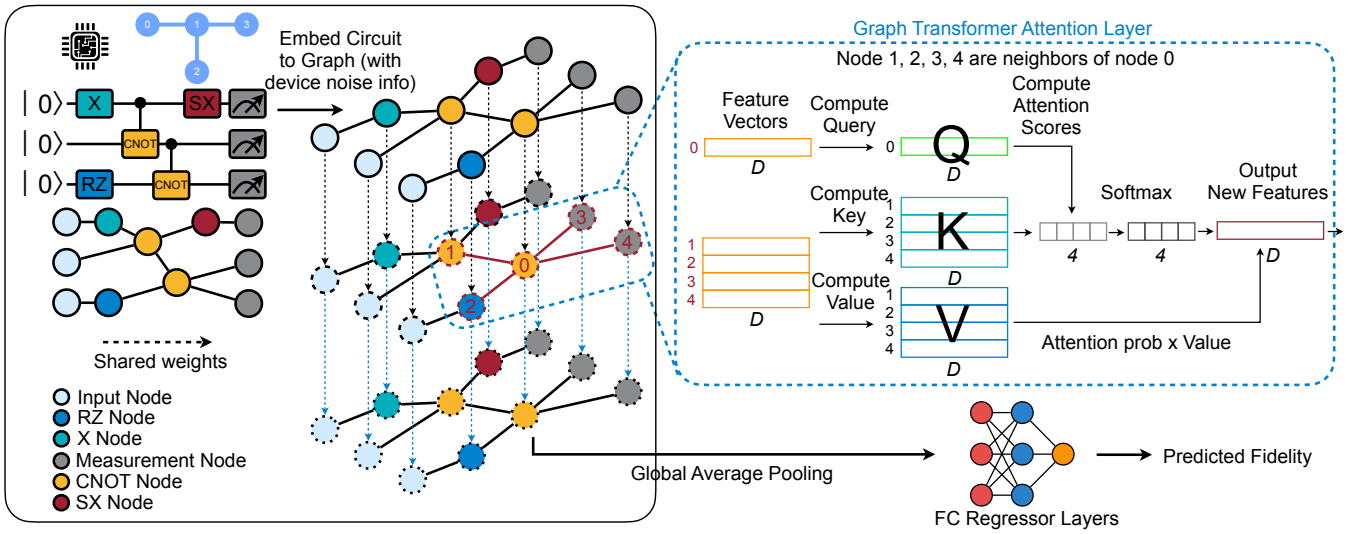

Figure 5: Overview of the Graph Transformer for fidelity prediction. (i) Generate the graph according to quantum circuit, and then generate the feature vector for each of the node according to the quantum device noise information. (ii) For one Graph Transformer layer, we perform graph attention layer to extract information and captures the neighboring correlations. The weight matrices are shared across all nodes. (iii) Finally, a regressor containing several FC layers regresses the circuit PST (an approximation of fidelity).

Algorithm 1: Attention in Graph Transformer

Input: Circuit graph: G with K nodes
Length of feature vector: D
Node features: $\mathbf{H} \in \mathbb{R}^{K \times D}$
Query, Key, Value weights $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V\} \in \mathbb{R}^{D \times D}$
$\mathbf{Q} = \mathbf{W}_Q \cdot \mathbf{H}$
$\mathbf{K} = \mathbf{W}_K \cdot \mathbf{H}$
$\mathbf{V} = \mathbf{W}_V \cdot \mathbf{H}$
do in parallel
  for $i = 0$ **to** K **do**
    Obtain neighbor nodes $\mathcal{N}_i$ according to G
    $attention\_score_{ij} = \mathbf{Q}_i \cdot \mathbf{K}_j^T, j \in \mathcal{N}_i$
    attention_score = **attention_score**/sqrt($|\mathcal{N}_i|$)
    attention_prob = Softmax(**attention_score**)
    $\mathbf{attention\_out}_i = \sum_{j \in \mathcal{N}_i} attention\_prob_{ij} \cdot V_j$
    $\mathbf{attention\_out}_i \in \mathbb{R}^D$
  end
end
Output: **attention_out** $\in \mathbb{R}^{K \times D}$

points. The performance of our graph transformer model on these circuits is demonstrated in Figure 8.

