



**Master in Quantum
Science and Technology
Barcelona**



Master Thesis in Quantum Science and Technology

Probabilistic Simulation of Deep Quantum Circuits and Applications to Shor's Algorithm

Author:

Helena Monsó Mas

Supervised by:

Edward Jiang, Tim Heightman, Antonio Acín

Universitat de Barcelona
Facultat de Física

Institut de Ciències Fotòniques
Quantum Information Theory Group

September 2026

Probabilistic Simulation of Deep Quantum Circuits and Applications to Shor's Algorithm

Helena Monsó Mas

Supervised by: Edward Jiang, Tim Heightman, Antonio Acín

Quantum Information Theory Group, ICFO, 08860, Castelldefels

September 2026

The simulation of deep quantum circuits is traditionally bottlenecked by the exponential scaling of the Hilbert space. This thesis presents a classical simulation framework that circumvents this barrier by mapping quantum states to classical probability distributions using informationally complete positive operator-valued measures (IC-POVMs). By leveraging an autoregressive transformer architecture, this work redefines quantum simulation as a high-dimensional generative modeling task, utilizing the self-attention mechanism to capture non-local multi-qubit correlations. To rigorously benchmark the model against volume-law entanglement, the circuits are generated dynamically using uncompiled logical primitives, including the Cuccaro ripple-carry adder routed through the Vedral-Barenco-Ekert (VBE) modular exponentiation architecture. Evaluated via a variance-stabilized mixed Monte Carlo framework, the transformer successfully simulated a 25-qubit quantum Fourier transform (QFT) applied to an entangled 1D cluster state, maintaining a classical fidelity of $F_c > 0.99$ over 325 gates. Furthermore, the model executed the complete, uncompiled Shor's algorithm to factor $N = 15$. Across a depth of 301 gates and 22 qubits, the network sustained a classical fidelity of $F_c > 0.97$. These results demonstrate that attention-based neural networks can robustly track complex, reversible quantum arithmetic without suffering exponential error accumulation, thereby establishing a stricter classical baseline for proving practical quantum advantage.

Keywords: Classical Simulation, Deep Quantum Circuits, Shor's Algorithm, Transformers, Autoregressive Models, IC-POVM, Modular Exponentiation, Quantum Fourier Transform.

Acknowledgements

Firstly, I would like to thank my supervisors, Edward and Tim, for all our meetings, your invaluable advice, and the knowledge you shared with me. I never doubted we would make a great team, and not once have I thought otherwise. Hearing “Great job, as always” at the end of each sync has had a greater impact on me than you probably realize, and I am truly grateful for all your encouragement and kind words.

To Toni, thank you for letting me be part of your incredible group, it has certainly been one of the highlights of my (still short) life, and a memory I will always cherish. And to the QIT group, thank you for welcoming me with open arms. Feeling comfortable in such a large group is not easy, but you made it happen. Special thanks to all my *fútbol* professors; I arrived as an amateur, and I leave having successfully mastered the slingshot! Hope you are proud.

To my friends, thank you for all the hours you spent listening to me, even when you couldn’t understand a word I was saying, and for always reminding me what life really is about.

To my parents, thank you for your unwavering support. The person you believe I am is the person I strive to become. There is no greater feeling than making you proud.

Marina, you are a constant inspiration to me. You might be the person who carries the greatest responsibilities in my life, yet somehow you make it all seem light. Your optimism and the way you face every challenge are qualities I will always admire. And to Tommy, my mentor, thanks for showing me the path. I now aspire to follow in your footsteps, if I make it even halfway, I’ll consider myself lucky.

And finally, to David: you made the best out of an unpredictable, overwhelming, and volatile year. Thank you for helping me make difficult decisions, for bringing calm when I needed it most, for cheering me on, and for always being by my side. I know I will always be by yours, no matter where.

Contents

1	Introduction	1
1.1	The Shifting Landscape of Quantum Advantage	1
1.2	Escaping the Hilbert Space: From Amplitudes to Probabilities	1
1.3	Neural Network Quantum States and the Transformer Architecture	2
1.4	Scope, Objectives and Structure of the Thesis	2
2	Theoretical Background	3
2.1	Informationally Complete POVMs	3
2.2	Translation to Probability Space	4
2.3	Deep Learning Framework: Transformers	5
2.4	Transformation to the Computational Z-Basis	8
2.5	Shor's Algorithm	9
2.6	Modular Exponentiation	10
2.7	Period Extraction and the QFT^{-1}	11
3	Probabilistic Simulation	13
3.1	Simulation Methodology and Implementation	13
3.1.1	QFT Benchmark	13
3.1.2	Modular Exponentiation Benchmark	14
3.2	Results and Discussion	14
3.2.1	QFT	15
3.2.2	Modular Exponentiation	16
3.2.3	Factoring $N=15$	18
4	Conclusion	19
	Bibliography	21
	References	22
A	Derivation of the Pseudo-Bistochastic Transition Matrix	23
B	Proof of Exact Sampling in Autoregressive Quantum Circuit Simulation	24
C	Basis Translation and Scalable Monte Carlo Estimation	25
C.1	The Translation Map: Deriving W_Z	25
C.2	The Curse of Dimensionality	25
C.3	The Monte Carlo Rescue	26
C.4	Marginalization: Extracting Local Observables	26
D	Simulation Procedure and Specifications	27
D.1	Framework and Precision Architecture	27
D.2	Algebraic Formulation	27
D.3	Neural Network Architecture	28
D.4	Optimization and Training Loop	28
D.5	JAX Implementation and Vectorization Details	29
D.6	Scalable Verification (OOM-Proof Branch Tracking)	29

1 Introduction

1.1 The Shifting Landscape of Quantum Advantage

The pursuit of *quantum advantage*, the definitive demonstration that a quantum device can perform a computational task beyond the reach of any classical supercomputer, remains the central objective of the second quantum revolution [1]. For decades, the theoretical foundation of this advantage has rested on the exponential nature of the Hilbert space. Because the state vector of an N -qubit system requires 2^N complex amplitudes to fully describe, classical simulation using direct matrix-vector multiplication rapidly becomes intractable. This exponential bottleneck has long been viewed as the impenetrable barrier safeguarding the quantum advantage promise.

However, the boundary between classical and quantum computational capabilities is not static; it is an active frontier subject to intense competition. As quantum hardware struggles with the decoherence and error rates inherent to the noisy intermediate-scale quantum (NISQ) era [2, 3], classical simulation techniques have grown remarkably sophisticated. Current state-of-the-art methods encompass a variety of approaches: tensor networks provide highly efficient ansätze for systems with low entanglement [4, 5, 6, 7], while techniques like Pauli propagation and stabilizer-based formalisms excel at simulating Clifford-dominated circuits [8, 9, 10]. Furthermore, neural network quantum states (NNQS) have emerged as a powerful paradigm for tackling both static and complex quantum dynamics [11]. Rather than conceding the quantum advantage, a growing sector of the research community focuses on identifying alternative mathematical representations and algorithmic shortcuts to efficiently simulate deep quantum circuits on classical hardware [12]. By pushing classical methods to their absolute limits, we establish the true, rigorous baseline that physical quantum devices must surpass to claim practical supremacy.

1.2 Escaping the Hilbert Space: From Amplitudes to Probabilities

The traditional roadblock to classical simulation is the necessity of tracking *complex* amplitudes to preserve quantum interference. To bypass this, we must reconsider how the quantum state is represented. The fundamental insight of this work is to step outside the standard state-vector formalism and re-cast quantum mechanics entirely within the domain of classical probability theory.

By utilizing an informationally complete positive operator-valued measure (IC-POVM), any quantum state can be mapped to a classical probability distribution over a set of measurement outcomes. For a system of N qubits, local IC-POVMs generate a probability distribution over 4^N possible outcomes. While the state space expands from 2^N to 4^N , this transformation provides a critical advantage: the state is no longer a superposition of complex amplitudes subject to unitary evolution, but a classical joint probability distribution subject to quasi-stochastic transitions [13].

By escaping the Hilbert space and entering the probability domain, the problem of quantum simulation is fundamentally redefined. The challenge is no longer performing linear algebra on exponentially large vectors; rather, it becomes a problem of high-dimensional generative modeling, a domain where modern machine learning excels.

1.3 Neural Network Quantum States and the Transformer Architecture

The idea of leveraging machine learning to approximate the many-body wavefunction was pioneered through the introduction of NNQS [14], which initially utilized restricted boltzmann machines (RBMs) to find ground states of quantum many-body systems [15]. Subsequently, NNQS frameworks were extended beyond static problems to simulate complex real-time quantum dynamics [16]. Other neural network architectures have been successfully adapted for quantum simulation, such as convolutional neural networks (CNNs), which have been employed to capture local correlations in 2D quantum lattices [17]. This paradigm was significantly advanced by Carrasquilla et al. [18], who demonstrated that autoregressive generative models could efficiently simulate quantum circuits by learning the IC-POVM probability distributions [19, 20, 21].

Building upon the foundation laid by [18], one can depart from traditional recurrent neural networks (RNNs) in favor of the transformer architecture. While traditional simulation methods like tensor networks [4] offer powerful, physically motivated ansatzes for systems with low entanglement (satisfying area laws), they struggle to efficiently contract deep circuits that generate volume-law entanglement, such as Shor’s algorithm. Conversely, while RNNs can model probability distributions sequentially, they suffer from vanishing gradients and struggle to capture the long-range, non-linear multipartite correlations generated by deep quantum arithmetic [22].

The transformer architecture, characterized by its self-attention mechanism [23], provides a unique solution. It does not enforce a rigid geometry like a tensor network, nor does it suffer from the sequential memory bottlenecks of an RNN. Instead, it maintains a global receptive field, allowing it to directly map the complex, non-local entanglement structures that arise during modular exponentiation.

1.4 Scope, Objectives and Structure of the Thesis

While previous works have successfully utilized NNQS to simulate shallow circuits or systems adhering to area-law entanglement [14, 18], this work investigates the capacity of transformer-based models to simulate *deep, highly entangled* quantum algorithms. The scope of this research is restricted to the uncompiled simulation of Shor’s factoring algorithm [24], encompassing both the quantum Fourier transform (QFT) and the modular exponentiation sequence. To force the network to navigate the extreme limits of state-space complexity, we establish two primary objectives:

- *Dynamic uncompiled circuit generation.* Moving beyond simplified lookup-table simulations by dynamically generating the quantum circuits using base logical primitives. This includes implementing the QFT as a foundational benchmark, followed by the rigorous Cuccaro ripple-carry adder [25] routed through the Vedral-Barenco-Ekert (VBE) modular exponentiation architecture [26].
- *Expansion of the IC-POVM autoregressive framework.* Accommodating multi-qubit arithmetic by deriving and implementing generic 3-qubit target transition matrices. This allows the network to natively learn complex multi-qubit correlations and simulate the Toffoli operations critical to the reversible addition phase of the algorithm.

By evaluating these implementations through a variance-stabilized mixed Monte Carlo framework, we demonstrate that transformers can successfully track the reversible algorithmic trajectory of Shor’s algorithm. This challenges the assumed boundaries of classical simulation and redefines the benchmarks required for true quantum advantage.

The remainder of this document is structured as follows. Chapter 2 provides the theoretical background, introducing the mathematical formulation of IC-POVMs, the principles of neural network quantum states, and the underlying logic of the transformer architecture. Chapter 3 details the proposed methodology, including the dynamic generation of the QFT and VBE arithmetic circuits, as well as presenting our performance metric, before showing the results. These results validate the model’s performance on the QFT and then analyze the fidelity dynamics for the full modular exponentiation for different system sizes ($N = 3, 4, 8$), culminating in the factorization of $N = 15$, implementing the whole Shor’s algorithm. Finally, Chapter 4 summarizes the conclusions drawn from this research and outlines potential avenues for future work in the pursuit of classical quantum simulation.

2 Theoretical Background

2.1 Informationally Complete POVMs

The representation of quantum dynamics via pseudostochastic maps is both equivalent to the regular Hilbert space representation [13], and enables classical probabilistic models to simulate them. Here we review the key theory that underpins this transition.

