# FLOWQC: CONDITIONAL FLOW MATCHING FOR EFFICIENT AND ROBUST QUANTUM CIRCUIT SYNTHESIS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Quantum circuit synthesis is crucial for near-term quantum computing, yet existing generative solutions built on diffusion models suffer from slow iterative sampling, limited conditional controllability, and high sensitivity to sampling steps and the guidance scales. To address these challenges, we propose flowQC, a conditional flow matching framework for efficient quantum circuit generation. To improve conditional generation quality, we introduce the Prompt-Aware Quantum Transformer (PAQ-former), which employs adaLN-Zero modulation and sigmoid residual gating to control the conditional signal dynamically in the generation trajectory. For dataset generation and verification, we firstly replace Qiskit's Density-Matrix representation with its Statevector implementation, for larger-scale training and evaluation. Experiments on conditional circuit generation tasks demonstrate that flowQC improves sampling efficiency and circuit quality over the previous diffusion-based baselines. For entanglement generation, flowQC achieves up to 31.44$\times$ inference acceleration and reduces generated circuit gate depths by 35.5% on average. Moreover, across varying numbers of sampling steps from 20 to 5 and guidance scales from 5 to 1, flowQC maintains more stable accuracy than the previous diffusion-based baseline. For unitary-conditioned generation, flowQC produces diverse candidate circuits validated by phase-invariant unitary equivalence evaluation. These results establish flowQC as an efficient and robust conditional generative framework for quantum circuit synthesis. The code and data will be made publicly available upon acceptance.

## 1 INTRODUCTION

Quantum computing has attracted interest in physics (Di Meglio et al., 2024), chemistry (Robledo-Moreno et al., 2025), and machine learning (Wang & Liu, 2024) due to its potential computational advantages. However, near-term quantum devices remain constrained by limited coherence time, hardware-specific gate sets, and restricted circuit depth (Zhang et al., 2022). These challenges motivate the development of automated methods for designing quantum circuits. Traditional circuit synthesis methods normally rely on optimized search, decomposition algorithms, and heuristic compilation techniques. Although these approaches can provide high-quality solutions, their computational cost increases rapidly as circuit size and search space grow (Makarova et al., 2026).

Generative models have emerged as a promising alternative for automated quantum circuit generation (Merilehto, 2026). By learning distributions over existing circuit structures, generative models can produce diverse candidate circuits without performing expensive explicit search. Fürrutter et al. (2024) introduced genQC, the first diffusion-based framework for quantum circuit synthesis, demonstrating the feasibility of learning circuit distributions with deep generative models. Then Chen & Tang (2025) proposed UDiTQC which explored transformer-based diffusion architectures to improve circuit modeling capability.

Despite these advances, existing generative approaches still face three key challenges. First, diffusion-based models require iterative denoising procedures, resulting in high sampling latency (Yu et al., 2026). Second, generating circuits under diverse quantum constraints remains challenging, as effective conditioning mechanisms are needed to guide the generation process (Mauro et al., 2026). Third, we observe the previous diffusion-based models are highly sensitive to the guid-



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

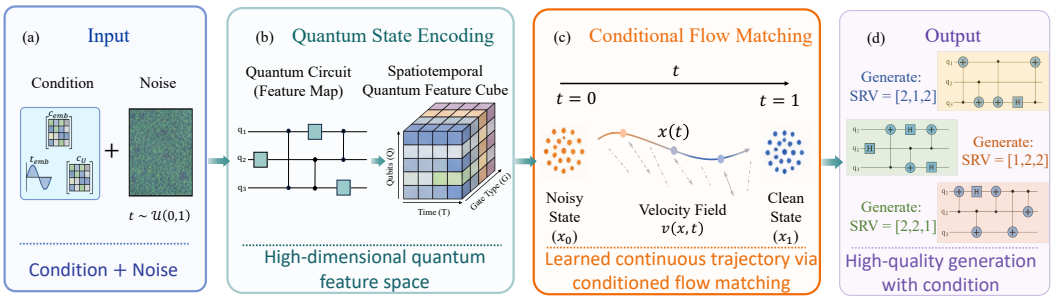



Figure 1: **Pipeline of flowQC.** We formulate quantum circuit synthesis as a conditional flow matching problem over continuous circuit encoding. **(a)** A condition signal is combined with random noise sampled at timestep $t$. **(b)** The input is encoded into a spatiotemporal quantum feature cube via a parameterized quantum circuit acting as a structured feature map. **(c)** A conditional flow matching module learns a velocity field $v(x,t)$ that defines a continuous trajectory from the noisy state $x_0$ to the clean state $x_1$, with adaptive condition coupling modulated by $t$. **(d)** The velocity field generates diverse, high-fidelity quantum circuits under the given condition.

ance scales and the number of sampling steps, limiting the practicality of diffusion-based circuit synthesis (Luo et al., 2024; Chen et al., 2024).

To address these challenges, we propose **flowQC**, a conditional flow matching framework (Lipman et al., 2023) for quantum circuit generation as illustrated in Figure 1. Unlike diffusion-based approaches that rely on iterative denoising, flowQC learns a continuous-time velocity field to directly model the transformation from noise to valid circuit representations.

To further optimize the model's awareness of the generation trajectory, we design a **Prompt-Aware Quantum Transformer** (PAQ-former) backbone. By incorporating the adaLN-Zero modulation (Peebles & Xie, 2023) and sigmoid residual gating (Lian et al., 2026), PAQ-former dynamically regulates the strength of conditional coupling based on timestep $t$ (Singh et al., 2025). This enables the model to prioritize global conditional guidance during the early stages of generation while transitioning toward local self-refinement in the later stages.

For dataset generation and verification, we firstly leverage existing Qiskit's Statevector (Javadi-Abhari et al., 2024). This established pure-state simulation tool provides practical support for the large-scale training and testing in this work.

The primary contributions of this work are summarized as follows:

(1) We propose flowQC, the first conditional flow matching framework for quantum circuit generation, enabling efficient few-step sampling compared with diffusion-based approaches.

(2) We design PAQ-former, a timestep-aware conditional architecture integrating adaptive normalization and gated residual connections to improve constraint-aware generation.

(3) Comparisons with diffusion-based baselines on entanglement generation and unitary circuit synthesis show that flowQC consistently improves generation quality, sampling efficiency and circuit compactness. Sensitivity analyses across sampling steps and guidance scales further demonstrate its greater robustness to inference hyperparameters.

## 2 Related Work

**Quantum Circuit Synthesis.** Quantum circuit synthesis aims to construct gate-level quantum circuits that implement desired quantum states or unitary transformations. Traditional synthesis approaches mainly rely on numerical optimization, heuristic search, and decomposition-based compilation methods. For example, Davis et al. (2020a) formulates circuit synthesis as a discrete search problem over possible gate sequences and employs efficient search strategies to identify high-quality implementations. Younis et al. (2021) introduces a hierarchical decomposition strategy to improve the scalability of unitary synthesis by recursively optimizing smaller circuit blocks. In addition,

practical quantum software frameworks such as Qiskit provides a collection of synthesis and optimization passes for compiling quantum circuits to hardware-compatible gate sets.

Although these approaches can achieve high-quality synthesis results, their computational cost often increases as the circuit size grows due to the expanding search space(Rasmussen et al., 2026; Yan et al., 2026). This motivates the exploration of learning-based approaches that can directly model distributions over existing quantum circuit structures and efficiently generate candidate solutions.

**Generative Models for Quantum Circuit Generation.** Recent advances in deep generative modeling have introduced a new paradigm for quantum circuit generation. Instead of explicitly searching over the exponentially large circuit space, generative approaches learn the underlying distribution of existing circuits and sample diverse candidates after training. Fürrutter et al. (2024) introduced genQC, the first diffusion-based generative framework for quantum circuit synthesis, demonstrating the feasibility of learning conditional circuit distributions with denoising diffusion models. UDiTQC explored a U-Net-style diffusion transformer architecture to improve global dependency modeling in quantum circuit representations and enhance generation efficiency (Chen & Tang, 2025).

Despite their promising performance, diffusion-based quantum circuit generators still suffer from iterative denoising overhead and require effective mechanisms for incorporating diverse quantum constraints during generation. Furthermore, constructing large-scale verified circuit datasets remains challenging because generated circuits must be repeatedly evaluated with quantum semantic metrics before being used for model training.

**Flow Matching and Conditional Generative Models.** Flow matching (Lipman et al., 2023) provides an alternative formulation for generative modeling by learning continuous-time vector fields that transport samples from a simple prior distribution toward target data distributions. Compared with diffusion models, flow-based approaches can enable efficient deterministic sampling with fewer integration steps while maintaining competitive generation quality.

Recent studies have demonstrated the effectiveness of flow matching and rectified flow in large-scale generative tasks, including image synthesis and structured data generation (Lipman et al., 2023; Esser et al., 2024). Transformer-based architectures with adaptive normalization and conditional modulation. DiT have further improved the scalability of conditional generative models by enabling flexible conditioning mechanisms (Peebles & Xie, 2023).

However, flow matching has not been explored for quantum circuit generation. In this work, we extend conditional flow matching to quantum circuit representations by introducing a quantum-specific generation framework and a timestep-aware conditional architecture.