4 Predictor

The dataset introduced in the previous section enables a data-driven approach to learning the PST from circuit and noise features. This section will continue to present a case study of a deep learning model, graph transformer, for circuit PST prediction. Figure 5 shows the overview of the framework. A gate graph is firstly extracted from the circuit. Then the node features are generated according to the gate type, noise information, etc. Next, a graph transformer containing attention operations is introduced to process the node features and neighboring relations. Finally, a PST regression layer outputs the predicted values.

4.1 Graph Construction

We firstly use directed acyclic graphs (DAG) to represent the topology of quantum circuits. Each node represents one qubit, quantum gate, or measurement. Edges represent the time-dependent order of different gates. One example of extracting the graph from the circuit is presented on the left of Figure 5. The connectivity can be encoded into an adjacent matrix. With the TorchQuantum framework, the DAG can be conveniently converted from the circuit.

4.2 Node Features

For each node in the graph, we generate a vector representing the features. The features include gate type, target qubit index, T1 and T2 of the target qubit, gate error, and gate index, as shown in Figure 6. In our experiments, we set the maximum qubit number to 10, and the feature vector has a length of 24. The first 6 numbers are one-hot vectors describing the gate type: initial input qubit, measurement, `RZ`, `X`, `SX`, or `CNOT`. Then we use 10 numbers to describe the target gate qubit(s). If this gate acts on the i^{th} qubit, the i^{th} number of the vector is set to 1 and otherwise 0. That also applies to multi-qubit gates. Then we use the following 7 numbers to describe the calibration information of the backend with the following format: [T1, T2 for the first target qubit, T1, T2 for the second target qubit, gate error rate, readout error10, readout error01]. If a feature is not applicable for a particular node, the corresponding value is set to 0. For example, `RZ` acts on only one qubit, so T1 and T2 for the second target qubit are set to 0. Since `RZ` is not a measurement, readout error10 and readout error01 are set to 0 also. The last number is used to encode the index of the node. The whole featur vector is illustrated in Figure 6.

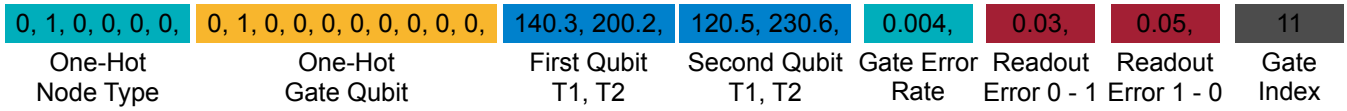

Figure 6: Node feature vector.