A quantum state, typically described by a density matrix ρ , can be uniquely associated with a probability distribution through *informationally complete positive operator-valued measures* or IC-POVMs for short.

Definition 1. A *POVM* is a set of m positive semidefinite operators $\{M_\alpha\}_{i=\alpha}^m$ satisfying $\sum_\alpha M_\alpha = \mathbb{I}$.

Definition 2. A POVM is *informationally complete* if its elements span the space of linear operators of our Hilbert space, $\mathcal{L}(\mathcal{H})$, meaning the outcome statistics $p_\alpha = \text{Tr}(\rho M_\alpha)$ unambiguously specify ρ [27].

This representation allows quantum states and their evolution to be described using the language of probability theory. By mapping ρ to a probability vector $\mathbf{p}$, we transition from complex Hilbert space to a *real probability space*.

Definition 3. For a d -dimensional system, a *symmetric IC-POVM* (SIC-POVM) is a minimal representation consisting of d^2 elements. These elements are constructed from normalized states $\{|\psi_\alpha\rangle\}_{\alpha=1}^{d^2}$ such that the POVM elements are $M_\alpha = \frac{1}{d}\Pi_\alpha$, where $\Pi_\alpha = |\psi_\alpha\rangle\langle\psi_\alpha|$. A *SIC-POVM* satisfies the specific symmetry condition regarding operator overlap:

$$\text{Tr}(\Pi_\alpha \Pi_\beta) = \frac{d\delta_{\alpha\beta} + 1}{d + 1}, \quad (2.1)$$

where $\delta_{\alpha\beta}$ stands for the Kronecker delta [28].

Note that the set of projectors $\{\Pi_\alpha\}_{\alpha=1}^{d^2}$ forms a basis in the space $\mathcal{L}(\mathcal{H})$ of linear operators acting on $\mathcal{H}$. Therefore, SIC-POVMs are particularly efficient because they contain no redundant information: the number of probabilities (d^2) is the absolute minimum required to reconstruct all elements of the density matrix. Although SIC-POVMs possess global equiangular symmetry, there are other minimal informationally complete POVMs (MIC-POVMs) that satisfy this dimensional lower bound without being symmetric [29].

For a single qubit ($d = 2$), we utilize the SIC-POVM projectors defined as:

$$\begin{aligned}\Pi_1 &= |\psi_1\rangle\langle\psi_1| = \frac{1}{2} \left[\mathbb{I} + \frac{1}{\sqrt{3}}(X - Y + Z) \right], \\ \Pi_2 &= |\psi_2\rangle\langle\psi_2| = \frac{1}{2} \left[\mathbb{I} + \frac{1}{\sqrt{3}}(X + Y - Z) \right], \\ \Pi_3 &= |\psi_3\rangle\langle\psi_3| = \frac{1}{2} \left[\mathbb{I} + \frac{1}{\sqrt{3}}(-X + Y + Z) \right], \\ \Pi_4 &= |\psi_4\rangle\langle\psi_4| = \frac{1}{2} \left[\mathbb{I} + \frac{1}{\sqrt{3}}(-X - Y - Z) \right].\end{aligned}\tag{2.2}$$

These form a regular tetrahedron inscribed within the Bloch sphere, as shown in Fig. 1.

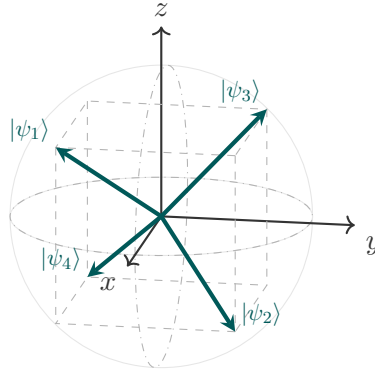


Figure 1: Bloch vectors corresponding to the SIC-POVM projectors $\{|\psi_\alpha\rangle\langle\psi_\alpha|\}_{\alpha=1}^4$ from Eq. (2.2).

2.2 Translation to Probability Space

From now on, Greek letters will indicate the indices of the POVM elements: $\alpha \in \{1, \dots, d^2\}$, and Latin alphabet letters will be used to indicate the qubit index $k \in \{1, \dots, N\}$.

To simulate quantum circuits, we must map quantum gates to transformations between probability distributions. For N qubits, we construct the IC-POVM basis as a tensor product of single-qubit elements: $\mathbb{M} = \{\bigotimes_{k=1}^N M_{\alpha_k}^{(k)}\}$. While this set remains informationally complete, it generally *loses the global symmetry* of a SIC-POVM.

Example 1. *The loss of symmetry.* While single-qubit SIC-POVMs are equiangular, their tensor products generally lose this global symmetry and yield varying off-diagonal overlaps. For instance, consider the projectors of a two-qubit system constructed via tensor products. While the diagonal overlap remains unity: $\text{Tr}(\Pi_1 \otimes \Pi_1 \cdot \Pi_1 \otimes \Pi_1) = 1$, the off-diagonal overlaps may vary depending on the number of mismatched indices:

$$\text{Tr}(\Pi_1 \otimes \Pi_1 \cdot \Pi_1 \otimes \Pi_2) = 1 \cdot \frac{1}{3} = \frac{1}{3}, \text{ and } \text{Tr}(\Pi_1 \otimes \Pi_2 \cdot \Pi_3 \otimes \Pi_4) = \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{9}.$$

In contrast, for a true 2-qubit SIC-POVM (where the system dimension is $D = d^2 = 4$), all off-diagonal overlaps must equal a strict constant of $1/(D + 1) = 1/5$. And since $1/3 \neq 1/9 \neq 1/5$, the tensor product basis is informationally complete but lacks the symmetry of a true SIC-POVM. Thus, when doing the translation into the probability space, one must keep in mind that one-qubit systems and higher-dimensional systems are two different situations.

The transition of a density matrix ρ to a probability vector $\mathbf{p}$ is handled via a linear transformation $|\rho\rangle\rangle = K\mathbf{p}$, which is specified in [13]. Under the action of a unitary gate U , the evolution is tracked as $\mathbf{p}^{\text{out}} = S\mathbf{p}^{\text{in}}$. For a d -dimensional system using SIC-POVMs, the *pseudo-bistochastic matrix* S is defined by

$$S_{\alpha,\beta} = (d+1)s_{\alpha,\beta} - \frac{1}{d}, \text{ where } s_{\alpha,\beta} = \frac{1}{d}\text{Tr}\left(U\Pi_\beta U^\dagger\Pi_\alpha\right), \quad (2.3)$$

the derivation of this formula can be found in Appendix A. S is pseudo-bistochastic because its columns and rows sum to unity, but it may contain negative elements, which serve as signatures of non-classicality. Additionally, and as a consequence of the symmetry, matrix S is also orthogonal.

When dealing with multi-qubit gates and IC-POVMs, we adopt the approach from [18], which employs an overlap matrix T to reconstruct the density matrix:

$$\rho = \sum_{\alpha,\beta} p_\beta T_{\alpha,\beta}^{-1} M_\alpha, \text{ with } T_{\alpha,\beta} = \text{Tr}(M_\alpha M_\beta). \quad (2.4)$$

The evolution after applying a quantum gate U is governed by the matrix O :

$$p_\alpha = \text{Tr}(U\rho U^\dagger M_\alpha) = \sum_\beta O_{\alpha,\beta} p_\beta, \text{ with } O_{\alpha,\beta} = \sum_\gamma \text{Tr}(U M_\gamma U^\dagger M_\alpha) T_{\gamma,\beta}^{-1}. \quad (2.5)$$

The O matrices remain pseudo-stochastic but are no longer orthogonal. This probabilistic formulation allows the simulation of a quantum circuit to be treated as a sequence-to-sequence task suitable for transformer architectures.

2.3 Deep Learning Framework: Transformers

Having mapped the dynamics of a quantum system into the evolution of discrete probability distributions, the simulation of a quantum circuit naturally frames itself as a sequence-to-sequence prediction task. Given an initial probability distribution over POVM outcomes, the goal is to predict the final distribution after the application of a sequence of gates.

We employ the *transformer* architecture for this job, as it excels at capturing long-range dependencies [23], a critical feature when simulating multi-qubit entanglement across distant physical qubits. This unique capability to process vast context windows and track complex relationships is precisely what makes transformers the foundational architecture behind modern large language models (LLMs) [30, 31].

In natural language processing, a transformer processes sequences of words by mapping them into discrete *tokens*. In the context of our pseudostochastic quantum simulation, the vocabulary consists of the discrete measurement outcomes of our IC-POVMs. Recalling the notation from the previous section, for an N -qubit system, the state is characterized by the measurement probabilities of tensor product $\bigotimes_{k=1}^N M_{\alpha_k}^{(k)}$. Here the specific outcome indices $\alpha_k \in \{1, 2, 3, 4\}$ act directly as our tokens. The probability of a specific sequence of outcomes is modeled *autoregressively* via the chain rule:

$$p_\theta(\alpha_1, \dots, \alpha_N) = \prod_{k=1}^N p_\theta(\alpha_k | \alpha_{<k}). \quad (2.6)$$

To enforce this causal structure during training, a *mask* is applied within the multi-head attention mechanism. This ensures that when predicting the probability of the outcome for qubit k , the network cannot access the information from subsequent outcomes $\alpha_{>k}$.

The application of a sequence of quantum gates defines a trajectory of probability distributions. Let $\mathbf{p}_0$ be the initial distribution, the exact application of a gate $U^{(i+1)}$ yields $\mathbf{p}_{i+1} = O^{(i+1)}\mathbf{p}_i$. However, tracking this exact probability distribution directly requires performing a matrix-vector multiplication of size $\mathcal{O}(4^N)$, which scales exponentially and quickly becomes intractable. To fix this, we approximate the intermediate states using parameterized transformer models $\mathbf{p}_{\theta_i}$. Because the transformer allows for exact sampling¹, we can optimize the model iteratively gate-by-gate using two primary training strategies which are shown in Fig. 2:

1. *Kullback-Leibler (KL) Divergence Optimization.* Assuming the model has accurately captured the state at step i , i.e. $\mathbf{p}_{\theta_i} \approx \mathbf{p}_i$, we generate the exactly evolved target distribution for the next step as $\mathbf{p}_{i+1}^{(e)} = O^{(i+1)}\mathbf{p}_{\theta_i}$. We then train the next model: $\mathbf{p}_{\theta_{i+1}}$, by minimizing the KL divergence [32],

$$KL(\mathbf{p}_{i+1}^{(e)} || \mathbf{p}_{\theta_{i+1}}) = - \sum_{\alpha} \mathbf{p}_{i+1}^{(e)}(\alpha) \ln \left(\frac{\mathbf{p}_{\theta_{i+1}}(\alpha)}{\mathbf{p}_{i+1}^{(e)}(\alpha)} \right). \quad (2.7)$$

2. *Forward-Backward algorithm.* A quantum gate can be decomposed as $U = U_1 U_1^\dagger$ with $U_1 = \sqrt{U}$. Thus, the corresponding POVM transformation O_1 acts as a “half-gate.” The forward evolution $O_1 \mathbf{p}_{\theta_i}$ must logically match the backward evolution $O_1^\dagger \mathbf{p}_{\theta_{i+1}}$. The model can also be optimized by minimising the L_1 norm of their difference, $C = \|O_1 \mathbf{p}_{\theta_i} - O_1^\dagger \mathbf{p}_{\theta_{i+1}}\|_1$.

