F2: OFFLINE REINFORCEMENT LEARNING FOR HAMILTONIAN SIMULATION VIA *Free-Fermionic* SUBROUTINE COMPILATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Compiling shallow and accurate quantum circuits for Hamiltonian simulation is challenging due to hardware constraints and the combinatorial complexity of minimizing gate count and circuit depth. Existing compilers rely on hand-engineered heuristics and cannot adapt to input-dependent structure, limiting optimization opportunities. We introduce **F2**, an offline reinforcement learning framework that exploits free-fermionic structure for quantum simulation. Across benchmarks spanning lattice models, protein fragments, and crystalline materials (12–222 qubits), F2—using a single trained model—reduces R_Z count by 28.8% on average relative to strong industry and academic baselines (IBM Qiskit, Google Cirq/OpenFermion, Berkeley BQSKit), while maintaining errors on the order of 10^{-7}. These results demonstrate that aligning reinforcement learning with algebraic structure can substantially improve scalable quantum circuit synthesis.

1 INTRODUCTION

Quantum computing provides a principled approach to simulating quantum many-body systems that are intractable for classical computation, with applications in chemistry, materials science, and condensed-matter physics (Feynman, 1982; Lloyd, 1996; Aspuru-Guzik et al., 2005; Peruzzo et al., 2014; Cao et al., 2019; McArdle et al., 2020). Realizing this potential requires translating high-level quantum dynamics, formalized as a Hamiltonian matrix, into executable quantum circuits composed of elementary gates. Such circuits are constructed from sequences of one- and two-qubit operations drawn from a fixed native gate set, which together approximate or implement the target unitary evolution (Nielsen & Chuang, 2010; Dawson & Nielsen, 2006; Ross & Selinger, 2014; Kliuchnikov et al., 2013).

Near-term quantum architectures remain severely resource-limited: gate fidelities and restricted qubit connectivity constrain both circuit depth and total gate count (Preskill, 2018). As a result, aggressive circuit optimization is essential for achieving practical quantum advantage on near-term and emerging hardware. However, many core compilation objectives are computationally intractable; in particular, minimizing circuit depth or overall circuit complexity is NP-hard (van de Wetering & Amy, 2024; Ito et al., 2023).

From an algorithmic perspective, quantum circuit synthesis is a challenging *long-horizon optimization problem*. A rich toolbox of compilers and rewrite systems exists, including industrial transpilers (e.g., Qiskit, t|ket⟩, Cirq) and diagrammatic ZX-calculus optimizers (Javadi-Abhari et al., 2024; Sivarajah et al., 2020; Duncan et al., 2020; Cirq Developers, 2024). While these tools achieve strong empirical performance, they rely primarily on hand-engineered heuristics and lack the ability to adapt to input-dependent structure.

Reinforcement learning (RL) offers a natural framework for sequential synthesis problems, but applying RL to quantum compilation presents several fundamental challenges. First, quantum systems are widely believed to exhibit exponentially large feature spaces, making naïve state representations intractable and generalization difficult. Second, circuit synthesis is a sparse-reward, long-horizon control problem, which significantly complicates exploration and credit assignment. Third, synthesis actions combine discrete structural choices with continuous parameters, resulting in large and heterogeneous action spaces that are difficult to learn effectively. As a result, existing learning-based approaches to quantum compilation have been limited in scalability and robustness (Ruiz et al., 2024; Tang et al., 2024).

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.

In this work, we show that these challenges can be overcome by aligning reinforcement learning with the algebraic structure of quantum dynamics. Many physically relevant Hamiltonians contain substructures whose evolution lies within classically simulatable Lie subgroups, most notably free-fermionic (quadratic) systems. By isolating these components, we obtain synthesis problems with compact polynomial-size representations, deterministic transitions, and exact time reversibility—properties that are highly favorable for offline reinforcement learning.

We introduce **F2**, an offline reinforcement learning framework for Hamiltonian simulation that exploits these properties to enable scalable learning-based compilation. F2 formulates unitary synthesis as a deterministic Markov decision process over a structured subgroup of the unitary group, where states admit polynomial-size representations and actions correspond to physically motivated generators. Crucially, we leverage the exact reversibility of unitary evolution to generate abundant, guaranteed-successful training data: by sampling trajectories backward from the identity, we construct offline datasets with dense supervision and well-defined optimal values, without requiring environment interaction or exploration.

To further stabilize learning, we introduce a geometric value regularizer that incorporates domain knowledge about the underlying group structure. Free-fermionic unitaries admit a canonical decomposition into independent rotations, yielding an intrinsic notion of distance to the identity. We parameterize this distance and learn its scaling jointly with the value function, providing a geometry-aware inductive bias that improves value calibration and approximation accuracy during offline training.

We evaluate F2 on a diverse set of Hamiltonian simulation benchmarks, including strongly correlated lattice models, crystalline materials, and biologically relevant protein fragments, with 12 to 222 qubits. Across all domains, F2 consistently outperforms state-of-the-art compilers, achieving substantial reductions in R_Z count while maintaining high-fidelity approximations. These results demonstrate that offline reinforcement learning with reversible, structure-aware environments can effectively address long-horizon program synthesis problems. Our major contributions can be summarized as follows:

- We formulate structured unitary synthesis as a deterministic, reversible Markov decision process with compact polynomial-size state representations.
- We introduce a trajectory-reversal data generation strategy that produces guaranteed-successful offline training data by exploiting exact reversibility.
- We propose a learned geometric regularizer for value functions that aligns reinforcement learning objectives with intrinsic group structure.
- We demonstrate significant empirical improvements over state-of-the-art quantum compilers on large-scale Hamiltonian simulation benchmarks.

2 F2 Design and Methodology

The goal of quantum circuit compilation is to minimize gate count and circuit depth under hardware constraints. In Hamiltonian Simulation a quantum computer performs an approximate simulation of quantum dynamics, starting from initial conditions described by a Hamiltonian matrix. This simulation is typically performed via a product formula decomposition called Trotterization. During simulation, Trotter steps are executed many times to approximate time evolution, which means that even slightly more efficient compiled circuits yield dramatic end-to-end efficiency savings. F2 focuses on making Hamiltonian simulation more efficient by optimizing circuits that implement Trotter steps.

The key insight underlying F2 is that many practically important Hamiltonians contain *classically simulatable* substructures called *free-fermionic* terms. Trotterization factorizes the global time-evolution operator into a product of simpler unitaries, making it possible to isolate these tractable components from genuinely interacting terms. While existing compilers typically treat all terms uniformly, F2 explicitly exploits this structural asymmetry: the clasically-simulatable *free-fermionic* terms are aggressively optimized using offline reinforcement learning, while the remaining non-free-fermionic terms are compiled using conventional heuristics.

Figure 1 provides an overview of the F2 workflow. The input Hamiltonian is first partitioned into free-fermionic and non-free-fermionic terms (Step 1). Interacting terms are compiled using state-of-the-art classical techniques, while free-fermionic terms are passed to an RL-based compiler that operates over a compact, classically simulatable representation (blocks on the right side in Figure 1). The resulting circuits are then composed to form a deeply optimized Trotter step. By concentrating learning capacity on the tractable subspace where deep optimization is possible, F2 uncovers circuit reductions that are largely inaccessible to prior approaches. A detailed description of the Hamiltonian

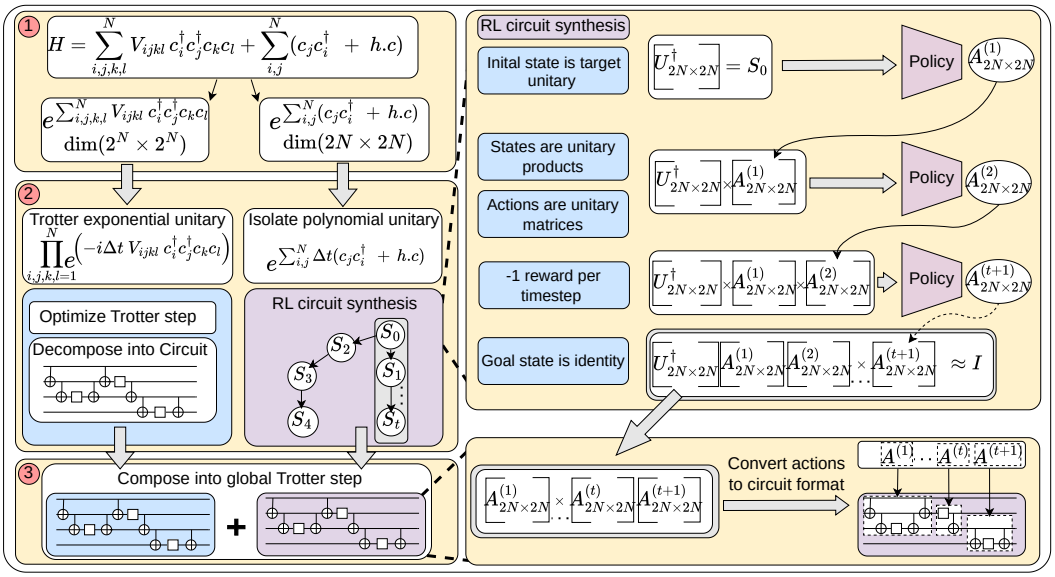


Figure 1: The goal is to optimize a Trotter step. 1) The input Hamiltonian is separated into classically representable terms and exponential terms. 2) Exponential unitaries are compiled using state-of-the-art techniques, while classically simulatable unitaries are decomposed with our RL algorithm. 3) The policy architecture constructs a trajectory by sequentially multiplying actions into the target unitary until the terminal state is reached. Actions have theoretical decompositions into circuit representations, creating a mapping between states and quantum circuits. After both compilers complete circuit synthesis, the resulting circuits are concatenated to produce a deeply optimized Trotter step.

partitioning and compilation procedure is in Appendix C. The remainder of this section describes our learning-based approach to compiling free fermionic terms.