# 3 METHOD

## 3.1 PROBLEM SETUP

Let $\mathcal{G} = \{g_1, \ldots, g_K\}$ denote the quantum gate pools, with $K = 2$ (H, CX) for entanglement generation and $K = 6$ (adding X, Z, SWAP, CCX) for unitary compilation. An $N$-qubit circuit of length $T$ is encoded as a discrete tensor $C \in \{-K, \ldots, K\}^{N \times T}$. Here, $C_{q,t} = 0$ indicates an idle qubit, positive integers denote single-qubit gates or multi-qubit targets (1 for H, 2 for CX target), and negative integers strictly specify control nodes (e.g., -2 for CX control). This discrete matrix is projected into a continuous tensor $x \in \mathbb{R}^{d_e \times N \times T}$ via an orthogonal embedding $E_{clr}$ to serve as the target state for flow matching, where $d_e$ is the embedded dimension. The generation is guided by a composite condition vector $c = [c_y \parallel c_U]$, where $c_y$ is the CLIP-encoded (Hafner et al., 2021) text prompt specifying the target Schmidt Rank Vector (SRV) (van Luijk et al., 2024), and $c_U$ is the encoded target unitary matrix. Formally, each training instance consists of $(x, c)$, and the goal is to learn the conditional generative model $p_\theta(x \mid c)$ that can be decoded into valid and condition-aligned quantum circuits.

## 3.2 MODEL OVERVIEW

As shown in Figure 2(a), flowQC parameterizes the conditional velocity field $v_\theta(x_t, t, c)$ using a conditional U-Net. The intermediate circuit tensor $x_t$ is projected by a $1 \times 1$ convolution and augmented with two-dimensional positional encodings (details in the appendix A.3). The encoder–decoder hier-

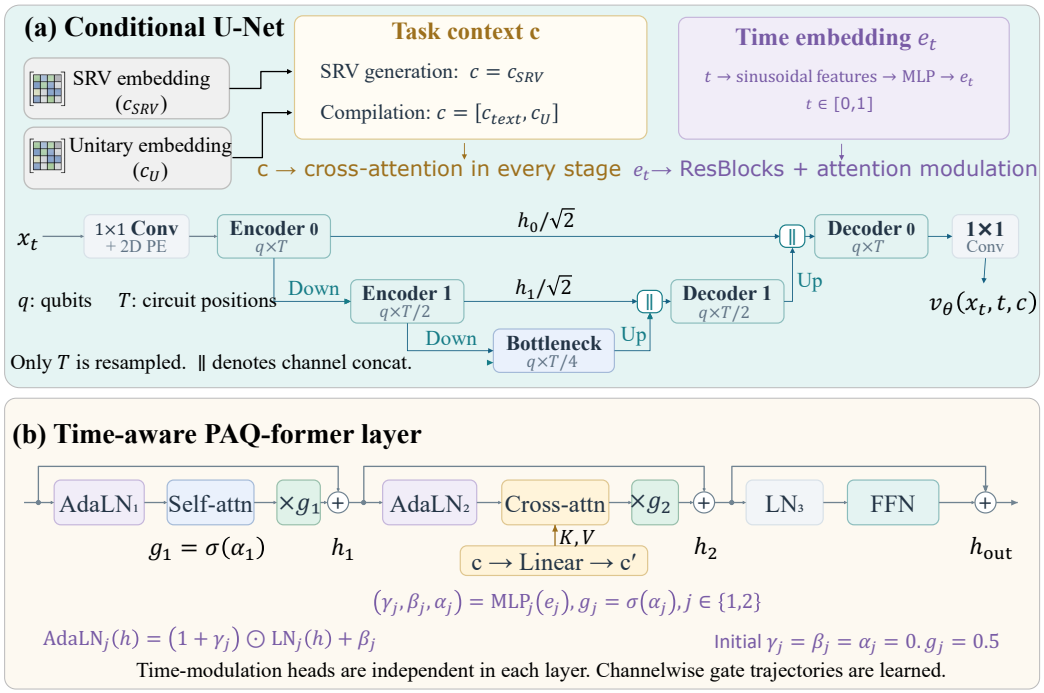



Figure 2: Architecture of **flowQC**. (a) The conditional U-Net preserves the qubit axis $q$ while resampling the circuit-position axis $T$, with scaled encoder features concatenated into the decoder. (b) A time-aware PAQ-former layer applies independently time-modulated and sigmoid-gated self-attention and cross-attention, followed by an unmodulated feed-forward residual update.

archy resamples only the circuit-position axis $T$, while preserving the qubit axis $q$. Encoder features are scaled by $1/\sqrt{2}$ and concatenated with the corresponding decoder features along the channel dimension. A final $1 \times 1$ convolution outputs the predicted velocity.

Task context and flow time are incorporated through separate pathways. The task context is $c = c_{\text{SRV}}$ for entanglement generation and $c = [c_{\text{text}}, c_U]$ for unitary compilation, and is supplied to cross-attention at every U-Net stage. The flow time $t \in [0, 1]$ is mapped through sinusoidal features and an MLP to obtain the time embedding $e_t$, which conditions the residual blocks and modulates the attention sublayers.

Figure 2(b) details the time-aware Prompt-Aware Quantum Transformer (PAQ-former) layer. Its self-attention and cross-attention sublayers each use time-conditioned adaptive layer normalization and a channelwise sigmoid-gated residual connection, with independent modulation heads in each layer. The task context is linearly projected to supply keys and values for cross-attention. These attention updates are followed by a feed-forward sublayer with standard layer normalization and an ungated residual connection. At initialization, the modulation outputs satisfy $\gamma_j = \beta_j = \alpha_j = 0$, yielding attention gates $g_j = \sigma(\alpha_j) = 0.5$, $j \in \{1, 2\}$. The channelwise gate trajectories are learned, enabling layer-specific, time-dependent control of self-attention and conditional attention.

### 3.3 Quantum Circuit Synthesis based on Condition Flow Matching

Flow Matching (FM) (Lipman et al., 2023) is a continuous-time generative modeling framework that learns a time-dependent velocity field to morph a source distribution into a target distribution along a continuous trajectory. In this work, we make the first attempt to apply the FM framework to the continuous tensor representation space of quantum circuits. Specifically, the source distribution $x_0 \sim q_0 = \mathcal{N}(0, I)$ is defined as the noisy circuit encoding distribution, and the target distribution $x_1 \sim q_1(\cdot \mid c)$ is defined as the valid circuit embedding distribution. The transformation evolves over a denoising time interval $t \in [0, 1]$, conditioned on the composite signal $c$ that may correspond to SRV or unitary constraints.

Formally, our goal is to learn a conditional velocity field $v_\theta(x_t \mid t_{\text{emb}}, c_{\text{emb}}, c_U)$, where $t_{\text{emb}}$ is the sinusoidal timestep embedding that embeds the flow time $t \in [0, 1]$, $c_{\text{emb}}$ is the SRV condition embedding that captures the geometric signature of the target circuit and $c_U$ is the unitary matrix serving as structural condition. This field characterizes the change of the circuit state at denoising time $t$. The state evolution follows the conditional ODE:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = v_\theta(X_t \mid t_{\text{emb}}, c_{\text{emb}}, c_U) \tag{1}$$

where $X_t$ denotes the generated circuit embedding state at time $t$.

During training, Independent Conditional Flow Matching (I-CFM) minimizes the discrepancy between the predicted velocity $v_\theta$ and the reference velocity $u$ induced by a predefined interpolation path $\pi_t(x_0, x_1)$, for which we adopt the linear form $\pi_t(x_0, x_1) = (1-t)x_0 + tx_1$. Given a conditioning constraint $c$, we define $x_0 \sim q_0$ as a noisy source tensor, and $x_1 \sim q_1(\cdot \mid c)$ as the corresponding valid circuit expression drawn from the target distribution. The training objective is given by:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_c \, \mathbb{E}_{\substack{x_0 \sim q_0, \\ x_1 \sim q_1(\cdot|c)}} \, \mathbb{E}_{t \sim \mathcal{U}(0,1)} \left[ \| v_\theta(x_t \mid t, c) - (x_1 - x_0) \|_2^2 \right]. \tag{2}$$

This formulation enables the model to learn the conditional transformation from noisy intermediate states to the valid quantum circuit states.

$$x_{t_{k+1}} = x_{t_k} + \Delta t \cdot v_\theta^{\text{cfg}}(x_{t_k}, t_k, c) \tag{3}$$

During inference, we integrate the learned velocity field using a uniform step size $\Delta t = (1 - t_{\text{start}})/K$.

In our study, an Euler solver with $K = 20$ steps is sufficient to produce high-quality circuits. For masking/editing, setting $t_{\text{start}} > 0$ and utilizing the noise-data interpolation as the initial value supports inpaint-style sampling.

### 3.4 Backbone: Prompt-Aware Quantum Transformer

While Flow Matching defines the optimization objective via a velocity field $v_\theta$, the backbone determines the fidelity and efficiency with which this field is learned. We propose the Prompt-Aware Quantum Transformer (PAQ-former) to address two limitations of the spatial Transformer under continuous-time training in the original genQC (Fürrutter et al., 2024): (i) Transformer sub-layers are agnostic to the denoising time step $t$, applying a uniform condition-coupling strength throughout the trajectory; and (ii) Deep stacking of standard Transformer blocks exhibits gradient instability when fitting the high-dimensional velocity field. PAQ-former resolves these issues by integrating *adaLN-Zero modulation* and *sigmoid residual gating* at every attention sub-layer.

**Backbone overview.** We adopt a U-Net topology optimized for quantum circuit tensors, featuring three encoder stages and symmetric decoder stages connected via $1/\sqrt{2}$ scaled skip connections. Within this structure, we replace the conventional bottleneck and mid-layer blocks with our PAQ-former blocks. Each block comprises a Self-Attention sub-layer (capturing qubit-wise dependencies), a Cross-Attention sub-layer (absorbing the multi-modal condition $c$) and a Feed-Forward (FFN) sub-layer, organized in a Pre-Norm configuration to ensure smooth gradient flow.

