Design and Implementation of a Quantum Machine Learning Inference Circuit

Prepared by: Andrija Paurevic
Supervised by: Dr. Glenn Gulak

Introduction

Introduction

Research Gap

- Classical Machine Learning (CML) is applied in various sectors of society (healthcare, financial, industrial, etc)
- CML is used in situations where the results must be reliable and room for error is extremely small (e.g. Will X default on their loan? Does X have a terminal illness?)
- The IDC claims the field of Quantum Computing (QC) currently has a market size of ~$500M and is projected to grow to ~$9B by 2027 [source]
- We are currently in the era of Noisy Intermediate Scale Quantum (NISQ) hardware
- NISQ hardware is composed of 10s - 100s noisy qubits each of which perform imperfect operations in a limited coherence time

Introduction

Research Gap

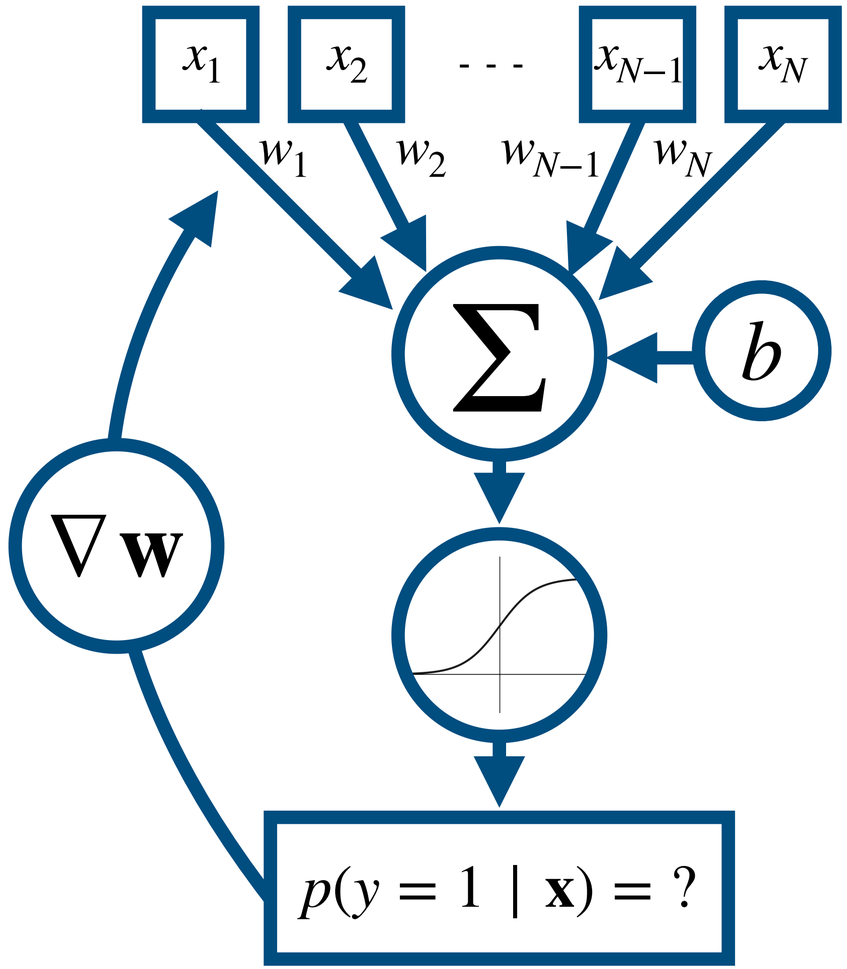
- Current researchers are focused on the development of theoretical Quantum Machine Learning (QML) algorithms
- Practical QML is lagging behind due to impairments introduced by NISQ hardware
- The objective of this research is to design and implement a QML inference circuit on NISQ hardware from an Engineering perspective
- The performance and scalability of practical quantum circuits were studied

Theoretical Background

Theoretical Background

Logistic Regression Algorithm

- Logistic Regression classifies an input data vector $(\mathbf{x})$, into one of two classes $(y = 0,\ y = 1)$.
- Classification is done with a sigmoid activation function
$$\sigma(\mathbf{x} \cdot \mathbf{w} + b) = \frac{1}{1 + \exp(-\mathbf{x} \cdot \mathbf{w} + b)}$$
with weight vector $(\mathbf{w})$ and bias (b)
- Sigmoid output gives the probability that the data is in class $y = 1$ or $y = 0$.
- Machine learning inference is using a pre-trained set of parameters to classify a test vector.