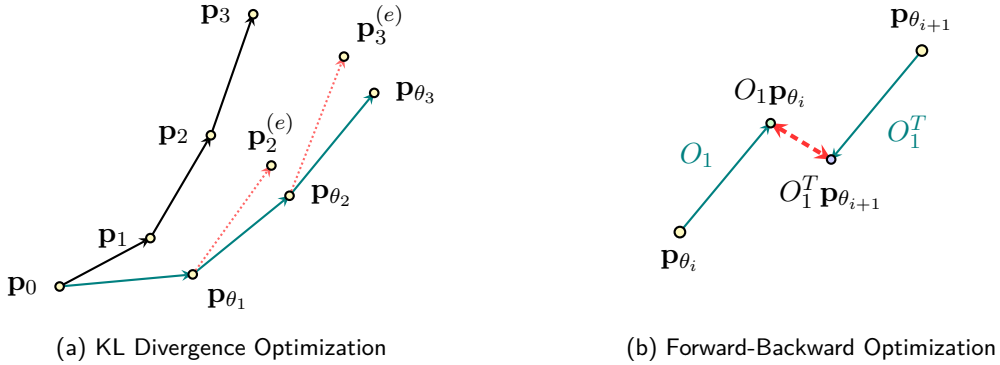


Figure 2: Schematics of the two training procedures. (a) KL divergence optimization. Exact theoretical trajectory (black), optimized models (green), and gate-evolved targets (orange) after applying $O^{(i+1)}$ to the previous model. (b) Forward-backward training. The L_1 distance between the forward-evolved $\mathbf{p}_{\theta_i}$ and the backward-evolved $\mathbf{p}_{\theta_{i+1}}$ is minimized.

By tracking the circuit evolution through these local, sample-efficient gradient updates, the transformer compresses the exponentially large probability distributions into a polynomial number of network parameters. This is because the pseudostochastic matrices $O^{(i+1)}$ are highly local (acting only on k qubits), this locality makes gradient updates efficient while the attention mechanism allows the network to represent the state effectively. More specifically, the gradients of these losses can be estimated using N_s sample generated exactly from the transformer. This drops the computational cost of an update from $\mathcal{O}(4^N)$ to $\mathcal{O}(N_s 4^k)$, where k is the locality of the gate.

¹This is shown in Appendix B.

The core innovation of the transformer is the *self-attention* mechanism [23]. Rather than processing the sequence of POVM tokens strictly left-to-right, self-attention allows the model to weigh the importance of every token in the sequence relative to every other token simultaneously. Mathematically, the input embeddings are linearly projected using *trainable* weight matrices: queries (Q), keys (K), and values (V). The attention weights are computed using the scaled dot-product:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V, \quad (2.8)$$

where d_k is the dimension of the key vectors. The softmax function ensures that the attention weights sum to one, conceptually aligning with our probabilistic framework.

In the context of our quantum circuit, if a CNOT gate entangles qubit 1 and qubit N , the self-attention mechanism allows the token α_1 to directly attend to the token α_N without the distance-based information loss, specifically the vanishing gradient problem, that severely limits classical sequential models [22].

To concretize the training process, consider the simplest possible scenario: a single-qubit system ($N = 1$) initialized in the state $\rho = |0\rangle\langle 0|$, subjected to a Hadamard gate ($U = H$).

Example 2. *A toy model: 1 qubit and 1 gate.* Recall our single-qubit SIC-POVM elements $M_\alpha = \frac{1}{2}\Pi_\alpha$ (for $\alpha \in \{1, 2, 3, 4\}$) defined in Eq. (2.2). The quantum simulation proceeds as follows:

1. Initialization (the input data). The initial state is mapped to a probability vector $\mathbf{p}_0$. By calculating $\mathbf{p}_0(\alpha) = \text{Tr}(\rho^{\text{in}} M_\alpha)$, we obtain a discrete distribution over our four vocabulary tokens:

$$\mathbf{p}_0 = (p_0(1), p_0(2), p_0(3), p_0(4))^T.$$

2. Exact evolution (the target). The Hadamard gate is analytically translated into a 4×4 pseudostochastic matrix $O^{(H)}$ using Eq. (2.3). The exact probability distribution after the gate is applied is simply the matrix-vector product:

$$\mathbf{p}_1^{(e)} = O^{(H)} \mathbf{p}_0.$$

Because the transition matrix $O^{(H)}$ contains negative entries, the evolution operates as a pseudo-stochastic map. However, because the unitary gate maps a valid quantum state to another valid state, the target $\mathbf{p}_1^{(e)}$ remains a true probability distribution.

3. Transformer optimization (the learning phase). Because $N = 1$, there is no sequence to mask. The transformer model $\mathbf{p}_{\theta_1}$ simply acts as a parameterized distribution over the 4 possible tokens. The model outputs its own predicted probability vector:

$$\mathbf{p}_{\theta_1} = (p_{\theta_1}(1), p_{\theta_1}(2), p_{\theta_1}(3), p_{\theta_1}(4))^T.$$

To train the model, we compute the exact KL divergence between the exact target $\mathbf{p}_1^{(e)}$ and the model's prediction $\mathbf{p}_{\theta_1}$, and the L_1 norm of the difference between the forward and backward evolution: $O_1 \mathbf{p}_{\theta_i}$ and $O_1^T \mathbf{p}_{\theta_{i+1}}$.

The network's weights are updated via gradient descent to minimize these losses until convergence. Finally, $\mathbf{p}_{\theta_1}$ mimics the quantum output.

2.4 Transformation to the Computational Z-Basis

Simulating quantum circuits within the IC-POVM formalism allows us to represent complex, entangled states as classical probability distributions. While this enables the use of autoregressive transformers, standard quantum algorithms require a final projective measurement in the computational Z -basis to yield meaningful classical bitstrings. Therefore, we require a computationally efficient protocol to translate the network’s predicted POVM distribution back into the Z -basis.

Fundamentally, an IC-POVM spans the entire operator space of a qubit. This property ensures that any physical observable can be reconstructed as a linear combination of the POVM effects. By geometrically isolating the Z -Pauli operator from the symmetric tetrahedron projectors, we can analytically derive the projectors $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$ strictly in terms of the POVM measurement elements. Applying the Born rule to these operators yields a deterministic, linear 2×4 transformation matrix, W_Z , which maps any local 4-dimensional POVM probability vector $\mathbf{p}$ directly to a 2-dimensional Z -basis probability vector: $P_Z = W_Z \mathbf{p}$ ².

The matrix W_Z perfectly translates a single qubit, but applying it naively to an N -qubit system requires evaluating the joint probability tensor $p(a_1, a_2, \dots, a_N)$ over all 4^N POVM outcomes. This exponential complexity renders direct translation intractable for large circuits. However, many algorithms (such as Shor’s) only require measuring a specific subset of q qubits, such as the control register. Translating this subset requires the marginalized probability distribution $p(a_1, \dots, a_q)$. Normally, marginalization requires summing over the 4^{N-q} discarded states. Yet, because our wavefunction is modeled by an autoregressive neural network, the joint probability strictly decomposes as a product of conditionals:

$$p(a_1, \dots, a_N) = \prod_{i=1}^N p_\theta(a_i | a_{<i}). \quad (2.9)$$

By the definition of conditional probability, the sum over all possible future discarded states inherently normalizes to 1. Provided that the causal ordering of the transformer is intentionally structured such that the q measured qubits (the control register) are generated before the non-measured qubits (the target register), the exact marginalized distribution simplifies to the truncated autoregressive product:

$$p_{\text{marg}}(a_1, \dots, a_q) = \prod_{i=1}^q p_\theta(a_i | a_{<i}). \quad (2.10)$$

This mathematical property is critical: it reduces the complexity of exact subset translation from $\mathcal{O}(4^N)$ to $\mathcal{O}(4^q)$.

Furthermore, for implementations where q is also prohibitively large, this exact tensor contraction is seamlessly replaced by Monte Carlo sampling [33]. The autoregressive network generates S samples from the marginalized q -qubit space in linear time $\mathcal{O}(S \cdot q)$. The analytical W_Z matrix acts as a classical weighting function, applied to the sampled POVM strings to yield an unbiased estimation of the Z -basis observables without ever instantiating the full exponential probability space.

²The full analytical derivation of W_Z , as well as the scalable framework required to apply this matrix efficiently to large multi-qubit registers, is detailed in Appendix C.

2.5 Shor's Algorithm

Current algorithms [34] state that the number of steps any classical computer requires in order to find the prime factors of an n -digit integer N increases super-polynomially with n . Factoring large integers has been conjectured to be intractable classically, and cryptographic codes have been built upon this foundation. This is why Shor's quantum factoring algorithm was so revolutionary and has become one of the central candidates for provable quantum advantage. Since Shor presented his idea in 1999 [24] many efforts have been directed to building a working quantum computer, able to make profound implications for modern cryptography.

Despite its promise, current quantum devices cannot handle enough qubits to make this a reality, nor are they expected to anytime soon. Nonetheless, there have been some important advances with regards to the experimental realization of Shor's algorithm [35, 36, 37], where *compiled* versions are used.

We begin by outlining the complete algorithm for factoring N , which is widely detailed in standard texts such as [38]. This hybrid algorithm, which relies on polynomial-time classical pre- and post-processing steps, proceeds as follows:

1. If N is even, return the factor 2.
2. Classically, determine if $N = p^q$ for $p \geq 1$ and $q \geq 2$, and if so return the factor p .
3. Choose a random number a such that $1 < a \leq N - 1$. Using Euclid's algorithm [39], determine if $\gcd(a, N) > 1$, and if so, return the factor $\gcd(a, N)$.
4. Use the *order-finding quantum algorithm* to find the order r of $a \pmod{N}$
5. If r is odd or r is even but $a^{r/2} \equiv -1 \pmod{N}$, then go back to step 3. Otherwise, compute $\gcd(a^{r/2} - 1, N)$ and $\gcd(a^{r/2} + 1, N)$. Test if one of these is a non-trivial factor of N (meaning is not 1 nor the number N) and return the factor.

The exponential quantum speed-up relies entirely on step 4, represented in Fig. 3.

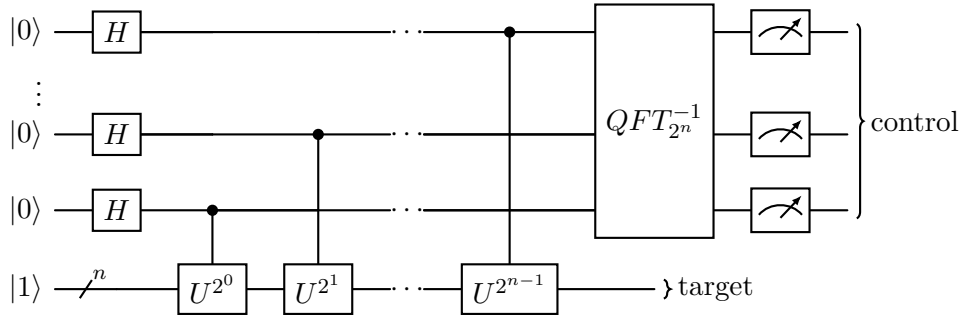


Figure 3: Phase estimation circuit used in Shor's algorithm for period finding.

The order finding algorithm circuit consists of three primary steps:

1. *Modular exponentiation* (U^{2^k}). Factoring a number N can be reduced to finding the period of the *modular exponentiation* function:

$$f(x) = a^x \pmod{N}, \quad \text{with } x \in \{0, 1, 2, \dots\}, \quad (2.11)$$

where the period r is defined as the smallest r positive integer such that

$$a^r \equiv 1 \pmod{N}. \quad (2.12)$$

2. *Inverse quantum Fourier transform* (QFT^{-1}). Applied to the control register to extract the phase information associated with the period r .
3. *Continued fractions algorithm*. To extract the period r from the measured phase, we utilize the continued fractions algorithm, a *classical* method for finding the best rational approximation of a real number (see Appendix A4.4 of [38] for a detailed explanation).

Thus, our work will focus on the quantum steps: the modular exponentiation and QFT^{-1} .

2.6 Modular Exponentiation

The objective of the modular exponentiation circuit in Shor’s algorithm is to evaluate the function $f(x) = a^x \pmod{N}$ for all possible values of x *simultaneously*. This requires two quantum registers: a *control* register of size t initialized in an equal superposition of all states $|x\rangle$. And a *target* register of size $n = \lceil \log_2(N) \rceil + 1$, initially prepared in the state $|1\rangle$.