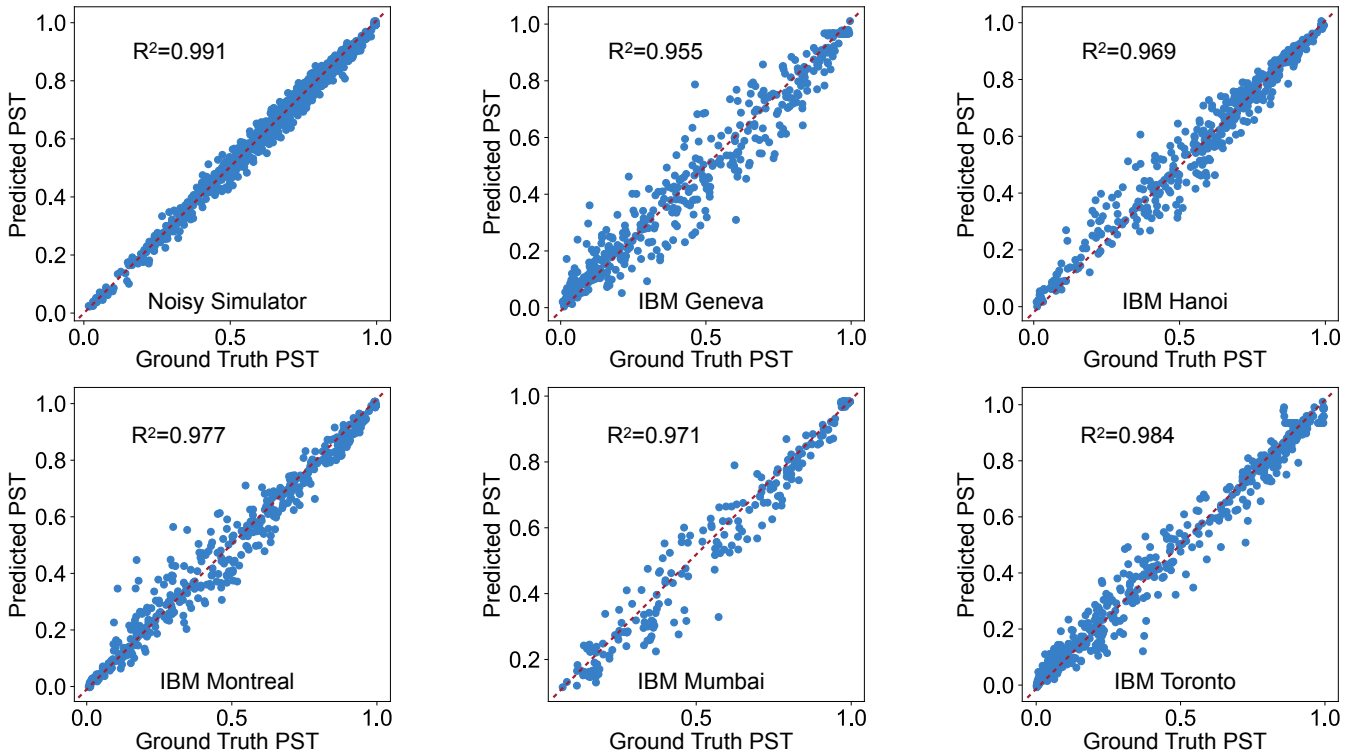

Figure 7: Scatter plots of PST of randomly generated circuits on noisy simulators and 5 real machines. Our transformer can provide accurate estimations of PST with R^2 higher than 0.95.

4.3 Graph Transformer

To process graphs with node features, we propose to use a graph transformer as shown in Figure 5 right. The transformer contains multiple layers, each containing the attention operation. The attention is described in Algorithm 1. the Query, Key, and Value vectors for each node are computed with shared weights. Then for one node, we fetch the Key vectors of its neighboring nodes and compute Query $\times$ Key^T. The outputs are attention scores which are then normalized according to the square root of the number of neighbors. Softmax is adopted to normalize the attention scores. The output is called attention probability because the values add up to one. The probability vector is then employed as weights to perform a weighted sum of the Value vectors of the neighboring nodes. The output has the same dimension as the input feature of the center node. After that, we perform a residual connection between input and output of attention with a layer normalization. The output will be the feature vector of the next layer. Note that computations on all nodes are done *simultaneously*.

After multiple transformer layers, we obtain a learned feature on each node, with its neighbors influenced. If deep enough, each node can access to features of all nodes in the graph. Finally, we perform a global average pooling of the node features and obtain an aggregated node feature vector. Then a regressor with three FC layers is appended to output the final regressed PST. Besides node feature, we also leverage *global features*, representing the circuit depth, width, and counts of `RZ`, `X`, `SX`, and `CNOT` gates. The global feature vector is concatenated with the aggregated node feature vector and fed to the regressor.

The computational complexity of the proposed graph transformer is polynomial to qubit number since the overall number of gates is typically polynomial to qubit number.