Theoretical Background

Quantum Mechanics

- Quantum mechanics tells us that at small scales, particles wave functions that obey Schrödinger's equation: $H(t)\,|\Psi(t)\rangle = i\hbar\frac{\partial}{\partial t}\,|\Psi(t)\rangle$
- Any particle's wave function is the linear superposition of its classical (measurable!) states ($|i\rangle$) given by: $|\Psi\rangle = \sum_{i=0}^{N} c_i\,|i\rangle,\ c_i \in \mathbb{C}$
- It is not until the particle is "measured" does it "collapse" into one of these classical states
- The probability of it collapsing into any given classical state $|i\rangle$ is given by Born's rule: $P\big(|\Psi\rangle = |i\rangle\big) = |c_i|^2$

Theoretical Background

Quantum Mechanics

- Consider a quantum coin; it is in a linear superposition of two classical states: heads and tails.
- Visualize the wave function as the coin mid-flip
- The coin's wave function is $|\Psi_{\text{QCoin}}\rangle = c_H|H\rangle + c_T|T\rangle$
- When "measured", it will collapse into one of the two classical states with respective probabilities: $P\big(|QCoin\rangle = |H\rangle\big) = |c_H|^2$ and $P\big(|QCoin\rangle = |T\rangle\big) = |c_T|^2$
- Can think of "measuring" the quantum state as when the coin hits the table

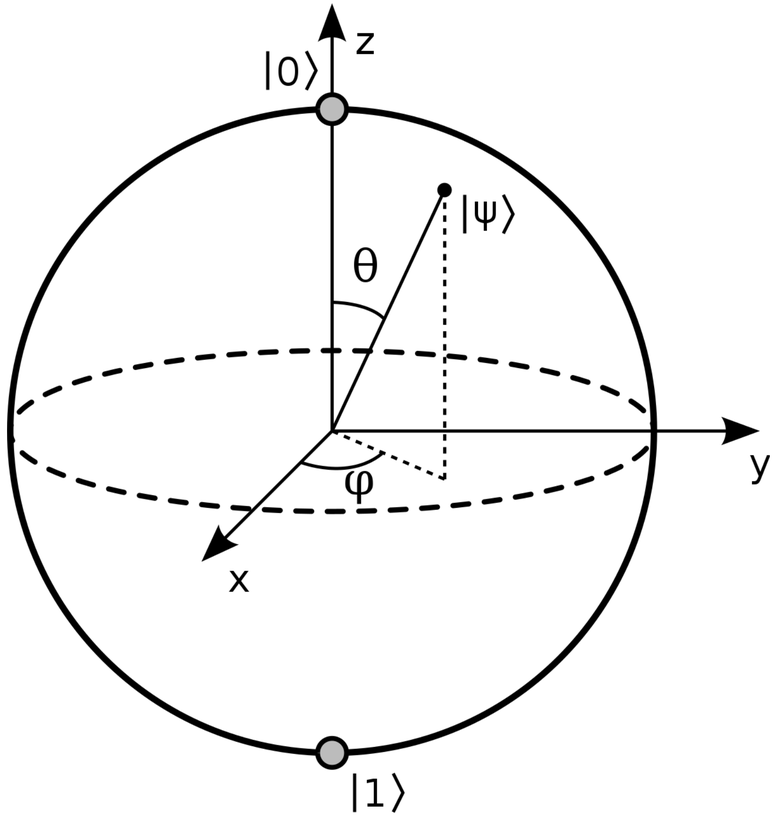
Theoretical Background

Quantum Computing

- A qubit is just a fancy quantum coin!
- The measurable states are the classical bits $|0\rangle$ and $|1\rangle$.
- We can represent a singular qubit with:

$$|\Psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

- This wave function spans the entire surface of a “Bloch Sphere”
- Every vector on the surface is a possible qubit state