2.1 Reinforcement Learning Environment for F2

We cast the compilation a free-fermionic term as a sequential decision making problem. The goal of free-fermionic circuit compilation is to select a sequence of quantum gates (*Pauli string exponentials*) that transition a target free-fermionic unitary matrix to the identity matrix. We model compilation as a deterministic Markov Decision Process environment with discrete states and actions.

Unlike previous approaches to RL for quantum algorithm synthesis, our environment restricts our model to walk along a Lie algebra of unitaries that have a guaranteed polynomial representation. Concretely, we expose the agent to a compact action alphabet built from exponentials of Pauli strings that map to these free-fermionic evolutions via standard qubit–fermion encodings and gate decompositions. Importantly, restriction also implies exact time reversibility of the environment dynamics, a property we exploit in Section 2.2 to generate guaranteed-successful offline training data.

State, action, and transition. Let U denote the target free-fermionic unitary we wish to synthesize, expressed as a matrix $U \in SO(2N)$ where N is the number of *free-fermionic modes.*[1] We define the set of environment states as $\mathcal{S} = SO(2N)$ and let $U \in \mathcal{S}$ be the initial state. At each timestep t the agent chooses an action $A_t \subseteq \mathcal{A} \subseteq SO(2N)$ from a discrete set of parameterized gates to extend the partial quantum circuit synthesis. Each action transitions the environment state according to left-multiplication; the state at each timestep t is given by

$$S_t \coloneqq A_{t-1} \cdots A_0 U.$$

Reward and termination. In our MDP environment, each episode represents an attempted quantum synthesis, which succeeds when the environment state is approximately equal to the identity matrix. Formally, for a fixed tolerance parameter $\epsilon > 0$ we terminate the episode at the first timestep t such that $\frac{1}{2N}|\mathrm{Tr}(S_t)| > 1 - \epsilon$. To encourage the agent to find small quantum circuits, we associate a reward of $r_t = -1$ with each timestep. We also include a fixed maximum episode horizon T_{max} and truncate unsuccessful episodes if they do not succeed in the first T_{max} steps.

[1] U is the target free-fermionic unitary's real Majorana form. See B for background on free-fermionic systems.

Action parameterization. Each action $\mathcal{A} \subseteq SO(2N)$ represents a quantum gate and can be compiled to a hardware-compatible circuit using conventional techniques. Concretely, each action is an element discrete set of parameterized exponentials of Pauli strings drawn from a free-fermionic generating set:

$$\mathcal{A} = \Big\{ A(P,\theta) = \exp\big(-i\,\tfrac{\theta}{2}\,P\big) \;\Big|\; P \in \mathcal{P},\ \theta \in \Theta \Big\},$$

$$\mathcal{P} = \Big\{ \{X_i X_{i+1},\, Y_i Y_{i+1} \,\Big|\, i{=}1,\ldots,N{-}1\Big\} \cup \Big\{ Z_i \,\Big|\, i{=}1,\ldots,N\Big\}, \quad \Theta = \Big\{ \pm\,\tfrac{\pi}{2^k} \,\Big|\, k{=}1,\ldots,16\Big\}.$$

This gives 34 generators ($N = 12$ on-site terms and 11 nearest-neighbour pairs $\times\{XX, YY\}$) and 32 angles, an action alphabet of $|\mathcal{A}| = 1088$. The state matrix and every action matrix are elements of $SO(2N)$; the agent never forms the 2^N-dimensional unitary. Since many gates do not commute, the ordering of actions materially affects S_t. We include more details about Pauli strings and Trotterization in A; the details are not needed to follow the rest of this paper.

Practical simulation and gate mapping. Although the environment is defined in full unitary form, all environment step dynamics are implemented in a polynomial-time free-fermion simulator. This makes *step*, *reset*, and *check-terminal* operations efficient. Each action $A(P,\theta)$ is compiled to a finite native gateset (e.g., CNOTs plus single-qubit R_z) using standard decompositions, so the agent's decisions lift directly to hardware-compatible circuits for downstream applications in chemistry and materials.

2.2 Trajectory-Reversal Data Generation

We exploit the exact reversibility of unitary dynamics to generate guaranteed-successful trajectories without environment interaction. Specifically, we *walk backwards* from the identity unitary by sampling arbitrary actions that (when inverted) define valid forward steps. These trajectories are then treated as off policy environmental interactions and used in offline RL techniques for generating an efficient policy.

We can express the trajectory reversal for data generation as follows. Recall from Section 2.1 that each action $A \in \mathcal{A}$ is a matrix in $SO(2N)$ that transitions the environment state according to left-multiplication, and the goal of circuit synthesis is to select a sequence of actions that transitions the environment state toward the identity matrix. Because action A satisfies the orthogonality condition $AA^\top = I$, any synthesis trajectory admits an exact time reversal. In other words, the environment's dynamics are invertible under each action. We can thus sample a guaranteed-successful trajectory of length T as follows: we first set the final state S_T to the identity I. We then sample a random sequence of actions $A_{T-1}, A_{T-2}, \ldots, A_0$. We then compute the state at each timestep $0 \le t < T$ of the trajectory as $S_t \coloneqq I A_t^\top A_{t+1}^\top \cdots A_{T-1}^\top$ and include the -1 per timestep reward as described above. We then include this trajectory in the offline dataset used to train F2. We additionally label each transition (S_t, A_t, r_t, S_{t+1}) with the action history $A_0, \ldots, A_{t-1}$ and the (undiscounted) return-to-go $G_t = r_t + \cdots + r_{T-1}$ for use with the Monte Carlo learning approach described in the following subsection.

2.3 Geometric Regularization and Training

F2 is pretrained with offline Monte Carlo learning to learn a value function V_ω parameterized by learnable parameters ω. The value function takes as input both the current state S and the history of actions. We augment the typical Monte Carlo target (return-to-go) G_t, which quantifies the number of steps left until the the trajectory terminates at the identity, with potential-based reward shaping (Ng et al., 1999) to incorporate a geometric notion of distance from the identity. We additionally include a learned *geometric regularizer* in the value network's loss function to further align F2 with unitary structure.

Geometric potential function. Our *geometric potential function* $\phi : \mathcal{S} \to \mathbb{R}$ captures a geometric notion of distance from identity that provides an inductive bias tailored to free-fermionic synthesis. The key observation is that each free-fermionic state S can be block-diagonalized into a series of independent rotations. A special case is the goal state I where the block diagonalized form of the matrix has no rotations on independent subspaces. A geometric potential function can then be a fitted squared sum of the independent rotation angles where the goal state I has the unique value of zero. To formalize, each state S can be factorized as follows:

$$\exists\, O \in SO(2N):\; O^\top S\, O \;=\; \bigoplus_{j=1}^{N} \begin{bmatrix} \cos\theta_j & \sin\theta_j \\ -\sin\theta_j & \cos\theta_j \end{bmatrix}, \tag{1}$$

We then define the geometric potential function as $\phi(S) = -\frac{1}{N\pi^2}\sum_{j=1}^{N}\theta_j^2$. Here, we include normalization so that for any state S we have $\phi(S) \leq 0$, with $\phi(S) = 0$ exactly when S is the goal unitary I. In the equation above, $\theta_1, \ldots, \theta_N \in [0, \pi]$ are the rotation angles of $S \in SO(2N)$, the single-particle map of the remaining unitary (the current, partially synthesized circuit composed with the target). In practice we obtain the rotations from the Cayley transform $K = (I - S)(I + S)^{-1}$, whose singular values come in N pairs σ_j, via $\theta_j = 2\arctan\sigma_j$.

For each trajectory in our offline dataset, we label each transition with the *shaped reward* $\tilde{r}_t = -1 + \eta(\phi(S_{t+1}) - \phi(S_t))$ and *shaped Monte Carlo return-to-go* $\tilde{G}_t = \sum_{t'=t}^{T-1}\tilde{r}_{t'}$. Here, η is a scalar hyperparameter that controls the weight of the potential-based-reward shaping; in our experiments we use $\eta = 2$.

Geometric regularizer. To further incorporate geometry informed heuristics, we incorporate a learned geometric regularizer directly into the value network's loss function. Our geometric regularizer is defined as

$$\Phi_{\alpha,\beta}(S) = \alpha + \beta\,\phi(S), \tag{2}$$

where the scalars α, β are learned jointly with the critic rather than set as hyperparameters, so that the regularizer can match the scale of the Monte Carlo targets. For a sampled transition with current state S_t, action history $A_{<t} = A_0, \ldots, A_{t-1}$, and shaped return-to-go $\tilde{G}_t$, we define the critic's loss as:

$$\mathcal{L}(\omega, \alpha, \beta) = \mathcal{H}_\delta\big(V_\omega(S_t; A_{<t}),\, \tilde{G}_t\big) + \lambda\left(V_\omega(S_t; A_{<t}) - \Phi_{\alpha,\beta}(S_t)\right)^2, \tag{3}$$

where ω are the value network's learnable parameters, $\mathcal{H}_\delta$ is the Huber loss with parameter δ, and λ is a weighting hyperparameter. The first term in the loss function is standard for Monte Carlo value learning. The second encourages value estimates to correlate with the intrinsic geometric distance to the target. In our experiments we use hyperparameters $\delta = 25$ and $\lambda = 0.01$.

Value network and training. F2 trains a value network V_ω, implemented as a dual-tower transformer. An axial-attention encoder over the $2N \times 2N$ state and a standard transformer over the action history, fused into a scalar value head. Appendix D gives the architecture, Figure 3, and all hyperparameters; Appendix E defines the compositional action embedding.

The value network is first pretrained using offline Monte Carlo learning and the objective in 3. Following Monte-Carlo pretraining for stability, we switch to TD updates to refine local consistency. During TD-based training, the geometric regularizer is dropped. The regimen of Monte Carlo pretraining prior to TD based learning was introduced in Jelley et al. (2024).

