# Extended Abstract

**Motivation** Quantum computers today are still incredibly noisy and often need to perform in environments which are not perfectly pure, so a circuit that is ideally and mathematically correct can perform quite poorly once the gates that comprise that circuit are exposed to real hardware errors. This project studies one specific formulation of the noise-aware compilation problem: given a target quantum operation, can a deep RL agent construct a short circuit (consisting of elementary quantum gates) that has high fidelity (which we will define more clearly in the paper) under some simulated noise? Instead of treating compilation as only an exact search problem that we can mathematically solve, we formulate it as a sequential MDP where every extra gate has a cost and the final score is measured after the noisy simulation.

**Method** We built a Gymnasium environment around IBM's open-source platform Qiskit Aer. The state contains the current ideal (noiseless) unitary (a special type of matrix that represents a qunatum operation), the desired target unitary, and the fraction of the maximum depth already used. The action space is a set of elementary gates containing single-qubit Clifford and rotation gates, two-qubit CNOTs, and a STOP action. When we start an episode, we initially have just the identity matrix as our circuit, and then at each step the agent appends a new gate until it STOPs or reaches its depth limit. Then, we compute reward

$$R = F_{\text{avg}}(C, U_{\text{target}}) - \lambda \cdot \text{depth},$$

where $F_{\text{avg}}$ is the average noisy state fidelity. This reward provides a balanced trade-off between correctness and circuit length (longer circuit length might mean you can fit more complex circuits but at the expense of being far more noise-prone). We trained a custom proximal policy optimization (PPO) agent with an MLP as our actor-critic network, generalized advantage estimation, entropy regularization, and clipped policy updates.

**Implementation** Our main engineering contribution was making the environment work well for the learning process. Computing fidelity naively would require us to simulate every possible input, which made training too slow. Instead, we batched together all of the basis-state circuits (which consist only of NOT gates) into one Aer simulation call as a density-matrix simulation which allows us to fully parallelize this. We also implemented random and Qiskit transpiler baselines, evaluation scripts, learning-curve plots, and noise-robustness plots. The experiments focused on Bell-state preparation, SWAP and a 2-qubit Quantum Fourier Transform (QFT) as two-qubit setups, and then we tested scalability on GHZ preparation as a three-qubit setup.

**Results** On the Bell target under the medium combined noise model, PPO learned the standard textbook two-gate solution and reached an average fidelity $0.9585$, matching the Qiskit baseline and clearly beating the random baseline, which averaged $0.2491$. This is the cleanest success case: the agent learned both to apply the right entangling structure and to stop at the right time. On the SWAP target, however, PPO learned to stop immediately, which produced a fidelity $0.4938$. That was better than the random average of $0.2462$ but does not match the expected construction of this circuit, which involves three CNOT gates. The GHZ training run showed unstable progress, ending with a training fidelity of $0.7645$, which shows that PPO can find useful partial structures but struggles to solve the complete construction problem as we scale up the number of qubits and the action space.

**Discussion** The main takeaway from this work is that a deep RL approach can work to build up small noise-aware circuits when the target circuit is relatively small and the reward system isn't too unpredictable. Bell-state preparation is a good example: $\pi_\theta$ is fairly compact, and PPO can build this circuit reliably. The harder examples also showed the limitations of on-policy PPO with sparse terminal rewards - e.g., learning an immediate STOP action for SWAP. For GHZ, the reward is delayed across more gates and a larger action space.

**Conclusion** Overall, in this project, we've built a working end-to-end noise-aware quantum circuit simulation, trained a PPO training pipeline and showed that this performs comparably to standard approaches today. We've also found natural next steps to follow up on this work, such as designing a better, denser reward function, and forcing more exploration before committing to trajectories.

# Reinforcement Learning for Noise-Aware Quantum Circuit Compilation

**Abhishek Shah**
Department of Computer Science
Stanford University
abhishek.shah@stanford.edu

**Vinav Shah**
Department of Computer Science
Stanford University
vinav@stanford.edu

## Abstract

Constructing quantum circuits from a set of elementary gates is a fundamental operation in quantum computing, yet there are no robust methods to achieve this under real-world noise constraints. In this project, we aim to use RL to construct a short circuit that implements a target unitary with high fidelity under a Qiskit Aer noise model. We treat this problem as a finite-horizon MDP where each action either appends another gate or stops, the state contains the current and target unitaries, and the terminal reward is average fidelity with a subtracted penalty based on the depth of the circuit. We implement a custom PPO agent and compare it against random search and Qiskit's transpiler on small targets. PPO successfully learns Bell-state preparation, reaching fidelity $0.9585$ and matching the standard compiler baseline. However, on SWAP it converges to the shallow STOP policy, and on GHZ training is unstable. These results show both the promise and the brittleness of simple policy-gradient methods for quantum compilation with sparse terminal rewards.

