IBM Qiskit Community Summer Jam Hackathon
(North Carolina)

Hybrid Neural Network with Qiskit and Pytorch

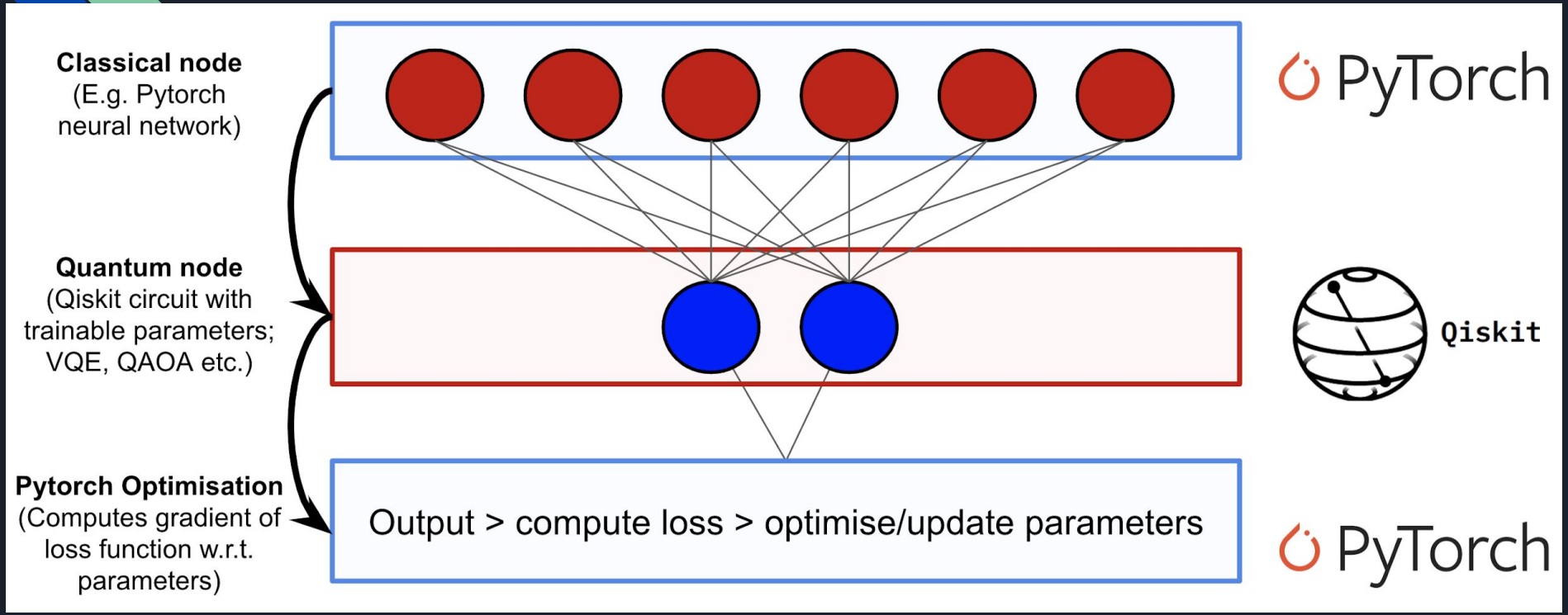
Hackathon Coach: Robert Loredo

Team Ube Pancake

Foundational Work

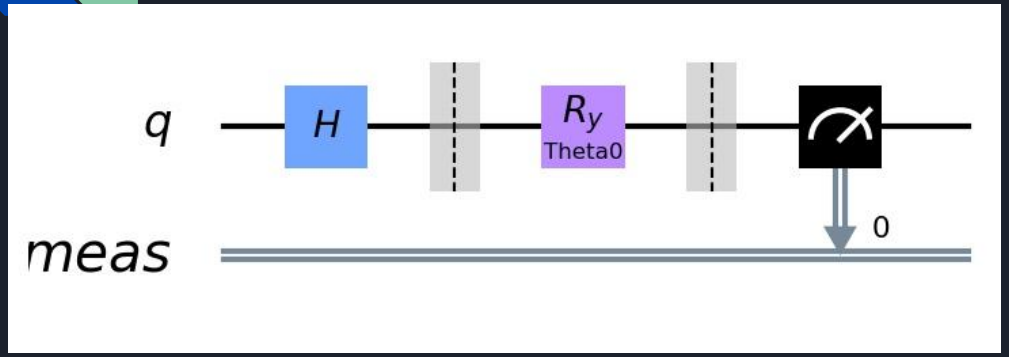
- Qiskit Textbook: Hybrid quantum-classical Neural Networks with PyTorch and Qiskit
  - https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html
  - Implemented a quantum neural network layer of 1 qubit with 1 trainable parameter into a classical neural network for the classification between 0 and 1
- Qiskit Camp Europe 2019 Hackathon: Team QizGloria
  - https://github.com/BoschSamuel/QizGloria
  - Built upon the Qiskit Textbook example and expanded to 1 qubit with 2 and 3 trainable parameters
  - Attempted at implementing a 2-qubit layer