5 Evaluation

5.1 Evaluation Methodology

Model and Training Setups. In the default setup, we use two layers of graph transformers. The embedding dimension is 24 since we have 24 features. The dimension for the Query, Key, and Value

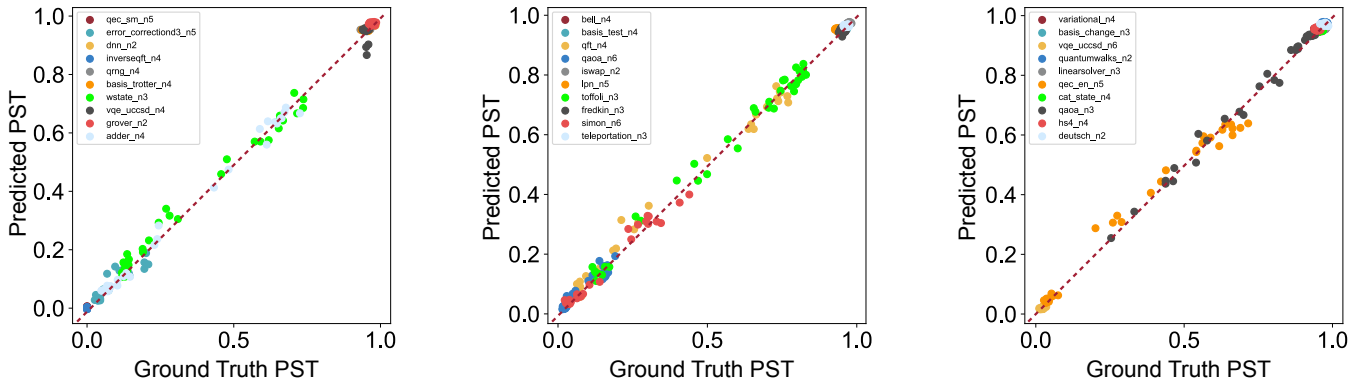

Figure 8: Scatter plots of circuit PST of 30 quantum algorithms on noisy simulators. Our transformer can provide accurate estimations of PST with R^2 0.99.

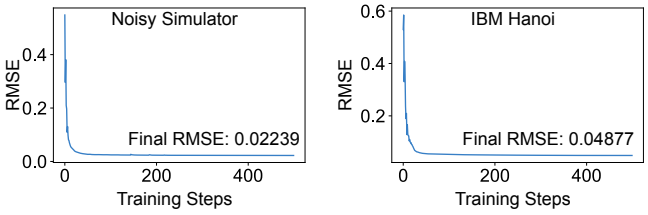

Figure 9: Training curves of transformer models on noisy simulators and IBM Hanoi datasets for random circuits.

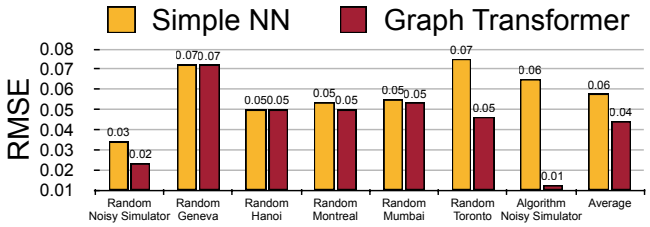

Figure 10: The proposed graph transformer-based model can outperform the simple NN model on various benchmarks.

Table 1: Prediction RMSE vs. Whether Using Global Features

Features	Noisy Simulator	IBM Geneva	IBM Hanoi
w/o Global Features	0.0239	0.0757	0.0506
w/ Global Features	**0.0232**	**0.0723**	**0.0500**

vectors is also 24. We use single-head attention layers. The global average pooling across nodes generates a single 24 dimensional vector as the aggregated feature for a circuit. If global features are enabled, we use two FC layers with hidden and output dimensions of 12 to pre-process and concatenate it with the aggregated node feature. The concatenated feature is processed with additional three FC layers with hidden dimension 128 and output dimension 1. This output is treated as the predicted PST value. We use ReLU activation.

We normalize the node features across the dataset by removing the mean and dividing the standard deviation. We then train the models with Adam optimizer for 500 epochs with a constant learning rate of 10^{-2}, weight decay 10^{-4}, batch size 2500 and MSE loss. Then we choose the model that performs best on the validation set to test on the test set.