The general circuit relies on the unitary operation U_a such that $U_a |y\rangle = |a \cdot y \pmod{N}\rangle$. By applying this operator x times, we obtain $U_a^x |y\rangle = |a^x \cdot y \pmod{N}\rangle$. However, constructing a monolithic U_a^x gate to perform this full exponentiation in one step is practically impossible. Instead, the computation is decomposed by expressing x in its binary representation:

$$x = x_{t-1}2^{t-1} + \dots + x_12^1 + x_02^0. \quad (2.13)$$

Using standard exponent rules, a^x is expanded into a product of sequential square powers:

$$a^x = a^{x_02^0} \cdot a^{x_12^1} \cdot \dots \cdot a^{x_{t-1}2^{t-1}} \pmod{N}. \quad (2.14)$$

Thus, the modular exponentiation is executed as a sequence of controlled modular multiplications. For each qubit x_i in the control register, if $|x_i\rangle = |1\rangle$, a controlled unitary is applied that multiplies the target register by the constant $a^{2^i} \pmod{N}$. Because the control register is in a superposition, this sequence generates a massive, highly entangled state: $\sum |x\rangle |a^x \pmod{N}\rangle$. The periodic mathematical pattern of the function is thereby physically encoded into the quantum amplitudes of the control register, allowing the period to be extracted later via the QFT^{-1} .

Modular exponentiation is widely considered the most space and time consuming component of Shor’s quantum factoring algorithm. While much work has been dedicated to lower the number of resources needed for a general instance [40, 41, 42], building a general multiplier that can handle *any* base a and *any* n -bit modulus N requires a prohibitive number of qubits and gate operations. Consequently, in both physical implementations on NISQ devices and standard classical simulations, this operation is often heavily optimized or compiled. In compiled circuits, the arithmetic is pre-calculated classically, and the circuit is reduced to a static lookup table of simplified X and CNOT gates. While this reduces gate depth, it abstracts away the algorithmic logic. The quantum computer, or the neural network emulating it, is no longer performing true quantum arithmetic but simply memorizing a hardcoded path.

To test the transformer architecture’s ability to learn true quantum operations, this work implements an uncompiled modular exponentiation circuit based on the architecture proposed by Vedral, Barenco, and Ekert (VBE) [26]. The VBE architecture fundamentally decomposes modular exponentiation into a series of controlled additions. To perform the multiplication, the unit executes three distinct phases:

1. Multiplication: The product $y \cdot a^{2^i} \pmod{N}$ is built in an accumulator register via sequential additions.
2. Swapping: The result in the accumulator is swapped into the target register y .
3. Uncomputation: To prevent the accumulation of intermediate data and preserve unitarity, the accumulator is cleanly reverted to $|0\rangle$ by applying the inverse arithmetic operations.

To perform the underlying arithmetic optimally, this implementation integrates the Cuccaro ripple-carry adder [25]. While traditional quantum addition circuits require a linear number of ancillary qubits, the Cuccaro adder minimizes the overhead to a single ancillary qubit by leveraging localized, in-place computations.

The total qubit requirement of the VBE-Cuccaro circuit scales linearly with the modulus bit-length n . Specifically, the system requires $3n + 2$ operational qubits, alongside $t = 2n$ control qubits for the phase estimation.

By feeding this uncompiled circuit into the transformer, the neural network is forced to dynamically emulate a functional quantum arithmetic logic unit. The model must preserve the unitary coherence of reversible arithmetic by successfully tracking carry bits through the computation layers and perfectly unweaving the entanglement during uncomputation. Evaluating this logic across a superposition of states provides a robust benchmark for the transformer’s attention mechanism in highly entangled quantum environments.

2.7 Period Extraction and the QFT^{-1}

The ultimate objective of the modular exponentiation sequence is to create a periodic correlation between the control and target registers. Once the computation is complete, the system resides in the highly entangled state:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |a^x \pmod{N}\rangle. \quad (2.15)$$

At this stage, the period r is physically hidden within the relative phases of the control qubits. To extract this periodicity and convert it into measurable probability amplitudes, we apply the QFT^{-1} to the control register. The general structure of the QFT^{-1} is depicted in Fig. 4, and it consists of Hadamard gates and controlled phase rotations defined as:

$$R_k^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & e^{-2\pi i/2^k} \end{bmatrix}. \quad (2.16)$$

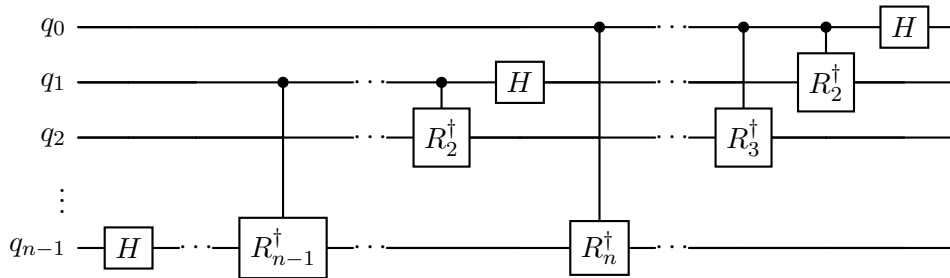


Figure 4: Structure of QFT^{-1} . Reversing the gate sequence and inverting phase angles yields the QFT.

By applying the QFT^{-1} , constructive interference peaks near fractions s/r (where s is an integer). Measuring the control register in the Z -basis yields a bitstring estimating $s/r \times 2^n$, rounded to the nearest integer. The classical continued fractions algorithm is then used to isolate the denominator and extract the period r , a process that often requires multiple measurements to refine the approximation. If r is an even integer, the final step of Shor’s algorithm is completed by classically computing the greatest common divisors, $\text{gcd}(a^{r/2} \pm 1, N)$, revealing the prime factors of N .

Benchmarking the Transformer: The 1D Cluster State

While the QFT and QFT^{-1} possess an elegant recursive structure allowing them to scale easily to larger qubit systems, benchmarking a classical neural network on these transforms presents a unique challenge. As a simulation benchmark, we are primarily interested in how a transformer tracks the spread of quantum entanglement. However, it is a well-established property that the QFT is an inherently low-entangling operator [43].

When applied to a computational basis state $|x\rangle$, the output of the QFT is always a fully separable tensor product state (up to a bit reversal):

$$\text{QFT} |x\rangle = \bigotimes_{j=1}^n \frac{|0\rangle + e^{i\phi_j(x)} |1\rangle}{\sqrt{2}}, \quad \text{where } \phi_j(x) = 2\pi x/2^j. \quad (2.17)$$

Because the resulting state is separable, computational basis states never become entangled under the action of the QFT . For instance, $\text{QFT} |0\rangle^{\otimes n} = |+\rangle^{\otimes n}$. Consequently, simulating the QFT on a standard basis state is a trivial task for an attention mechanism, as there are no complex non-local correlations to track.

To rigorously test the transformer’s capacity to model entanglement dynamics, we must apply the QFT to an initial state that is *already* entangled. This ensures the transform redistributes and preserves the entanglement rather than operating trivially. For this benchmark, we initialize the system in a *1D cluster state* [44].

A 1D cluster state is an entangled many-body state characterized by local, nearest-neighbor entanglement. For n qubits arranged linearly, the state is constructed by preparing all qubits in the $|+\rangle$ superposition and applying a controlled- Z (CZ) gates across all neighboring pairs:

$$|C_n\rangle = \left(\prod_{j=1}^{n-1} \text{CZ}_{j,j+1} \right) |+\rangle^{\otimes n}, \quad (2.18)$$

where the $\text{CZ}_{j,j+1}$ operator applies a -1 phase shift exclusively to the $|11\rangle$ component of adjacent qubits.

Schematically, the cluster state can be viewed as an equal superposition of all computational basis states, with correlated phases introduced by the CZ gates:

$$|C_n\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle, \quad (2.19)$$

where $f(x)$ encodes the nearest-neighbor correlations. When the QFT is applied to this state, it acts coherently on all basis components simultaneously:

$$\text{QFT} |C_n\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} \text{QFT} |x\rangle. \quad (2.20)$$

While QFT $|x\rangle$ is a product state, the linear combination of these exponentially many phase-correlated product states results in a highly complex, non-local entangled state. This occurs because the phase function $f(x)$ of the initial cluster state is strictly non-separable; it encodes correlated nearest-neighbor interactions. Consequently, the resulting superposition of QFT amplitudes cannot be factored into a simple tensor product of independent qubits, dynamically forcing the entanglement to spread across the entire register.

By utilizing the 1D cluster state, the QFT acts as an entanglement transport mechanism. The neural network must accurately simulate how the local, nearest-neighbor entanglement of the cluster state is dynamically spread across the entire multi-qubit register. This provides a substantially more rigorous evaluation of the transformer’s ability to model many-body quantum interference than testing the QFT on a trivial, unentangled basis state.

3 Probabilistic Simulation

3.1 Simulation Methodology and Implementation

The complete computational setup, hyperparameters, and hardware specifications for these simulations are detailed in Appendix D. To conduct these experiments, a custom pipeline was developed. The complete source code for this pipeline can be accessed on GitHub [45].

Before detailing the simulation benchmarks, we must establish our primary performance metric. We evaluate the model’s representational accuracy using the classical fidelity (F_c) [38], mathematically defined as the overlap between the exact target distribution $p^{(e)}$ and the network’s predicted distribution p_θ :

$$F_c = \sum_{\alpha} \sqrt{p^{(e)}(\alpha)p_\theta(\alpha)}. \quad (3.1)$$

A fidelity of $F_c = 1.0$ indicates that the transformer has perfectly captured the physical quantum state.

3.1.1 QFT Benchmark

To empirically validate the transformer’s ability to track the dynamic redistribution of entanglement, we first simulate a 25-qubit QFT applied to an initial 1D cluster state. The simulation is executed step-by-step using our autoregressive transformer architecture, tracking the quantum state via the IC-POVM framework. The methodology consists of four primary phases:

1. *Cluster state initialization via MPS.* Because a 1D cluster state features bounded, nearest-neighbor entanglement, its exact POVM probability distribution can be computed efficiently using a matrix product state (MPS) contraction. The transformer is trained on this exact distribution until the loss converges, ensuring the network begins the QFT sequence with a near-perfect internal representation of the initial entanglement structure.
2. *Autoregressive transformer architecture.* The state distribution is modeled using a causal quantum transformer (4 layers, 8 attention heads, embedding dimension 64). The autoregressive nature of the model allows it to sample multi-qubit measurement strings step-by-step, sequentially capturing the non-local correlations induced by the quantum gates.

3. *Step-wise gate execution and hybrid optimization.* The 25-qubit QFT is decomposed into 325 fundamental gate operations (25 Hadamards and 300 controlled-phase gates). For each gate, the parameters are updated using a hybrid loss function. This function combines a KL divergence penalty targeting the forward transition with a backward consistency penalty, preventing representation collapse and maintaining unitarity.
4. *Log-space Monte Carlo fidelity tracking.* To circumvent numerical underflow in high-dimensional probability spaces, both model predictions and target matrices are evaluated in log-space using 64-bit precision. After every gate application, the classical transition fidelity (F_c) is evaluated via Monte Carlo sampling.