Policy Inference. During inference, at each timestep t the F2 policy observes the current state S_t and action history $A_0, \ldots A_{t-1}$. F2 leverages the fact that dynamics are known and deterministic to perform greedy policy inference with respect to the learned value function. F2 selects the action $A_t = \mathrm{argmax}_{A \in \mathcal{A}}\, V_\omega(AS_t; A_0, \ldots A_{t-1}, A)$.

3 Experiments

In this section, we evaluate F2 on a diverse set of Hamiltonian simulation benchmarks designed to test scalability, circuit efficiency, and approximation accuracy across physics, materials science, and computational chemistry.

3.1 Experimental Setup

F2 Training. All compilation results are obtained using a *single* F2 model trained offline on $10{,}000$ synthetic trajectories comprising $1{,}502{,}443$ frames generated via trajectory reversal with actions chosen uniformly at random. The trained policy is reused across all benchmark instances and domains, without per-instance or per-family retraining. During compilation, inference consists of greedy action selection using the learned value function, yielding a deterministic synthesis procedure with no online environment interaction. Training is performed once for the shared model and amortized across all compilation tasks. The optimizer used for training is Adam (Kingma & Ba, 2017). Critics are trained for 6 epochs with batch size 32 with learning rate 2×10^{-4} and weight decay 10^{-4}, using a single shuffled loader over the full dataset. Offline training was performed on a single NVIDIA A100 GPU and an AMD EPYC 9654P 96-core CPU. Our implementation uses PyTorch 2.5.1 with CUDA 12.1.

Table 1: Number of R_Z rotations in each compiled circuit. Averaged across benchmarks, F2 uses 28.8% fewer R_Z rotations than the best other compiler on each benchmark.

Benchmark	Qubits	F2	Google-Cirq	IBM-base	IBM-rustiq	BQSKit-QFactor	IBM-ai
Heisenberg 1D	12	**67**	563	264	113	221	264
Heisenberg 2D	12	**109**	605	408	229	263	408
Fermi-Hubbard 1D	12	**38**	450	198	179	198	198
Fermi-Hubbard 2D	12	**94**	506	452	208	254	763
t-J 1D	12	**162**	574	881	166	322	1392
t-J 2D	12	262	674	1302	**216**	422	1898
Heisenberg 1D	72	**437**	3413	1704	504	1361	1704
Fermi-Hubbard 1D	72	**264**	3372	1368	1160	1320	1368
t-J 1D	72	**1148**	4256	6191	1546	2204	10092
Heisenberg 1D	144	**881**	6833	3432	972	2729	3432
Fermi-Hubbard 1D	144	**540**	6756	2772	2330	2652	2772
t-J 1D	144	**2340**	8556	12563	3007	4452	20532
PD1 fragment	28	**111**	202	428	169	211	598
PD1 fragment +	74	**929**	2495	3385	1280	1608	5447
PD1 fragment ++	222	**4077**	10661	14463	4896	6609	23751
ABL fragment	84	**777**	914	2495	923	1088	3865
La_2CuO_4	16	**2496**	14464	9216	2513	6320	9216
La_2CuO_4 (Hubbard)	32	**11648**	51328	53808	15302	26464	—*
La_2CuO_4 (spectral-fit)	32	**11648**	51328	53808	15304	26464	—*
κ-$(ET)_2Cu_2(CN)_3$	32	**2224**	7184	9771	2273	4076	15729

Table 2: Depth of each compiled circuit. F2 has the lowest depth on 5 of the 20 benchmarks and is within 4% of the best compiler on 13 of 20; the remaining seven, where it is 1.2–2.2× deeper. d ominate any average.

Benchmark	Qubits	F2	Google-Cirq	IBM-base	IBM-rustiq	BQSKit-QFactor	IBM-ai
Heisenberg 1D	12	129	390	429	269	**73**	429
Heisenberg 2D	12	308	567	429	369	**251**	429
Fermi-Hubbard 1D	12	**60**	207	156	244	62	156
Fermi-Hubbard 2D	12	**211**	372	299	356	227	495
t-J 1D	12	**562**	680	1557	592	564	2322
t-J 2D	12	841	959	2064	**809**	843	2815
Heisenberg 1D	72	157	400	2769	1439	**82**	2769
Fermi-Hubbard 1D	72	110	381	1026	1713	**62**	1026
t-J 1D	72	3571	3839	10947	4721	**3549**	16633
Heisenberg 1D	144	165	405	5577	2843	**87**	5577
Fermi-Hubbard 1D	144	110	381	2070	3477	**62**	2070
t-J 1D	144	7197	7465	22215	9386	**7175**	33805
PD1 fragment	28	**167**	179	250	185	179	364
PD1 fragment +	74	2182	2321	2664	4058	**2175**	3852
PD1 fragment ++	222	15899	16156	22321	33703	**15894**	29692
ABL fragment	84	**1901**	1906	2827	1904	1965	3721
La_2CuO_4	16	6096	10992	**2808**	11286	5635	**2808**
La_2CuO_4 (Hubbard)	32	22019	30819	38127	78104	**21792**	—*
La_2CuO_4 (spectral-fit)	32	22019	30819	38127	78103	**21792**	—*
κ-$(ET)_2Cu_2(CN)_3$	32	6481	7266	9475	13441	**6460**	13194

*Exceeded the 320 GB memory cap.

Benchmarks. We evaluate our method across three domains spanning both foundational science and commercially relevant workloads: (i) strongly correlated physics models, (ii) crystalline materials, and (iii) protein fragments. First, we include the Fermi–Hubbard, Heisenberg, and t–J models, which arise as effective electronic Hamiltonians in appropriate limits and are widely used to study quantum phases such as spin liquids and quantum glasses Hubbard (1963); Anderson (1959); Chao et al. (1977). Second, we evaluate on two experimentally important materials—La_2CuO_4 and κ-$(ET)_2Cu_2(CN)_3$—whose low-energy behavior can be mapped onto Hubbard- or Heisenberg-type models with parameters obtained from *ab initio* and fitted calculations Lane et al. (2018); Zheng et al. (2005); Coldea et al. (2001); Nakamura et al. (2009). Finally, we include fragments of biologically relevant proteins. Modeling full proteins requires thousands of qubits, but focusing on functional hotspots yields tractable subsystems while preserving biochemical relevance Berman et al. (2000). We use fragments from PD-1 and ABL1, two human membrane-associated proteins involved in immune regulation and cell-cycle control, and key targets in cancer therapeutics. Across all domains, problem sizes range from 12 to 222 qubits and were implemented for compilation using OpenFermion with our neural network having a maximum error allowance of 10^{-6}. All benchmarks are compiled using the same trained F2 model, allowing us to evaluate generalization across Hamiltonian families, system sizes, and application domains.

Baselines We compare against publicly available industry baselines from IBM Qiskit 2.3.1 (Qiskit Development Team, 2024) and and Google Cirq 1.5.0 Developers (2024) . We compare F2 against five strong baselines from industry and academia: **IBM-base** synthesises the full Hamiltonian as a first-order Trotter circuit via IBM Qiskit's (version 2.3.1) (Qiskit Development Team, 2024)PauliEvolutionGate with `LieTrotter` (Li et al., 2021), transpiled at optimization level 3. **IBM-rustiq** synthesises the same Trotter gate through Qiskit's `PauliEvolutionSynthesisRustiq` plugin (de Brugière & Martiel, 2024) with default options (count-optimising, order-preserving), then transpiles at level 3. **Google-Cirq** is Google's Givens-rotation compiler (`bogoliubov_transform`) (McClean et al., 2020; Jiang et al., 2018), run in Cirq and transpiled at level 3. **QFactor** is an approximate circuit-synthesis optimiser from Lawrence Berkeley National Laboratory (Kukliansky et al., 2023). Finally, **Pauli-RL** is IBM's reinforcement learning synthesiser for Pauli networks (Dubal et al., 2025b).

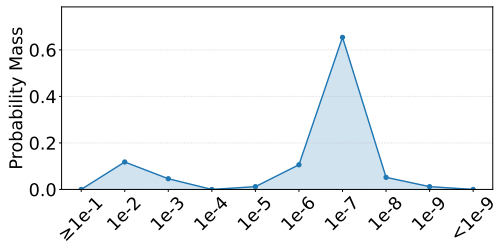
Evaluation pipeline. In each experiment, every method compiles the same targets and is scored by the same metric; the methods differ only in how the free-fermionic part is synthesised. For the circuit-cost experiments: (1) the Hamiltonian is built with Google OpenFermion (version 1.7.0) (McClean et al., 2017) and split into 12-mode windows (Appendix C) for every baseline as a high level representation; (2) IBM-base, IBM-ai and IBM-rustiq compile the whole Hamiltonian as a Trotter circuit, while Google-Cirq, Qfactor and F2 compile the free fermionic part, as mentioned in 1 and add then use IBM rustiq for compilation of the non-free fermionic components (3) every circuit is transpiled to the basis $\{R_z, \mathrm{CX}, H, S, S^\dagger\}$; (4) we report R_Z count (which measure of size), and depth. For all experiments we capped the memory usage at 320GB and if the method went over in memory we leave as an asterisk in our results.

Metrics. We evaluate compiler performance using three quantitative metrics. The **circuit depth** D measures the (parallelism) depth of the circuit after applying standard commutation and gate-merging passes, reflecting temporal efficiency. The R_Z **count** G is the number of R_Z rotations in the compiled circuit. Finally, we compute the **fidelity** with respect to the target unitary U as $F := |\frac{1}{d}\operatorname{Tr}(U^\dagger \hat{U})|$ where d is the Hilbert-space dimension and $\hat{U}$ is the compiled unitary.

