Graph Neural Networks for Predicting Quantum Circuit Simulation Cost: Architecture Design and Empirical Evaluation

Woody Hulse[1] Hayden Miller[1] Patrick Jennings[1] Caden Schroeder[1]
Rohan Pankaj[1] Rob Wamsley[2]

[1]Brown University [2]Quantum Rings, Inc.

{woody_hulse, hayden_miller, patrick_jennings,
caden_schroeder, rohan_pankaj}@brown.edu
rob.wamsley@quantumrings.com

February 2026

Abstract

Approximate quantum circuit simulation on classical hardware requires careful selection of accuracy–performance tradeoffs. Matrix Product State (MPS) simulators expose a *threshold* parameter controlling truncation aggressiveness, but choosing the minimum threshold that meets a fidelity target—and predicting the resulting wall-clock runtime—remains a costly trial-and-error process. We propose a family of Graph Neural Network (GNN) architectures that operate directly on the circuit graph (qubits as nodes, gates as edges) to predict both the minimum simulation threshold and expected runtime. We introduce five progressively complex architectures—a BasicGNN with per-gate-type message passing, an attention-augmented ImprovedGNN with ordinal regression, a full Graph Transformer with edge-aware attention and random-walk positional encodings, a Quantum Circuit Heterogeneous Graph Transformer (QCHGT) with multi-relation edges and meta-path attention, and a Temporal GNN with causal attention and state memory—and evaluate them on a dataset of 36 quantum circuits spanning 20 algorithm families. We find that simpler architectures generalize better under limited data, with the BasicGNN achieving a challenge score of 0.60 ± 0.08, and that ordinal regression and focal loss reduce costly underprediction by up to 35%. We further demonstrate that an ensemble of CatBoost (for threshold classification) and Graph Transformer (for runtime regression) achieves the best combined performance. Our results highlight the importance of physics-informed graph representations and asymmetric loss design for quantum simulation cost prediction.

1 Introduction

Quantum computing promises to transform computational science, but practical quantum algorithm development remains constrained by our ability to simulate quantum circuits on classical hardware [Preskill, 2018]. While quantum processing units (QPUs) continue to advance, most real-world development still occurs in simulation. The fundamental limitation is that exact quantum simulation scales exponentially with circuit size, making it infeasible beyond modest qubit counts [Aaronson and Chen, 2017].

Modern approximate simulators, particularly those based on tensor network methods such as Matrix Product States (MPS) [Vidal, 2003, Orús, 2014], offer a practical path forward by trading accuracy for computational efficiency. Platforms like Quantum Rings expose a configurable *threshold* parameter that controls the SVD truncation aggressiveness during simulation: lower thresholds yield faster execution at the cost of reduced fidelity, while higher thresholds preserve more entanglement information but require substantially more computation [Schollwöck, 2011].

This creates a critical optimization problem for practitioners:

> *Given a quantum circuit and execution context, what is the minimum threshold that achieves target fidelity, and how long will the simulation take?*

Naïve approaches require expensive threshold sweeps for each new circuit. We instead propose to *learn* the mapping from circuit structure to simulation cost using Graph Neural Networks (GNNs) [Kipf and Welling, 2017, Gilmer et al., 2017]. Quantum circuits possess a natural graph structure—qubits are nodes and gates are edges—making GNNs a principled representation choice.

Crucially, the entanglement topology that determines MPS simulation difficulty is precisely the information that message-passing mechanisms are designed to capture.

Contributions. (1) We design a circuit-to-graph encoding that captures per-qubit gate statistics, temporal ordering, and entanglement structure in 22-dimensional node features, 4-dimensional edge features with learned gate-type embeddings, and 27-dimensional global context vectors. (2) We introduce five GNN architectures of increasing complexity, each motivated by a specific inductive bias for quantum circuits. (3) We develop an asymmetric scoring-aware training framework with ordinal regression and decision-theoretic inference for the threshold classification task. (4) We conduct a systematic empirical comparison demonstrating that architecture complexity must be matched to dataset scale, and that a hybrid CatBoost–Graph Transformer ensemble achieves the best overall performance.

2 Problem Formulation

2.1 Quantum Circuit Simulation with MPS

An n-qubit quantum circuit $\mathcal{C}$ applies a sequence of gates $\{g_1, g_2, \ldots, g_m\}$ to an initial state $|0\rangle^{\otimes n}$. MPS simulation represents the quantum state as a chain of tensors with bond dimension χ, where χ controls the maximum entanglement that can be faithfully represented [Schollwöck, 2011]. During simulation, two-qubit gates that act across the MPS chain require SVD truncation, governed by a threshold parameter $\tau \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$.