Dataset Setup. For noisy simulators datasets, we have 10000 random circuits and 350 circuits for 30 quantum algorithms each. For real machine datasets, we collect 5000, 5000, 5450, 2750, and 6750 random circuits for IBM Geneva, IBM Hanoi, IBM Montreal, IBM Mumbai, and IBM Toronto, respectively. We split the dataset into three parts, the training set includes 70% data, the validation set includes 20% data, and the test set consists of the last 10%.

5.2 Experimental Results

Figure 7 shows the scatter plots of transformer predicted PST vs. the ground truth PST for randomly generated circuits on the test set. The red dash line is the $y = x$ line. We train one separate model for each of the backend settings. For results on noisy simulators, the points are close to the $y = x$ line with an R^2 value of 0.991. On real machines, the difficulty is greater than on noisy simulators. Although the predictor R^2 is lower than noisy simulators, they are still higher than 0.95. Furthermore, as in Figure 8, we select 30 representative quantum algorithms as benchmarks and show the scatter plots for predicted PST on the test set. Each color represents one algorithm circuit under different noise models. We train one common model for the 30 algorithm circuits. The transformer model can effectively track the PST value, especially for those spanning a wide range of PST. The overall R^2 for 30 benchmarks is 0.9985. We also show the two representative training curves on noisy simulators and the real quantum machine IBM Hanoi in Figure 9. The training loss converges after around 200 steps. The convergence speed on real machine data is slightly slower than the noisy simulator data and has a higher final RMSE (around 0.05).

Besides, we also compare our transformer-based model with the simple NN model adapted from [30] as in Figure 10. The simple NN model only takes 116 features as input, which include circuit depth, width, and counts of `RZ`, `X`, `SX`, and `CNOT` gates, single-qubit gate counts on each qubit, and two-qubit gate counts on each qubit pair. It uses 3 FC layers with hidden dimension 128 and ReLU activation to regress the PST. We compare the RMSE on the test set for random circuits on 6 benchmarks and 30 algorithm circuits on

Table 2: Importance Comparison of Node Features

Features	Noisy Simulator	IBM Geneva	IBM Hanoi
All Features	0.0232	0.0723	0.0500
w/o Gate Error Rate	0.0235	0.0732	0.0501
w/o Gate Index	**0.0247**	0.0730	0.0497
w/o Gate Type	0.0236	**0.0742**	**0.0512**
w/o Qubit Index	**0.0239**	**0.0736**	**0.0514**
w/o T1&T2	0.0239	0.0707	0.0491

Table 3: Prediction RMSE vs. Transformer Layer Number

# Layers	Noisy Simulator	IBM Geneva	IBM Hanoi
1	**0.0230**	0.0720	**0.0491**
2	0.0232	0.0723	0.0500
3	0.0232	**0.0719**	0.0500

noisy simulators. On average, the RMSE of the transformer model is 0.02 better than the simple NN model. On the algorithm circuit, the gap is even more apparent – up to 0.05. The R^2 on algorithm circuits with transformer is also much higher than simple NN (0.9985 vs. 0.9110). That shows the effectiveness of involving circuit graph information in the model.

5.3 Analysis

In Table 1, we show the effectiveness of concatenating the global features to the aggregated node features. Adding global features can reduce the RMSE loss on the test set with negligible computational overhead. The effectiveness is especially significant in IBM Geneva, where the RMSE is reduced by around 0.003.

Table 2 further performs an ablation study on the importance of each feature in the node feature vectors. We remove one feature while keeping all other features in each experiment and then train the model again to obtain the results and report the RMSE loss on the test set. The bold values mark the largest two losses when removing different features. We can see that removing 'Qubit Index' severely degrades the accuracy in all three backends. This may be because the qubit index helps the transformer model know the location of the gate. Removing 'Gate Type' also has a substantial negative impact since the model will not know the node type. We also observe that removing some features even improves the accuracy. This only happens on the real machine backend and maybe because of the large fluctuations of noise on the real backend.

Table 3 shows the relationship between the number of transformer layers with the prediction performance. We find that different model sizes do not greatly impact accuracy. On the noisy simulator and IBM Hanoi datasets, the one-layer model slightly outperforms the others, while on the IBM Geneva dataset, the three-layer model is the best. Therefore, in most of our experiments, we use a two-layer model as a trade-off.