Quantum-classical Hybrid Neural Network



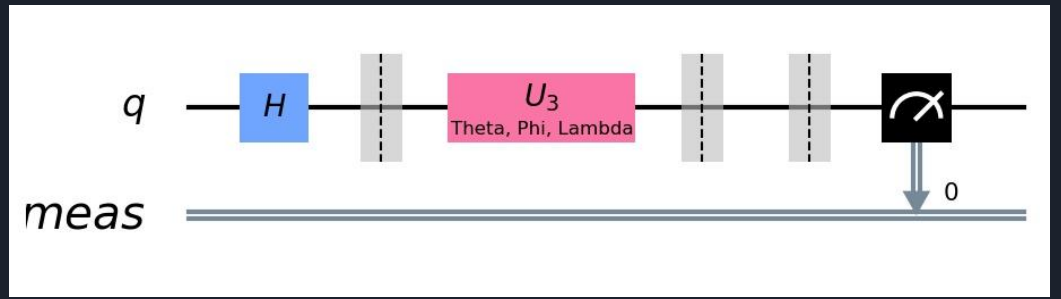
Reference: Qiskit Textbook Hybrid quantum-classical Neural Networks with PyTorch and Qiskit (https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html)

Quantum Layer Circuit

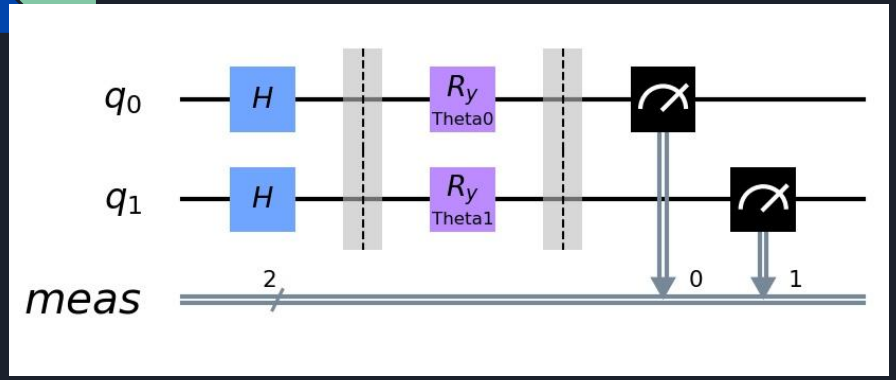


1-qubit circuit with a y-rotation that has 1 trainable parameter/qubit (ry)

1-qubit circuit with a unitary3 gate that has 3 trainable parameters/qubit (u3)