## 1 Introduction

Quantum circuit compilation is the problem of translating a desired quantum operation into a sequence of gates that can actually be run on hardware. In the ideal version of the problem, we mainly care whether the circuit implements the same unitary as the target. On real devices, however, that is not enough. Gates are noisy, two-qubit operations are especially expensive, and longer circuits usually lose more information before they finish. A compiler that ignores noise will usually return a circuit that is mathematically correct but performs quite poorly empirically.

This project asks whether reinforcement learning can be used as a noise-aware compiler for small quantum circuits. In practice, this involves an agent that starts with an empty circuit and repeatedly chooses a gate to append. At the end, the circuit is evaluated under a noisy simulator. Good circuits get high fidelity and short circuits avoid a depth penalty, which lets the agent optimize the actual quantity we care about (performance under noise), instead of only reproducing an ideal gate sequence.

There are two reasons this is a natural RL problem. Firstly, circuit construction can be modeled as a sequential MDP. A gate may look useless by itself but become important after later gates are added, especially for entangling operations (which is a quantum process that ties together the state of two independent qubits). Also, exact search becomes expensive quickly because even a small set of gates to choose from creates an exponentially large number of possibilities. RL gives us a way to learn a policy sequentially over half-complete circuits rather than exhaustively enumerate all possibilities.

At the same time, this is not an easy RL problem. The reward is mostly terminal, so the agent may take many zero-reward actions before learning whether the circuit was useful. The action space includes many gates that are irrelevant for a given target. Finally, the reward includes a noisy simulation, which is much slower than a normal toy RL environment. A major part of the project was not just

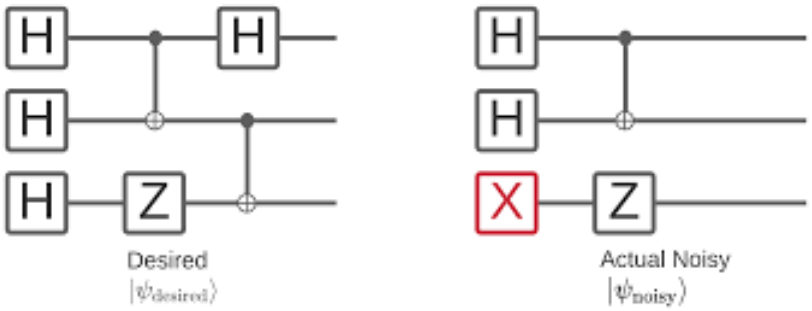


Figure 1: The above image presents a concrete example of a theoretically desired quantum circuit and the noisy circuit we see in practice. Suppose in both cases that our state starts at $|000\rangle$. In the desired circuit, a Hadamard (H) gate is applied to all 3 qubits simultaneously to create a perfect superposition of all 8 possible binary states from $|000\rangle$ to $|111\rangle$. Then, a Z gate is applied to the third qubit to change its phase, and then two CNOTs are applied: first on the middle qubit controlled by the top, and then on the bottom qubit controlled by the middle. Finally, we have another Hadamard gate on the first qubit to cause some interference that achieves our final state of $\psi_{\text{desired}} = \frac{1}{2}(|000\rangle - |011\rangle + |110\rangle + |101\rangle)$. However, since the $H$ gate is internally a rotation in the Hilbert space where these superimposed qubit states live (known as the "Bloch sphere"), a slight perturbation from hardware noise or impurities in the environment can lead to a different type of rotation gate being applied, as we see in the noisy state where the Hadamard on the third qubit is replaced by an $X$ gate. As a result, instead of an equal superposition of all 8 states, after the first set of gates, we only see a superposition of 4 states, and in fact, tracking these "hardware mutations" over the course of the circuit, our final state ends up being $\psi_{\text{noisy}} = [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)] \otimes [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)] \otimes (-|1\rangle)$. Instead of our perfectly entangled desired state, we see *no entanglement* at all once noise is applied, and so our circuit will not yield the desired results.

implementing PPO, but building an environment and evaluation pipeline where training was fast enough to run a large number of experiments.