Furthermore, we show the performance differences under different numbers of shots in noisy simulators as in Table 4. As the shots

Table 4: Prediction RMSE vs. Number of Shots

Shots	IBM Jakarta	IBM Lima	IBM Manila
512	**0.0287**	0.0266	0.0440
1024	0.0352	0.0246	0.0403
2048	0.0305	**0.0217**	0.0410
4096	0.0294	0.0250	**0.0399**

Table 5: Runtime of Simulation vs. Transformer Predictor

	Simulation	Predictor (bsz=1)	Predictor (bsz=10)
Latency (s)	5.57E-1	2.79E-3	3.28E-4

increase, the precision of the ground truth PST in the training set will be improved and will converge to the true PST when the shots are infinity. However, counter-intuitively, we find that increasing shot number does not guarantee better model accuracy.

Finally, besides theoretical proof of lower computation complexity of our model, we also perform empirical runtime comparisons as shown in Table 5. We run both the circuit simulator and the graph transformer on an Nvidia 3090 GPU with 24GB memory for 1000 sampled circuits from the random circuit dataset, and report average runtime. We select batch size 1 or 10 for the graph transformer predictor. The predictor achieves 200× and 1.7K× speedup over classical simulators to obtain the PST for batch size 1 and 10, respectively. That demonstrates the much higher efficiency of our graph transformer-based predictor.

6 Conclusion

Using machine learning to optimize quantum system problems is promising. This paper presents a case study of the ML for Quantum part of TorchQuantum library. We are inspired by that *a quantum circuit is a graph* and propose to leverage a *graph transformer* model to predict the circuit fidelity under the influence of quantum noise. First, we collect a large dataset of randomly generated circuits and algorithm circuits, and measure their fidelity on simulators and real machines. A graph with feature vectors for each node is constructed according to the circuit. The graph transformer processes the circuit graph and calculates the anticipated fidelity value for the circuit. Instead of the exponential cost of performing whole circuit simulations, we can effectively evaluate the fidelity under polynomial complexity. The datasets and models have been integrated into the TorchQuantum library, and we hope they can accelerate research in the ML and Quantum field.

Acknowledgment

We thank National Science Foundation, MIT-IBM Watson AI Lab, and Qualcomm Innovation Fellowship for supporting this research. This work is funded in part by EPiQC, an NSF Expedition in Computing, under grants CCF-1730082/1730449; in part by STAQ under grant NSF Phy-1818914; in part by DOE grants DE-SC0020289 and DE-SC0020331; and in part by NSF OMA-2016136 and the Q-NEXT DOE NQI Center. We acknowledge the use of IBM Quantum services for this work.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014).

[2] Charles H Bennett, Gilles Brassard, Claude Crépeau, et al. 1993. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. Physical review letters 70, 13 (1993), 1895.

[3] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. Nature (2017).

[4] Colin D Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M Sage. 2019. Trapped-ion quantum computing: Progress and challenges. Applied Physics Reviews 6, 2 (2019), 021314.

[5] Marco Cerezo, Alexander Poremba, Lukasz Cincio, and Patrick J Coles. 2020. Variational quantum fidelity estimation. Quantum 4 (2020), 248.

[6] Ranyiliu Chen, Zhixin Song, Xuanqiang Zhao, and Xin Wang. 2021. Variational quantum algorithms for trace distance and fidelity estimation. Quantum Science and Technology 7, 1 (2021), 015019.

[7] Jinglei Cheng, Haoqing Deng, and Xuehai Qia. 2020. Accqoc: Accelerating quantum optimal control based pulse generation. In ISCA (2020). IEEE, 543–555.

[8] Don Coppersmith. 2002. An approximate Fourier transform useful in quantum factoring. arXiv preprint quant-ph/0201067 (2002).

[9] Poulami Das, Christopher A Pattison, Srilatha Manne, Douglas M Carmean, Krysta M Svore, Moinuddin Qureshi, and Nicolas Delfosse. 2022. AFS: Accurate, Fast, and Scalable Error-Decoding for Fault-Tolerant Quantum Computers. In HPCA (2022). IEEE, 259–273.