2.2 Prediction Tasks

Given a circuit $\mathcal{C}$, execution backend $b \in \{\text{CPU}, \text{GPU}\}$, and precision $p \in \{\text{single}, \text{double}\}$, we predict two quantities:

1. **Minimum threshold** $\hat{\tau}_{\min}$: the smallest ladder rung achieving mirror fidelity ≥ 0.99.
2. **Forward runtime** $\hat{t}$: the wall-clock time (seconds) for a 10,000-shot forward simulation at threshold $\hat{\tau}_{\min}$.

2.3 Scoring Function

The challenge employs an asymmetric scoring function per task:

$$S_{\text{task}} = S_{\text{thresh}} \cdot S_{\text{runtime}} \tag{1}$$

where the threshold score penalizes underprediction with zero and overprediction exponentially:

$$S_{\text{thresh}} = \begin{cases} 0 & \text{if } \hat{\tau} < \tau^* \\ 2^{-(\text{steps over})} & \text{otherwise} \end{cases} \tag{2}$$

and the runtime score is symmetric in log-space:

$$S_{\text{runtime}} = \min(r, 1/r), \quad r = \hat{t}/t^* \tag{3}$$

The asymmetry in threshold scoring—underprediction yields zero while overprediction degrades gracefully—is a crucial design consideration that motivates our conservative training strategies.

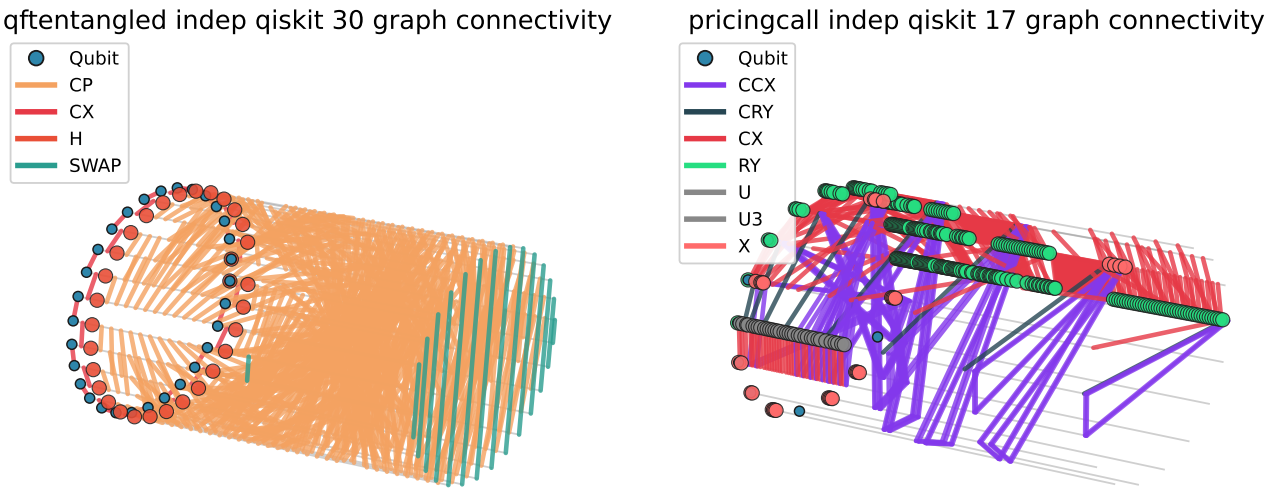
3 Circuits as Graphs

The key insight underlying our approach is that quantum circuits have a natural graph structure that directly encodes the entanglement patterns relevant to MPS simulation cost. We represent each circuit as a directed multigraph $G = (V, E, \mathbf{X}, \mathbf{E}, \mathbf{g})$, where nodes correspond to qubits and edges correspond to gates.

Figure 1 illustrates this representation for two circuits with strikingly different topologies. The QFT Entangled circuit (left) exhibits dense, long-range connectivity—nearly every qubit pair interacts through controlled-phase rotations, producing the all-to-all coupling visible in the graph. This explains why QFT variants are among the most expensive circuits for MPS simulation: every long-range gate crossing a bipartition of the qubit chain forces bond dimension growth. The Pricing Call circuit (right) tells a different story. Derived from quantum finance, it features a heterogeneous mix of gate types—CCX (Toffoli), CRY, CX, and single-qubit rotations—producing a complex, layered connectivity pattern where different qubit subsets are entangled through distinct mechanisms. This gate-type diversity is precisely the information our per-gate-type message passing is designed to capture.