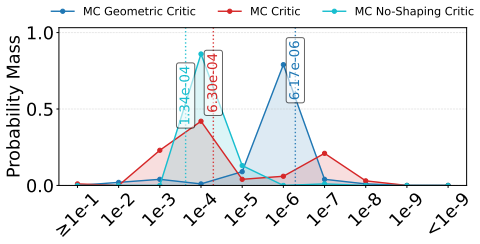
3.2 Results

R_Z count and depth. Tables 1 and 2 summarize R_Z count and depth across the suite. Averaging across benchmarks, our approach reduces R_Z count by 28.8% (relative to the strongest baseline on each benchmark). The largest reductions are on the Fermi–Hubbard and Heisenberg lattice models (up to 79%), the smallest on the materials benchmarks (1–24%); F2 uses more R_Z rotations than the best baseline on one benchmark (t-J 2D, 12 qubits). On depth, F2 is the best compiler on 5 of 20 benchmarks and within 4% of the best on 13 of 20; on the seven remaining benchmarks, it is 1.2–2.2$\times$ deeper. A potential explanation for this is that the reward function penalizes long sequences but does not finely differentiate operations with very different downstream parallel structure.

Compilation accuracy. Our approach targets *approximate* unitaries within a prescribed tolerance rather than analytical identities. Given hardware noise, many applications only require bounded error (e.g., domain-specific "chemical accuracy" thresholds Louvet et al. (2023)) rather than exact synthesis.



(a) F2 Compilation Fidelity.



(b) Value regularization study (no TD postraining)

Figure 2: Fidelity distributions grouped by error $(1 - F)$. **Left (2a):** 500 random unitaries with small angle rotations compiled by F2. **Right (2b):** 100 random unitaries with small angle rotations compiled by Monte Carlo pretrained F2 three critics, dotted lines at each median. "MC No-Shaping Critic" is trained on plain labels with no geometric signal (median 1.34×10^{-4}); "MC Critic" on labels shaped by the geometric potential at flat scaling $\eta = 2$ (median 6.30×10^{-4}); "MC Geometric Critic" on the same shaped labels with the learned value-loss regularizer of equation 3 (median 6.17×10^{-6}).

Figure 2a reports compilation fidelity across random unitaries found in Trotter steps; accuracy remains within target tolerances while improving R_Z count. Here, Hamiltonians and coefficients are sampled from our benchmark with the range of Δt between $(-0.02, 0.02)$, which is commonly cited as the target Trotter step size interval Heyl et al. (2019); Babbush et al. (2014); Yi & Crosson (2022); Poulin et al. (2015). We distinguish two quantities that are easily conflated. The *acceptance threshold* is the largest error we admit: a compilation would be rejected if its error exceeds 10^{-6}. The *achieved* error is what F2 typically attains within that budget, on the order of 10^{-7} (Figure 2a). For our work and the current trajectory of quantum computing, an error on the order of 10^{-7} is routinely attained and justified for many near and intermediate-term Hamiltonian simulations without compromising accuracy targets.

Effect of the geometric regularizer. This experiment isolates the effect of geometric value regularization within the shared F2 model, holding training data and architecture fixed. Now the performance changes due to the learned regularizer of equation 3. Three critics are compared, differing only in how the geometric signal enters training. The first ablates both the potential-based reward shaping and the geometric regularizer: it is trained on plain return-to-go labels $G_t = -(T - t)$ and receives no geometric signal at all. The second ablates just the geometric regularizer and is trained on the shaped return-to-go labels $\tilde{G}_t$ of Section 2.2, in which each reward is augmented by a potential-based term with $\Phi(S_t)$ the normalized geometric distance of Section 2.3 and $\eta = 2$. The third is trained on the shaped return-to-go labels *and* carries the learned value-loss regularizer of equation 3. Comparing the first two asks whether the geometric signal helps at all; comparing the last two asks whether lifting it into the value loss adds anything beyond shaping the labels. In our environment, future rewards are not discounted, so there is no discount factor. The rewards regressed on were Monte Carlo rewards only, and we did not perform a TD-based fine-tuning step afterwards. Next, over 100 random unitaries, the models performed compilation, and the approximation accuracy was be recorded. The approximation error distributions along with the medians are plotted in Figure 2b. We find that including both shaped return-to-go labels and the geometric regularizer significantly improves compilation fidelity.

Effect of training-set size. Retraining the full F2 critic on nested quarter- and half-size subsets of the training set at a fixed gradient-step budget shows that median approximation error falls monotonically by a factor of 430 from 2,500 to 10,000 trajectories. Appendix G gives the curve, the exact statistics, and the replication details.

4 Related Work

Trajectory data augmentation and generation. Sparse-reward RL has pursued (i) reversible or reverse-time augmentation, (ii) goal relabeling, (iii) curricula over start states, and (iv) learned backward/inverse models. TSDA synthesizes reverse-time transitions under approximate time reversibility but is outcome-agnostic Barkley et al. (2024). HER densifies rewards by relabeling goals using achieved states, but in doing so changes the task goal rather than producing success for an unmet fixed goal Andrychowicz et al. (2017). RCG expands the start-state distribution outward from the goal under a reset-anywhere assumption Florensa et al. (2018). Offline approaches learn

reverse/inverse-dynamics models from datasets to improve coverage or data efficiency Höftmann et al. (2023); Wang et al. (2021). In contrast, we keep the original goal fixed and exploit unitary-group structure with a geometry-aware shaping signal to synthesize *forward* successful rollouts from diverse initial states—improving coverage near and far from the terminal manifold without goal relabeling or reverse-time samples.

Value function regularization. Potential-based reward shaping densifies sparse rewards via a hand-designed potential while preserving optimality Ng et al. (1999). Auxiliary-task methods add extra objectives (often via multiple heads) to learn representations useful for control Jaderberg et al. (2016); Laskin et al. (2020); Yarats et al. (2021); Hafner et al. (2023). Value-regularization methods, especially in offline/conservative RL, instead penalize value estimates to reduce over-optimism on out-of-distribution states Kumar et al. (2020). Closest in spirit, heuristic-guided approaches such as HuRL and HUBL incorporate heuristics to reduce variance and the effective decision horizon Cheng et al. (2021); Geng et al. (2023). Our approach differs by using a single value head whose objective is directly modified by a learned geometric regularizer: we lift the heuristic into the value loss in addition to using it as a potential-based shaping term Ng et al. (1999), and learn its scale rather than fixing it, avoid multi-head auxiliary formulations Jaderberg et al. (2016); Laskin et al. (2020); Yarats et al. (2021); Hafner et al. (2023), and target task-specific off-policy generalization rather than task-agnostic robustness regularization Kumar et al. (2020).

Deep learning for quantum compilation and routing. Early RL-based synthesis targeted single-qubit unitaries Zhang et al. (2020). Later multi-qubit work typically assumes circuit-form inputs and does not inject quantum-specific inductive biases Kölle et al. (2024); Rietsch et al. (2024). Specialized encoders have been proposed for exact Clifford+T compilation Ruiz et al. (2024), while recent RL synthesis trends emphasize online learning without explicit structural priors over the unitary Broersma et al. (2024); Nogúe et al. (2025); Kolle et al. (2024); Dubal et al. (2025a). For routing, CNN-based online RL similarly presumes a circuit representation Tang et al. (2024); Kremer et al. (2025). In contrast, we exploit classically simulatable kernels to obtain compact, structure-aware state representations that share information across action subspaces.

Non–AI compiler methods for Hamiltonian simulation. Classical compilation began with simultaneous diagonalization of commuting Pauli terms Cowtan et al. (2020); van den Berg & Temme (2020), then moved to reordering-driven cancellation strategies Li et al. (2021); Gui et al. (2020); Anastasiou et al. (2024). Pauli-network synthesis further reduces Trotter-step depth de Brugière & Martiel (2024); Paykin et al. (2023); Schmitz et al. (2024); Decker et al. (2025). QuCLEAR explores Clifford extraction/absorption for Hamiltonian simulation, but requires updating the observable while leaving other circuit components reusable Liu et al. (2024). Grouping non-commuting terms can improve approximation accuracy yet does not necessarily improve per-step gate count over state of the art Decker et al. (2025).

5 Conclusion

This work demonstrates that offline RL can be used for gate-efficient compilation of quantum algorithms. Our work has shown average R_Z count reductions of 28.8% compared to strong industry and academic baselines. While this progress is promising, several directions remain for future work. An important extension is applying this paradigm to other classically tractable subroutines, such as those described by tensor-network representations. Incorporating continuous action parameters and hardware-aware architectural constraints may further improve synthesis quality. More broadly, combining offline learning with limited online refinement presents a promising direction for scalable learning-based quantum compilation.

Our work focuses on making quantum simulation more efficient, and thus inherits all of the societal impacts, both positive (e.g. novel drug discovery) and negative (e.g. compromising encryption schemes) of quantum computing.

6 AI Use Statement

In this work we used generative AI tools for the following tasks with required disclosure: designing and providing feedback on experiments, in the form of written experiment protocols for the added baselines, the ablations and the timing study; implementing methods, namely the harnesses that run the BQSKit-QFactor and Qiskit AI-synthesis baselines, the compilation-time measurements and the data-scaling evaluation against our existing pipeline; and interpreting results, in the form of first

drafts of the sentences in Section 3 that summarize Tables 1 and 2. We did not use generative AI tools to generate the training data, to formulate the hypotheses or the method, or to write proofs, and the remaining tasks with required disclosure are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: editing the paper for readability and notational consistency, restructuring tables and figures, drafting parts of the experimental section and this statement, identifying and checking related literature, and creating and editing software code.

We have reviewed all AI-assisted work. All AI-generated code was read by the authors, and every baseline circuit it produced was verified against the target unitary before its gate counts were reported. All numbers in the paper were recomputed by the authors from the raw experiment records. AI-drafted text was revised by the authors, and literature suggested by AI tools was read in the original before being cited.

REFERENCES

Panagiotis G. Anastasiou, Yanzhu Chen, Nicholas J. Mayhall, Edwin Barnes, and Sophia E. Economou. Tetris-adapt-vqe: An adaptive algorithm that yields shallower, denser circuit ansatz-es. *Physical Review Research*, 2024. Published 2024; arXiv:2209.10562.

