Day - 14

Key Reasons Why Qiskit-Based QNN is Significantly Slower than Cirq-Based Version

1. Backend Type and Execution Model

Cirq:

- Uses cirq.Simulator() — a fast local simulator optimized for classical simulation of quantum circuits.
- Execution is lightweight and happens on CPU with minimal overhead.

Qiskit:

- Uses qiskit_aer.primitives.Sampler, which internally relies on **Aer backends** (e.g., qasm_simulator).
- These simulate quantum measurements (shots), but with **higher overhead per call**, especially with large number of measurements or full circuit measurements (measure_all()).

2. Measurement Overhead

Qiskit circuit measures all qubits (qc.measure_all()), while in Cirq we measured only qubit 0:

- Measuring all qubits adds unnecessary complexity and result processing overhead.
- Extracting marginal distributions from the full 4-qubit readout in Qiskit is slower than just counting qubit 0 outcomes like in Cirq.

Fix: Replace qc.measure_all() with qc.measure(0) to mimic Cirq behavior more closely.

3. Sampler API and Job Overhead

- The Sampler API is more general and supports quasi-distributions and sampling strategies, which adds **latency**.
- Internally it uses qiskit_aer.Sampler.run() which wraps jobs as batch jobs, even for single circuits.

Fix: Use qiskit_aer.AerSimulator directly and call .run(transpile(qc, simulator)) for more control and speed.

4. Circuit Transpilation and Parameter Binding

- Qiskit's parameter binding and transpilation are **not lightweight**. Every time you change params, the circuit gets **rebuilt** and possibly **transpiled** again.
- In Cirq, you use NumPy-level operations — faster with fewer layers of abstraction.

Fix: Use ParameterizedCircuit.bind_parameters() outside of recompilation steps if possible.

5. Result Processing Time

- Cirq returns a histogram dictionary directly — fast and native.
- Qiskit's Sampler.result() returns a list of QuasiDistribution objects, requiring formatting and extraction per run, which is relatively slow.

Summary: Why Qiskit is Slower

Factor	Cirq (Faster)	Qiskit (Slower)
Simulator	Fast local simulator	General-purpose Aer with overhead
Measurement	Single qubit	All qubits, extra processing
Circuit Compilation	Lightweight	Parameter binding + transpilation
API Overhead	Minimal	Higher due to Sampler design
Result Interpretation	Direct count	Quasi-distribution post-processing

Recommendations to Speed Up Qiskit Code

1. **Measure only required qubit:**
   - qc.measure(0)

2. **Use AerSimulator() directly:**
   - from qiskit_aer import AerSimulator
     simulator = AerSimulator()
     compiled_circuit = transpile(qc, simulator)
     job = simulator.run(compiled_circuit, shots=100)

3. **Avoid full recompilation by reusing circuits.**

4. **Switch to analytic expectation value simulation (like StatevectorSimulator) if not using shot-based learning.**