3.1.2 Modular Exponentiation Benchmark

We now address the most computationally demanding phase of Shor’s algorithm: the modular exponentiation sequence. We test the architecture across three different factoring regimes: $N = 3$, $a = 2$ (12 qubits, 81 gates); $N = 4$, $a = 3$ (17 qubits, 114 gates); and $N = 8$, $a = 3$ (22 qubits, 207 gates). Simulating these highly entangled arithmetic circuits expands upon the prior framework in three critical ways:

1. *Dynamic circuit generation.* Rather than relying on standard lookup-table compilations, the circuit is generated dynamically as a sequence of purely uncompiled logical primitives. The arithmetic core utilizes the Cuccaro ripple-carry adder routed through the VBE architecture to build the product before uncomputing the workspace.
2. *Integration of 3-qubit POVM dynamics.* The uncompiled ripple-carry heavily relies on the Toffoli gate. To accommodate this, the transformer’s target logic was expanded to compute 3-qubit transition distributions, requiring the simultaneous calculation of IC-POVM overlaps across 64 distinct local state configurations. The loss functions were consequently upgraded to evaluate these 3-body matrix operations at each step.
3. *Variance-stabilized mixed Monte Carlo evaluation.* Assessing the accuracy of the neural network across an exponentially large Hilbert space requires stochastic evaluation. To circumvent exploding variance spikes caused by near-zero model predictions, we implement a mixture importance sampling framework. By evaluating the fidelity over a theoretical 50/50 mixture distribution, $Q(x) = \frac{1}{2}(p^{(e)}(\alpha) + p_\theta(\alpha))$, the estimator mathematically transforms into the ratio of the geometric mean to the arithmetic mean:

$$F \approx \mathbb{E}_{x \sim Q} \left[\frac{2\sqrt{p^{(e)}(\alpha)p_\theta(\alpha)}}{p^{(e)}(\alpha) + p_\theta(\alpha)} \right]$$

By the arithmetic mean-geometric mean inequality, the term inside this expectation is strictly bounded within $(0, 1]$, guaranteeing numerical stability. In practice, this evaluation is performed over 16 independent random seeds generating batches of 1,024 strings. By taking the ensemble average, the fidelity metric evaluates 16,384 distinct configurations per gate, leveraging the law of large numbers to ensure high convergence across the deep circuit.

3.2 Results and Discussion

With the methodological framework and architectural optimizations established, this section evaluates the performance of the transformer model.

3.2.1 QFT

The results of the 25-qubit cluster state QFT simulation are presented in Fig. 5. The figure utilizes a dual-axis representation. The scatter points on the left axis track the fidelity of the transformer’s state representation across the cumulative sequence of 325 applied quantum gates. The color mapping denotes the specific target qubit block being processed, progressing from the first qubit (red) to the final 25th qubit (purple). Because the overall fidelity remains extremely close to unity, a standard linear scale obscures minor fluctuations. Therefore, the right axis plots the log infidelity, $\log_{10}(1 - F_c)$, in gray. This logarithmic scale magnifies minute variations, making the sharp spikes in accuracy immediately visible.

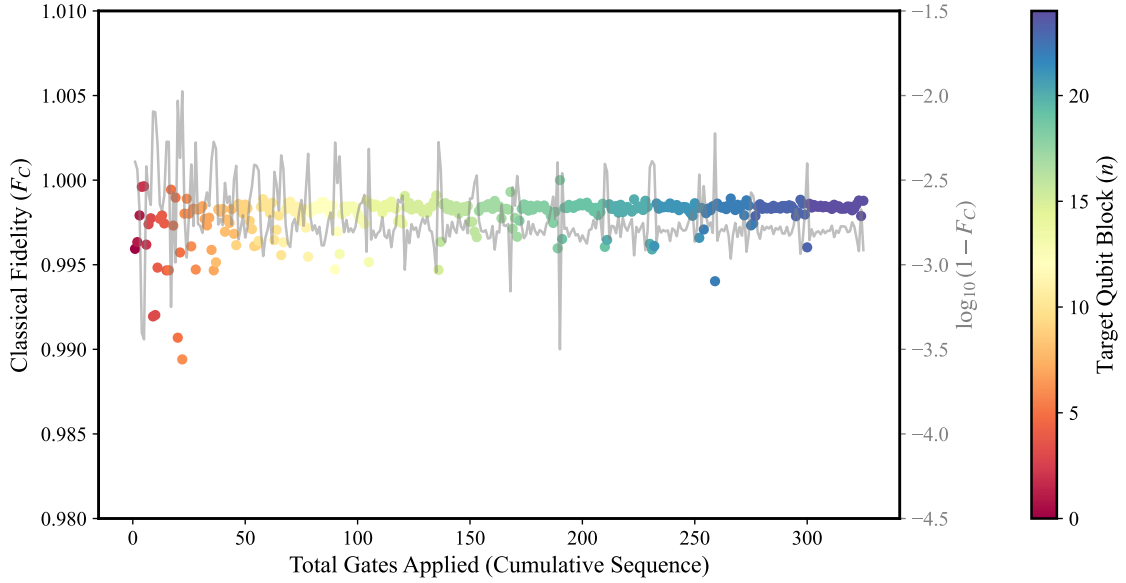


Figure 5: Classical fidelity and log infidelity progression during the dynamic simulation of a 25-qubit, 325 gate QFT applied to an initial 1D cluster state. The scatter points (left axis) track F_c , color-coded by the target qubit block. The solid gray line (right axis) tracks the log infidelity illustrating the bounded error of the transformer and highlighting momentary spikes.

The data reveals several critical insights regarding the transformer’s capacity for quantum simulation. First, the simulation demonstrates robust coherence and a notable lack of exponential decay. In standard classical simulations, approximation errors tend to compound exponentially with circuit depth. Remarkably, the classical fidelity consistently hovers above 0.995 throughout all 325 operations, with the absolute minimum remaining above 0.989. By applying the QFT to a 1D cluster state, strictly local, nearest-neighbor entanglement is forced to dynamically spread across the entire multi-qubit register. The sustained high fidelity proves that the transformer’s self-attention layers are highly capable of mapping and transporting these non-local correlations as the state complexity grows.

Looking at the overarching trajectory of the simulation, the fidelity exhibits significantly more oscillatory behavior in the early stages of the computation compared to the end. We attribute this macroscopic trend directly to our initial state preparation. Because the input is a highly entangled 1D cluster state, the network is immediately tasked with processing complex correlations from the very first gate. During these early stages, the transformer must rapidly learn to map and propagate this pre-existing entanglement across the expanding register, which manifests as wider fluctuations in fidelity.

Furthermore, the data exhibits distinct oscillatory gate-level dynamics within individual target qubit blocks. This behavior directly reflects the QFT’s layered structure and the network’s parameter initialization strategy. Because the optimized network parameters from a previous gate serve as the starting point for the next, we believe sudden physical shifts in the circuit logic cause temporary fidelity dips. Specifically, the sharpest spikes in error occur when transitioning from a non-trivial Hadamard gate to the first controlled-phase rotation. However, as the sequence progresses through the subsequent two-qubit phase gates (R_k to R_{k+1}), the parametric distance between successive operations shrinks, allowing the transformer to seamlessly adapt and rapidly recover from these initial dips. This recovery transitions into an asymptotic stabilization in the later stages of the sequence. The controlled-phase rotations apply a phase shift of $2\pi/2^k$, as the target block advances and the physical distance between the control and target qubits increases, these operators rapidly approach the identity matrix ($R_k \approx \mathbb{I}$). For the neural network, learning and executing a near-identity transformation is mathematically simpler than modeling large-angle rotations, resulting in higher fidelity retention and a tightening of the variance in the later algorithmic stages.

Ultimately, this benchmark demonstrates that, at the 25-qubit scale, a deep quantum circuit can be robustly simulated using an attention-based neural network. Crucially, the architecture successfully tracks entanglement distribution without succumbing to the depth-induced exponential error decay that traditionally limits classical simulation methods.

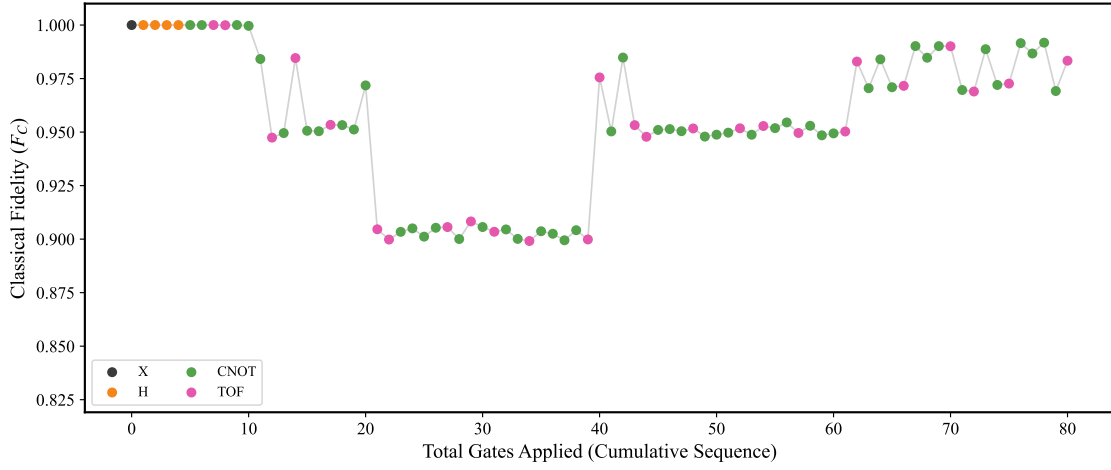
3.2.2 Modular Exponentiation

The performance of the transformer on the uncompiled modular exponentiation circuits is illustrated in Fig. 6. The plots track F_c for three increasingly complex factoring regimes.

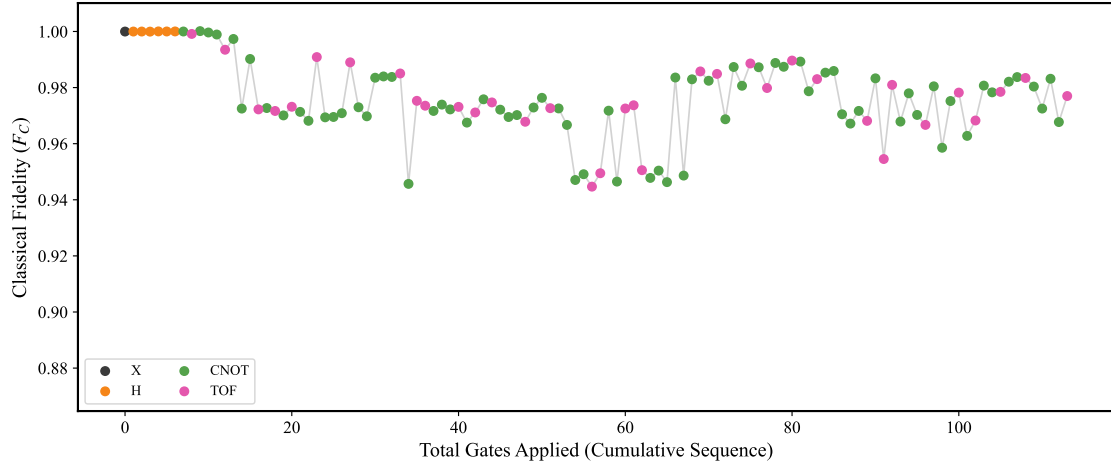
Analyzing the network’s behavior across these arithmetic circuits reveals distinct physical dynamics compared to the QFT benchmark. In the QFT simulation, deep non-local correlations are already established during the cluster state initialization. The network adapts to this complexity early on, meaning it primarily learns to transport and redistribute pre-existing two-body entanglement. Conversely, the uncompiled modular exponentiation sequence begins with a simple, unentangled product state and must dynamically generate massive entanglement mid-circuit. This sudden expansion of state complexity is driven by dense blocks of highly entangling CNOTs and 3-qubit Toffoli gates. Forcing the network to abruptly learn these emerging multi-body correlations creates a significantly more difficult probability landscape to navigate. Consequently, across all three plots, the fidelity exhibits sharp drops, falling to roughly 0.90 for $N = 3$, and hovering near 0.94 for $N = 4$ and 8.

In standard classical approximation schemes, this dense multi-partite entanglement typically causes an unrecoverable collapse in fidelity. The transformer, however, averts this fate. While the fidelity dips during intensive computation steps, it successfully rebounds. We believe this resilience is driven by the VBE architecture, which explicitly uncomputes the workspace after each addition. As the circuit cleans the ancilla qubits and reduces overall entanglement, the transformer naturally recovers its representational accuracy.