**adaLN-Zero modulation.** The conditional velocity field $v_\theta(x_t, t, c)$ is heterogeneous across $t$: at $t \to 0$, $x_t$ is nearly pure noise, requiring the network to rely on $c$ for global orientation; at $t \to 1$, $x_t$ approaches a valid circuit, necessitating self-dominated local refinement. To allow per-layer adaptation, we replace standard LayerNorm with an adaLN-Zero (Peebles & Xie, 2023) module. The scale $\gamma$, shift $\beta$, and gating parameter $\alpha$ are parameterized as $t$-dependent affine projections of the sinusoidal time embedding $t_{\text{emb}}$:

$$\gamma(t),\ \beta(t),\ \alpha(t) \quad = \quad W\ \texttt{SiLU}(t_{\text{emb}}) + b \tag{4}$$

$$\texttt{adaLN}(h, t) \quad = \quad \big(1 + \gamma(t)\big) \odot \texttt{LN}(h) + \beta(t) \tag{5}$$

Crucially, the zero initialization of $W$ and $b$ ensures that $\gamma$ and $\beta$ vanish at $t = 0$, reducing the module to standard LayerNorm. Combined with the sigmoid gate ($\sigma(0) = 0.5$), the block is initialized as a half-weight Pre-Norm Transformer. The configuration provides superior gradient stability in deep regimes compared to post-norm alternatives (Xiong et al., 2020).

**Sigmoid residual gating.** We apply a sigmoid-gated residual connection:

$$h_{\text{out}} = h_{\text{in}} + \sigma(\alpha(t)) \odot \texttt{Attention}\big(\texttt{adaLN}(h_{\text{in}}, t)\big). \tag{6}$$

This design choice is motivated by two factors. First, $\sigma(\alpha(t)) \in (0, 1)$ provides a probabilistic interpretation of condition coupling strength, representing information flow into residual stream. Second, the sigmoid eliminates pathological regimes where $\alpha < 0$ would cause sign-inverted attention, but the effect lacks physical meaning in circuit synthesis. We employ independent MLPs $\phi_s$ and $\phi_c$ for self- and cross-attention, allowing the network to learn decoupled trajectories for structural self-refinement and condition alignment.

PAQ-former enables flowQC to learn a $t$-varying coupling strategy directly from the velocity field dynamics. Empirically, the cross-attention gate $\sigma(\alpha_c(t))$ converges to a sigmoidal decay, prioritizing condition guidance at $t \approx 0$ and transitioning to structural refinement at $t \approx 1$.

### 3.5 Dataset Generation and Verification

As reported in genQC (Fürrutter et al., 2024) and UDiTQC (Chen & Tang, 2025), the scale of generative quantum circuit synthesis has so far been limited to 8 qubits. We identify the key bottleneck in the generation and verification of the training datasets: labeling each sampled circuit with its SRV requires characterizing the entanglement structure of the output state. The conventional pipeline constructs the full density matrix $\rho = |\psi\rangle\langle\psi|$ of the $N$-qubit output state and derives the SRV from reduced density matrices via partial trace, which severely limits the scale of dataset construction and is the key factor behind the 8-qubit ceiling of prior generative approaches.

In this work, we resolve this bottleneck by leveraging the Statevector class from qiskit.quantum_info (Javadi-Abhari et al., 2024). This reduces the memory footprint of dataset generation from $\mathcal{O}(4^N)$ to $\mathcal{O}(2^N)$ and lowers the per-sample verification cost.

## 4 Experiment

### 4.1 Experimental Setup

We evaluate flowQC on three conditional circuit generation tasks: (1) **Entanglement Generation.** We assess generation efficiency, generation accuracy, and circuit compactness under text-based SRV conditioning. (2) **Unitary Compilation.** We evaluate the diversity of valid circuits under matrix-based unitary conditioning and an expanded gate set. (3) **Sensitivity Analysis.** We examine the robustness of flowQC versus the previous diffusion-based baseline to varying sampling steps and guidance scales, and their impact on generation quality.

**Methods.** (1) genQC (baseline): A DDPM-based diffusion model strictly following the implementation in (Fürrutter et al., 2024). The backbone is the original Spatial Transformer, where the time step $t$ is injected additively without dynamic attention modulation. (2) UDiTQC(baseline): An U-Net-style Diffusion Transformer architecture (Chen & Tang, 2025) which combines U-Net's strengths with the Transformer's ability. (3) flowQC: our full proposed framework, which upgrades the backbone to the PAQ-former, featuring adaLN-Zero modulation and Sigmoid residual gating. (4) flowQC−PAQ-former: based on genQC, flowQC−PAQ-former replaces the $\epsilon$-prediction objective with I-CFM velocity field prediction ($v_\theta \to x_1 - x_0$). The backbone remains the original Spatial Transformer. This variant is used to isolate the contribution of the Flow Matching objective itself.

**Fairness Statement on Reproducibility.** All models are trained from scratch on the same datasets and same U-Net, using shared sample distributions and class balance to control for differences in training data. During inference, all models use $K$=20 denoising/integration steps: genQC employs DDIM (Song et al., 2021) sampling, while flowQC and flowQC−PAQ-former use Euler ODE integration. This ensures that all comparisons are conducted under an identical computational budget.

**Datasets.** Datasets for two core tasks: (1) Entanglement Generation: The gate pool is restricted to $\mathcal{G} = \{\text{H, CX}\}$. The datasets cover $N = 3 \sim 12$ qubits. Data is split into training, validation, and test sets in an 8:1:1 ratio, while maintaining a strict balance across SRV classes to prevent evaluation bias; (2) Unitary Compilation: We expand the gate pool to $K = 6$ (H, CX, X, Z, SWAP, CCX).

Table 1: Generation speedup of **flowQC** relative to the previous diffusion-based baseline across different qubit scales and target entanglement levels. Values report the multiplicative speedup factor ($\times$), with larger values indicating greater generation efficiency.

| **Qubits** | **Entanglement Level** (Speedup $\times$) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| $N = 3$ | 4.83× | 6.24× | 5.67× | — | — | — | — | — | — | — | — | — |
| $N = 4$ | 5.08× | 5.34× | 5.17× | 4.60× | — | — | — | — | — | — | — | — |
| $N = 5$ | 7.19× | 7.64× | 5.92× | 6.46× | 7.58× | — | — | — | — | — | — | — |
| $N = 6$ | 8.44× | 8.07× | 5.87× | 5.98× | 6.38× | 7.65× | — | — | — | — | — | — |
| $N = 7$ | 10.08× | 9.79× | 7.73× | 7.36× | 7.74× | 9.04× | 8.88× | — | — | — | — | — |
| $N = 8$ | 11.48× | 11.24× | 9.06× | 8.31× | 8.29× | 8.93× | 9.62× | 9.05× | — | — | — | — |
| $N = 9$ | 14.20× | 12.57× | 11.16× | 10.38× | 9.51× | 10.47× | 10.11× | 11.45× | 11.75× | — | — | — |
| $N = 10$ | 15.28× | 13.95× | 12.53× | 11.50× | 10.33× | 10.11× | 11.06× | 11.90× | 12.94× | 11.94× | — | — |
| $N = 11$ | 17.20× | 15.94× | 14.81× | 14.98× | 12.77× | 11.75× | 13.39× | 15.09× | 17.09× | 20.17× | 26.48× | — |
| $N = 12$ | 18.30× | 17.86× | 17.17× | 18.49× | 18.99× | 19.45× | 18.07× | 17.57× | 21.39× | 23.20× | 31.44× | 30.75× |

Table 2: Average accuracy across different qubit scales in flowQC, genQC and UDiTQC.

| **Method** | **Number of Qubits** | | | | | | | | | | **Overall** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | |
| genQC(Fürrutter et al., 2024) | 58.67 | 69.25 | 74.20 | 72.67 | 72.71 | 71.88 | 67.78 | 64.90 | 64.27 | 63.75 | 67.67 |
| UDiTQC(Chen & Tang, 2025) | 66.00 | 77.00 | 78.00 | 76.83 | 75.14 | 74.25 | 71.56 | 68.40 | 68.45 | 67.83 | 71.63 |
| **flowQC (ours)** | **84.67** | **85.50** | **82.60** | **77.67** | **76.71** | **77.38** | **73.22** | **72.70** | **73.09** | **72.92** | **75.95** |

### 4.2 Entanglement Generation

We evaluate the core performance of flowQC through the entanglement generation task. We discuss the results across two dimensions: generation efficiency and circuit quality.

For all circuit-generation experiments involving systems from 3 to 12 qubits, we use a classifier-free guidance scale of $g = 5$, 20 sampling steps, $k = 512$.

**Generation efficiency.** We compare the sampling efficiency of **flowQC** with diffusion-based baselines by measuring the wall-clock time required to generate the same number of valid quantum circuits satisfying the target condition. Specifically, we define the joint efficiency cost $T_{\text{joint}}$ as $T_{\text{joint}}(k) = k \cdot T_{\text{total}}/(N_{\text{total}} - N_{\text{error}} - N_{\text{invalid}})$, where $T_{\text{total}}$ is the total sampling time, $N_{\text{total}}$ is the total number of generated circuits, $N_{\text{error}}$ denotes the number of circuits that fail to satisfy the target condition, and $N_{\text{invalid}}$ denotes the number of invalid circuits. The parameter $k$ specifies the desired number of valid, condition-compliant circuits. Therefore, $T_{\text{joint}}(k)$ estimates the time required to obtain $k$ successful samples, with a lower value indicating higher sampling efficiency.

In order to highlight the performance of flowQC compared with the previous diffusion-based models, we define the speed up as $\text{Speedup} = \text{T}_{\text{diffu}}/\text{T}_{\text{flowQC}}$, where $T_{\text{diffu}}$ and $T_{\text{flowQC}}$ are respectively the running times of diffusion-based method and flowQC.