P. W. Anderson. New approach to the theory of superexchange interactions. *Physical Review*, 115(1): 2–13, 1959. doi: 10.1103/PhysRev.115.2.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/7090-hindsight-experience-replay.pdf.

Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005. doi: 10.1126/science.1113479.

Ryan Babbush, Jarrod McClean, Dave Wecker, Alán Aspuru-Guzik, and Nathan Wiebe. Chemical basis of trotter error in quantum chemistry simulation. *arXiv preprint arXiv:1410.8159*, 2014. URL https://arxiv.org/abs/1410.8159.

Ben Barkley, Amy Zhang, and David Fridovich-Keil. An investigation of time reversal symmetry in reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, PMLR, 2024. URL https://proceedings.mlr.press/v242/barkley24a/barkley24a.pdf.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, Tejas Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000. doi: 10.1093/nar/28.1.235.

Sergey Bravyi. Lagrangian representation for fermionic linear optics. *Quantum Information & Computation*, 5(3):216–238, 2005.

J. Broersma, V. Bergholm, J. Gacon, and V. Dunjko. Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning. *arXiv preprint arXiv:2405.13196*, 2024. URL https://arxiv.org/abs/2405.13196.

Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, 2019. doi: 10.1021/acs.chemrev.8b00803.

K. A. Chao, J. Spalek, and A. M. Oles. Kinetic exchange interaction in a narrow band. *Journal of Physics C: Solid State Physics*, 10(10):L271–L276, 1977. doi: 10.1088/0022-3719/10/10/002.

Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning, 2021. URL https://arxiv.org/abs/2106.02757.

Cirq Developers. Cirq, 2024. URL https://zenodo.org/doi/10.5281/zenodo.4062499. Python library for quantum circuits.

R. Coldea, S. M. Hayden, G. Aeppli, T. G. Perring, C. D. Frost, T. E. Mason, S.-W. Cheong, and Z. Fisk. Spin waves and electronic interactions in la_2cuo_4. *Physical Review Letters*, 86(23): 5377–5380, 2001. doi: 10.1103/PhysRevLett.86.5377.

Alexander Cowtan, Will Simmons, and Ross Duncan. A generic compilation strategy for the unitary coupled cluster ansatz, 2020. Preprint, 27 Aug 2020.

Christopher M. Dawson and Michael A. Nielsen. The solovay–kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.

Timothée Goubault de Brugière and Simon Martiel. Faster and shorter synthesis of hamiltonian simulation circuits. *arXiv preprint*, arXiv:2404.03280, 2024. Preprint, 4 Apr 2024.

Ethan Decker, Lucas Goetz, Evan McKinney, Erik Gustafson, Junyu Zhou, Yuhao Liu, Alex K. Jones, Ang Li, Alexander Schuckert, Samuel Stein, Eleanor Crane, and Gushu Li. Kernpiler: Compiler optimization for quantum hamiltonian simulation with partial trotterization, 2025. URL https://arxiv.org/abs/2504.07214.

Cirq Developers. Cirq: A Python Framework for Creating, Editing, and Invoking Noisy Intermediate Scale Quantum (NISQ) Circuits, 2024. URL https://doi.org/10.5281/zenodo.11398048.

A. Dubal, D. Kremer, S. Martiel, V. Villar, H. Wang, and J. Cruz-Benito. Pauli network circuit synthesis with reinforcement learning. *arXiv preprint arXiv:2503.14448*, 2025a. URL https://arxiv.org/abs/2503.14448.

Ayushi Dubal, David Kremer, Simon Martiel, Victor Villar, Derek Wang, and Juan Cruz-Benito. Pauli network circuit synthesis with reinforcement learning. *arXiv preprint arXiv:2503.14448*, 2025b.

Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. *Quantum*, 4:279, 2020. doi: 10.22331/q-2020-06-04-279.

Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982. doi: 10.1007/BF02650179.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning, 2018. URL https://arxiv.org/abs/1707.05300.

Sinong Geng, Aldo Pacchiano, Aldo Pacchiano, A. Kolobov, and Ching-An Cheng. Improving offline rl by blending heuristics. In *ICLR 2024*, May 2023. URL https://www.microsoft.com/en-us/research/publication/improving-offline-rl-by-blending-heuristics/.

Kaiwen Gui, Teague Tomesh, Pranav Gokhale, Yunong Shi, Frederic T. Chong, Margaret Martonosi, and Martin Suchara. Term grouping and travelling salesperson for digital quantum simulation. *arXiv preprint*, arXiv:2001.05983, 2020. Preprint, 16 Jan 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Naomichi Hatano and Masuo Suzuki. Finding exponential product formulas of higher orders. *Lecture Notes in Physics*, 679:37–68, 2005. doi: 10.1007/11339117_2.

Markus Heyl, Philipp Hauke, and Peter Zoller. Quantum localization bounds trotter errors in digital quantum simulation. *Science Advances*, 5(10):eaau8342, 2019. doi: 10.1126/sciadv.aau8342. URL https://arxiv.org/abs/1806.11123.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019. URL https://arxiv.org/abs/1912.12180.

John Hubbard. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A*, 276(1365):238–257, 1963. doi: 10.1098/rspa.1963.0204.

Marc Höftmann, Jan Robine, and Stefan Harmeling. Backward learning for goal-conditioned policies. *arXiv preprint arXiv:2312.05044v1*, 2023. URL https://arxiv.org/abs/2312.05044v1.

Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Algorithmic theory of qubit routing, 2023. URL https://arxiv.org/abs/2305.02059.

Max Jaderberg, Volodymyr Mnih, Wojciech M. Czarnecki, et al. Reinforcement learning with unsupervised auxiliary tasks. In *Advances in Neural Information Processing Systems*, 2016.

Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jonathan 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. URL https://arxiv.org/abs/2405.08810.

Adam Jelley, Trevor McInroe, Sam Devlin, and Amos Storkey. Efficient offline reinforcement learning: The critic is critical, 2024. URL https://arxiv.org/abs/2406.13376.

Zhang Jiang, Kevin J. Sung, Kostyantyn Kechedzhi, Vadim N. Smelyanskiy, and Sergio Boixo. Quantum algorithms to simulate many-body physics of correlated fermions. *Physical Review Applied*, 9:044036, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by clifford and t circuits. *Physical Review Letters*, 110:190502, 2013. doi: 10.1103/PhysRevLett.110.190502.

M. Kolle, T. Schubert, P. Altmann, M. Zorn, J. Stein, and C. Linnhoff-Popien. A reinforcement learning environment for directed quantum circuit synthesis. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART)*. SCITEPRESS, 2024. URL https://www.scitepress.org/Papers/2024/123832/123832.pdf.

Michael Kölle, Tom Schubert, Philipp Altmann, Maximilian Zorn, Jonas Stein, and Claudia Linnhoff-Popien. A reinforcement learning environment for directed quantum circuit synthesis. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024)*. SCITEPRESS, 2024. doi: 10.5220/0012383200003636.

David Kremer, Ali Javadi-Abhari, and Priyanka Mukhopadhyay. Optimizing the non-clifford-count in unitary synthesis using reinforcement learning, 2025. URL https://arxiv.org/abs/2509.21709.

Alon Kukliansky, Ed Younis, Lukasz Cincio, and Costin Iancu. QFactor: A domain-specific optimizer for quantum circuit instantiation. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 814–824, 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Christopher Lane, Aron J. Cohen, Michael J. Wieduwilt, Sandro Sorella, and George H. Booth. Assessing the accuracy of scan and its use in the hubbard u correction for transition metal oxides. *Physical Review B*, 98(125140), 2018. doi: 10.1103/PhysRevB.98.125140.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5639–5650, 2020.

Gushu Li, Anbang Wu, Yunong Shi, Ali Javadi-Abhari, Yufei Ding, and Yuan Xie. Paulihedral: A generalized block-wise compiler optimization framework for quantum simulation kernels, 2021. URL https://arxiv.org/abs/2109.03371.

Ji Liu, Alvin Gonzales, Benchen Huang, Zain H. Saleem, and Paul Hovland. Quclear: Clifford extraction and absorption for significant reduction in quantum circuit size. *arXiv preprint*, 2024. Preprint, 23 Aug 2024.

Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996. doi: 10.1126/science.273.5278.1073.

Thibaud Louvet, Thomas Ayral, and Xavier Waintal. On the feasibility of performing quantum chemistry calculations on quantum computers. *arXiv preprint arXiv:2306.02620v3*, 2023. URL https://arxiv.org/abs/2306.02620. Last revised 2024-10-03.

Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020. doi: 10.1103/RevModPhys.92.015003.

J. R. McClean et al. Openfermion: The electronic structure package for quantum computers, 2017.

Jarrod R. McClean, Nicholas C. Rubin, Kevin J. Sung, Ian D. Kivlichan, et al. OpenFermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 5(3): 034014, 2020.

Kazuma Nakamura, Yoshihide Yoshimoto, Taichi Kosugi, Ryotaro Arita, and Masatoshi Imada. Ab initio derivation of low-energy model for κ-(et)$_2x$. *arXiv e-prints*, 2009.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287, 1999.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge, 2010. ISBN 9781107002173.

J. Nogúe, N. Klingler, A. Martinez, J. van de Wetering, and A. Kissinger. Reinforcement learning based quantum circuit optimization via zx-calculus. *Quantum*, 9:1758, 2025. doi: 10.22331/q-2025-05-28-1758. URL https://doi.org/10.22331/q-2025-05-28-1758.

Jennifer Paykin, Albert T. Schmitz, Mohannad Ibrahim, Xin-Chuan Wu, and A. Y. Matsuura. Pcoast: A pauli-based quantum circuit optimization framework. *arXiv preprint*, arXiv:2305.10966, 2023. Preprint, 16 May 2023.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5:4213, 2014. doi: 10.1038/ncomms5213.