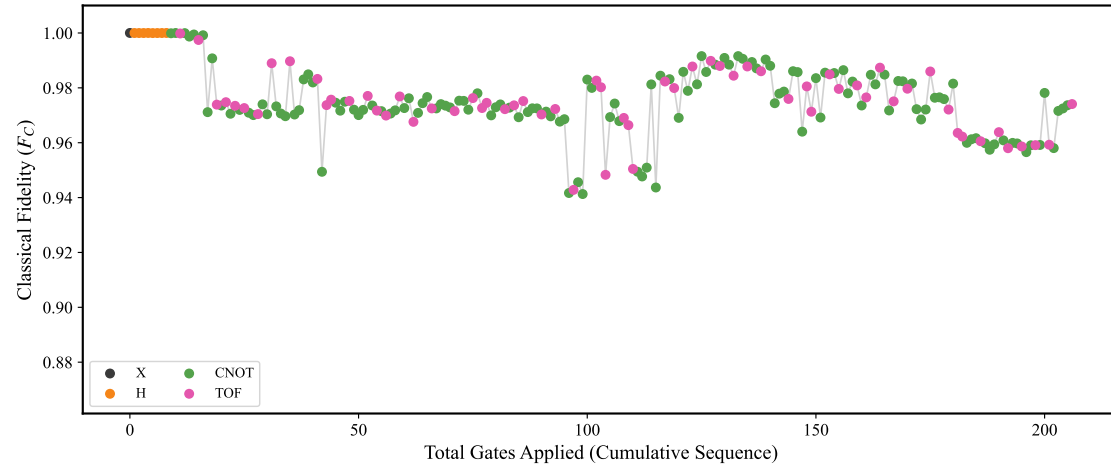
Finally, these arithmetic simulations demonstrate the model’s architectural scalability and depth invariance. Moving from 12 qubits ($N = 3$) to 22 qubits ($N = 8$) provides a definitive demonstration of the model’s scalability. The $N = 8$ regime requires simulating roughly 200 cumulative gates inside a vastly larger probability space. Despite this intense computational overhead, the baseline fidelity does not suffer a macroscopic collapse, oscillating comfortably between 0.94 and 1.0.



(a) Modular Exponentiation for $N = 3, a = 2$ requiring a 12-qubit VBE circuit.



(b) Modular Exponentiation for $N = 4, a = 3$ requiring a 17-qubit VBE circuit.



(c) Modular Exponentiation for $N = 8, a = 3$ requiring a 22-qubit VBE circuit.

Figure 6: Classical fidelity (F_c) progression during the uncompiled modular exponentiation sequences. The pronounced oscillatory dips correspond to the highly entangling Toffoli gates within the Cuccaro adder blocks. Notably, the transformer successfully bounds the error, recovering representational coherence during the circuit's uncomputation phases.

3.2.3 Factoring $N=15$

Having validated the transformer on both dense arithmetic and non-local phase rotations, we integrated these primitives to simulate the complete Shor’s algorithm for factoring $N = 15$ ($n = 22$ qubits, reaching a depth of 301 cumulative gates). The full progression of the circuit’s performance is shown in Fig. 7.

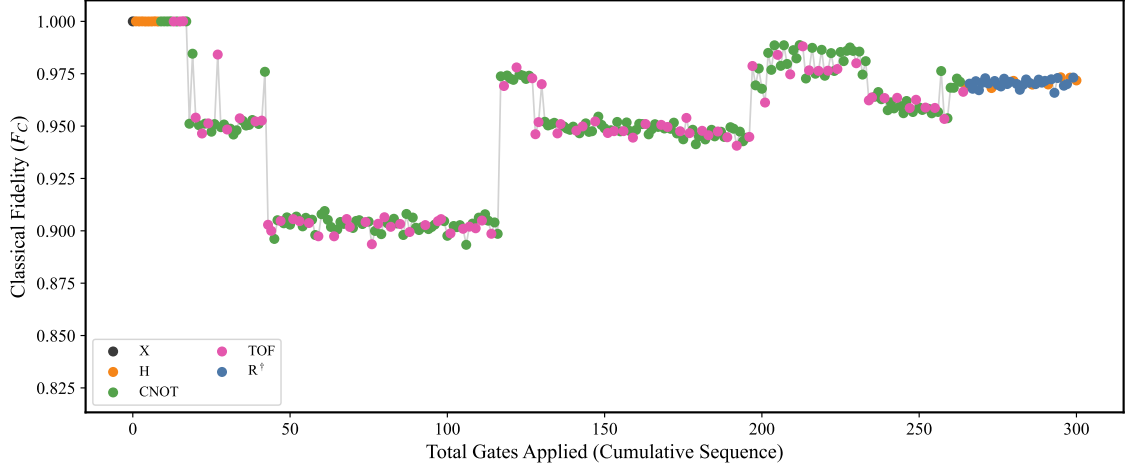


Figure 7: Classical fidelity (F_c) progression during the Shor’s algorithm for $N = 15$, $a = 4$ requiring a 22-qubit circuit. The circuit begins with the initialization (grey/orange X and H gates), is followed by the modular exponentiation sequence (green/pink CNOTs and Toffolis), and concludes with the QFT^{-1} (blue/orange $R^\dagger$ and H gates).

The plot in Fig. 7 reveals three distinct phases of the algorithmic simulation. The first phase corresponds to initialization and state preparation. During the first ~ 30 gates, the circuit constructs the initial superpositions and prepares the target register. The network handles this largely unentangled generation effortlessly, maintaining a fidelity near 1.0.

Following initialization, the circuit enters the modular exponentiation phase, which introduces deep, volume-law entanglement. The core of the algorithm consists of dense blocks of highly entangling Toffoli and CNOT gates. As observed in the isolated benchmarks, the fidelity drops to a floor of roughly 0.900. However, the transformer successfully bounds this error by leveraging the algorithmic uncomputation phases to recover representational coherence.

The final phase of the simulation, spanning gates 266 to 300, applies the QFT^{-1} to the control register to manage and extract the phase information. Notably, during this stage, the fidelity stabilizes remarkably well, remaining flat and consistently high. Because the deep arithmetic entanglement has already been established by the Cuccaro adder, the QFT^{-1} merely decodes this structure via local phase rotations. The transformer’s ability to maintain a flat trajectory proves that its self-attention layers successfully “lock in” and manage pre-existing complex correlations without leaking information.

To verify the physical validity of the simulation, we translated the final probability distribution of the $m = 8$ control qubits back into the computational Z -basis. The top measured decimal phase outcomes are as follows:

$$|0\rangle : 48.03\% \quad |128\rangle : 26.83\% \quad |64\rangle : 13.89\% \quad \dots$$

In Shor’s algorithm, a measurement outcome of $y = 0$ is considered a trivial failure, as it yields no information about the period (producing $s/r = 0$, leading to $r = 1$). We therefore discard $|0\rangle$ and proceed with the highest non-trivial peak, $y = 128$. We use the continued fractions algorithm to extract the period r . Given our control register size of $m = 8$ qubits, the total number of states is $Q = 2^8 = 256$. The measured phase is approximated by the fraction:

$$\frac{y}{Q} = \frac{128}{256} = \frac{1}{2}.$$

This fraction reduces exactly, therefore the continued fraction expansion is trivial, immediately yielding a period of $r = 2$. With the period found, we apply classical number theory to extract the prime factors. We first verify that r is even (2 is even) and check that $a^{r/2} \not\equiv -1 \pmod{N}$:

$$4^{2/2} \pmod{15} = 4 \not\equiv -1 \pmod{15}.$$

Since the conditions are met, the prime factors p and q are guaranteed to be found using the greatest common divisor (gcd) of $a^{r/2} \pm 1$ and N :

$$\begin{aligned} p &= \gcd(4^1 - 1, 15) = \gcd(3, 15) = 3, \\ q &= \gcd(4^1 + 1, 15) = \gcd(5, 15) = 5. \end{aligned}$$

Our probabilistic transformer successfully simulates the full depth of Shor’s algorithm, yielding the correct prime factors 3 and 5 from the factorization of $N = 15$.

4 Conclusion

This thesis successfully demonstrates that deep, highly entangled quantum algorithms can be rigorously simulated on classical hardware by transitioning away from the standard state-vector formalism and into the domain of classical probability theory. By utilizing IC-POVMs, the exponential complexity of the Hilbert space was recast as a generative sequence-to-sequence task, which was effectively managed by the global receptive field of a transformer neural network.

Summary of Findings

The findings of this research highlight the robustness and scalability of the proposed approach. First, the transformer architecture proved highly capable of tracking the dynamic redistribution of entanglement, maintaining a classical fidelity above 0.989 during the simulation of a 25-qubit QFT applied to a 1D cluster state. Furthermore, by moving beyond simplified, compiled lookup tables, the network successfully simulated true quantum arithmetic by dynamically emulating the Cuccaro ripple-carry adder and VBE architecture. Within these arithmetic circuits, the model demonstrated remarkable resilience against error accumulation. While dense multi-partite CNOT and Toffoli gates caused temporary fidelity drops, the transformer successfully rebounded by learning to dynamically unweave entanglement during the uncomputation phases. Ultimately, the scalability of the methodology was validated by simulating the complete Shor’s algorithm for $N = 15$ across 22 qubits and over 300 cumulative gates with a final fidelity of $F_c > 0.97$, accurately extracting the correct prime factors 3 and 5.

Future Work

Building upon these results, several avenues for future research emerge. Subsequent studies should investigate the computational limits of this simulation framework by scaling to higher qubit counts and greater circuit depths in order to factor larger composite numbers. Alongside this physical scaling, a systematic threshold analysis is required to identify the specific breaking point (namely, the ratio of qubits to gates) where the classical fidelity permanently drops below $F_c < 0.9$. Additionally, the transformer’s performance should be explored and tested against alternative quantum algorithms that prioritize different hardware constraints, such as shallower depths or higher qubit counts. Finally, extensive optimization of the network’s hyperparameters, including architectural adjustments, learning rate schedules, and optimizers, is necessary to guarantee robust reproducibility across different random seeds.

Bibliography

- [1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, and D. A. Buell, *Nature* **574**, 505 (2019).
- [2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, *Rev. Mod. Phys.* **94**, 015004 (2022).
- [3] J. Preskill, *Quantum* **2**, 79 (2018).
- [4] R. Orús, *Ann. Phys.* **349**, 117 (2014).
- [5] G. Vidal, *Phys. Rev. Lett.* **91**, 147902 (2003).
- [6] G. Vidal, *Phys. Rev. Lett.* **101**, 110501 (2008).
- [7] I. L. Markov and Y. Shi, *SIAM J. Comput.* **38**, 963 (2008).
- [8] S. Aaronson and D. Gottesman, *Phys. Rev. A* **70**, 052328 (2004).
- [9] M. S. Rudolph, T. Jones, Y. Teng, A. Angrisani, and Z. Holmes, *arXiv:2505.21606* (2025).
- [10] M. S. Rudolph, E. Fontana, Z. Holmes, and L. Cincio, *arXiv:2308.09109* (2023).
- [11] H. Lange, A. Van de Walle, A. Abedinnia, and A. Bohrdt, *Quantum Sci. Technol.* **9**, 040501 (2024).
- [12] T. Heightman, E. Jiang, R. Mora-Soto, M. Lewenstein, and M. Płodzień, *arXiv:2507.12117* (2025).
- [13] E. Kiktenko, A. Malyshev, A. Mastiukova, V. Man'ko, A. Fedorov, and D. Chruściński, *Phys. Rev. A* **101**, 052320 (2020).
- [14] G. Carleo and M. Troyer, *Science* **355**, 602 (2017).
- [15] G. Carleo, Y. Nomura, and M. Imada, *Nat. Commun.* **9**, 5322 (2018).
- [16] I. L. Gutiérrez and C. B. Mendl, *Quantum* **6**, 627 (2022).
- [17] K. Choo, T. Neupert, and G. Carleo, *arXiv:1903.06713* (2019).
- [18] J. Carrasquilla, D. Luo, F. Pérez, A. Milsted, B. K. Clark, M. Volkovs, and L. Aolita, *Phys. Rev. A* **104**, 032610 (2021).
- [19] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, *Nat. Mach. Intell.* **1**, 155 (2019).
- [20] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, *Phys. Rev. Lett.* **124**, 020503 (2020).
- [21] K. Donatella, Z. Denis, A. Le Boité, and C. Ciuti, *Phys. Rev. A* **108**, 022210 (2023).
- [22] R. Pascanu, T. Mikolov, and Y. Bengio, in *International Conference on Machine Learning* (PMLR, 2013) pp. 1310–1318.
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Adv. Neural Inf. Process. Syst.* **30** (2017).
- [24] P. W. Shor, *SIAM Rev.* **41**, 303 (1999).
- [25] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, *arXiv:quant-ph/0410184* (2004).
- [26] V. Vedral, A. Barenco, and A. Ekert, *Phys. Rev. A* **54**, 147 (1996).
- [27] G. d'Ariano, P. Perinotti, and M. Sacchi, *J. Opt. B: Quantum Semiclass. Opt.* **6**, S487 (2004).
- [28] J. M. Renes, R. Blume-Kohout, A. J. Scott, and C. M. Caves, *J. Math. Phys.* **45**, 2171 (2004).
- [29] V. Yashin, E. Kiktenko, A. Mastiukova, and A. Fedorov, *New J. Phys.* **22**, 103026 (2020).
- [30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askill, *Adv. Neural Inf. Process. Syst.* **33**, 1877 (2020).