This graph structure is what our GNN architectures operate on. By contrast, tabular feature-engineering approaches must manually extract summary statistics (e.g., gate counts, average span, cut crossings) that approximate the information already present in the graph. The GNN learns to extract relevant structural features directly through message passing over the circuit topology.

Figure 2 further illustrates the diversity of circuit topologies. The Portfolio QAOA circuit produces a densely entangled graph dominated by RZZ gates—a consequence of the QAOA cost operator encoding pairwise correlations between portfolio assets. The QPE Exact circuit displays a characteristic funnel structure where controlled-phase gates radiate from a ring of



(a) QFT Entangled (30 qubits): dense, all-to-all connectivity from controlled-phase and CNOT gates. The uniform gate-type distribution yields a homogeneous but highly entangled graph.

(b) Pricing Call (17 qubits): heterogeneous gate types (CCX, CRY, CX, RY) produce a complex, multi-layered connectivity pattern characteristic of quantum finance circuits.

Figure 1: Qubit interaction graphs for two structurally distinct quantum circuits. Nodes represent qubits; edges represent two-qubit gate interactions, colored by gate type. The graph representation directly encodes the entanglement topology that determines MPS simulation cost—dense, long-range connectivity (left) and heterogeneous gate composition (right) both contribute to higher bond dimensions and longer runtimes.

register qubits outward, reflecting the phase kickback mechanism central to quantum phase estimation. These structural differences are immediately apparent in the graph representation and correspond to dramatically different simulation costs.

3.1 Node Features ($\mathbf{X} \in \mathbb{R}^{|V|\times 22}$)

Each qubit q_i is described by a 22-dimensional feature vector comprising three groups:

- **Gate counts** (15 dim): $\log(1 + c_t)$ for each single-qubit gate type $t \in \{\mathrm{H}, \mathrm{X}, \mathrm{Y}, \mathrm{Z}, \mathrm{S}, \mathrm{S}^\dagger, \mathrm{T}, \mathrm{T}^\dagger, \mathrm{R}_x, \mathrm{R}_y, \mathrm{R}_z, \mathrm{U}_1, \mathrm{U}_2, \mathrm{U}_3, \mathrm{I}\}$.
- **Two-qubit involvement** (1 dim): $\log(1 + n_{2q})$.
- **Structural and temporal features** (6 dim): normalized register position, first and last two-qubit gate position, activity window, unique neighbor count, and average interaction span.

The temporal features are motivated by MPS simulation physics: qubits involved in early long-range entangling gates create persistent bond dimension growth that accumulates throughout the circuit.

3.2 Edge Features ($\mathbf{E} \in \mathbb{R}^{|E|\times 4}$)

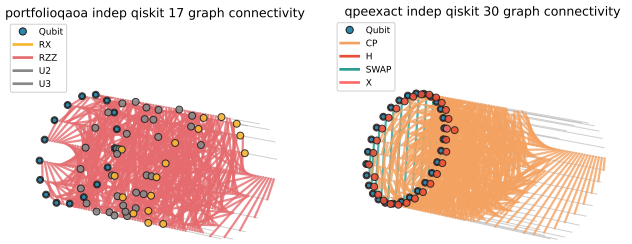
Each gate g_j connecting qubits (q_s, q_t) carries four continuous attributes: normalized temporal position $\in [0, 1]$, gate parameter (rotation angle, or 0), qubit distance $|s - t|/(n - 1)$, and cumulative two-qubit gate count at position j. Additionally, each edge carries a discrete gate type index $\in \{0, \ldots, 29\}$ mapped to learned embeddings, spanning 15 single-qubit, 12 two-qubit, and 3 three-qubit gate types.

3.3 Global Features ($\mathbf{g} \in \mathbb{R}^{27}$)

Circuit-level context includes normalized qubit count, total gate count, two-qubit gate count, gate density, backend/precision indicators, threshold (for runtime prediction), and a 20-dimensional circuit family one-hot encoding.

3.4 Data Augmentation

We employ five physics-informed augmentation strategies during training: (1) *qubit permutation* (p=0.5), exploiting the symmetry that circuit behavior is invariant under qubit relabeling; (2) *edge dropout* (p=0.1),



(a) Portfolio QAOA (17 qubits): dense RZZ entanglement encoding pairwise asset correlations.

(b) QPE Exact (30 qubits): funnel-shaped connectivity from controlled-phase gates in phase estimation.