[10] Alexey Dosovitskiy, Lucas Beyer, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020).

[11] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028 (2014).

[12] András Gilyén and Alexander Poremba. 2022. Improved Quantum Algorithms for Fidelity Estimation. arXiv preprint arXiv:2203.15993 (2022).

[13] Lov K Grover. 1996. A fast quantum mechanical algorithm for database search. In STOC (1996). 212–219.

[14] Jeremy Hsu. 2018. Intel49. In Intel.

[15] Zhirui Hu, Peiyan Dong, Zhepeng Wang, Youzuo Lin, Yanzhi Wang, and Weiwen Jiang. 2022. Quantum Neural Network Compression. ICCAD (2022).

[16] Fei Hua, Yanhao Chen, Yuwei Jin, Chi Zhang, Ari Hayes, Youtao Zhang, and Eddy Z Zhang. 2021. Autobraid: A framework for enabling efficient surface code communication in quantum computing. In Micro (2021). 925–936.

[17] Hsin-Yuan Huang, Richard Kueng, and John Preskill. 2020. Predicting many properties of a quantum system from very few measurements. Nature Physics 16, 10 (2020), 1050–1057.

[18] IBM. 2022. IBM Unveils Breakthrough 127-Qubit Quantum Processor. In IBM.

[19] Qiskit IBM. 2021. https://qiskit.org/textbook/ch-quantum-hardware/calibrating-qubits-pulse.html

[20] Weiwen Jiang, Jinjun Xiong, and Yiyu Shi. 2021. A co-design framework of neural networks and quantum circuits towards quantum advantage. Nature communications 12, 1 (2021), 1–13.

[21] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. 2017. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature (2017).

[22] Julian Kelly. 2018. A Preview of Bristlecone, Google's New Quantum Processor. In Google.

[23] Emanuel Knill, Dietrich Leibfried, Rolf Reichle, Joe Britton, R Brad Blakestad, John D Jost, Chris Langer, Roee Ozeri, Signe Seidelin, and David J Wineland. 2008. Randomized benchmarking of quantum gates. Physical Review A 77, 1 (2008).

[24] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. 2019. A quantum engineer's guide to superconducting qubits. Applied Physics Reviews 6, 2 (2019), 021318.

[25] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, et al. 2022. Realizing repeated quantum error correction in a distance-three surface code. Nature 605, 7911 (2022), 669–674.

[26] Zhiding Liang, Jinglei Cheng, Hang Ren, Hanrui Wang, Fei Hua, Yongshan Ding, Fred Chong, Song Han, Yiyu Shi, and Xuehai Qian. 2022. PAN: Pulse Ansatz on NISQ Machines. arXiv preprint arXiv:2208.01215 (2022).

[27] Zhiding Liang, Hanrui Wang, Jinglei Cheng, Yongshan Ding, Hang Ren, Xuehai Qian, Song Han, Weiwen Jiang, and Yiyu Shi. 2022. Variational quantum pulse learning. QCE (2022).

[28] Zhiding Liang, Zhepeng Wang, Junhuan Yang, Lei Yang, Yiyu Shi, and Weiwen Jiang. 2021. Can noise on qubits be learned in quantum neural network? a case study on quantumflow. In ICCAD (2021). IEEE, 1–7.

[29] Daniel A Lidar and Todd A Brun. 2013. Quantum error correction. Cambridge university press.

[30] Ji Liu and Huiyang Zhou. 2020. Reliability Modeling of NISQ- Era Quantum Computers. In IISWC (2020). 94–105.

[31] Easwar Magesan, Jay M Gambetta, and Joseph Emerson. 2011. Scalable and robust randomized benchmarking of quantum processes. Physical review letters 106, 18 (2011), 180504.

[32] Easwar Magesan, Jay M Gambetta, and Joseph Emerson. 2012. Characterizing quantum gates via randomized benchmarking. Physical Review A 85, 4 (2012), 042311.