David Poulin, Matthew B. Hastings, Dave Wecker, Nathan Wiebe, Andrew C. Doherty, and Matthias Troyer. The trotter step size required for accurate quantum chemistry. *Quantum Information & Computation*, 15(5-6):361–384, 2015. URL https://arxiv.org/abs/1406.4920.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018. doi: 10.22331/q-2018-08-06-79.

Qiskit Development Team. Qiskit: An open-source framework for quantum computing. https://qiskit.org/, 2024. Version 1.3.2.

Sebastian Rietsch, Abhishek Y. Dubey, Christian Ufrecht, Maniraman Periyasamy, Axel Plinge, Christopher Mutschler, and Daniel D. Scherer. Unitary synthesis of clifford+t circuits with reinforcement learning. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 824–835. IEEE, September 2024. doi: 10.1109/qce60285.2024.00102. URL http://dx.doi.org/10.1109/QCE60285.2024.00102.

Neil J. Ross and Peter Selinger. Optimal ancilla-free clifford+t approximation of z-rotations, 2014. URL https://arxiv.org/abs/1403.2975.

Francisco J. R. Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco J. H. Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, Alhussein Fawzi, Konstantinos Meichanetzidis, and Pushmeet Kohli. Quantum circuit optimization with alphatensor, 2024. URL https://arxiv.org/abs/2402.14396.

Albert T. Schmitz, Nicolas P. D. Sawaya, Sonika Johri, and A. Y. Matsuura. Graph optimization perspective for low-depth trotter-suzuki decomposition. *Physical Review A*, 109(4):042418, 2024. doi: 10.1103/PhysRevA.109.042418. Received 26 May 2023; Accepted 13 March 2024; Published 18 April 2024.

Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t|ket⟩: A retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003, 2020. doi: 10.1088/2058-9565/ab8e92.

Wei Tang, Yiheng Duan, Yaroslav Kharkov, Rasool Fakoor, Eric Kessler, and Yunong Shi. Alpharouter: Quantum circuit routing with reinforcement learning and tree search, 2024. URL https://arxiv.org/abs/2410.05115.

John van de Wetering and Matthew Amy. Optimising quantum circuits is generally hard, 2024. URL https://arxiv.org/abs/2310.05958.

Ewout van den Berg and Kristan Temme. Circuit optimization of hamiltonian simulation by simultaneous diagonalization of pauli clusters. *Quantum*, 4:322, September 2020. ISSN 2521-327X. doi: 10.22331/q-2020-09-12-322. URL http://dx.doi.org/10.22331/q-2020-09-12-322.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *arXiv preprint arXiv:2110.00188*, 2021. URL https://arxiv.org/abs/2110.00188.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

Changhao Yi and Elizabeth Crosson. Spectral analysis of product formulas for quantum simulation. *npj Quantum Information*, 8(57), 2022. doi: 10.1038/s41534-022-00548-w. URL https://www.nature.com/articles/s41534-022-00548-w.

Yuan-Hang Zhang, Pei-Lin Zheng, Yi Zhang, and Dong-Ling Deng. Topological quantum compiling with reinforcement learning. *Physical Review Letters*, 125(17):170501, 2020. doi: 10.1103/PhysRevLett.125.170501. URL https://doi.org/10.1103/PhysRevLett.125.170501.

Weihong Zheng, Rajiv R. P. Singh, and Jaan Oitmaa. Series studies of the hubbard model. *Physical Review B*, 71(184440), 2005. doi: 10.1103/PhysRevB.71.184440.

A Pauli Strings and Trotterization

This appendix collects the quantum-simulation background that the main text assumes.

In an n-qubit system, a *Pauli string* is a length-n tensor product of operators from a set of unitary operations $\{X, Y, Z, I\}$. Each factor acts on a different qubit and rotates along the X, Y or Z axis. An example of a Pauli String would be the tensor product X_1X_2 which has an X rotation on qubit 1 and qubit 2 simultaneously. Pauli strings arise naturally in quantum Hamiltonians and form a convenient basis for both analysis and implementation. The time evolution generated by a single Pauli string, e^{-iPt}, can be synthesized exactly using Pauli gates, CNOTs, and a single-qubit Z-rotation Nielsen & Chuang (2010).

Pauli strings form a basis for the real vector space of Hermitian operators on n qubits. Consequently, any Hamiltonian can be written as $H = \sum_i w_i P_i,\ w_i \in \mathbb{R}$, where P_i are Pauli strings Nielsen & Chuang (2010). For notational simplicity, we absorb coefficients into the operators and write $H = \sum_i H_i$. While exponentiating a single term e^{-iH_it} is straightforward, synthesizing the exponential of a sum, $\exp\big(it\sum_i H_i\big)$, is substantially more challenging. Closed-form circuit decompositions are

generally unavailable, motivating approximation techniques based on product formulas. A standard approach is *Trotterization*, based on the Lie–Trotter formula Hatano & Suzuki (2005):

$$e^{\,t\sum_{k=1}^{M} H_k} \;\approx\; \left(\prod_{k=1}^{M} e^{\,\frac{t}{r}H_k}\right)^{r}, \tag{4}$$

where t is the total evolution time and r is the number of Trotter steps. The approximation error depends on commutators $[H_a, H_b]$ and decreases linearly with r. (We reserve N for the number of fermionic modes throughout; see Section 2.1.) Optimizing the circuit structure of these product formulas is the central focus of this work. For a comprehensive treatment of quantum computing fundamentals, we refer readers to Nielsen & Chuang (2010).

B Free Fermionic Systems

Free-fermionic (quadratic) Hamiltonians form an important class of quantum systems that admit efficient classical simulation; see Bravyi (2005) for details. A fermionic mode may be either empty or occupied by a single fermion. Creation and annihilation operators $(c_j^\dagger, c_j)$ satisfy the canonical anti-commutation relations $\{c_j, c_k^\dagger\} = \delta_{jk}$ and $\{c_j, c_k\} = 0$.

A general quadratic fermionic Hamiltonian takes the form

$$H = \sum_{j,k} A_{jk}\, c_j^\dagger c_k + \tfrac{1}{2}\big(B_{jk}\, c_j^\dagger c_k^\dagger + B_{jk}^*\, c_j c_k\big). \tag{5}$$

The Heisenberg evolution of an operator $O(t)$ obeys

$$\frac{dO(t)}{dt} = i[H, O(t)]. \tag{6}$$

For creation and annihilation operators, this equation closes linearly. Defining

$$\mathbf{c}(t) = \big(c_1(t), \ldots, c_{n_f}(t), c_1^\dagger(t), \ldots, c_{n_f}^\dagger(t)\big)^\top,$$

the dynamics can be written as

$$\frac{d\mathbf{c}(t)}{dt} = K\,\mathbf{c}(t), \tag{7}$$

where K is a $2n_f \times 2n_f$ matrix determined by (A, B). (Here n_f is the number of fermionic modes, written N in the main text; the two are the same quantity.) The solution is

$$\mathbf{c}(t) = e^{Kt}\,\mathbf{c}(0) = U^\dagger(t)\,\mathbf{c}(0)\,U(t), \tag{8}$$

which admits the Bogoliubov–de Gennes form

$$\mathbf{c}(t) = \begin{pmatrix} U_t & V_t \\ V_t^* & U_t^* \end{pmatrix} \mathbf{c}(0), \tag{9}$$

with $U_t, V_t \in \mathbb{C}^{n_f \times n_f}$ satisfying canonical fermionic constraints Bravyi (2005).

Each operator evolves as a linear combination

$$c_j(t) = \sum_k \big(U_{t,jk}\, c_k + V_{t,jk}\, c_k^\dagger\big). \tag{10}$$

As a result, free-fermionic dynamics are fully specified by polynomial-size matrices (U_t, V_t) rather than exponentially large state vectors, and can be simulated classically using $O(n^2)$ parameters.

Mapping to qubits. To implement free-fermionic dynamics on quantum hardware, one applies a fermion–qubit mapping such as the Jordan–Wigner transformation,

$$c_j = \Big(\prod_{k<j} Z_k\Big)\frac{X_j - iY_j}{2},\; c_j^\dagger = \Big(\prod_{k<j} Z_k\Big)\frac{X_j + iY_j}{2}. \tag{11}$$

This mapping converts quadratic fermionic terms into Pauli strings acting on qubits while preserving fermionic statistics. Consequently, free-fermionic subsystems admit compact circuit representations and can be isolated as classically tractable components within a larger quantum Hamiltonian.

C HAMILTONIAN PARTITIONING AND COMPILATION

Our compilation pipeline begins with a Hamiltonian specified in second-quantized form, as is standard in quantum chemistry and many-body physics. The input Hamiltonian may be represented either directly in terms of fermionic creation and annihilation operators or as a sum of Pauli strings acting on qubits. In both cases, the goal is to transform this abstract description of quantum dynamics into an efficient quantum circuit suitable for execution on hardware. The algorithm is shown in Algorithm 1.

When the Hamiltonian is provided in a Pauli-string representation, we first apply an inverse fermion-to-qubit mapping to recover an equivalent fermionic description. Concretely, we reverse the Jordan–Wigner transformation to map Pauli operators back into fermionic operators. This step allows us to reason about the Hamiltonian in terms of fermionic modes, where structural properties such as quadratic (free-fermionic) versus interacting terms are explicit and easy to identify.

Once expressed in fermionic form, the Hamiltonian is partitioned according to operator degree. Terms that are quadratic in fermionic operators correspond to free-fermionic dynamics and admit a compact polynomial-size representation. In contrast, higher-order terms (such as quartic interactions) represent genuinely interacting fermionic processes that fall outside the free-fermionic regime. We isolate the quadratic component of the Hamiltonian by grouping terms based on the weight of their fermionic operators, separating those that are bilinear from those that involve higher-order products.