- [31] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altschmidt, S. Altman, and S. Anadkat, arXiv:2303.08774 (2023).
- [32] S. Kullback and R. A. Leibler, Ann. Math. Stat. **22**, 79 (1951).
- [33] G. García-Pérez, M. A. Rossi, B. Sokolov, F. Tacchino, P. K. Barkoutsos, G. Mazzola, I. Tavernelli, and S. Maniscalco, PRX Quantum **2**, 040342 (2021).
- [34] J. P. Buhler, H. W. Lenstra, and C. Pomerance, *The Development of the Number Field Sieve* (Springer Berlin Heidelberg, Berlin, Heidelberg, 1993) pp. 50–94.
- [35] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature **414**, 883 (2001).
- [36] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O’Brien, Nat. Photonics **6**, 773 (2012).
- [37] U. Skosana and M. Tame, Sci. Rep. **11**, 16599 (2021).
- [38] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
- [39] T. L. Heath, *The Thirteen Books of Euclid’s Elements* (Courier Corporation, 1956).
- [40] S. Beauregard, arXiv:quant-ph/0205095 (2002).
- [41] Y. Takahashi and N. Kunihiro, Quantum Inf. Comput. **6**, 184 (2006).
- [42] C. Zalka, arXiv:quant-ph/0601097 (2006).
- [43] J. Chen, E. Stoudenmire, and S. R. White, PRX Quantum **4**, 040318 (2023).
- [44] H. J. Briegel and R. Raussendorf, Phys. Rev. Lett. **86**, 910 (2001).
- [45] H. Monsó, “Transformers Shor’s Algorithm,” GitHub Repository, <https://github.com/helena-monso/Transformers-Shor-s-Algorithm> (2026).

A Derivation of the Pseudo-Bistochastic Transition Matrix

In this appendix, we derive the exact form of the pseudo-bistochastic transition matrix S (presented in Eq. (2.3)), which dictates the evolution of the IC-POVM probability vector under a unitary quantum gate.

We begin with the general framework established by Kiktenko *et al.* [13] for tracking quantum states through Completely Positive Trace-Preserving (CPTP) maps. Consider a general CPTP map $\Phi : \mathcal{L}(\mathcal{H}_{\text{in}}) \rightarrow \mathcal{L}(\mathcal{H}_{\text{out}})$, which maps linear operators from a d_{in} -dimensional Hilbert space to a d_{out} -dimensional one. The action of Φ on an input state ρ_{in} is described by its Kraus representation:

$$\rho_{\text{out}} = \Phi(\rho_{\text{in}}) = \sum_k A_k \rho_{\text{in}} A_k^\dagger \quad (\text{A.1})$$

where the Kraus operators satisfy the trace-preserving condition $\sum_k A_k^\dagger A_k = \mathbb{I}_{\text{in}}$.

Let $\{\Pi_\beta^{\text{in}}\}_{\beta=1}^{d_{\text{in}}^2}$ and $\{\Pi_\alpha^{\text{out}}\}_{\alpha=1}^{d_{\text{out}}^2}$ be the SIC-POVM projectors for the input and output spaces, respectively. The map Φ transforms the input probability vector $\mathbf{p}^{\text{in}}$ into the output vector $\mathbf{p}^{\text{out}} = S\mathbf{p}^{\text{in}}$. The elements of this generalized transition matrix S are given by:

$$S_{\alpha,\beta} = (d_{\text{in}} + 1)s_{\alpha,\beta} - \frac{1}{d_{\text{out}}} \text{Tr} \left(\Pi_\alpha^{\text{out}} \sum_k A_k A_k^\dagger \right), \quad (\text{A.2})$$

where the transition overlaps $s_{\alpha,\beta}$ are defined as:

$$s_{\alpha,\beta} = \frac{1}{d_{\text{out}}} \sum_k \text{Tr} \left(A_k \Pi_\beta^{\text{in}} A_k^\dagger \Pi_\alpha^{\text{out}} \right). \quad (\text{A.3})$$

In the context of our simulation, we are strictly concerned with the evolution of a closed system under pure unitary quantum gates. This introduces three critical mathematical simplifications to the general framework:

1. Dimensionality: The input and output spaces are identical, meaning $d_{\text{in}} = d_{\text{out}} = d$. Consequently, the input and output POVM sets are also identical ($\Pi^{\text{in}} = \Pi^{\text{out}} = \Pi$).
2. Unitary Evolution: The CPTP map is defined by a single Kraus operator: the unitary gate itself ($A_1 = U$).
3. Unitarity Condition: Because U is a unitary matrix, it satisfies $UU^\dagger = \mathbb{I}$.

Applying these physical constraints to Eq. (A.2), the summation $\sum_k A_k A_k^\dagger$ collapses directly to $UU^\dagger = \mathbb{I}$. The subtractive trace term consequently simplifies to $\text{Tr}(\Pi_\alpha \mathbb{I}) = \text{Tr}(\Pi_\alpha)$. By definition, the trace of any individual unnormalized SIC-POVM projector Π_α in dimension d is exactly 1. Therefore, the entire subtractive term reduces strictly to $\frac{1}{d}$. Substituting this constant, alongside our single operator U , into the matrix definition directly yields our functional transition matrix:

$$S_{\alpha,\beta} = (d + 1)s_{\alpha,\beta} - \frac{1}{d}, \quad (\text{A.4})$$

with the overlap coefficients cleanly reducing to:

$$s_{\alpha,\beta} = \frac{1}{d} \text{Tr} \left(U \Pi_\beta U^\dagger \Pi_\alpha \right). \quad (\text{A.5})$$

This completes the derivation of Eq. (2.3), providing the exact pseudo-bistochastic matrix used to evolve the target probability vectors in our simulation.

B Proof of Exact Sampling in Autoregressive Quantum Circuit Simulation

In the context of our quantum circuit simulation, sampling from the simulated quantum state involves drawing IC-POVM outcome configurations $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_N)$ from the classical probability distribution $p(\alpha)$. Recall that N denotes the number of qubits, and $\alpha_k \in \{1, 2, 3, 4\}$ represents the discrete POVM measurement outcome for the k -th qubit.

Traditional energy-based models (such as boltzmann machines) represent probabilities via an energy function $E(\alpha)$:

$$p(\alpha) = \frac{1}{Z} e^{-E(\alpha)}, \quad (\text{B.1})$$

where the partition function $Z = \sum_{\alpha} e^{-E(\alpha)}$ acts as the global normalization constant. Computing Z requires summing over all 4^N possible configurations in the expanded probability space, which becomes computationally intractable for large N . Consequently, these architectures must rely on approximate Markov Chain Monte Carlo (MCMC) sampling, which suffers from long mixing times and autocorrelation biases.

In contrast, an autoregressive transformer evades the intractable partition function by decomposing the joint probability distribution $p_{\theta}(\alpha)$ into a product of local conditional probabilities via the *probability chain rule*:

$$p_{\theta}(\alpha) = p_{\theta}(\alpha_1, \alpha_2, \dots, \alpha_N) = \prod_{k=1}^N p_{\theta}(\alpha_k \mid \alpha_{<k}). \quad (\text{B.2})$$

The structural design of the transformer guarantees that this decomposition is mathematically exact and computationally tractable through three key pillars:

1. **Causal Attention Masking:** The self-attention layers apply a strict upper-triangular causal mask to the attention score matrix. For any qubit index k , the representation cannot attend to future qubit outcomes $j > k$. Therefore, the output logits for qubit k are mathematically deterministic functions exclusively dependent on the prefix $\alpha_{<k} = (\alpha_1, \dots, \alpha_{k-1})$.
2. **Tractable Local Normalization:** At each generation step k , the model outputs raw scores (logits) z_{k, α_k} for the four possible local POVM outcomes $\alpha_k \in \{1, 2, 3, 4\}$. The conditional probability is normalized locally via the softmax function:

$$p_{\theta}(\alpha_k \mid \alpha_{<k}) = \frac{e^{z_{k, \alpha_k}}}{\sum_{\gamma=1}^4 e^{z_{k, \gamma}}}. \quad (\text{B.3})$$

Because the denominator only sums over the local single-qubit POVM outcome space ($|V| = 4$) rather than the full 4^N probability space, the normalization constant is completely tractable and computed in $\mathcal{O}(1)$ time per site.

3. **Ancestral Sampling:** Since every local conditional distribution $p_{\theta}(\alpha_k \mid \alpha_{<k})$ is normalized and explicitly known, a classical computer can draw a sample for α_k directly using standard categorical sampling (e.g., rolling a biased 4-sided die with probabilities dictated by $p_{\theta}(\alpha_k \mid \alpha_{<k})$).

By sampling the POVM outcomes sequentially from $k = 1$ to N , the resulting configuration α is a direct, independent, and unbiased draw from the exact joint distribution $p_{\theta}(\alpha)$. This eliminates the need for any burn-in periods, rejection steps, or MCMC equilibration, fulfilling the definition of *exact sampling* for gate-by-gate quantum circuit simulations.

C Basis Translation and Scalable Monte Carlo Estimation

In this appendix, we formalize the readout phase of our autoregressive simulation. We derive the exact linear mapping matrix, W_Z , required to translate 4D POVM distributions into the 2D computational Z -basis. Furthermore, we demonstrate how applying this map globally suffers from the *Curse of Dimensionality*, and how Monte Carlo sampling elegantly circumvents this scaling limit using local quasi-probabilities.