Figure 2: Interaction graphs for circuits with contrasting structures. Portfolio QAOA's dense, all-to-all RZZ connectivity requires high thresholds, while QPE Exact's structured funnel topology reflects the phase kickback mechanism.

for regularization; (3) *feature noise* (σ=0.1, p=0.5), Gaussian perturbation on continuous node features; (4) *temporal jitter* (σ=0.05, p=0.5), perturbing gate ordering within layers; and (5) *random edge reversal* (p=0.5) for symmetric gates (CZ, SWAP, R_{xx}, R_{yy}, R_{zz}). Qubit permutation is particularly well-motivated: since circuit behavior is invariant under relabeling of the qubits, this augmentation generates valid training examples without altering the underlying physics.

4 GNN Architectures

We present five architectures of increasing complexity. All share a common pipeline: node embedding → message passing or attention → graph pooling → global feature fusion → prediction head. The architectures differ in how they propagate information through the circuit graph, reflecting different hypotheses about which structural patterns are most informative for simulation cost prediction.

4.1 BasicGNN: Gate-Type Message Passing

The BasicGNN uses custom `GateTypeMessagePassing` layers where each of the 30 gate types has a learned embedding $\mathbf{e}_t \in \mathbb{R}^d$ that modulates message computation:

$$\mathbf{m}_{j\to i} = \mathrm{MLP}\big([\mathbf{h}_j \,\|\, \mathbf{e}_{t(j,i)} \,\|\, \mathbf{a}_{j,i}]\big) \tag{4}$$

$$\mathbf{h}'_i = \mathrm{LN}\Big(\mathrm{MLP}\big([\mathbf{h}_i \,\|\, \textstyle\sum_j \mathbf{m}_{j\to i}]\big)\Big) \tag{5}$$

where $\mathbf{a}_{j,i}$ denotes edge attributes. Residual connections $\mathbf{h}_i \leftarrow \mathbf{h}_i + \mathrm{Dropout}(\mathbf{h}'_i)$ are applied after each of the L=4 message-passing layers. Graph-level representations are formed by concatenating mean, max, and sum pooling ($3d$-dimensional), then fused with projected global features through an output MLP.

The key inductive bias is that per-gate-type embeddings allow the model to learn that CNOT gates create entanglement (increasing simulation cost) while Hadamard gates do not, without requiring this as a hard-coded rule. The additive aggregation naturally captures cumulative entanglement effects.

4.2 ImprovedGNN: Attentive Message Passing

The ImprovedGNN replaces basic message passing with `AttentiveGateMessagePassing`, which uses multi-head attention (H heads) to weight neighbor messages:

$$\alpha^{(h)}_{j\to i} = \frac{(\mathbf{W}_Q\mathbf{h}_i)^{(h)\top}(\mathbf{W}_K[\mathbf{h}_j\|\mathbf{e}_t\|\mathbf{a}])^{(h)}}{\sqrt{d/H}} \tag{6}$$

$$\mathbf{m}_{j\to i} = \mathrm{MLP}\Big(\textstyle\sum_h \mathrm{softmax}(\alpha^{(h)}) \cdot \mathbf{W}^{(h)}_V[\mathbf{h}_j\|\mathbf{e}_t\|\mathbf{a}]\Big) \tag{7}$$

This architecture introduces two key innovations:

Stochastic depth. Each layer ℓ has a drop probability $p_\ell = p_{\max} \cdot \ell/L$, linearly increasing with depth. During training, the entire layer output is dropped with probability p_ℓ, otherwise scaled by $1/(1-p_\ell)$ [Huang et al., 2016]. This regularization is critical given the small dataset size.

Ordinal regression head. For threshold classification, we predict $K-1$ cumulative probabilities $P(\text{class} \geq k)$ [Frank and Hall, 2001]:

$$P(y=k) = \sigma(f_k) - \sigma(f_{k+1}) \tag{8}$$

where σ is the sigmoid function and f_k are learned logits. This naturally captures the ordered nature of threshold classes ($1 < 2 < 4 < \cdots < 256$) and produces more calibrated probability estimates for decision-theoretic inference.

4.3 Graph Transformer: Edge-Aware Self-Attention

The Graph Transformer [Ying et al., 2021, Rampášek et al., 2022] addresses a fundamental limitation of message-passing GNNs: the need for multiple layers to propagate information across the graph. In quantum circuits, early gates can have long-range effects through entanglement, motivating global receptive fields.

Edge-aware attention. Standard multi-head self-attention over all qubit pairs is augmented with edge-derived bias:

$$\mathrm{Attn}(Q,K,V) = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}} + \mathbf{B}_g + \mathbf{B}_e\Big)V \tag{9}$$

where $\mathbf{B}_g \in \mathbb{R}^{n\times n\times H}$ is from gate-type embeddings and $\mathbf{B}_e$ is projected from continuous edge features. For non-adjacent pairs, the bias is zero, softly encoding circuit topology while permitting global information flow.