The free-fermionic portion of the Hamiltonian is then passed to our learning-based compiler, which is specifically designed to exploit the algebraic structure and classical simulability of quadratic fermionic systems. This component is optimized using our reinforcement-learning framework to produce shallow, gate-efficient circuit implementations. All remaining not compiled or efficiently solvable by our model are compiled using a standard baseline compiler. In our implementation, we rely on Qiskit rustiq de Brugière & Martiel (2024) to synthesize these interacting terms using state-of-the-art heuristic optimization passes.

Algorithm 1 Hybrid Hamiltonian Partitioning and Compilation

Require: Hamiltonian H (second-quantized fermionic form or Pauli-string form), evolution time step Δt
Require: Learning-based free-fermion compiler $\mathcal{C}_{\text{F2}}$, baseline compiler $\mathcal{C}_{\text{base}}$ (e.g., Qiskit Rustiq)
Ensure: Quantum circuit C implementing one compiled Trotter step for $e^{-iH\Delta t}$ (up to tolerance)

/* Normalize representation */
if H is given as Pauli strings **then**
  $H_f \leftarrow \textsc{InverseJordanWigner}(H)$ ▷ Map Pauli-string Hamiltonian back to fermionic operators
else
  $H_f \leftarrow H$
end if
/* Partition by fermionic operator degree */
$H_{\text{quad}} \leftarrow 0$, $H_{\text{int}} \leftarrow 0$
for all terms h in H_f **do**
  $d \leftarrow \textsc{FermionicDegree}(h)$ ▷ # of creation/annihilation operators in h
  if $d = 2$ **then**
    $H_{\text{quad}} \leftarrow H_{\text{quad}} + h$ ▷ Quadratic (free-fermionic) term
  else
    $H_{\text{int}} \leftarrow H_{\text{int}} + h$ ▷ Interacting / higher-order term (e.g., quartic)
  end if
end for
/* Compile each partition */
$C_{\text{quad}} \leftarrow \mathcal{C}_{\text{F2}}(H_{\text{quad}}, \Delta t)$ ▷ RL-based compilation exploiting free-fermionic structure
$C_{\text{int}} \leftarrow \mathcal{C}_{\text{base}}(H_{\text{int}}, \Delta t)$ ▷ Baseline synthesis/optimization for non-free terms
/* Compose into a single Trotter step */
$C \leftarrow \textsc{ComposeCircuits}(C_{\text{quad}}, C_{\text{int}})$ ▷ Concatenate or interleave per chosen product formula ordering
return C

Finally, the circuits produced by the two compilation pathways are composed to form a complete Trotter step for the full Hamiltonian evolution. This hybrid strategy allows us to aggressively optimize the classically tractable subroutines while maintaining correctness and generality for the interacting

parts of the algorithm. The result is an end-to-end compilation pipeline that leverages learning where structure permits and falls back on robust classical compilation where it does not, as described in the main paper

Windowing for larger systems. The network has a fixed input size of $N = 12$ modes: this fixes the 24×24 state input and its axial position embeddings, the site vocabulary, and the size of the action output. Larger systems are handled by cutting them into non-overlapping 12-mode windows and compiling each window separately. Concretely, the modes are relabelled along a spanning chain of the coupling graph and the chain is cut into windows $[12k, 12(k+1))$. A free-fermionic term is assigned to a window only if all of its modes lie in that window. Terms that cross a window boundary, together with all non-free-fermionic terms, are collected into a residual that is compiled by Rustiq at optimization level 0; they are not compiled by F2. Each window is shifted to start at qubit 0, compiled by F2, and mapped back to its global qubits; a window with fewer than 12 modes is padded with the identity up to 24 dimensions. The window circuits and the residual circuit are then concatenated, and the whole block is repeated once per Trotter step. This chunking is across qubits; it does not split a single window's move sequence over time.

D Architecture Details

We design an architecture that jointly encodes the *quantum state*—represented as the product of unitaries already applied—and the *action sequence* that produced it. Formally F2 is a dual-tower transformer composed of an *axial-attention* Ho et al. (2019) unitary encoder and a *standard transformer* sequence encoder, fused into a lightweight value head. The two-tower pattern is inspired from AlphaTensor-style designs that pair structured tensor encoders with sequence models Ruiz et al. (2024), while axial attention provides an efficient factorization of attention over multi-axis arrays Ho et al. (2019). Figure 3 illustrates the dual-tower architecture and the compositional action embeddings used by F2. A formal definition of the compositional action embedding and additional ablation results are provided in Appendix E.

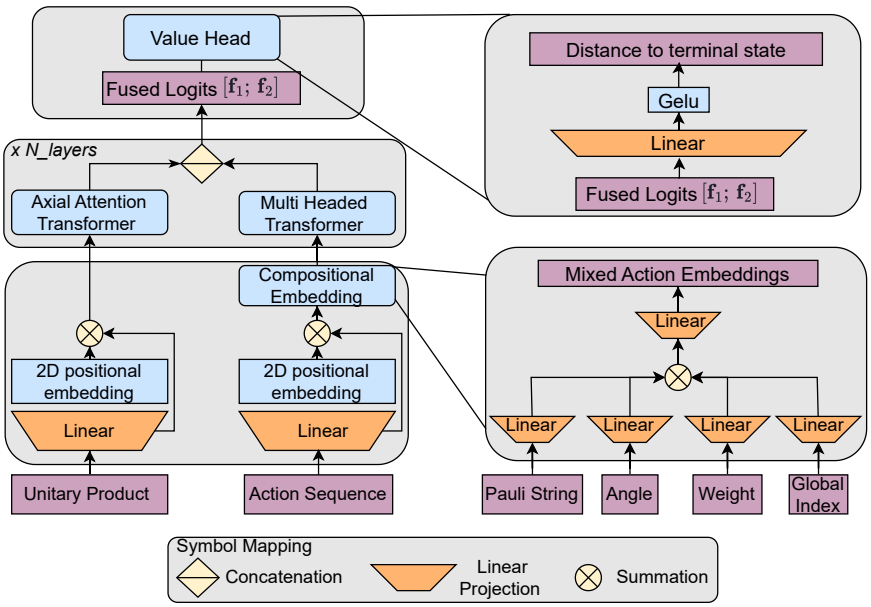


Figure 3: Our two-tower neural network architecture using custom compositional embeddings. The compositional embeddings factorize each feature of the action into a custom embedding and then mix these features together for the final embedding.

Core dimensions. The shared model width is $d_{\text{model}} = 128$. Both the sequence and axial towers use 4 layers each. Each tower uses 4 attention heads, and dropout is set to 0.01. The fused representation dimension is 256, corresponding to $2 \times d_{\text{model}}$.

Representations and encoders. We use the Majorana (free-fermionic) representation for the residual unitary state. For the unitary encoder, the axial-attention tower operates over a structured matrix representation whose positional indexing is parameterized by the number of qubits. In our

reported configuration, the number of qubits is set to 12, so the learned row and column positional embeddings each have size $2 \cdot \text{num_qubits} = 24$.

The sequence tower encodes the action history using a Transformer encoder with batch-first inputs and pre-normalization. The internal feed-forward dimension and activation function follow the PyTorch defaults. Action embeddings are normalized using a square-root scaling.

Sequence length. The maximum action sequence length is set to 100. We append an explicit end-of-sequence (EOS) token, yielding a positional encoding length of 101. The EOS token is excluded from the pooling operation.

Hyperparameters. See Table 3 below.

Component	Value
Hidden size (d_{model})	128
Fused input dimension	256
Sequence encoder layers	4
Sequence attention heads	4
Dropout	0.01
Axial-attention layers	4
Axial-attention heads	4
Axial MLP expansion	4
Normalization	LayerNorm
Maximum sequence length	100
End-of-sequence token	enabled
EOS included in pooling	disabled
Representation	Majorana
Number of qubits	12
Unitary positional embedding size	$2 \times \texttt{num_qubits} = 24$

Table 3: Resolved model hyperparameters used in all experiments.

E Compositional Embedding

Naïve one-hot action embeddings discard rich structure shared across operations (e.g., rotation axis, target qubits, angle magnitude). We instead use a novel *compositional* representation that factorizes each action into semantically meaningful components which was co-designed with the environment. Concretely, write an action as

$$a \;=\; (P, \theta), \qquad P \in \{I, X, Y, Z\}^{\otimes n}$$

where P is a Pauli string identifying which operator acts on which qubit and θ is the rotation parameter (discretized during training as in our environment). We embed each component separately—(i) Pauli string, (ii) Angle of rotation, (iii) the total length of the Pauli string, and (iv) the global index of the action for auxiliary features.

Formally, the embedding can be expressed through the following function:

$$\phi(a_i) = \text{TypeEmb}(\text{type}_i) + \text{AngleEmb}(\alpha_i) \tag{12}$$

$$+ \text{WeightEmb}(|a_i|) + \text{GlobalEmb}(\text{idx}_i) \;\in\; \mathbb{R}^d. \tag{13}$$

Here, d is the embedding vector dimension and all of the embedding functions are learned during training. This yields an embedding space in which actions that differ only by, say, angle or a single target index lie close to one another, providing the inductive bias needed to efficiently learn over large move sets with many overlapping features.

Ablation. Here we show the efficacy of our custom action embedding compared to a naive embedding architecture over the action space.

Our experiment compares loss curves for predicting distance to terminal state based on Monte Carlo based labels and comparisons were done for varying sized action sets and varying feature space dimensions. Results of this experiment are captured in Figure 4.

What is noticed is that as our action space is increasing, there is more learning efficiency observed our proposed architecture. This suggests a favorable scaling behavior of our neural network to handle even larger unitaries, as gains are already seen at a small scale.