Table 1 shows that flowQC achieves generation speedups over the previous diffusion-based models across evaluated settings. The advantage increases with system size, with the average speedup rising from $5.58\times$ at $N = 3$ to $21.06\times$ at $N = 12$, and reaching a maximum of $31.44\times$. The generative gain remains consistently large, for large systems and more complex target conditions.

**Generation accuracy.** As shown in Table 2, flowQC achieves higher exact SRV accuracy than both genQC and UDiTQC across all evaluated system sizes, ranging from 3 to 12 qubits. A detailed breakdown of the exact SRV accuracy for each target SRV is provided in the Table 6 of the appendix A.1.

**Circuit quality.** High-quality quantum circuits must be not only semantically correct but also compact to minimize decoherence errors during hardware execution. We analyze the circuit depth in valid generated circuits (Table 3). Empirical results show that flowQC generates shorter circuits across all test conditions, achieving an average gate saving of almost 35.5% and 37.9% compared to genQC and UDiTQC alternatively. This directly addresses the typical deficiency of diffusion mod-

Table 3: Comparison of generative circuit depths (mean ± std) among flowQC, genQC and UDiTQC across varying qubit scales and entanglement levels.

| Qubits | Method | Entanglement Level | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0 | 2 | 3 | 4 | 5 | 6 |
| 3 | genQC(Fürrutter et al., 2024) | $12.60 \pm 9.22$ | $14.96 \pm 8.60$ | $19.14 \pm 11.25$ | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | $13.23 \pm 8.92$ | $13.60 \pm 9.37$ | $18.53 \pm 9.57$ | | | |
| | **flowQC(ours)** | $\mathbf{7.30 \pm 1.57}$ | $\mathbf{8.49 \pm 1.63}$ | $\mathbf{9.66 \pm 1.67}$ | – | – | – |
| 4 | genQC(Fürrutter et al., 2024) | $10.77 \pm 11.21$ | $16.82 \pm 10.72$ | $21.33 \pm 10.53$ | $37.95 \pm 8.26$ | – | – |
| | UDiTQC(Chen & Tang, 2025) | $10.46 \pm 10.42$ | $15.92 \pm 10.29$ | $22.96 \pm 11.42$ | $35.41 \pm 8.43$ | – | – |
| | **flowQC(ours)** | $\mathbf{6.44 \pm 1.26}$ | $\mathbf{8.43 \pm 2.30}$ | $\mathbf{12.70 \pm 2.26}$ | $\mathbf{15.28 \pm 1.76}$ | – | – |
| 5 | genQC(Fürrutter et al., 2024) | $10.34 \pm 10.78$ | $15.77 \pm 10.88$ | $20.40 \pm 11.16$ | $31.13 \pm 10.99$ | $43.69 \pm 10.34$ | – |
| | UDiTQC(Chen & Tang, 2025) | $12.37 \pm 11.25$ | $14.93 \pm 9.87$ | $19.92 \pm 10.74$ | $32.37 \pm 11.35$ | $42.92 \pm 10.27$ | – |
| | **flowQC(ours)** | $\mathbf{5.92 \pm 1.76}$ | $\mathbf{9.53 \pm 3.64}$ | $\mathbf{16.70 \pm 3.75}$ | $\mathbf{20.64 \pm 2.63}$ | $\mathbf{22.90 \pm 1.72}$ | – |
| 6 | genQC(Fürrutter et al., 2024) | $9.13 \pm 9.92$ | $16.75 \pm 11.69$ | $22.02 \pm 11.82$ | $30.02 \pm 12.58$ | $39.34 \pm 11.06$ | $46.91 \pm 7.43$ |
| | UDiTQC(Chen & Tang, 2025) | $8.39 \pm 8.97$ | $17.38 \pm 11.25$ | $23.42 \pm 11.64$ | $34.84 \pm 13.77$ | $39.93 \pm 12.31$ | $47.43 \pm 7.92$ |
| | **flowQC(ours)** | $\mathbf{5.70 \pm 1.49}$ | $\mathbf{10.79 \pm 4.37}$ | $\mathbf{20.35 \pm 5.52}$ | $\mathbf{28.01 \pm 4.20}$ | $\mathbf{32.09 \pm 2.86}$ | $\mathbf{34.53 \pm 2.15}$ |

Table 4: Generative accuracy among successfully decoded circuits under different sampling steps and classifier-free guidance scales $g$, averaged over the 3-12 qubits evaluation setting.

| | flowQC (ours) | | | | | Diffusion-based model | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Steps** | $g=1$ | $g=2$ | $g=3$ | $g=4$ | $g=5$ | $g=1$ | $g=2$ | $g=3$ | $g=4$ | $g=5$ |
| 5 | **0.415** | **0.615** | **0.679** | **0.683** | **0.675** | 0.255 | 0.369 | 0.426 | 0.473 | 0.505 |
| 8 | **0.460** | **0.654** | **0.712** | **0.733** | **0.741** | 0.278 | 0.415 | 0.488 | 0.544 | 0.580 |
| 10 | **0.496** | **0.680** | **0.731** | **0.752** | **0.753** | 0.312 | 0.476 | 0.559 | 0.604 | 0.632 |
| 12 | **0.466** | **0.672** | **0.724** | **0.746** | **0.757** | 0.345 | 0.503 | 0.587 | 0.634 | 0.666 |
| 14 | **0.512** | **0.699** | **0.746** | **0.760** | **0.767** | 0.364 | 0.514 | 0.597 | 0.647 | 0.680 |
| 16 | **0.526** | **0.704** | **0.750** | **0.766** | **0.776** | 0.375 | 0.532 | 0.615 | 0.664 | 0.694 |
| 18 | **0.519** | **0.703** | **0.753** | **0.772** | **0.780** | 0.386 | 0.545 | 0.624 | 0.677 | 0.705 |
| 20 | **0.535** | **0.711** | **0.759** | **0.776** | **0.785** | 0.398 | 0.562 | 0.640 | 0.687 | 0.716 |

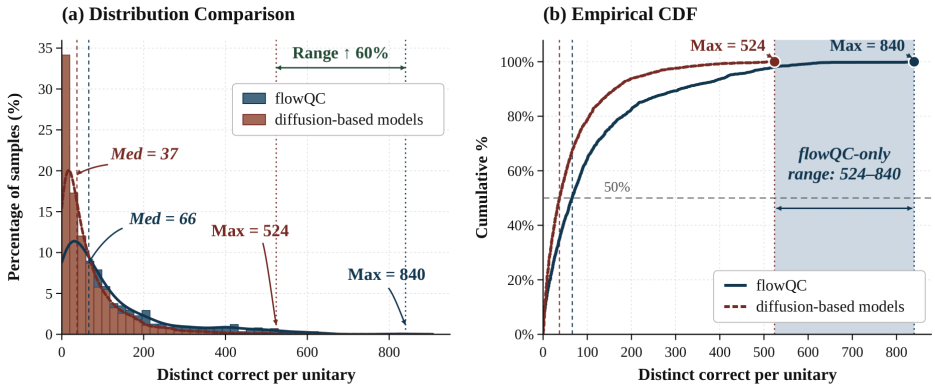



Figure 3: Distribution of distinct correct circuits per unitary for flowQC and genQC.

els, namely the tendency to introduce redundant gates irrelevant to the target condition. However, we admit that with the number of qubits increases, the difference will become less obvious. At present, it only has obvious effects on quantum circuits with low qubits.

Table 4 evaluates conditional correctness among successfully decoded circuits. flowQC consistently outperforms the previous diffusion-based model across all evaluated combinations of sampling steps and guidance scales. flowQC is also less sensitive to inference hyperparameters in terms of condition-aligned circuit generation.

### 4.3 Unitary Compilation

To evaluate the diversity of synthesized circuits, we measure the number of distinct correct circuits generated per target unitary. As shown in Figure 3(a), flowQC consistently produces more diverse solutions than genQC across the entire distribution: the median rises from 37 to 66 (a 78% increase),

Table 5: Comparison of unitary synthesis performance across different methods. ↑ indicates higher is better, while ↓ indicates lower is better.

| Method | Success Rate ↑ | Time / Target ↓ | CX Count ↓ | Circuit Depth ↓ |
|---|---|---|---|---|
| Qiskit QS (Shende et al., 2006) | **100.00%** | **0.017 s** | 19 | 39 |
| QSearch (Davis et al., 2020b) | 38.24% | 2.513 s | 6 | 13 |
| LEAP (Smith et al., 2023) | 47.43% | 2.668 s | 7 | 15 |
| genQC (Fürrutter et al., 2024) | 98.16% | 1.573 s | 8 | 13 |
| **flowQC(ours)** | 99.28% | 1.232 s | **5** | **12** |

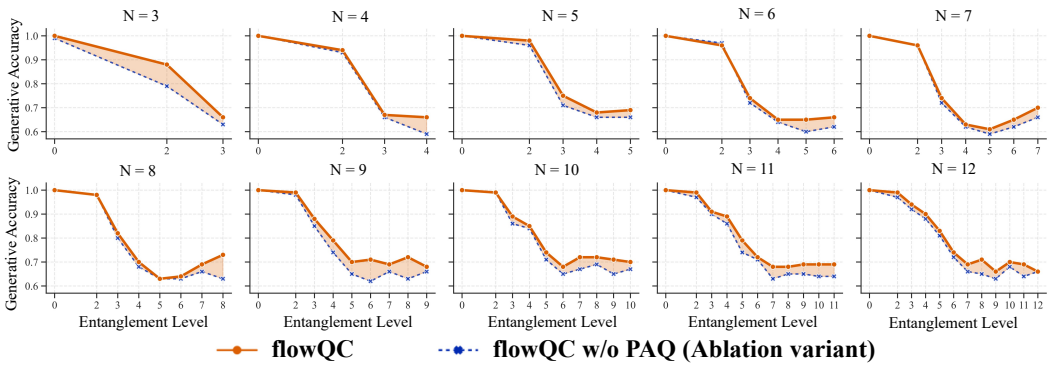



Figure 4: Ablation of architectural components. The table reports the average SRV Hit Rate across all entanglement levels for each system size ($N$). Full fine-grained results are in the Appendix.

and the maximum extends from 524 to 840, representing a 60% expansion in solution-space coverage. The empirical CDF in Figure 3(b) confirms that flowQC dominates genQC at every percentile, with a substantial exclusive range of 524–840 that genQC cannot reach at all. These results show that conditional flow matching enables flowQC to explore a significantly broader and more diverse set of valid circuit implementations for each target unitary.