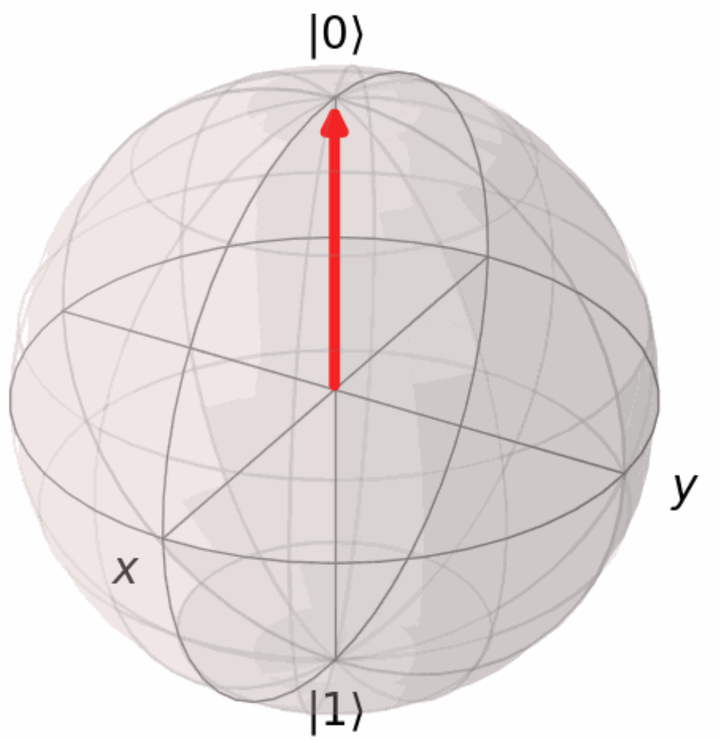


Bloch Sphere [source]

Theoretical Background

Quantum Computing

- Classical logic gates (AND, XOR, NOT, etc) act on classical bit(s) to manipulate their state in a strategic manner
- Quantum logic gates, U, act on qubit(s) in the same way
- Quantum gates are unitary, meaning they preserve probability densities and are their own inverses
- Can visualize a quantum logic gate as the operation required to move the qubit along the Bloch surface
- Famous gate is the “Hadamard” gate that creates a linear superposition: $H|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$