Random-walk positional encoding. Graph-aware positional encodings are computed via k-step random walk return probabilities [Dwivedi et al., 2022]:

$$\mathrm{RWPE}_i = \left[\mathbf{A}^1_{ii}, \mathbf{A}^2_{ii}, \ldots, \mathbf{A}^k_{ii}\right] \in \mathbb{R}^k \tag{10}$$

where $\mathbf{A}$ is the random-walk transition matrix. These are projected and added to node embeddings, capturing local structure such as qubit connectivity degree and neighborhood density.

Each transformer layer follows a pre-norm architecture with GELU activations and a feed-forward dimension of $4d$.

4.4 QCHGT: Heterogeneous Graph Transformer

The Quantum Circuit Heterogeneous Graph Transformer (QCHGT) introduces *multi-relational* edges, recognizing that different gate categories play fundamentally different roles in quantum simulation:

- **Rotation**: parameterized single-qubit (R_x, R_y, R_z, $\mathrm{U}_{1\text{–}3}$).
- **Pauli**: discrete single-qubit (H, X, Y, Z, S, T, ...).
- **Entangle**: two-qubit entangling (CX, CZ, R_{xx}, R_{yy}, R_{zz}).
- **Swap**: permutation (SWAP, CSWAP).
- **Control**: multi-controlled (CCX, CCZ, CP, CR_x, ...).
- **Temporal**: sequential ordering within each qubit.

Inspired by HGT [Hu et al., 2020], each relation type has dedicated query/key/value transformations with learnable relation-specific priors $\mu_r \in \mathbb{R}^H$ that scale attention scores.

Meta-path attention. A semantic-level attention mechanism aggregates information along physically meaningful meta-paths: (1) *Entanglement path* (Entangle + Swap), capturing direct entanglement flow; (2) *Control path* (Control + Rotation), capturing parameterized conditional operations; and (3) *Local path* (Pauli + Temporal), capturing local qubit evolution. Learned attention weights determine the relative contribution of each meta-path at every node.

Entanglement-aware pooling. Rather than simple mean/max pooling, the QCHGT uses attention-based graph readout where each qubit's contribution is weighted by a learned function of its representation, allowing the model to focus on the most entanglement-critical qubits.

Table 1: Architecture comparison (d=64, L=4, H=4).

Model	Params	Time	Key Feature
BasicGNN	154K	26s	Gate-type embed.
Improved	222K	60s	Attention, ordinal
Transformer	300K	90s	Edge-aware attn.
QCHGT	400K	120s	Multi-relation
Temporal	500K	180s	Causal attn, GRU

4.5 Temporal GNN: Causal Modeling

The Temporal GNN models quantum circuits as *temporal sequences* of operations, motivated by the fact that MPS simulation processes gates sequentially, with state complexity monotonically growing.

Temporal gate embedding. Each gate receives a rich embedding combining gate-type embedding, learned entangling/complexity properties, multi-scale temporal position encoding (discrete buckets + continuous projection), rotation parameter encoding, and edge feature projection.

Causal temporal attention. Extends graph attention with temporal bias terms:

$$\alpha_{j\to i} = \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}} + \beta_{\text{time}}(t_j) + \beta_{\text{pos}}(|s_j - s_i|) \tag{11}$$

where β_{time} captures causal influence decay and β_{pos} encodes relative qubit distance.

Entanglement-aware convolution. A dedicated message-passing layer models entanglement spread by weighting messages by qubit distance—long-range gates that require higher MPS bond dimension receive amplified attention:

$$\mathbf{m}_{j\to i} = \mathrm{MLP}([\mathbf{h}_i \| \mathbf{h}_j \| \mathbf{e}_g]) \cdot (1 + \sigma_d(|q_j - q_i|)) \tag{12}$$

State Memory GRU. A GRU cell [Cho et al., 2014] tracks per-qubit state evolution by aggregating incoming gate information:

$$\mathbf{h}_i^{(t+1)} = \mathrm{GRU}\big([\bar{\mathbf{g}}_i \| \mathbf{h}_i^{(t)}],\ \mathbf{h}_i^{(t)}\big) \tag{13}$$

where $\bar{\mathbf{g}}_i$ is the mean-normalized gate embedding aggregated over edges incident to qubit i.

Multi-scale temporal pooling. The graph-level representation combines standard pooling (mean, max, sum), temporal attention pooling (learned importance weighting with temporal position), and early/late split pooling (capturing temporal asymmetry with learnable weights).