Table 5 compares flowQC against genQC and conventional synthesis methods. flowQC improves synthesis success rate to 99.28%, cuts generation latency by 21.7%, and produces shallower circuits with fewer CX gates than genQC. Search-based methods yield compact circuits but low success rates; Qiskit QS is fast and reliable at the cost of heavy gate. The results highlight flowQC's balanced performance across reliability, sampling speed, and circuit compactness. The details are on the appendix A.1 of the Figure 5.

### 4.4 Ablation Study

To isolate the independent contribution of the PAQ-former under the Flow Matching framework, we conduct a controlled ablation study. We compare our full model (**flowQC**) against an ablation variant (**flowQC w/o PAQ**), which retains the I-CFM training objective but utilizes the original U-Net backbone (lacking adaLN-Zero modulation and sigmoid residual gating). From the Figure 4, the integration of the PAQ-former yields a strict performance improvement across all evaluated system sizes from $N = 3$ to 12, delivering an overall absolute gain of $+2.4$ percentage points.

## 5 Conclusion

We presented flowQC, a conditional flow matching framework for efficient quantum circuit synthesis, which replaces the previous diffusion-based noise prediction with continuous-time conditional velocity-field learning and ODE-based sampling. We further introduced the time-aware PAQ-former, which applies time-conditioned adaptive normalization and sigmoid residual gating to improve conditional generation. throughout the whole training process, we replaced the Qiskit DensityMatrix-based pipeline used in prior work with a Statevector-based implementation, supporting larger-scale training and evaluation of generated circuits across 3–12 qubits.

## AI USE STATEMENT

A large language model assisted the authors in polishing the language of the manuscript. The authors verified that all claims, proofs, mathematical formulations, and reported values are valid, checked them against the implementation and results, and take full responsibility for the manuscript.

## REFERENCES

Minshuo Chen, Song Mei, Jianqing Fan, and Mengdi Wang. An overview of diffusion models: Applications, guided generation, statistical rates and optimization. *arXiv preprint arXiv:2404.07771*, 2024.

Zhiwei Chen and Hao Tang. UDiTQC: U-net-style diffusion transformer for quantum circuit synthesis. *arXiv preprint arXiv:2501.16380*, 2025.

Marc G Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Towards optimal topology aware quantum circuit synthesis. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 223–234. IEEE, 2020a.

Marc Grau Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi, and Costin Iancu. Towards optimal topology aware quantum circuit synthesis. In *IEEE International Conference on Quantum Computing and Engineering, QCE 2020, Denver, CO, USA, October 12-16, 2020*, pp. 223–234. IEEE, 2020b. doi: 10.1109/QCE49297.2020.00036. URL https://doi.org/10.1109/QCE49297.2020.00036.

Alberto Di Meglio, Karl Jansen, Ivano Tavernelli, Constantia Alexandrou, Srinivasan Arunachalam, Christian W. Bauer, Kerstin Borras, Stefano Carrazza, Arianna Crippa, Vincent Croft, Roland de Putter, Andrea Delgado, Vedran Dunjko, Daniel J. Egger, Elias Fernández-Combarro, Elina Fuchs, Lena Funcke, Daniel González-Cuadra, Michele Grossi, Jad C. Halimeh, Zoë Holmes, Stefan Kühn, Denis Lacroix, Randy Lewis, Donatella Lucchesi, Miriam Lucio Martinez, Federico Meloni, Antonio Mezzacapo, Simone Montangero, Lento Nagano, Vincent R. Pascuzzi, Voica Radescu, Enrique Rico Ortega, Alessandro Roggero, Julian Schuhmacher, Joao Seixas, Pietro Silvi, Panagiotis Spentzouris, Francesco Tacchino, Kristan Temme, Koji Terashi, Jordi Tura, Cenk Tüysüz, Sofia Vallecorsa, Uwe-Jens Wiese, Shinjae Yoo, and Jinglei Zhang. Quantum computing for high-energy physics: State of the art and challenges. *PRX Quantum*, 5(3):037001, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=FPnUhsQJ5B.

Florian Fürrutter, Gorka Muñoz-Gil, and Hans J. Briegel. Quantum circuit synthesis with diffusion models. *Nature Machine Intelligence*, 6(5):515–524, 2024.

Markus Hafner, Maria Katsantoni, Tino Köster, James Marks, Joyita Mukherjee, Dorothee Staiger, Jernej Ule, and Mihaela Zavolan. Clip and complementary methods. *Nature Reviews Methods Primers*, 1(1):20, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

Junbo Jacob Lian, Feng Xiong, Yujun Sun, Kaichen Ouyang, Zong Ke, Mingyang Yu, Shengwei Fu, Zhong Rui, Zhang Yujun, and Huiling Chen. Twistnet-2d: Learning second-order channel interactions via spiral twisting for texture recognition. *arXiv preprint arXiv:2602.07262*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Yimin Luo, Qinyu Yang, Yuheng Fan, Haikun Qi, and Menghan Xia. Measurement guidance in diffusion models: Insight from medical image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):7983–7997, 2024.

Mariia Makarova, Davide Pastorello, Valter Cavecchia, Enrico Blanzieri, Kamil Khadiev, and Alexander Rumyantsev. Quantum circuit optimization techniques: A literature survey. *Quantum Information & Computation*, 26(1):154–179, 2026.

Francesco Mauro, Francesca De Falco, Lorenzo Papa, Andrea Ceschini, Alessandro Sebastianelli, Paolo Gamba, Massimo Panella, and Silvia Ullo. Enriching earth observation labeled data with quantum conditioned diffusion models. *IEEE Transactions on Geoscience and Remote Sensing*, 2026.

Juhani Merilehto. Generative AI for quantum circuits and quantum code: A technical review and taxonomy. *arXiv preprint arXiv:2603.16216*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Christian Rasmussen, Jason Perez, Ed Younis, Wibe Albert de Jong, and Samah Saeed. *Stability and Reproducibility in Heuristic Unitary Synthesis for Quantum Circuits*, pp. 856–861. Association for Computing Machinery, New York, NY, USA, 2026. ISBN 9798400724312. URL https://doi.org/10.1145/3787109.3816401.

Javier Robledo-Moreno, Mario Motta, Holger Haas, Ali Javadi-Abhari, Petar Jurcevic, William Kirby, Simon Martiel, Kunal Sharma, Sandeep Sharma, Tomonori Shirakawa, Iskandar Sitdikov, Rong-Yang Sun, Kevin J. Sung, Maika Takita, Minh C. Tran, Seiji Yunoki, and Antonio Mezzacapo. Chemistry beyond the scale of exact diagonalization on a quantum-centric supercomputer. *Science Advances*, 11(25):eadu9991, 2025.

V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006. ISSN 1937-4151. doi: 10.1109/tcad.2005.855930. URL http://dx.doi.org/10.1109/TCAD.2005.855930.

R Satya Rajendra Singh, PV Arun, and B Krishna Mohan. Future trends in explainable ai for geospatial applications. In *Explainable AI for Earth Observation Data Analysis*, pp. 258–277. CRC Press, 2025.

Ethan Smith, Marc Grau Davis, Jeffrey Larson, Ed Younis, Lindsay Bassman Oftelie, Wim Lavrijsen, and Costin Iancu. Leap: Scaling numerical optimization based synthesis using an incremental approach. *ACM Transactions on Quantum Computing*, 4(1):1–23, February 2023. ISSN 2643-6817. doi: 10.1145/3548693. URL http://dx.doi.org/10.1145/3548693.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Lauritz van Luijk, René Schwonnek, Alexander Stottmeister, and Reinhard F Werner. The schmidt rank for the commuting operator framework. *Communications in Mathematical Physics*, 405(7): 152, 2024.

Yunfei Wang and Junyu Liu. A comprehensive review of quantum machine learning: from NISQ to fault tolerance. *Reports on Progress in Physics*, 87(11):116402, 2024.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning (ICML)*, 2020.

Ge Yan, Wenjie Wu, Yuheng Chen, Kaisen Pan, Xudong Lu, Zixiang Zhou, Yuhan Wang, Ruocheng Wang, and Junchi Yan. Ai-empowered quantum circuit synthesis and compilation optimization: Overview and prospects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2026. doi: 10.1109/TPAMI.2026.3715237.

Ed Younis, Koushik Sen, Katherine Yelick, and Costin Iancu. Qfast: Conflating search and numerical optimization for scalable quantum circuit synthesis. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 232–243, 2021. doi: 10.1109/QCE52317.2021.00041.

King Yiu Yu, Aritra Sarkar, Erbing Hua, Maximilian Rimbach-Russ, Ryoichi Ishihara, and Sebastian Feld. From characterization to construction: Generative quantum circuit synthesis from gate set tomography data. *arXiv preprint arXiv:2605.01367*, 2026.