Our results are mixed but we can derive many important insights from them. PPO learns the Bell-state preparation circuit under combined depolarizing and amplitude damping noise, matching the Qiskit baseline and far outperforming random search. On the 2-qubit QFT (Quantum Fourier Transform), PPO far outperforms our baselines using fewer qubits on every noise model we tested on. However, PPO does not learn the SWAP circuit in our setup; it discovers that stopping immediately is a decent local solution (since this minimizes the depth penalty, and fidelity isn't horrible since the noise model can perform a random SWAP). A three-qubit GHZ run shows some learning but is not fully reliable. These outcomes are useful because they show where the simple formulation works and where better exploration, reward shaping, or model-based methods are probably needed.

## 2 Related Work

Standard quantum compilers such as Qiskit use rule-based decomposition, transpilation passes, routing, and gate cancellation to map circuits into a hardware-supported basis. This approach is very effective and is the default for many circuits. However, it is usually built around deterministic optimization passes rather than learning a policy from noisy execution feedback. Our Qiskit baseline represents this traditional compiler style.

Noise-aware compilation has become more important in the NISQ era, where hardware errors are large enough that small circuit differences can matter. Preskill's discussion of NISQ devices emphasizes that useful near-term quantum computation has to account for imperfect gates and finite coherence times (2). Other work has studied approximate synthesis, hardware-aware mapping, and circuit optimization under physical constraints. Our project is much smaller in scope, but it follows the same motivation: the compiler should care about performance after noise, not only ideal equivalence.

Reinforcement learning has also been applied to program synthesis and combinatorial optimization, where agents build structured objects step by step. Quantum circuit synthesis fits that pattern because a circuit is a sequence of discrete choices with delayed payoff. PPO is a common policy-gradient algorithm because it is relatively stable and easy to implement, and improves upon many of the shortcomings of vanilla policy gradients (3).

Our primary novel contribution in this work is to model this problem as a sequential MDP, which led us to produce an end-to-end implementation and analysis of a compact noise-aware circuit synthesis setting: a Gymnasium environment, Qiskit Aer noisy fidelity rewards, a PPO training loop, and comparisons against random search and Qiskit. On the circuits where PPO did not perform optimally, we also analyzed the modes in which it fails, which is important because small quantum compilation tasks can look solved until the agent is tested on a target like SWAP.

## 3 Method

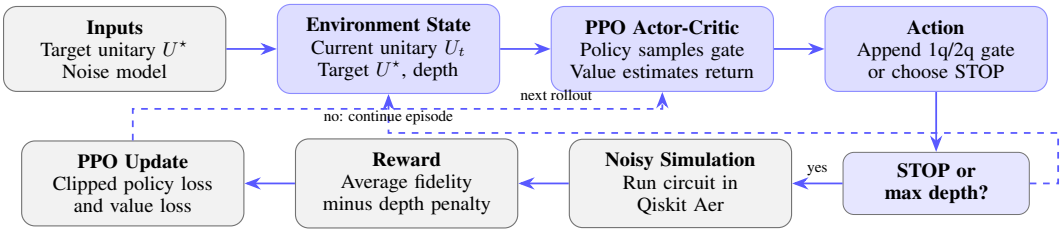


Figure 2: Overview of our noise-aware RL compilation pipeline. The agent builds a circuit one gate at a time, stops when it chooses STOP or reaches the depth limit, and then receives a noisy fidelity reward used for the PPO update.

### 3.1 MDP Formulation

In this context, an episode is defined as the process of constructing one candidate circuit. The environment starts with an empty $n$-qubit circuit, so the current unitary starts out as the identity matrix. The target unitary is one of Bell preparation, SWAP, GHZ, or QFT.

Each observation is a real-valued vector containing the real and imaginary parts of the current unitary $U_t$, the real and imaginary parts of the target unitary $U^\star$, and a scalar depth fraction $d_t/d_{\max}$. For $n$ qubits, the unitary dimension is $2^n$, so the observation size is $4(2^n)^2 + 1$. This representation is not hardware scalable, but it is appropriate for this project because it gives the agent exact information about the partial circuit.