[33] Shin Nishio, Yulu Pan, Takahiko Satoh, Hideharu Amano, and Rodney Van Meter. 2020. Extracting success from ibm's 20-qubit machines using error-aware compilation. JETC (2020) 16, 3 (2020), 1–25.

[34] Alberto Peruzzo, Jarrod McClean, et al. 2014. A variational eigenvalue solver on a photonic quantum processor. Nature communications 5, 1 (2014), 1–7.

[35] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. Quantum 2 (2018), 79.

[36] Gokul Subramanian Ravi, Kaitlin N Smith, Pranav Gokhale, Andrea Mari, Nathan Earnest, Ali Javadi-Abhari, and Frederic T Chong. 2022. Vaqem: A variational approach to quantum error mitigation. In HPCA (2022). IEEE, 288–303.

[37] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. 2014. Quantum support vector machine for big data classification. Physical review letters 113, 13 (2014), 130503.

[38] rigetti. 2021. Rigetti Quantum. In rigetti.

[39] Seongok Ryu, Jaechang Lim, Seung Hwan Hong, and Woo Youn Kim. 2018. Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network. arXiv:1805.10988 (2018).

[40] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient attention: Attention with linear complexities. In WACV (2021).

[41] Kok Chuan Tan and Tyler Volkoff. 2021. Variational quantum algorithms to estimate rank, quantum entropies, fidelity, and Fisher information via purity minimization. Physical Review Research 3, 3 (2021), 033251.

[42] Swamit S Tannu and Moinuddin Qureshi. 2019. Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes. In Micro (2019). 253–265.

[43] Swamit S Tannu and Moinuddin K Qureshi. 2019. Not all qubits are created equal: a case for variability-aware policies for NISQ-era quantum computers. In ASPLOS (2019). 987–999.

[44] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735 (2018).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).

[47] Hanrui Wang et al. 2020. Efficient algorithms and hardware for natural language processing. Massachusetts Institute of Technology (2020).

[48] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. 2022. Quantumnas: Noise-adaptive search for robust quantum circuits. In HPCA (2022). IEEE, 692–708.

[49] Hanrui Wang, Jiaqi Gu, Yongshan Ding, Zirui Li, Frederic T Chong, David Z Pan, and Song Han. 2022. QuantumNAT: Quantum Noise-Aware Training with Noise Injection, Quantization and Normalization. DAC (2022).

[50] Hanrui Wang, Zirui Li, Jiaqi Gu, Yongshan Ding, David Z Pan, and Song Han. 2022. QOC: Quantum On-Chip Training with Parameter Shift and Gradient Pruning. DAC (2022).

[51] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. 2020. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. In DAC 2020.

[52] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. Hat: Hardware-aware transformers for efficient natural language processing. ACL (2020).

[53] Hanrui Wang, Jiacheng Yang, Hae-Seung Lee, and Song Han. 2018. Learning to design circuits. NeurIPS MLSys Workshop (2018).

[54] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In HPCA 2021.

[55] Qisheng Wang, Zhicheng Zhang, Kean Chen, Ji Guan, Wang Fang, and Mingsheng Ying. 2021. Quantum algorithm for fidelity estimation. arXiv preprint arXiv:2103.09076 (2021).

[56] Zhepeng Wang, Zhiding Liang, Shanglin Zhou, Caiwen Ding, Yiyu Shi, and Weiwen Jiang. 2021. Exploration of quantum neural architecture by mixing quantum neuron designs. In ICCAD (2021). IEEE, 1–7.

[57] Xiao-Dong Yu, Jiangwei Shang, and Otfried Gühne. 2022. Statistical Methods for Quantum State Verification and Fidelity Estimation. Advanced Quantum Technologies (2022), 2100126.

[58] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. NeurIPS 32 (2019).

[59] Xiaoqian Zhang, Maolin Luo, Zhaodi Wen, Qin Feng, Shengshi Pang, Weiqi Luo, and Xiaoqi Zhou. 2021. Direct fidelity estimation of quantum states using machine learning. Physical Review Letters 127, 13 (2021), 130503.