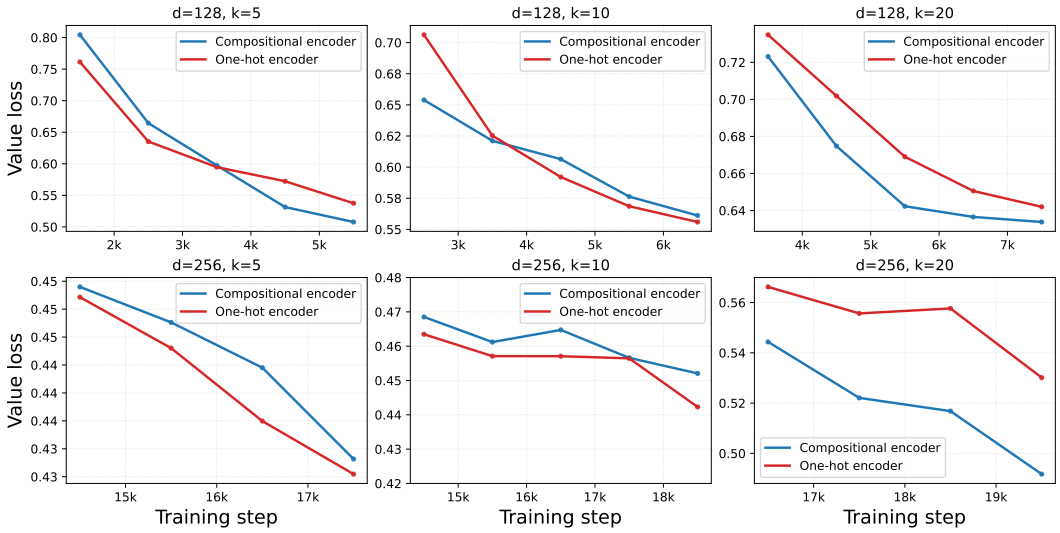


Figure 4: Comparing the learning efficiency over one epoch on distance to identity regression. Architectures used are naive embeddings for action space vs compositional embeddings.. X axis is on the order of 1000 steps

F Compilation Time

F2 compiles a 12-qubit benchmark in 11–24 s and a 144-qubit benchmark in 141–295 s on one A100, against under 1.3 s for every library baseline on one CPU core: three to four orders of magnitude, unchanged across the size range. Almost all of F2's time (98.9%) is forward passes of the value network. This appendix gives the protocol and the measurements.

F.1 Setup

We time one Trotter step of each Table 1 benchmark under the pipeline of Section 3, for F2 on one A100 and for the three baselines on one CPU core. Times cover synthesis of the circuit; the final basis transpile adds under 0.6 s in every case and is excluded. F2 decodes each window until the normalized chordal distance $\|Q - I\|_F/(2\sqrt{2n})$ to the identity falls to 3×10^{-4} (an infidelity near 10^{-6} on 12 modes), or after 300 moves; it reached the target on every window of every benchmark. Each measurement is a wall-clock reading around the call with GPU work synchronised, after one discarded warm-up run; baselines are repeated five times per benchmark and F2 once or twice, with run-to-run variation under 0.1%. Checkpoint loading is excluded. Hardware: one shared node with an AMD EPYC 9654P and two NVIDIA A100 80 GB; PyTorch 2.5.1, CUDA 12.4, Qiskit 2.3.1, cirq-core 1.5.0, OpenFermion 1.7.0.

F.2 Results

Table 4: Compilation time for one Trotter step of each benchmark, one thread per library, F2 on GPU. Seconds; "Windows" is the number of 12-qubit free-fermion windows in the partition.

Benchmark	Qubits	Windows	F2	Google-Cirq	IBM-base	IBM-rustiq
Fermi–Hubbard 1D	12	1	10.8	0.029	0.012	0.0057
Fermi–Hubbard 2D	12	1	10.8	0.032	0.017	0.0088
Heisenberg 1D	12	1	23.7	0.046	0.013	0.0065
Heisenberg 2D	12	1	23.7	0.038	0.020	0.012
t–J 1D	12	1	10.8	0.039	0.025	0.014
t–J 2D	12	1	10.7	0.046	0.035	0.021
Fermi–Hubbard 1D	72	6	70.4	0.235	0.056	0.061
Heisenberg 1D	72	6	146	0.228	0.062	0.070
t–J 1D	72	6	70.7	0.330	0.150	0.235
Fermi–Hubbard 1D	144	12	141	0.538	0.156	0.291
Heisenberg 1D	144	12	295	0.559	0.152	0.294
t–J 1D	144	12	142	1.26	0.348	0.998

F2's time is linear in the number of windows, about 11.8 s per window for Fermi–Hubbard and t–J and 24 s for Heisenberg, because windows are compiled one after another; one model call takes 0.532 s and a window takes roughly 20–45 of them. Window synthesis is essentially the whole cost: partitioning, the Rustiq remainder, assembly and transpile together stay under one second even at 144 qubits. Batching several windows into one model call does not help: the call time scales linearly with the batch, so a single window already saturates the A100.

G Effect of Training-Set Size

Retraining the full F2 critic on nested quarter- and half-size subsets of the training set, at a fixed gradient-step budget, lowers median approximation error by a factor of 430 across the range and shows no plateau at the full 10,000 trajectories. Training loss barely distinguishes the three critics; compilation quality does. This appendix gives the curve, the exact statistics, and enough detail to rerun it.

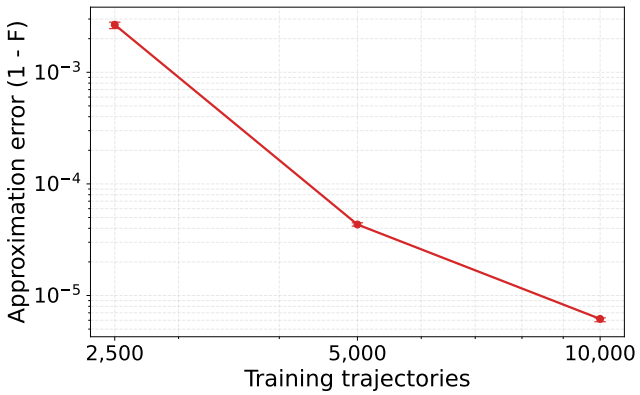


Figure 5: Approximation error on the 100 held-out unitaries against training-set size. Each critic uses the full F2 recipe (shaped labels and the learned value-loss regularizer) and the same gradient-step budget: 24, 12 and 6 epochs over 2,500, 5,000 and 10,000 trajectories respectively, so the points differ in data seen, not in training compute. The median is plotted because the error distributions are long-tailed. Error bars are 95% percentile bootstrap confidence intervals for the median, from 10,000 resamples of the 100 per-target errors; they are asymmetric by construction.

G.1 Setup

Three critics, all trained with the published F2 recipe: shaped labels (Section 2.2) and the learned value-loss regularizer of equation 3 with $\lambda_{\text{al}} = 0.01$, Adam, batch size 32, learning rate 2×10^{-4}, weight decay 10^{-4}. They differ only in how many trajectories they see.

Critic	Trajectories	Frames	Epochs
Quarter	2,500	377,126	24
Half	5,000	750,972	12
Full (published)	10,000	1,502,443	6

Step budget. The epoch counts are chosen so that every critic receives the same number of parameter updates as the published full-data run (six epochs over 10,000 trajectories). Holding epochs fixed instead would give the larger sets proportionally more training, confounding data diversity with compute. The trade-off is that the quarter critic revisits each trajectory four times as often as the full critic, so part of its gap may be memorisation rather than coverage; the two are not separable at a single seed.

Nesting. All three sets are generated from the same seed, and the quarter set is the first 2,500 trajectories of the half set, which is the first 5,000 of the full set (verified on the stored trajectories). Each point therefore differs from the next only by added data. All three carry the shaped labels of Section 2.2 (residual against $-(T - t)$ has slope -2 on every file).

Training metrics. The learned regularizer scalars landed near $c_0 \approx 47$, $c_1 \approx -48$ on both smaller runs against 42.8, -45.5 on the full run. Final value losses were 23.94 (quarter) and 23.73 (half), with target correlations 0.916 and 0.919: the three critics are nearly indistinguishable on training metrics.

G.2 Evaluation

Identical to Figure 2b: 100 targets drawn with seeds $10 + i$, $i = 0, \ldots, 99$, as $\exp(H)$ at $\tau = 1$ for random 1D Heisenberg or Fermi–Hubbard models with couplings in ± 0.02, projected to $SO(24)$; greedy decoding with the published stopping rule and a 100-step limit; the reported quantity is the best process fidelity F_{pro} reached at any decoding step (Section 3). The full-data point reuses the published critic's evaluation; the same script, targets and settings produced all three. Per-target error values are stored, not only summaries.

Table 5: Approximation error $1 - F_{\text{pro}}$ over the 100 held-out targets, per critic. The 10,000-trajectory row is the same checkpoint and the same evaluation as the "MC Geometric Critic" of Figure 2b.

Trajectories	**Median**	**Mean**	**Std. dev.**
2,500	2.67×10^{-3}	2.97×10^{-3}	1.57×10^{-3}
5,000	4.33×10^{-5}	3.10×10^{-3}	8.37×10^{-3}
10,000	6.17×10^{-6}	5.51×10^{-4}	2.39×10^{-3}

Error bars. The bars in Figure 5 are 95% percentile bootstrap confidence intervals for the median, from $10{,}000$ resamples of the 100 per-target errors; they are asymmetric by construction.

G.3 Results

The median falls monotonically by a factor of 430 and has not flattened at the largest size. The mean does not track it: it rises slightly from 2,500 to 5,000 before falling, because the half-data critic splits the targets, compiling most of them two orders of magnitude better than the quarter critic while failing a few outright, which inflates both its spread and its mean. The full critic recovers the typical case and thins the tail. Medians are reported for this reason.

The three critics' training losses agree to within about one percent while their median compilation error differs by two and a half orders of magnitude. Training loss on shaped labels is a poor proxy for compilation quality; coverage of the unitary space is what the downstream metric responds to.