The action space is a discrete set of elementary quantum gates. For each qubit, the agent can choose the Hadamard $H$, $X$, $Y$, $Z$, $S$, $T$, and rotations $R_x$, $R_y$, $R_z$ with angles $\pi/4$, $\pi/2$, and $\pi$. For every ordered pair of distinct qubits, the agent can choose a CNOT. The final action is STOP. If STOP is selected or the maximum allowed depth is reached, the episode terminates.

The terminal reward is

$$R(C) = F_{\text{avg}}(C, U^\star) - \lambda \cdot |C|,$$

where $|C|$ is the number of gates and $\lambda$ is the depth penalty. The fidelity term is computed as the average state fidelity over all computational basis inputs:

$$F_{\text{avg}}(C, U^\star) = \frac{1}{2^n} \sum_i F\left(U^\star|i\rangle,\ \mathcal{E}_C(|i\rangle\langle i|)\right).$$

We define $\mathcal{E}_C$ as the noisy channel that is induced by running circuit $C$ in Qiskit Aer. In the main experiments, we let non-terminal rewards be 0, a sparse reward environment. We also experimented with using dense reward functions, but this paper reports the sparser terminal reward because we deemed that they more directly match the final objective.

### 3.2 Noise Models

A "combined medium" noise model is used in the main experiments. We assign single-qubit gates to receive depolarizing noise with probability $p_1 = 0.005$ alongside amplitude damping with $\gamma = 0.01$. Two-qubit CNOT gates receive depolarizing noise with probability $p_2 = 0.02$ and independent amplitude damping on both qubits. This is due to the noise inherent in these gates when they occur in a real, non-simulated environment. We also implemented weak and strong depolarizing and combined noise settings for robustness experiments.

Compared to the types of noise models that might occur in real-world scenarios, this model is still somewhat simplified. For example, real-world quantum circuits have connectivity constraints, measurement error, or gate-specific device data. Regardless, this simulated noise captures the main tradeoff: while adding gates is necessary to improve the ideal unitary match, they also can expose the state to more noise.

### 3.3 PPO Agent

We represent the policy as a categorical distribution over the discrete gate actions. A shared MLP with three hidden layers of size 256 and $\tanh$ activations, followed by separate policy and value heads is utilized. We train PPO with generalized advantage estimation (GAE), clipped policy updates with $\epsilon = 0.2$, value loss weight $0.5$, gradient clipping, and entropy regularization.

The agent collects a rollout of 256 environment steps for each update, the default learning rate is $3 \cdot 10^{-4}$, the discount factor is $0.99$, and the GAE parameter is $0.95$. Bell, QFT and SWAP use entropy coefficient $0.03$. The Bell run used 200,000 environment steps and maximum depth 15. The SWAP and GHZ runs used 150,000 environment steps, with maximum depths 20. The QFT run used 300,000 environment steps, with maximum depth 20.

### 3.4 Efficient Fidelity Evaluation

The slowest part of the environment is computing terminal fidelity. For $n$ qubits, the metric requires simulating all $2^n$ computational basis inputs. At first, this would naturally be implemented as separate simulator calls. That was too slow for RL because we'd need to compute the terminal reward thousands of times during training.

The implemented environment instead creates all basis-input circuits and sends them to Qiskit Aer as one batched density-matrix job. Aer can parallelize over the batch internally, which substantially improves throughput and avoids all the extra Python overhead (instead, this is all handled in Aer's C backend). In the benchmark run, the environment reached roughly 950–1200 steps per second for a short two-qubit setting. Full Bell training ran at about 472 steps per second, while the three-qubit GHZ run was slower at about 173 steps per second because it requires eight basis states per terminal evaluation instead of four.

## 4 Experimental Setup

We evaluated three methods. The random baseline uniformly samples gate actions and uses a small STOP probability. This baseline is intentionally weak, but it serves to shows how hard the problem is without learning.

The Qiskit baseline constructs the known ideal circuit for the target, transpiles it into the allowed basis, and evaluates the resulting circuit under the same noise model. We limit the optimization level on the Qiskit approach to 1, because higher optimization levels usually lead to Qiskit using tricks that make it a bad baseline for comparison (e.g., it turns SWAP into an internal permutation that happens within the layout itself, and so this produces an empty circuit with depth of 0, which we can't compare against).