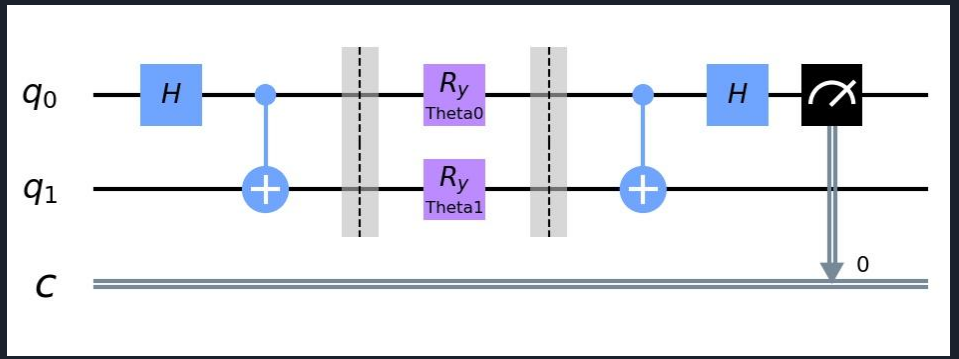


Quantum Layer Circuit

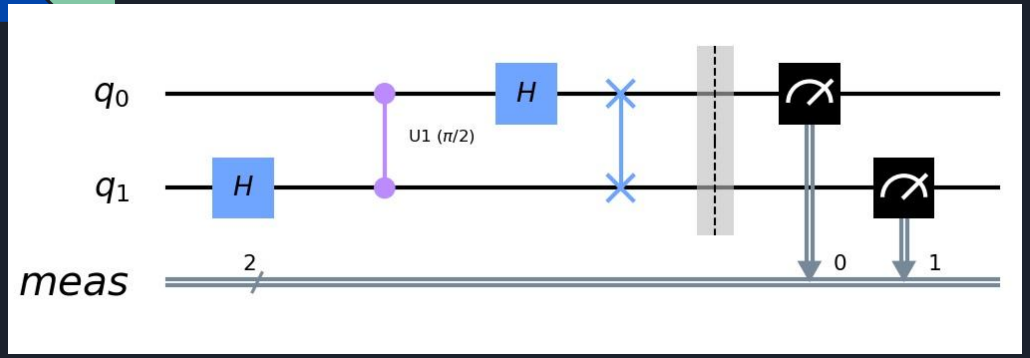


N-qubit y-rotation typed circuit with 1 trainable parameter/qubit (ryN)

Bell state circuit with n-qubit y-rotation with 1 trainable parameter/qubit (bell)



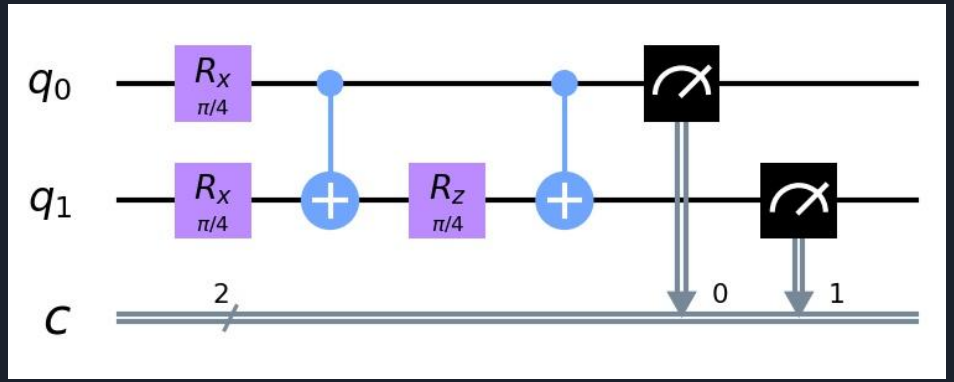
Quantum Layer Circuit (in progress)



N-qubit Quantum Fourier Transform Circuit (qft)

Qiskit Quantum Fourier Transform: https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html

N-qubit QAOA Circuit (QAOA)