Kun Zhang, Kwangmin Yu, and Vladimir Korepin. Quantum search on noisy intermediate-scale quantum devices. *Europhysics Letters*, 140(1):18002, 2022.

# A APPENDIX

## A.1 SUPPLEMENTARY EXPERIMENT

**Generation accuracy** From the Table 6, we can see the performace about the flowQC compared with other methods.

Table 6: Exact generation accuracy of flowQC, genQC, and UDiTQC across different qubit scales and entanglement levels.

| Qubits | Method | Entanglement Level | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 3 | genQC(Fürrutter et al., 2024) | 0.87 | 0.46 | 0.43 | – | – | – | – | – | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | 0.89 | 0.57 | 0.52 | – | – | – | – | – | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.88** | **0.66** | – | – | – | – | – | – | – | – | – |
| 4 | genQC(Fürrutter et al., 2024) | 0.89 | 0.73 | 0.57 | 0.58 | – | – | – | – | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | 0.92 | 0.82 | 0.72 | 0.62 | – | – | – | – | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.98** | **0.75** | **0.69** | – | – | – | – | – | – | – | – |
| 5 | genQC(Fürrutter et al., 2024) | 0.95 | 0.84 | 0.68 | 0.63 | 0.61 | – | – | – | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.92 | 0.71 | 0.62 | 0.65 | – | – | – | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.98** | **0.76** | **0.70** | **0.69** | – | – | – | – | – | – | – |
| 6 | genQC(Fürrutter et al., 2024) | 0.97 | 0.86 | 0.69 | 0.62 | 0.61 | 0.61 | – | – | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | **0.96** | 0.72 | 0.63 | 0.60 | **0.70** | – | – | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.96** | **0.74** | **0.65** | **0.65** | 0.66 | – | – | – | – | – | – |
| 7 | genQC(Fürrutter et al., 2024) | 0.98 | 0.89 | 0.74 | 0.63 | 0.61 | 0.61 | 0.63 | – | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.94 | 0.73 | 0.69 | 0.62 | 0.63 | 0.65 | – | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.95** | **0.77** | **0.70** | **0.64** | **0.65** | **0.66** | – | – | – | – | – |
| 8 | genQC(Fürrutter et al., 2024) | 0.97 | 0.91 | 0.79 | 0.68 | 0.60 | 0.59 | 0.59 | 0.62 | – | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.92 | 0.80 | 0.72 | 0.64 | 0.65 | 0.63 | 0.58 | – | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.96** | **0.81** | **0.76** | **0.69** | **0.66** | **0.66** | **0.65** | – | – | – | – |
| 9 | genQC(Fürrutter et al., 2024) | 0.98 | 0.92 | 0.79 | 0.64 | 0.61 | 0.57 | 0.54 | 0.51 | 0.54 | – | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.94 | 0.83 | 0.72 | 0.63 | 0.60 | 0.55 | 0.58 | **0.59** | – | – | – |
| | **flowQC(ours)** | **1.00** | **0.96** | **0.84** | **0.75** | **0.65** | **0.61** | **0.60** | **0.61** | 0.57 | – | – | – |
| 10 | genQC(Fürrutter et al., 2024) | 0.96 | 0.93 | 0.77 | 0.67 | 0.62 | 0.55 | 0.53 | 0.49 | 0.47 | 0.50 | – | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | **0.98** | 0.82 | 0.73 | 0.65 | 0.58 | 0.52 | 0.51 | 0.52 | 0.53 | – | – |
| | **flowQC(ours)** | **1.00** | 0.97 | **0.86** | **0.79** | **0.69** | **0.64** | **0.61** | **0.61** | **0.56** | **0.54** | – | – |
| 11 | genQC(Fürrutter et al., 2024) | 0.97 | 0.92 | 0.76 | 0.65 | 0.60 | 0.57 | 0.54 | 0.53 | 0.51 | 0.50 | 0.52 | – |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.95 | 0.81 | 0.78 | 0.69 | 0.61 | 0.57 | 0.54 | 0.52 | 0.51 | 0.55 | – |
| | **flowQC(ours)** | **1.00** | **0.97** | **0.88** | **0.83** | **0.73** | **0.66** | **0.63** | **0.61** | **0.59** | **0.58** | **0.56** | – |
| 12 | genQC(Fürrutter et al., 2024) | 0.98 | 0.91 | 0.78 | 0.69 | 0.62 | 0.59 | 0.55 | 0.53 | 0.51 | 0.49 | 0.47 | 0.53 |
| | UDiTQC(Chen & Tang, 2025) | **1.00** | 0.93 | 0.77 | 0.75 | 0.71 | 0.65 | 0.61 | 0.58 | 0.57 | 0.52 | 0.51 | 0.54 |
| | **flowQC(ours)** | **1.00** | **0.97** | **0.89** | **0.86** | **0.77** | **0.70** | **0.65** | **0.62** | **0.59** | **0.58** | **0.57** | **0.55** |

**Unitary compilation** From the Figure 5, We can see the specific differences between flowQC and other methods. Compared with other methods, the quantum circuit generated by flowQC will be more compact.

## A.2 UNITARY-SYNTHESIS EVALUATION PROTOCOL AND METRICS

**Target-unitary equivalence.** For an $n$-qubit target unitary $U \in \mathbb{C}^{d\times d}$ with $d = 2^n$, a candidate circuit $C$ induces a unitary $V(C)$. Since global phase is not physically observable, we regard $C$ as correct when $V(C)$ implements $U$ up to a global phase. We use the phase-invariant process infidelity

$$\epsilon_U(U, C) = 1 - \frac{\left|\mathrm{Tr}\big(U^\dagger V(C)\big)\right|^2}{d^2}, \tag{7}$$

and accept a candidate if $\epsilon_U(U, C) \le 10^{-6}$. If $V(C) = e^{\mathrm{i}\phi}U$, then $\epsilon_U = 0$ for any global phase $\phi$. This criterion compares the complete unitary transformation rather than the output state obtained from a single input state.

**Target-level synthesis success.** For a benchmark containing $M$ target unitaries, let $\mathcal{C}_i$ be the set of candidates returned for target $U_i$. The target-level synthesis success rate is

$$\mathrm{SuccessRate} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}\left[\min_{C\in\mathcal{C}_i} \epsilon_U(U_i, C) \le 10^{-6}\right]. \tag{8}$$

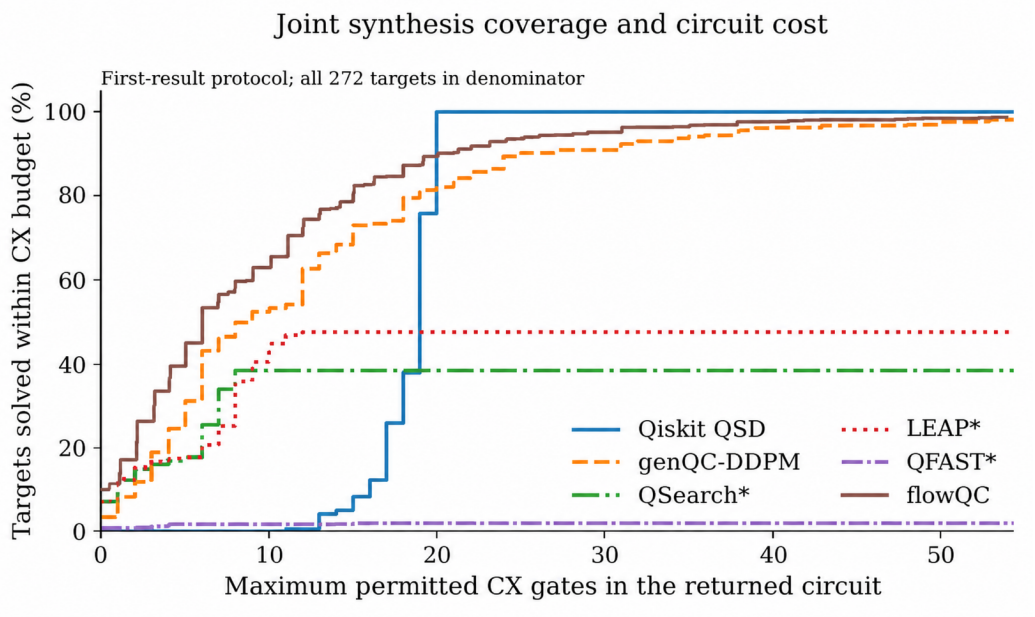



Figure 5: Joint synthesis coverage and circuit cost for the archived conventional-baseline benchmark. The denominator is fixed to all 272 targets, so unsuccessful runs contribute zero. Asterisks mark BQSKit workflows whose recorded coverage is affected by backend execution failures; their plateaus should therefore not be interpreted as intrinsic algorithmic limits.

Thus, the success rate measures target coverage and is distinct from the fraction of individual generated samples that are correct.

**Common circuit-cost basis.** To make circuit costs comparable across synthesis paradigms, verified circuits are translated to the common basis $\{u, \mathrm{CX}\}$ with Qiskit optimization level 0, all-to-all connectivity, and no routing. We report the number of CX gates and circuit depth after this common-basis translation. Using optimization level 0 avoids introducing method-dependent high-level optimization into the cost comparison.

**First-result latency.** For target $U_i$, $T_i^{\mathrm{first}}$ denotes the elapsed online time until the first verified candidate is obtained. For the generative model, this time includes batch sampling, decoding, and verification up to the first accepted candidate; model loading is excluded. For BQSKit-based methods, compiler startup is excluded. The reported "Time / Target" is the median $T_i^{\mathrm{first}}$ over successfully synthesized targets and should not be interpreted as single-candidate generation latency.