PPO is evaluated greedily by taking the argmax action from the trained policy.

The main targets are:

- **Bell, 2 qubits:** $H$ on qubit 0 followed by CNOT$(0, 1)$.
- **SWAP, 2 qubits:** exchange the two qubits, normally decomposed into three CNOT gates.

- **QFT, 2 qubits:** the two-qubit Quantum Fourier Transform, consisting of a Hadamard gate, a controlled phase rotation, a second Hadamard gate, and a final SWAP to reverse qubit order.
- **GHZ, 3 qubits:** $H$ on qubit 0 followed by a chain of CNOTs.

For Bell, PPO was evaluated over 50 deterministic rollouts. For SWAP, PPO was evaluated over 30 deterministic rollouts. Since the trained policy is deterministic at evaluation time and the simulator returns exact density matrices, PPO's standard deviation is essentially zero in these results.

# 5 Results

## 5.1 Bell-State Preparation

Bell-state preparation is the strongest result after the QFT. PPO learns the two-gate circuit and reaches the same fidelity as the Qiskit baseline. Table 1 shows that PPO achieves average fidelity $0.9585$ with mean depth 2.0, while random search only reaches $0.2491$ on average. The learning curve in Figure 3 also shows a clear rise during training, ending at the optimal two-gate solution.

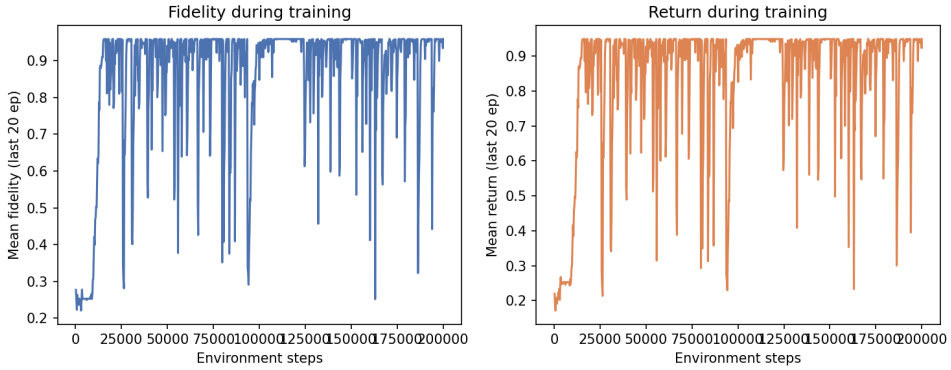


Figure 3: Bell training curve. PPO eventually learns a high-fidelity, short circuit and ends at the same fidelity as the Qiskit baseline.

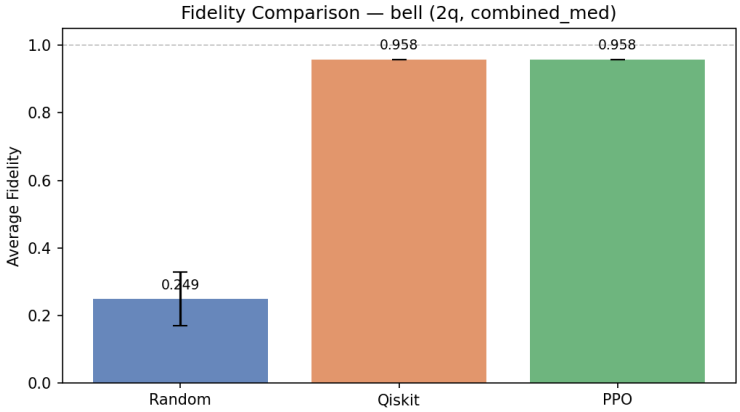


Figure 4: Bell evaluation under the medium combined noise model. PPO matches Qiskit and strongly outperforms the random baseline.

This result matters because Bell preparation is not just a single-qubit target. The agent has to discover that an entangling gate is needed, and it has to stop after the useful gates rather than continuing to add noisy operations. The learned policy therefore captures the basic idea of noise-aware synthesis: correctness and shortness both matter.