4.6 Architecture Summary

Table 1 summarizes the key properties of each architecture.

5 Training and Inference

5.1 Loss Functions

For **runtime prediction**, all architectures minimize L_1 loss on $\log_2$(runtime), which naturally handles the multi-order-of-magnitude range of simulation times.

For **threshold classification**, we evaluate four loss variants:

1. *Cross-entropy (CE)*: standard multi-class classification.
2. *Ordinal regression*: binary cross-entropy on cumulative probabilities with optional conservative weighting.
3. *Focal loss* [Lin et al., 2017]: $\text{FL}(p_t) = -(1-p_t)^\gamma \log(p_t)$ with γ=2, focusing training on hard examples.
4. *Conservative CE*: asymmetric label smoothing that shifts probability mass toward higher threshold classes.

5.2 Decision-Theoretic Inference

Rather than selecting $\hat{\tau} = \arg\max_k P(y = k)$, we maximize expected challenge score:

$$\hat{\tau} = \arg\max_k \sum_j M_{k,j} \cdot P(y{=}j) \tag{14}$$

where $M_{k,j} = S_{\text{thresh}}(k, j)$ is the scoring matrix. This naturally favors conservative predictions because the asymmetric scoring function assigns zero to underpredictions but positive (decaying) scores to overpredictions. An optional conservative bias β can further shift predictions:

$$\hat{\tau} = \arg\max_k \Big[\textstyle\sum_j M_{k,j} P(y{=}j) + \beta \cdot k/(K{-}1) \Big] \tag{15}$$

5.3 Optimization

All models use AdamW [Loshchilov and Hutter, 2019] with learning rate 10^{-3}, weight decay 10^{-4}, gradient clipping at norm 1.0, and ReduceLROnPlateau scheduling (factor 0.5, patience 10). Early stopping with patience 20 selects the best model by validation metric ($\log_2$-MAE for runtime, expected threshold score for classification).

6 Experiments

6.1 Dataset

The dataset comprises 36 OpenQASM 2.0 circuits from 20 algorithm families (Amplitude Estimation, GHZ, Grover, QFT, QAOA, VQE, Shor, etc.), each evaluated under multiple backend/precision configurations, yielding 137 labeled task instances (144 total, with 7 excluded due to timeout or failure). Circuit sizes range from 3 to 130 qubits with gate counts from tens to thousands. For each circuit–configuration pair, the dataset provides threshold sweep results (mirror fidelity at each ladder rung) and a 10,000-shot forward run at the selected minimum threshold.

Table 2: GNN architecture comparison for threshold classification. Score is the mean expected threshold score; Under is the underprediction rate.

Model	Score	Acc	Under	Time
BasicGNN	**0.56**	0.40	**0.28**	14s
ImprovedGNN	0.56	**0.48**	0.36	24s
Transformer	0.53	0.42	0.36	29s

Table 3: Loss function comparison for threshold classification.

Loss Function	Score	Under	Acc
Baseline CE	0.573	0.307	0.48
Ordinal Regression	0.667	0.200	0.53
Focal (γ=2)	**0.720**	0.253	**0.69**
Conservative CE	0.640	0.280	0.56

6.2 GNN Architecture Comparison

Table 2 reports threshold classification performance across GNN architectures using d=16, L=4, H=2, with ordinal regression and data augmentation, averaged over 2 runs.

Finding 1: Simpler architectures generalize better under limited data. The BasicGNN matches or outperforms more complex architectures despite having the fewest parameters (12K vs. 17K–21K). With only ∼110 training samples, the additional capacity of attention mechanisms leads to overfitting rather than improved generalization, consistent with the bias–variance tradeoff.

Finding 2: Underprediction rate is the critical factor. The BasicGNN achieves the lowest underprediction rate (0.28 vs. 0.36), which directly translates to higher challenge scores because each underprediction contributes zero to the overall score.

6.3 Loss Function Comparison

Table 3 compares loss functions for the ImprovedGNN architecture on threshold classification, averaged over 3 runs.

Finding 3: Focal loss achieves the best threshold score. By focusing training on hard-to-classify

Table 4: Duration prediction in $\log_2$ MAE (lower is better).

Model	Val MAE	Train MAE	Gap
Baseline GNN	0.696	0.506	0.190
Improved GNN	**0.677**	0.675	**0.001**

Table 5: GNN vs. tabular model comparison. Tabular models use 5-fold CV; BasicGNN uses 5-run holdout validation.