**Sample-level correctness and diversity.** For a full batch of $S$ generated candidates for target $U_i$, we define the correct-sample rate as

$$A_i = \frac{1}{S} \sum_{j=1}^{S} \mathbf{1}\left[\epsilon_U(U_i, C_{ij}) \leq 10^{-6}\right]. \tag{9}$$

The denominator contains all generated samples, including samples that fail to decode. We additionally report $D_i$, the number of distinct correct native gate sequences. Distinctness is defined syntactically using the ordered gate name, acted-on qubit indices, and gate parameters; barriers and global phase are ignored. Consequently, $D_i$ measures gate-sequence diversity and should not be interpreted as the number of physically inequivalent unitary transformations.

### A.3 Quantum Circuit Encoding

In this section, we present the procedure for encoding quantum circuits into continuous tensors. We first establish a comprehensive gate vocabulary $\mathcal{V}$ to map discrete operations into an integer index

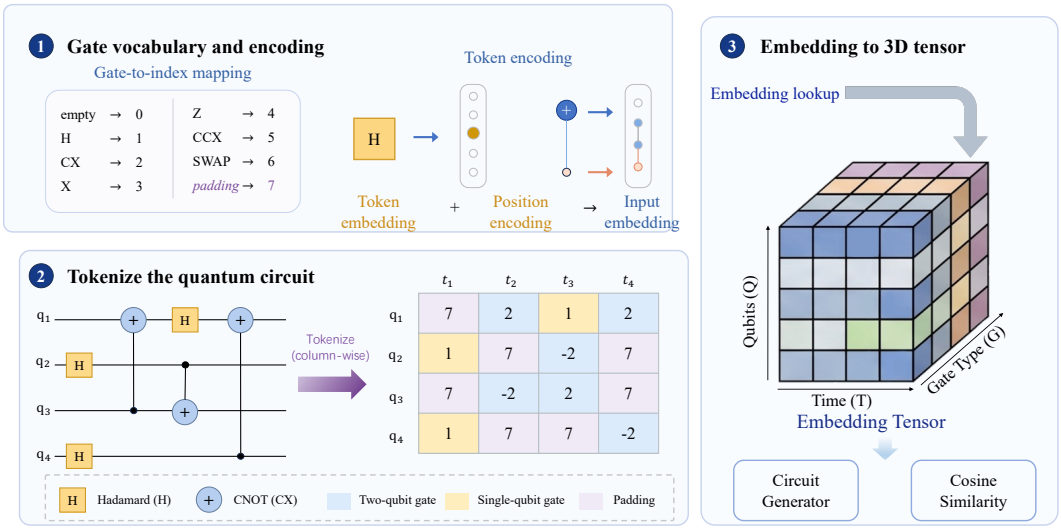



Figure 6: **The pipeline for quantum circuit encoding.** The process converts discrete gates into a 2D tokenized matrix $\mathbf{X}$ based on a predefined vocabulary, which is then mapped into a continuous 3D tensor $\mathcal{Z}$ via token and position embeddings.

representation. As illustrated in Figure 6, the vocabulary assigns specific integers to gate types: H $\Rightarrow$ 1, CX $\Rightarrow$ 2, Y $\Rightarrow$ 3, Z $\Rightarrow$ 4, CCX $\Rightarrow$ 5, and SWAP $\Rightarrow$ 6. Structural tokens are defined with "empty" as 0 and "padding" as 7. For multi-qubit gates, the control and target nodes are distinguished using opposite signs: the target node uses the positive index (e.g., 2), while the control node uses the negative index (e.g., -2). The quantum circuit is tokenized into a 2D matrix $\mathbf{X} \in \mathbb{Z}^{N\times T}$, where $N$ and $T$ denote qubits and time steps, respectively. Each element $x_{i,j}$ is defined as follows:

$$x_{i,j} = \begin{cases} k & \text{if qubit } i \text{ is the target of gate type } k \text{ at time } j \\ -k & \text{if qubit } i \text{ is the control of gate type } k \text{ at time } j \\ 0 & \text{if the position is empty} \\ 7 & \text{if the position is padding} \end{cases} \tag{10}$$

To facilitate neural network processing, the discrete matrix is mapped to a continuous 3D tensor $\mathcal{Z} \in \mathbb{R}^{N\times T\times d}$ by summing an orthogonal embedding and a spatiotemporal position encoding:

$$\mathbf{Z}_{i,j} = \text{Emb}_{token}(x_{i,j}) + \text{Emb}_{pos}(i,j) \tag{11}$$

The decoding process reconstructs the tokenized matrix from the model's continuous output $\mathbf{v}_{gen}$ using Cosine Similarity $S_C$:

$$S_C(\mathbf{v}_k, \mathbf{v}_{gen}) = \frac{\mathbf{v}_k \cdot \mathbf{v}_{gen}}{\|\mathbf{v}_k\|\|\mathbf{v}_{gen}\|} \tag{12}$$

The gate type $\tilde{k}$ and its specific node role $k$ are determined by:

$$\tilde{k} = \arg\max_{k\in\mathcal{V}} |S_C(\mathbf{v}_k, \mathbf{v}_{gen})|, \quad k = \tilde{k} \cdot \text{sign}\left(S_C(\mathbf{v}_k, \mathbf{v}_{gen})\right) \tag{13}$$

### A.4 Conditioning

As a conditional flow matching framework, flowQC requires robust encoding mechanisms to inject specific target features (such as entanglement properties or unitary matrices) into the model. This conditioning actively steers the vector field prediction during the generation process. In this section, we detail the embedding strategies for two distinct downstream tasks.

### A.4.1 TEXT-CONDITIONED SRV EMBEDDING

For the entanglement generation task, the objective is to synthesize quantum circuits that conform to a specific Schmidt Rank Vector (SRV). Unlike traditional conditional models that rigidly map dataset labels into discrete, finite integer classes via a standard label embedder, we formulate this task within a text-to-circuit paradigm.

Specifically, the target SRV is represented as a natural language text prompt (e.g., the string "[1,2,2]"). We leverage a pre-trained OpenCLIP (ViT-B-32) text encoder to process these strings, projecting the discrete SRV conditions into a rich, continuous semantic vector space. This prompt-based embedding not only enhances the semantic richness of the condition but also facilitates better generalization. Furthermore, to enable Classifier-Free Guidance (CFG) Ho & Salimans (2021) during the sampling phase, we employ a condition dropout strategy, randomly discarding the text embedding with a probability of $p_{drop} = 0.1$ during training.

### A.4.2 MATRIX-CONDITIONED UNITARY EMBEDDING

For the unitary compilation task, the condition is a dense, complex-valued unitary matrix $\mathbf{U}$. To effectively ingest this mathematical structure, we design a dedicated Unitary Encoder (U-enc).

The pipeline begins by decoupling the input unitary matrix into its real and imaginary components, constructing a two-channel spatial tensor. Following an initial convolutional projection, a 2D positional encoding layer is applied to explicitly preserve the absolute spatial coordinates of the matrix elements. Subsequently, the feature map is processed by a Transformer encoder. The self-attention mechanism within the Transformer is strictly necessary here, as the operational semantics and phase information of a unitary matrix are highly non-local and globally correlated. Finally, a $2 \times 2$ downsampling layer and subsequent convolutional blocks are utilized to compress the global features, ensuring the extracted unitary representation perfectly aligns with the hidden dimensions of the primary flow matching backbone.

## A.5 DATASETS

In this section, we descirbe the dataset generation process for the tasks about entanglement generation and unitary compilation.

To successfully train continuous-time generative models, we require massive datasets comprising hundreds of thousands of quantum circuits paired with their corresponding conditions (i.e., SRV labels for entanglement generation and target unitary matrices for unitary compilation). However, generating and verifying quantum semantic features at such a scale presents a severe computational bottleneck. Traditional dataset generation paradigms typically rely on full density matrix simulations . This approach suffers from an intractable exponential spatial and temporal complexity of $\mathcal{O}(4^N)$. we replaced the Qiskit DensityMatrix-based pipeline used in prior work with a Statevector-based implementation, supporting larger-scale training and evaluation of generated circuits across 3–12 qubits.

### A.5.1 GENERATION OF CIRCUITS

For the entanglement generation task, we systematically generate random quantum circuits for a fixed number of qubits $N$. The generation pipeline begins by sampling the circuit depth from a uniform distribution bounded by specified minimum and maximum limits. Subsequently, quantum operations are uniformly sampled from a predefined gate pool and assigned to randomly selected qubits. To ensure the structural validity and non-triviality of the generated dataset, we apply a lightweight algebraic optimization pass to eliminate redundant adjacent operations. Once the optimized circuit is constructed, we leverage the highly efficient Schmidt-Gram Contraction (SGC) strategy to compute its corresponding Schmidt Rank Vector (SRV). By operating directly on the pure statevector array and avoiding the explicit construction of the global density matrix, the SGC strategy rapidly isolates the $2 \times 2$ reduced density matrices to determine the exact entanglement ranks. Finally, the resulting SRV is formatted into a discrete text string, such as `"[1,2,2]"`, which is seamlessly embedded as a semantic condition to guide the subsequent training of the generative models.

For the unitary compilation task, we generate all possible subsets of gates from the gate pool using lexicographical ordering, where these subsets correspond to the labels in flowQC. Based on the chosen gate subsets, circuits are generated randomly for a given number of qubits and gates, and the corresponding unitary is computed as an additional condition for flowQC.

### A.5.2 Dataset Balancing Strategies

An inherent challenge in uniform random circuit generation is the severe class imbalance within the resulting SRV distributions. Specifically, circuits characterized by trivial or maximally entangled states (e.g., SRVs consisting entirely of 1s or 2s) naturally emerge with disproportionately high frequencies. To prevent the generative models from collapsing into these majority modes, we implement a strict balancing protocol prior to training. We downsample the over-represented categories such that each distinct SRV class maintains an identical number of unique training samples, thereby guaranteeing an unbiased learning objective for the flow matching model.