Table 1: Evaluation results under the medium combined noise model.

| Target | Method | Mean Fidelity | Std. Fidelity | Mean Depth |
|---|---|---|---|---|
| Bell | Random | 0.2491 | 0.0797 | 7.56 |
| Bell | Qiskit | 0.9585 | 0.0000 | 2.00 |
| Bell | PPO | 0.9585 | 0.0000 | 2.00 |
| SWAP | Random | 0.2462 | 0.1412 | 7.53 |
| SWAP | Qiskit | 0.4938 | 0.0000 | 0.00 |
| SWAP | PPO | 0.4938 | 0.0000 | 0.00 |

### 5.2 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is the most challenging benchmark in this paper and produces the strongest overall result. Unlike Bell-state preparation, QFT requires a sequence of operations which in total implement a change-of-basis rather than simply creating entanglement. PPO consistently discovers circuits that outperform both the Qiskit baseline and random search across all the noise settings we tested with.

| Noise | PPO Fid. | Qiskit Fid. | Random Fid. | PPO Depth | Qiskit Depth | PPO vs Qiskit |
|---|---|---|---|---|---|---|
| combined_weak | 0.4216 | 0.3686 | 0.243 | 3 | 7 | +14.4% |
| combined_med | 0.4173 | 0.3630 | 0.243 | 3 | 7 | +15.0% |
| combined_strong | 0.3907 | 0.3283 | 0.246 | 3 | 7 | +19.0% |
| depolarizing_strong | 0.4065 | 0.3370 | 0.245 | 3 | 7 | +20.6% |

Table 2: QFT evaluation across multiple noise models. PPO consistently achieves higher fidelity than Qiskit while producing substantially shorter circuits.

Table 5.2 shows that PPO outperforms the Qiskit baseline in every noise regime. Under the medium combined noise model, PPO reaches fidelity 0.4173 compared with 0.3630 for Qiskit and 0.243 for random search, a relative improvement of approximately 15%. The advantage grows as noise increases, reaching 20.6% under strong depolarizing noise.

Another important result is the circuit depth. PPO consistently discovers depth-3 circuits, whereas the Qiskit implementation has depth 7. The learned policy therefore reduces circuit depth by more than a factor of two while also improving fidelity. This behavior is exactly what the reward function was designed to incentivize: under noisy execution, shorter circuits can achieve higher fidelity even when they differ from the standard textbook decomposition.

These results demonstrate the main advantage of reinforcement learning for this problem. Rather than reproducing a fixed theoretical circuit, the agent learns implementations that are better adapted to the noise model and the available gate set. As circuit complexity increases, these opportunities for noise-aware optimization become more important, which helps explain why the performance gains for QFT are larger than those observed for Bell-state preparation or SWAP.

### 5.3 SWAP

The SWAP target, however, shows us one of the main weakness in the current formulation. PPO converges to STOP immediately, giving fidelity $0.4938$ and depth 0. This is better than the random average, but it is not a true SWAP circuit. The learning curve in Figure 5 shows that the agent settles into this local solution and stays there.

This failure is understandable from the reward. A true SWAP requires multiple two-qubit gates, and two-qubit gates are noisy and penalized by depth. Early in training, random long circuits are usually much worse than stopping. Once the policy learns that STOP is safe, the exploration phase ends and PPO never discovers the three-CNOT sequence we were looking for. Often, the Qiskit implementation of SWAP also has depth 0, but this is because the compiler can represent SWAP as just an internal layout permutation without needing any gates at all, which isn't captured by our current evaluation metrics. For that reason, the Bell and QFT Qiskit results are the cleaner compiler comparison, and the SWAP result should mainly be read as a PPO failure case. However, interestingly,

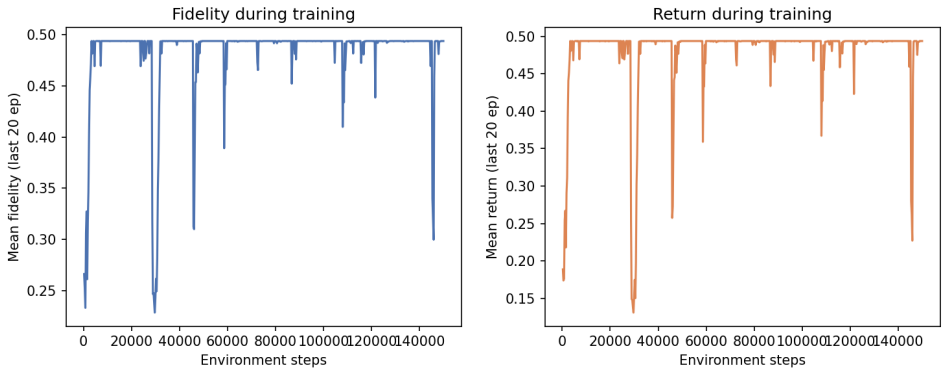


Figure 5: SWAP training curve. PPO converges to the immediate STOP policy, which receives moderate fidelity but does not synthesize the true SWAP.