C.1 The Translation Map: Deriving W_Z

We employ the SIC-POVM forming a tetrahedron inside the Bloch sphere from Eq. (2.2). The four unnormalized projectors are Π_i , and the physical measurement elements are $E_i = \frac{1}{2}\Pi_i$. To isolate the Z -Pauli operator, we pair these projectors to cancel the X and Y components. Adding Π_1 and Π_3 isolates the positive Z axis, while Π_2 and Π_4 isolate the negative Z axis:

$$(\Pi_1 + \Pi_3) - (\Pi_2 + \Pi_4) = \frac{2}{\sqrt{3}}Z \implies Z = \sqrt{3}(E_1 - E_2 + E_3 - E_4). \quad (\text{C.1})$$

The standard projective measurement for $|0\rangle$ is $|0\rangle\langle 0| = \frac{1}{2}(I + Z)$. Recognizing that $\mathbb{I} = \sum E_i$, we write the projector strictly in the POVM basis:

$$|0\rangle\langle 0| = \frac{1}{2} \sum_{i=1}^4 E_i + \frac{\sqrt{3}}{2}(E_1 - E_2 + E_3 - E_4). \quad (\text{C.2})$$

Applying the Born rule, $p_i = \text{Tr}(\rho E_i)$, yields the analytical translation formula:

$$p(|0\rangle) = \frac{1}{2} + \frac{\sqrt{3}}{2}(p_1 - p_2 + p_3 - p_4). \quad (\text{C.3})$$

To convert this affine equation into a linear matrix transformation, we utilize the axiom that POVM probabilities sum to unity ($\sum p_i = 1$), substituting this into the isolated $\frac{1}{2}$ term:

$$p(|0\rangle) = \frac{1}{2}(p_1 + p_2 + p_3 + p_4) + \frac{\sqrt{3}}{2}(p_1 - p_2 + p_3 - p_4). \quad (\text{C.4})$$

Factoring the coefficients for each p_i , and doing the same for $p(|1\rangle) = 1 - p(|0\rangle)$, yields the 2×4 transformation matrix, W_Z :

$$W_Z = \begin{pmatrix} \frac{1+\sqrt{3}}{2} & \frac{1-\sqrt{3}}{2} & \frac{1+\sqrt{3}}{2} & \frac{1-\sqrt{3}}{2} \\ \frac{1-\sqrt{3}}{2} & \frac{1+\sqrt{3}}{2} & \frac{1-\sqrt{3}}{2} & \frac{1+\sqrt{3}}{2} \end{pmatrix} \approx \begin{pmatrix} 1.366 & -0.366 & 1.366 & -0.366 \\ -0.366 & 1.366 & -0.366 & 1.366 \end{pmatrix}. \quad (\text{C.5})$$

For any single qubit, translation is executed via matrix-vector multiplication:

$$P_Z = W_Z \mathbf{p}_{POVM}. \quad (\text{C.6})$$

C.2 The Curse of Dimensionality

For a system of N qubits, the global translation matrix is the Kronecker product:

$$W_{\text{global}} = W_Z^{\otimes N}. \quad (\text{C.7})$$

This matrix maps a 4^N -dimensional vector to a 2^N -dimensional vector. If we simulate $N = 50$, W_{global} has dimensions $2^{50} \times 4^{50}$, making direct analytical translation physically impossible.

C.3 The Monte Carlo Rescue

To extract global probabilities without constructing W_{global} , we employ Monte Carlo sampling. Let $\mathbf{a} = (a_1, \dots, a_N)$ be a string of POVM outcomes sampled from the trained Transformer, $P_\theta(\mathbf{a})$. The exact probability of a specific Z -basis string $\mathbf{z} = (z_1, \dots, z_N)$ is approximated by drawing S random samples:

$$P(\mathbf{z}) \approx \frac{1}{S} \sum_{s=1}^S \left[\prod_{j=1}^N (W_Z)_{z_j, a_j^{(s)}} \right]. \quad (\text{C.8})$$

Example 3. *Monte Carlo Quasi-Probabilities (2 Qubits).* Suppose we want $P(\mathbf{z} = |01\rangle)$. If the network samples $\mathbf{a} = (1, 2)$, the local translation product is: $(W_Z)_{0,1} \times (W_Z)_{1,2} = 1.366 \times 1.366 = 1.866$. Notice that $1.866 > 1$, which is physically impossible for a true probability, indicating a quasi-probability. However, if the AI samples $\mathbf{a} = (2, 3)$, the contribution is $(-0.366) \times (-0.366) = 0.134$. By the Law of Large Numbers, summing thousands of these quasi-probabilities and dividing by S perfectly cancels the extremes, converging exactly on the physically bounded true joint probability without constructing the 16×4 global matrix.

C.4 Marginalization: Extracting Local Observables

To estimate the local Z -basis probability for the k -th qubit without summing over 4^{N-1} terms, we perform implicit marginalization. Because the samples $\mathbf{a}^{(s)}$ capture the correlated joint distribution, we apply W_Z exclusively to the k -th outcome of each sampled string, ignoring the rest:

$$p(z_k) \approx \frac{1}{S} \sum_{s=1}^S (W_Z)_{z_k, a_k^{(s)}}. \quad (\text{C.9})$$

Example 4. *Fast Marginalization* Suppose we want $p(|0\rangle_{Q0})$ with $S = 3$ generated strings: $\mathbf{a}^{(1)} = (1, 3, 2, \dots)$, $\mathbf{a}^{(2)} = (4, 0, 1, \dots)$, $\mathbf{a}^{(3)} = (1, 2, 2, \dots)$. Isolating Qubit 0 yields $\{1, 4, 1\}$. Applying the W_Z coefficients for $|0\rangle$:

$$p(|0\rangle_{Q0}) \approx \frac{1}{3} [1.366 - 0.366 + 1.366] \approx 0.789 \text{ (or 78.9\%)}.$$

This extracts highly accurate local quantum observables in $\mathcal{O}(S)$ time, independent of N .

D Simulation Procedure and Specifications

In this section, we outline the numerical and algorithmic specifications for simulating deep quantum circuits using an autoregressive Transformer neural network. The simulation circumvents exponential state-vector scaling by mapping quantum states to classical probability distributions using Symmetric Informationally Complete Positive Operator-Valued Measures (SIC-POVMs), leveraging JAX for hardware acceleration and Flax for network architecture.

D.1 Framework and Precision Architecture

The simulation relies on a high-performance, differentiable computing stack:

- JAX & Flax (`flax.linen`): Utilized for Just-In-Time (JIT) compilation, automatic differentiation, and constructing the causal Transformer architecture.
- Optax: Supplies the gradient descent optimization algorithms and learning rate schedules.
- Precision Split: To balance physical exactness with GPU memory limits, the simulation employs a dual-precision strategy. The exact theoretical transition matrices are pre-computed on the CPU using NumPy & SciPy in 64-bit precision (`complex128` and `float64`) to prevent floating-point drift. However, the neural network and JAX tensor operations are executed in 32-bit precision (`float32`) to maximize processing speed and memory efficiency during the training loop.

D.2 Algebraic Formulation

SIC-POVMs

Instead of complex state vectors, qubits are tracked via probability distributions over four Symmetric Informationally Complete POVM outcomes. The unnormalized projectors Π_k are defined using the Pauli matrices (I, X, Y, Z) and a normalization constant $c = 1/\sqrt{3}$:

$$\begin{aligned}\Pi_1 &= \frac{1}{2}(I + c(X - Y + Z)), \\ \Pi_2 &= \frac{1}{2}(I + c(X + Y - Z)), \\ \Pi_3 &= \frac{1}{2}(I + c(-X + Y + Z)), \\ \Pi_4 &= \frac{1}{2}(I + c(-X - Y - Z)).\end{aligned}\tag{D.1}$$

Transition Matrices (Math Oracles)

For any quantum gate U , the state evolution is dictated by a classical transition matrix O . To enable bidirectional learning, the principal matrix square root $V = \sqrt{U}$ is calculated using `scipy.linalg.sqrtm`. Three matrices are generated for each gate subset (1-qubit, 2-qubit, and 3-qubit operations like Toffoli):

- O_U : The full forward transition matrix.
- O_{fwd} : The halfway-forward matrix (derived from V).
- O_{bwd} : The halfway-backward matrix (derived from $V^\dagger$).

D.3 Neural Network Architecture

The core model is the `QuantumTransformer`, an autoregressive neural network built with `Flax`.

Hyperparameters

Parameter	Value
Model Dimension (<code>d_model</code>)	64
Attention Heads (<code>num_heads</code>)	8
Transformer Layers (<code>num_layers</code>)	4
Training Batch Size	1,024
Vocabulary Size	5 (0-3 for POVM, 4 for START)

Table 1: Transformer Architectural Specifications

Data Flow

1. **Embedding:** Discrete POVM outcomes and the START token are mapped to 64-dimensional continuous vectors, summed with learned positional embeddings initialized via a normal distribution ($\sigma = 0.02$).
2. **Causal Masking:** A strict causal mask ensures the prediction for qubit k attends exclusively to qubits $< k$.
3. **Self-Attention:** 8 parallel heads process local entanglement features inside the `TransformerBlock`.
4. **Feed-Forward Network:** A position-wise dense block expands and compresses the feature space: $\text{Dense}(128) \rightarrow \text{ReLU} \rightarrow \text{Dense}(64)$.
5. **Output Head:** A final $\text{Dense}(4)$ layer outputs raw logits, converted to normalized log-probabilities via `log_softmax`.

D.4 Optimization and Training Loop

Optimizer Configuration

The network is optimized using the Adam algorithm with an exponential decay learning rate schedule to enable rapid initial exploration followed by fine-tuning:

- **Learning Rate Schedule:** Initial $\eta = 0.001$, decaying by 50% every 5,000 steps, down to a floor of 10^{-5} .
- **Max Steps per Gate:** 20,000 iterations.
- **Early Stopping (Patience):** Training proceeds to the next gate if the loss remains below the convergence threshold (10^{-4}) for 2 consecutive steps.

Autoregressive Sampling and Hybrid Loss

For each training step, the network generates 1,024 sample sequences. The model minimizes a hybrid loss combining KL-divergence and a forward-backward symmetry penalty:

$$\mathcal{L}(\theta) = \mathcal{L}_{KL} + \mathcal{L}_{FB}$$

1. KL Divergence Penalty: Aligns the network’s predicted probabilities P_θ with the target distribution generated by the exact transition matrix O_U . To maintain stable gradients, we use an importance sampling formulation with a dynamic baseline μ :

$$\mathcal{L}_{KL} = -\mathbb{E}_{a \sim P_\theta} \left[\text{stop_grad} \left(\frac{P_{\text{target}}(a)}{P_\theta(a)} - \mu \right) \log P_\theta(a) \right] \quad (\text{D.2})$$

In our JAX implementation, the baseline μ is explicitly calculated as the batch mean of the probability ratios, naturally reducing gradient variance.

2. Forward-Backward Symmetry Penalty: Forces the prior state (θ_{old}) propagated halfway forward to match the newly updated state (θ_{new}) propagated halfway backward via a Mean Squared Error (MSE) penalty on the intermediate states:

$$\mathcal{L}_{FB} = \mathbb{E}_a \left[(O_{fwd} P_{\theta_{old}}(a) - O_{bwd} P_{\theta_{new}}(a))^2 \right] \quad (\text{D.3})$$

D.5 JAX Implementation and Vectorization Details

Vectorized State Manipulation (`jax.vmap`)

To compute the $\mathcal{L}_{FB}$ penalty, the model calculates the expected probability of the sequence if the target qubits were forced into all possible local permutations. The script utilizes `jnp.tile` to duplicate the sampled sequences (e.g., 64 clones for a 3-qubit Toffoli gate), injects the discrete local states, and passes the augmented batch through the Transformer in parallel via `jax.vmap`.

KL Divergence and Numerical Stability

Computing the exact KL divergence dynamically is computationally prohibitive. Instead, the simulation utilizes an importance sampling ratio between the exact physics target and the model’s current probability. To maintain numerical stability and avoid log-zero errors, exact targets are clipped at a minimum of 10^{-30} , and the logarithmic ratio is strictly bound between $[-10.0, 10.0]$.

Crucially, the algorithm must differentiate only with respect to the model’s generative probabilities. This is achieved using `jax.lax.stop_gradient` on the centered ratio, ensuring the physical transition targets act as fixed anchors during backpropagation.

D.6 Scalable Verification (OOM-Proof Branch Tracking)

Evaluating the fidelity of a 22-qubit state involves a Hilbert space of dimension 2^{22} , making exact dense state-vector evaluation prone to Out-Of-Memory (OOM) errors.

To conduct gate-by-gate verification without memory overflow, the simulation implements an explicit amplitude branch-tracking algorithm. The tracker recursively updates only the non-zero probability amplitudes through the sequence of $X, H, CNOT, CS^\dagger$, and *Toffoli* gates, collapsing and aggregating branches dynamically.

During the fidelity evaluation phase, network weights are frozen. The model generates 1,024 samples, which are compared against the exact amplitudes derived from the branch tracker. To smooth the variance of IC-POVM sampling, the simulation loops this Monte Carlo evaluation across 16 independent random seeds, effectively calculating the transition fidelity $\mathcal{F}_c$ across 16,384 distinct configurations per gate without overflowing the GPU.