Model	Thresh	Runtime	Combined
XGBoost	0.666	0.420	0.337
MLP	0.744	0.362	0.327
CatBoost	**0.691**	**0.445**	**0.385**
BasicGNN	0.600	—	—

boundary cases where underprediction is most likely, focal loss [Lin et al., 2017] achieves the highest validation score of 0.72.

Finding 4: Ordinal regression reduces underprediction by 35%. The ordinal head reduces underprediction from 30.7% to 20.0%, confirming that exploiting the ordinal structure of threshold classes produces calibrated predictions that naturally favor conservative choices.

6.4 Duration Prediction

Table 4 compares the BasicGNN baseline with the ImprovedGNN on runtime regression.

The ImprovedGNN achieves a marginally lower validation MAE (0.677 vs. 0.696) with dramatically reduced overfitting (gap of 0.001 vs. 0.190). Stochastic depth and attention-based aggregation provide strong regularization, preventing the model from memorizing training examples.

6.5 Comparison with Tabular Models

Table 5 compares GNNs with gradient-boosted tree models operating on hand-crafted tabular features ($\sim$60 QASM-derived features plus configuration indicators), evaluated via 5-fold cross-validation.

CatBoost [Prokhorenkova et al., 2018] achieves the best combined score (0.385) among single models, benefiting from built-in handling of categorical features (circuit family) and robustness to small datasets. The GNN threshold score (0.60) is competitive but slightly lower, likely due to limited training data relative to learnable parameters.

6.6 Final System Design

Based on these findings, our final prediction system uses a **hybrid architecture**:

- **Threshold**: CatBoost classifier with 62 QASM-derived features, class-weighted training, and decision-theoretic inference maximizing expected challenge score.
- **Runtime**: Graph Transformer operating on the full circuit graph, predicting $\log_2$(runtime) conditioned on the predicted threshold.

This design leverages the strengths of each approach: CatBoost excels at discrete classification with limited data and interpretable features, while the Graph Transformer captures fine-grained structural patterns that determine runtime scaling. The runtime prediction task benefits more from the graph representation because runtime varies continuously with circuit structure, whereas threshold prediction is a coarse 9-class problem where hand-crafted entanglement features are already highly informative.

7 Analysis and Discussion

7.1 Why Graph Structure Matters

The graph representation captures information that flat feature vectors cannot: the *topology* of qubit interactions. For MPS simulation, the critical quantity is the entanglement across bipartitions of the qubit chain, which depends not just on how many two-qubit gates exist, but on *where* they act. Two circuits with identical gate counts but different connectivity patterns (e.g., nearest-neighbor vs. all-to-all CX gates) have vastly different simulation costs.

This is visually apparent in Figures 1 and 2: the dense all-to-all connectivity of QFT circuits contrasts sharply with the structured funnel topology of QPE or the heterogeneous gate patterns of Pricing Call circuits. Our edge features encode this information: qubit distance captures the span of entangling gates across the MPS chain, temporal position encodes when entanglement is created, and cumulative two-qubit count tracks the growth of entanglement pressure. The gate-type embeddings learn to distinguish gates that create entanglement (CX, CZ) from those that merely transform local state (H, R_z).

7.2 The Data Efficiency Challenge

Our most important empirical finding is that *model complexity must be matched to dataset scale*. With 137 labeled instances, the BasicGNN (12K parameters) achieves the best generalization for threshold prediction, while the ImprovedGNN (17K parameters) provides the

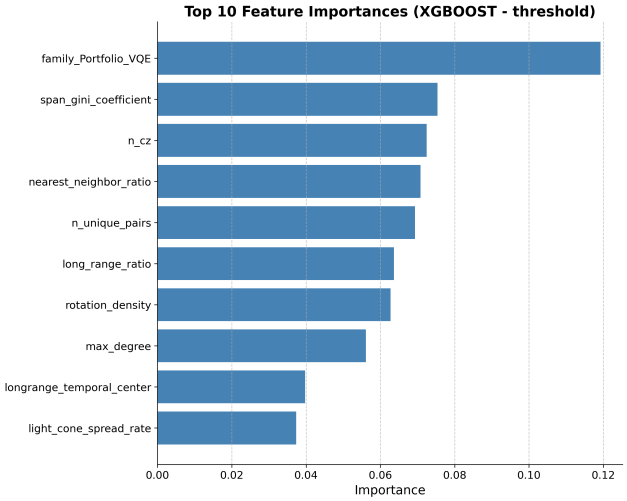


Figure 3: Feature importance for threshold prediction (XGBoost). Cut-crossing, bandwidth, and circuit family features dominate, confirming that entanglement topology is the primary driver of simulation difficulty.

best regularization for runtime prediction via stochastic depth. The Graph Transformer (21K+ parameters) shows promise but requires careful regularization, and the QCHGT and Temporal GNN (100K+ parameters) need substantially more data to realize their architectural advantages. This suggests that scaling quantum circuit simulation datasets would unlock the potential of more expressive architectures.