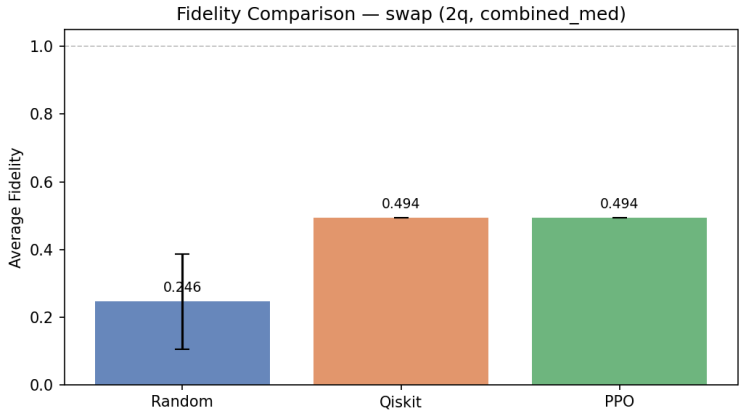


Figure 6: SWAP evaluation. PPO beats the random mean but only by learning to stop at depth 0.

these failure modes were fixed when we trained SAC on the same problem. Under strong depolarizing noise, SAC reaches a fidelity of 0.712, far better than Qiskit's 0.493, and we find it does indeed learn the 3 CNOT gate circuit we'd expect. We hypothesize that its ability to use the replay buffer and make use of old circuits it discovered rather than just throwing them away helps it keep track of more trajectories than PPO, which very quickly converges to the STOP action.

### 5.4 GHZ Scaling Run

The three-qubit GHZ run was included to test whether the approach scales beyond two qubits. The final training log reached a mean fidelity of $0.7645$, but the training was unstable: nearby updates varied from around $0.56$ to $0.87$. This instability is not surprising. Moving from two to three qubits doubles the number of basis states used in the fidelity calculation, increases the observation size from 65 to 257, and expands the gate catalogue. The target also requires a longer entangling sequence, which makes sparse terminal credit assignment harder.

Still, GHZ was a useful target because it showed that although the environment and training pipeline can handle three-qubit targets, the plain PPO recipe is not enough for reliable larger-circuit synthesis.

### 5.5 Noise Robustness

For Bell, we also evaluated the trained agent across multiple noise settings. The robustness plot in Figure 7 compares PPO against random search as the noise model changes. The main qualitative pattern is that PPO remains much better than random across the tested settings, but fidelity drops as noise becomes stronger. This matches the expected behavior: the learned Bell circuit is short and

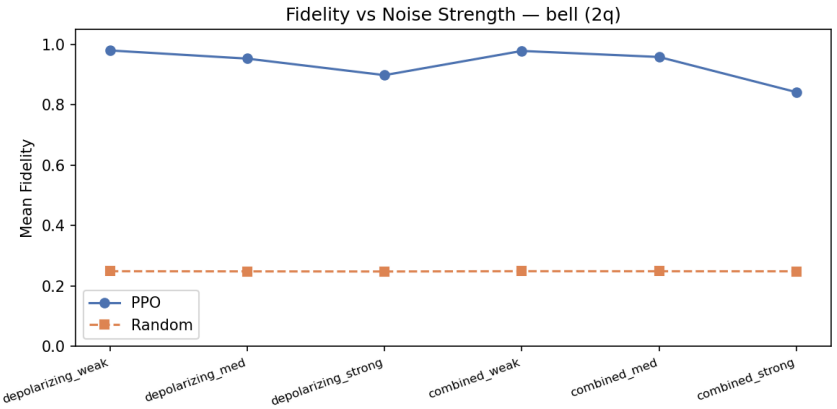


Figure 7: Bell noise robustness evaluation. The learned PPO policy stays well above random across noise settings, with lower fidelity as noise increases.

structurally correct, so it remains good, but no policy can avoid all gate noise when entanglement requires a CNOT.