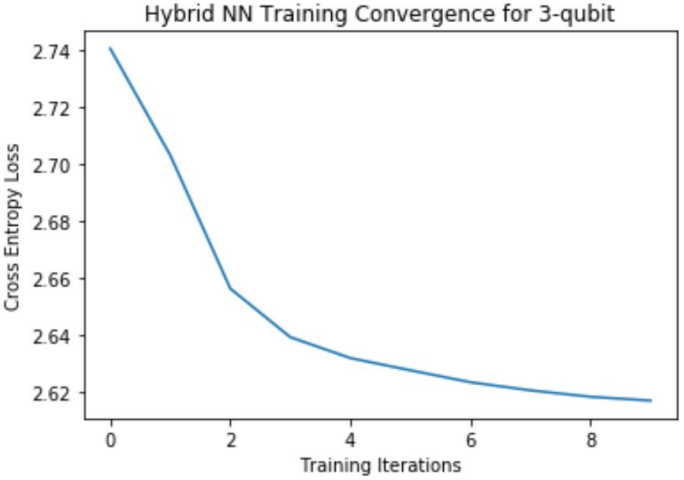


A Quantum Approximate Optimization Algorithm: https://arxiv.org/abs/1411.4028

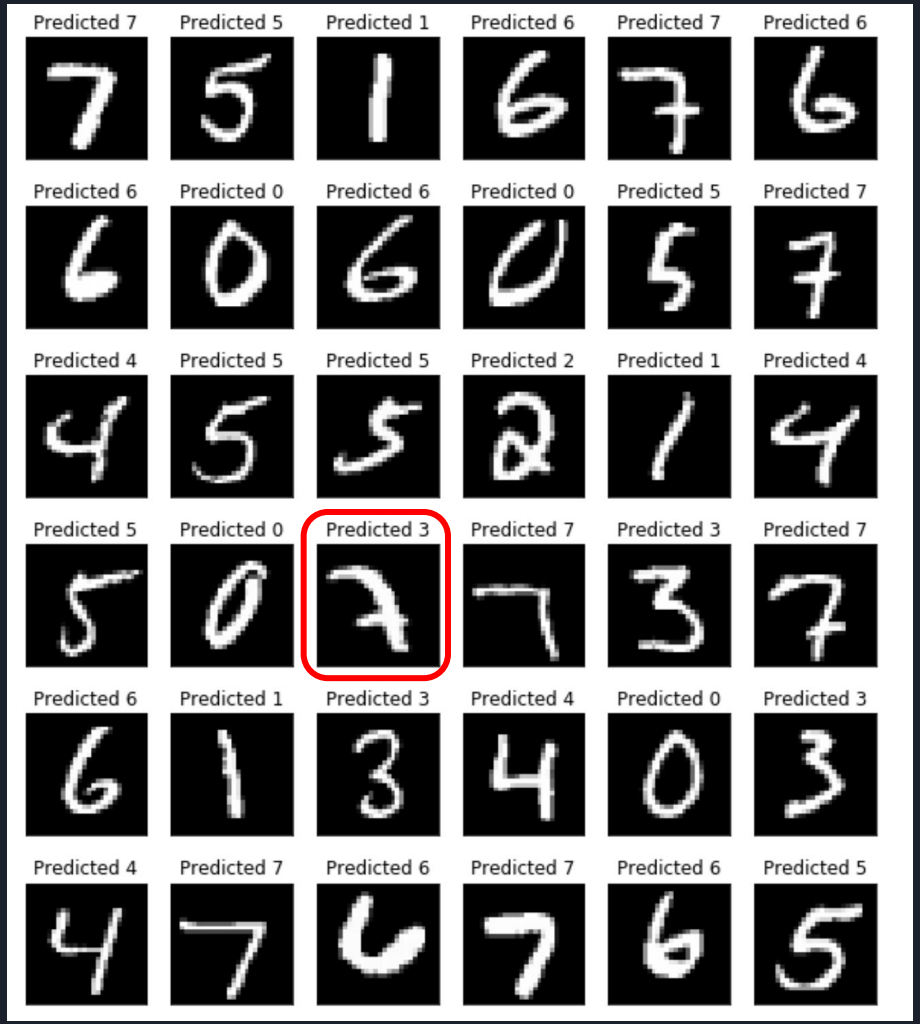
Result Highlights

	TASK	BACKEND	N_QUBITS	CIRC_TYPE	ACCURACY
	Classify 0-1	QASM Simulator	2	ryN	100%
	Classify 0-1	QASM Simulator	2	bell	99.8%
	Classify 0-5	QASM Simulator	3	ryN	91%
	Classify 0-7	QASM Simulator	3	ryN	95%
	Classify 0-9	QASM Simulator	4	ryN	86%

More results to be found in out README: https://github.com/liangqiyao990210/Quantum-Deep-Learning




```
Training [10%]  Loss: 2.7405
Training [20%]  Loss: 2.7030
Training [30%]  Loss: 2.6562
Training [40%]  Loss: 2.6391
Training [50%]  Loss: 2.6318
Training [60%]  Loss: 2.6275
Training [70%]  Loss: 2.6233
Training [80%]  Loss: 2.6204
Training [90%]  Loss: 2.6182
Training [100%] Loss: 2.6168
```



What we have tried and learned...

- Hybrid NNs could have potential applications for NISQ devices!! (Qiskit and Pytorch)
- It doesn't seem like the more qubits we use (hence more trainable parameters), the better results we get for classifying numbers (to be investigated)
- We have tried running our IBM Q Rome but it took too long…
- Don't try to run everything the night before...

Future Direction:

1. Try out different classification problems:
    a. MNIST Fashion, Cats and dogs, etc.
2. Try out different circuits:
    a. Investigate the differences between multiple parameters per single qubit versus multiple qubits each with a single parameter
    b. Try out more complex circuits, composite circuits, entanglement-generating circuits, n-controlled unitary, etc. (in progress)
    c. Explore suitable circuits for specific problems
3. Implement CUDA for GPU acceleration of training
4. Implement on NISQ devices (IBM quantum hardware):
    a. Try training our neural nets on the actual IBM machine (have attempted)
    b. Compare results from an actual hardware with that from the simulator
    c. Implement QECCs