For the unitary compilation task, the dataset must be balanced across different user-specified gate pools (subsets). We establish a target capacity (balance size) for each specific gate configuration. For highly expressive subsets, we randomly sub-sample the generated circuits to meet this exact threshold. Conversely, for highly constrained gate subsets where the total number of topologically unique circuits falls short of this predefined target, we strictly retain all available non-duplicate circuits to maximize data utilization without introducing synthetic redundancy.

### A.5.3 Dataset Construction

Table 7: Training set parameters for tasks.

| Task | Gate Pool | Qubits | Min Gates | Max Gates | Labels | Balanced Size | Total Circuits |
|---|---|---|---|---|---|---|---|
| Entanglement generation | H, CX | 3 | 2 | 16 | 5 | 69200 | 346000 |
| | | 4 | 3 | 20 | 12 | 36000 | 432000 |
| | | 5 | 4 | 28 | 27 | 20000 | 540000 |
| | | 6 | 5 | 40 | 58 | 9600 | 556800 |
| | | 7 | 6 | 52 | 121 | 4920 | 595320 |
| | | 8 | 7 | 52 | 248 | 2500 | 620000 |
| | | 9 | 8 | 52 | 503 | 1280 | 643840 |
| | | 10 | 9 | 52 | 1014 | 640 | 648960 |
| | | 11 | 10 | 52 | 2037 | 360 | 733320 |
| | | 12 | 11 | 52 | 4084 | 180 | 735120 |
| Unitary Compilation | H, CX, Z, X, CCX, SWAP | 3 | 2 | 12 | - | - | 500000 |

The generation of dataset were performed on a cluster equipped with NVIDIA 5090 GPUs.

## A.6 Training and Testing Setting for FlowQC

**flowQC training setup.** We train our proposed flowQC and the ablated flowQC−PAQ-former using the following shared configuration on a single NVIDIA A100-SXM4-80GB GPU. The baseline genQC differs solely in its generative paradigm and inference sampling.

- **Optimizer & LR schedule:** Adam ($\beta_1 = 0.9, \beta_2 = 0.999$) with an initial learning rate $\eta_{\text{init}} = 3 \times 10^{-4}$, smoothly decayed by a step-wise CosineAnnealingLR schedule to $\eta_{\text{min}} = 10^{-6}$ over the entire training process.
- **Batch size & Dataloader:** The batch size is set to 256. To ensure stable GPU memory allocation and prevent runtime errors during distributed processing, the dataloader is explicitly configured with `pin_memory=False` and `num_workers=0`.
- **Training length:** 100 epochs.
- **Loss function & Time sampling:** Flow matching MSE Loss ($\|v_\theta - (x_1 - x_0)\|_2^2$) with time steps uniformly sampled as $t \sim \mathcal{U}(0, 1)$.

- **Conditioning & CFG:** Text prompts are encoded using OpenCLIP ViT-B-32. To enable classifier-free guidance during inference, condition dropout is applied with a probability of $p_{\text{drop}} = 0.1$.
- **Backbone width/depth:** The U-Net backbone features 3 downsampling scales with `model_features` = $[128, 128, 256]$. The integrated Transformer blocks have depths of $[1, 2, 1]$ and corresponding attention heads of $[8, 8, 2]$. Residual blocks are configured as $[2, 2, 4]$, utilizing a Max Padding strategy.

**flowQC testing setup.** In this part, we describe in detail the hyperparameters in each experimental part of the text. We use NVIDIA A100-SXM4-80GB GPU for reasoning test:

- **Entanglement Generation Efficiency:** max_gates = 32, guidance_scale = 5, num_samples =512
- **Circuit Quality:** max_gates = 52, guidance_scale= 1-5, num_samples = 1024, sampling_steps = 5-20
- **Unitary compilation:** samples = 1024, num_of_U = 272, system_size = 12, gate_pool = [X, Z, CX, H, CCX, SWAP], num_of_qubits = 3, max_gates = 12, guidance_scale = 7.5, rnd_min_gates = 3, rnd_max_gates = 12
- **Ablation Study:** max_gates = 52, guidance_scale= 7.5, num_samples = 1024

### A.7 Consistency under randomized evaluation

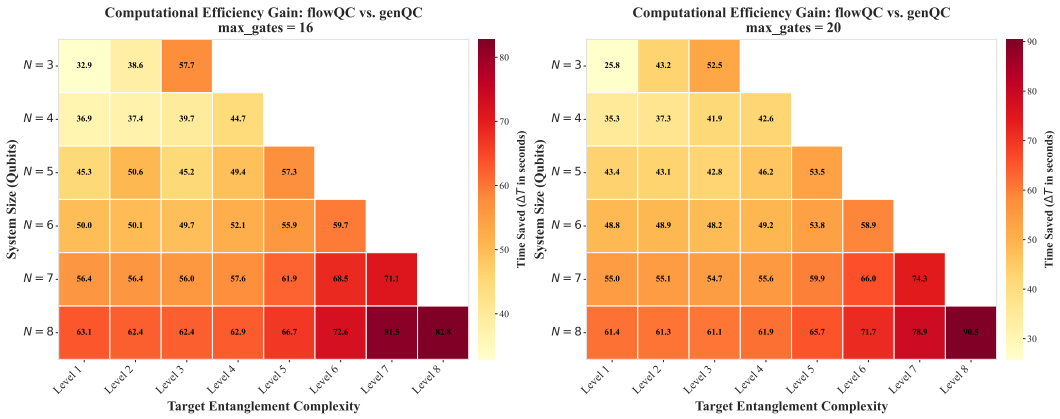



Figure 7: **max_gates=16,max_gate=20.**Comparison of joint generation efficiency ($T_{\text{cost}}$) across varying system sizes and entanglement levels.

In generative models for quantum circuits, the hyperparameter `max_gates` plays a pivotal role during the inference phase. It explicitly defines the upper bound of the generated sequence length (e.g., constraining the circuit to contain a maximum of 16 gates). From a systemic perspective, this parameter dictates the spatial dimensionality of the generated spatiotemporal tensor. Consequently, larger `max_gates` values inherently expand the search space, typically leading to significant inference latency and a degradation in sampling efficiency for traditional frameworks.

To rigorously validate that the efficiency of our proposed **flowQC** is not merely an artifact of a specific sequence length setting, we conduct a supplementary evaluation across randomized length constraints. In this section, we systematically vary the `max_gates` hyperparameter and evaluate the models using the *joint efficiency cost* metric $T_{\text{cost}}$. Our objective is to investigate whether flowQC consistently maintains its substantial efficiency superiority over the baseline **genQC**, regardless of the maximum generation length imposed during inference.

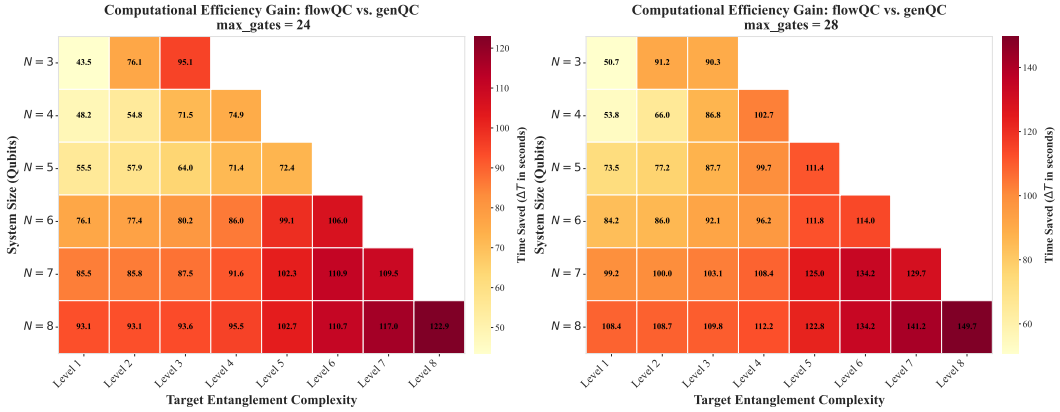



Figure 8: **max_gates=24,max_gate=28.**Comparison of joint generation efficiency ($T_{\text{cost}}$) across varying system sizes and entanglement levels.

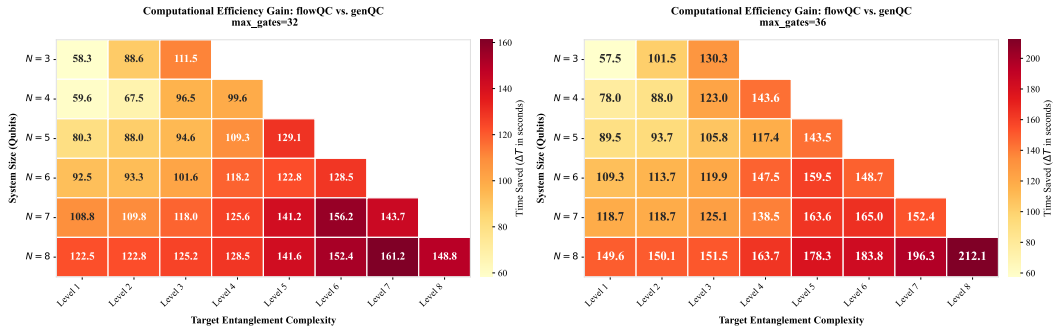



Figure 9: **max_gates=32,max_gate=36.**Comparison of joint generation efficiency ($T_{\text{cost}}$) across varying system sizes and entanglement levels.