## 6 Discussion

The biggest positive result is that the environment is capable of teaching a policy a real quantum circuit under a noisy reward. PPO did not simply memorize a string from supervision; it interacted with a simulator, received delayed feedback, and learned to stop at the correct depth. For Bell preparation, this worked about as well as we could hope.

The biggest limitation is sparse reward. The Bell target is short enough that PPO can eventually stumble into the right structure. SWAP is only one gate longer in the standard decomposition, but it is much harder because the intermediate circuits do not obviously look like the goal and the STOP action is too attractive. This suggests that future versions should use better reward shaping, curriculum learning, or search-guided exploration. One possible direction is to train first with noiseless unitary fidelity as a dense shaping signal and then fine-tune with noisy fidelity. Another is to use hindsight experience replay, since we can directly multiply matrices to find the exact noiseless unitaries we need.

The state representation is also a limitation. Storing full unitaries is fine for two or three qubits, but it scales exponentially. A more realistic compiler would need a representation based on circuit structure, Pauli transfer matrices, hardware topology, or learned embeddings. The current representation was chosen because it keeps the small-scale problem clean and lets us focus on the RL formulation.

Finally, the noise model is simplified. Real hardware has qubit-specific error rates, connectivity constraints, calibration drift, and measurement errors. Adding those details would make the project more realistic, but it would also make the learning problem harder. The current simulator should be viewed as a controlled first step.

## 7 Conclusion

This project built and tested a reinforcement learning pipeline for noise-aware quantum circuit compilation. The main accomplishment is a working Gymnasium/Qiskit environment where an agent constructs circuits gate by gate and is rewarded by noisy fidelity minus depth cost. A custom PPO agent successfully learned Bell-state preparation, matching the Qiskit baseline at fidelity $0.9585$, and outperformed the baseline for the Quantum Fourier Transform. At the same time, PPO failed to synthesize SWAP and showed unstable behavior on GHZ, which makes its limitations clear. In this setting, plain PPO is enough for very small, clean targets, but harder circuits need better credit assignment and exploration, and likely a more off-policy algorithm like SAC (which showed much more promising results on SWAP).

## 8 Team Contributions

- **Abhishek:** Abhi was the one in charge of researching and implementing the simulated environment, which includes choosing the final set of target circuits (e.g., GHZ, Bell, etc.) to evaluate our approach on, figuring out how to implement noise models and set up a good way to compute the final evaluation metric (which in our case was fidelity with some penalties). Abhi also helped with debugging on the RL algorithms themselves.
- **Vinav Shah:** Vinav was the primary team member in charge of implementing the PPO algorithm (and experimenting with a few other techniques to measure relative success, such as SAC). Once these algorithms were implemented and training was complete, Vinav combined this with the evaluation/simulation framework Abhi developed to lead data analysis. Both of us then worked together to draw conclusions and try to explain some of the behaviors we were seeing.

## 9 AI Use Disclosure

ChatGPT, Claude, and Cursor were used extensively. They were used for boilerplate code, debugging the data pipelines, and research assistance for some of the quantum mechanics. The PPO and SAC algorithms, which was the primary algorithmic part, were written by us, in consultation with prior course homework and other resources. We also did the ideation.

**Changes from Proposal** For the most part, both of our roles remained similar to what we had planned, although there was less of a strict separation and much more fluidity (for instance, when things weren't working as they were supposed to on the simulation, Vinav would help look through alternative approaches, even though it wasn't strictly his role in the initial project proposal).

## References


[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[2] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.

[4] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2024. `https://www.ibm.com/quantum/qiskit`

[5] Qiskit Aer contributors. Qiskit Aer: High performance simulators for Qiskit. `https://qiskit.github.io/qiskit-aer/`


## A Hyperparameters

Table 3: Main PPO hyperparameters.

| Hyperparameter | Value |
|---|---|
| Learning rate | $3 \cdot 10^{-4}$ |
| Hidden size | 256 |
| Rollout length | 256 |
| PPO epochs per update | 10 |
| Batch size | 64 |
| Clip parameter | 0.2 |
| Discount factor | 0.99 |
| GAE $\lambda$ | 0.95 |
| Depth penalty | 0.005 for Bell/SWAP/QFT, 0.003 for GHZ |