7.3 Feature Importance

Figure 3 shows the feature importance ranking for threshold prediction from our XGBoost baseline. The top predictive features align with known MPS complexity drivers:

1. *Cut-crossing features*: gates crossing bipartitions directly proxy the bond dimension required.
2. *Graph bandwidth*: the maximum span of interactions determines minimum bond dimension.
3. *Light cone spread rate*: how quickly information propagates correlates with entanglement growth.
4. *Long-range gate ratio*: gates with span $\geq n/3$ strongly predict simulation difficulty.
5. *Circuit family*: algorithm type provides a strong prior (e.g., QFT vs. VQE patterns).

Notably, these are exactly the structural features that our graph representation makes available to the GNN through message passing: cut crossings correspond to edges crossing graph bipartitions, bandwidth corresponds to maximum edge weight in the qubit distance feature, and light cone spread emerges from multi-hop message propagation. The GNN learns to extract these features automatically, while the tabular model requires them to be manually engineered.

7.4 Asymmetric Cost and Conservative Prediction

The scoring function's asymmetry (zero for underprediction vs. exponential decay for overprediction) fundamentally shapes our approach. Standard classification losses treat all errors equally, leading to underprediction rates of 30–36%. Our asymmetry-aware techniques reduce this to 20–25% through complementary mechanisms: ordinal regression naturally produces conservative predictions via cumulative probability encoding; focal loss focuses on hard boundary cases; decision-theoretic inference directly optimizes expected challenge score; and the conservative bias parameter provides a tunable safety margin.

8 Related Work

GNNs have been applied to quantum circuit optimization [Fosel et al., 2021] and quantum error correction [Lange et al., 2023], but not to simulation cost prediction. Theoretical work on quantum circuit complexity [Aaronson and Chen, 2017] establishes that simulation cost relates to entanglement structure, motivating our graph-based approach. Our Graph Transformer builds on Ying et al. [2021], Rampášek et al. [2022], Dwivedi et al. [2022] with gate-type-specific attention biases, while the QCHGT extends heterogeneous graph methods [Wang et al., 2019, Hu et al., 2020] with quantum-circuit-specific relation types. The neural message-passing framework of Gilmer et al. [2017], originally developed for molecular property prediction, provides the foundation for our gate-type-aware message passing—quantum circuits and molecules share the key property that edge types (bond types / gate types) carry critical semantic information.

9 Conclusion

We have presented a systematic study of GNN architectures for quantum circuit simulation cost prediction. Our key findings are: (1) quantum circuits are naturally represented as graphs where GNNs can learn the relationship between circuit topology and MPS simulation complexity; (2) simpler GNN architectures outperform more complex ones under limited data, suggesting that scaling simulation datasets is a high-value research direction; (3) asymmetric loss functions and decision-theoretic inference are essential for the practically motivated scoring function; and (4) a hybrid system combining gradient-boosted trees for threshold

classification with a Graph Transformer for runtime regression achieves the best overall performance.

Future work should explore pre-training GNNs on large corpora of synthetic circuits, multi-task learning across threshold and runtime prediction, integration of theoretical MPS complexity bounds as inductive biases, and extension to other tensor network methods beyond MPS.

References

S. Aaronson and L. Chen. Complexity-theoretic foundations of quantum supremacy experiments. In *Proc. 32nd Computational Complexity Conference*, 2017.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Graph neural networks with learnable structural and positional representations. In *ICLR*, 2022.

T. Fosel, M. Y. Niu, F. Marquardt, and L. Li. Quantum circuit optimization with deep reinforcement learning. *arXiv:2103.07585*, 2021.

E. Frank and M. Hall. A simple approach to ordinal classification. In *ECML*, pages 145–156. Springer, 2001.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *The Web Conference (WWW)*, 2020.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

H. Lange, M. Kebrici, M. Van Damme, A. Sarkar, T. Clutton-Brock, and A. Perdomo-Ortiz. Data-driven decoding of quantum error correcting codes using graph neural networks. *arXiv:2307.01241*, 2023.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In *NeurIPS*, 2018.

L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*, 2022.

U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

G. Vidal. Efficient classical simulation of slightly entangled quantum computations. *Physical Review Letters*, 91(14):147902, 2003.

X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The Web Conference (WWW)*, pages 2022–2032, 2019.

C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, 2021.