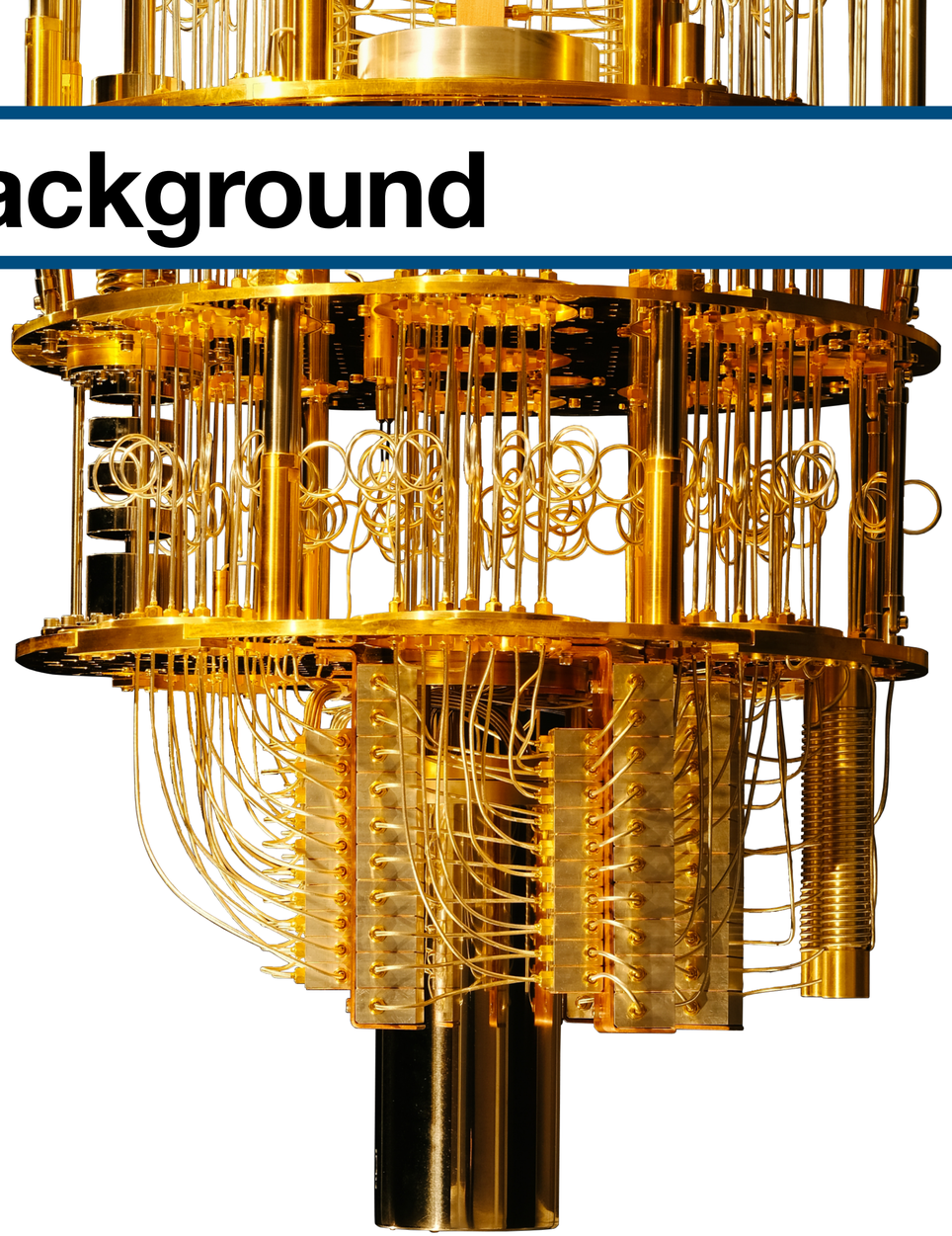


Transition of the $|0\rangle$ qubit state after a Hadamard gate

Theoretical Background

Quantum Hardware

- Many types of Quantum Computers exist today: Superconducting, Ion Trap, Photonic …
- IBM's superconducting machines were used
- Advantages:
    1. Manufacturing process leads to scalable design
    2. Fast Circuit Logic Operations Per Second (CLOPS)
    3. Open source quantum programming languages (Qiskit, Pennylane, etc) exist that can communicate with these machines
- Disadvantages:
    1. Fast qubit decoherence times (leads to noise)
    2. Expensive and inconvenient, requires extremely low temperatures to operate (below 100mK)



Superconducting Quantum Computer [source]

Theoretical Background

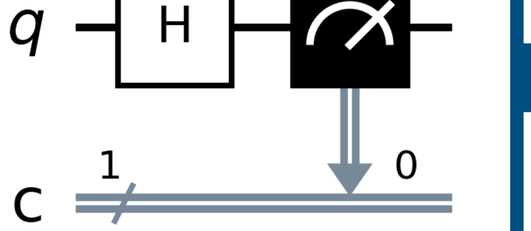
Quantum Programming

- Qiskit is an open source SDK used for creating and executing quantum circuits
- Python code gets compiled into Quantum Assembly (QASM) code and executed on hardware

```
# Create Quantum
Circuit
num_of_qubits = 1
num_of_cbits = 1
qc =
QuantumCircuit(1,1)

# Apply Hadamard
qc.h(qubit=0)

# Measure
qc.measure(0, 0)
```

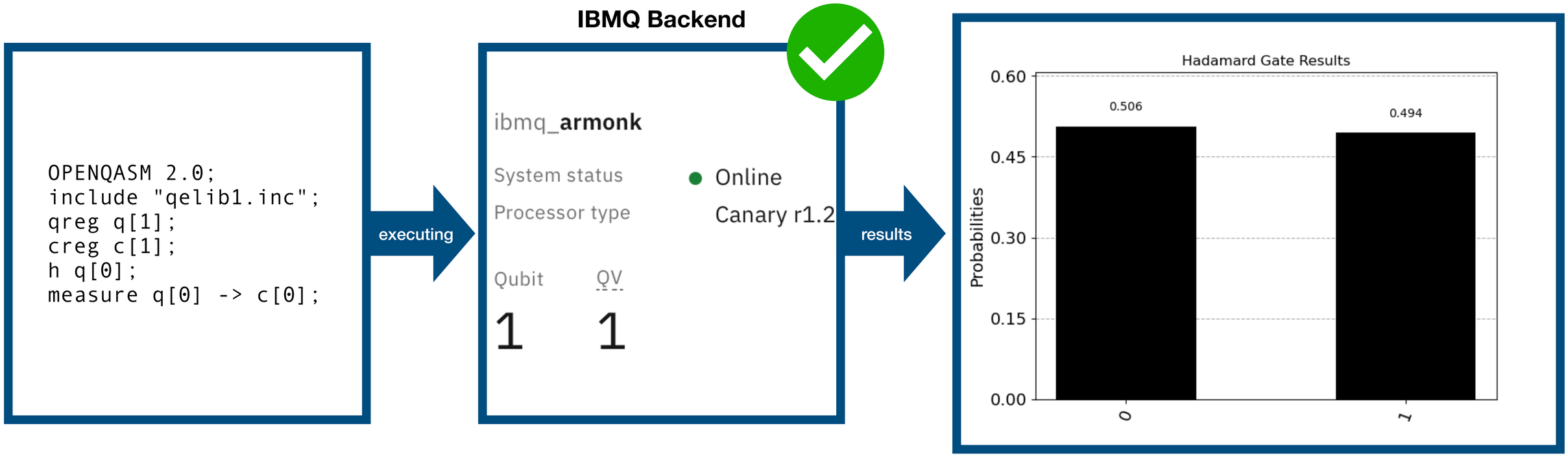



```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[1];
creg c[1];
h q[0];
measure q[0] -> c[0];
```

Theoretical Background

Quantum Programming

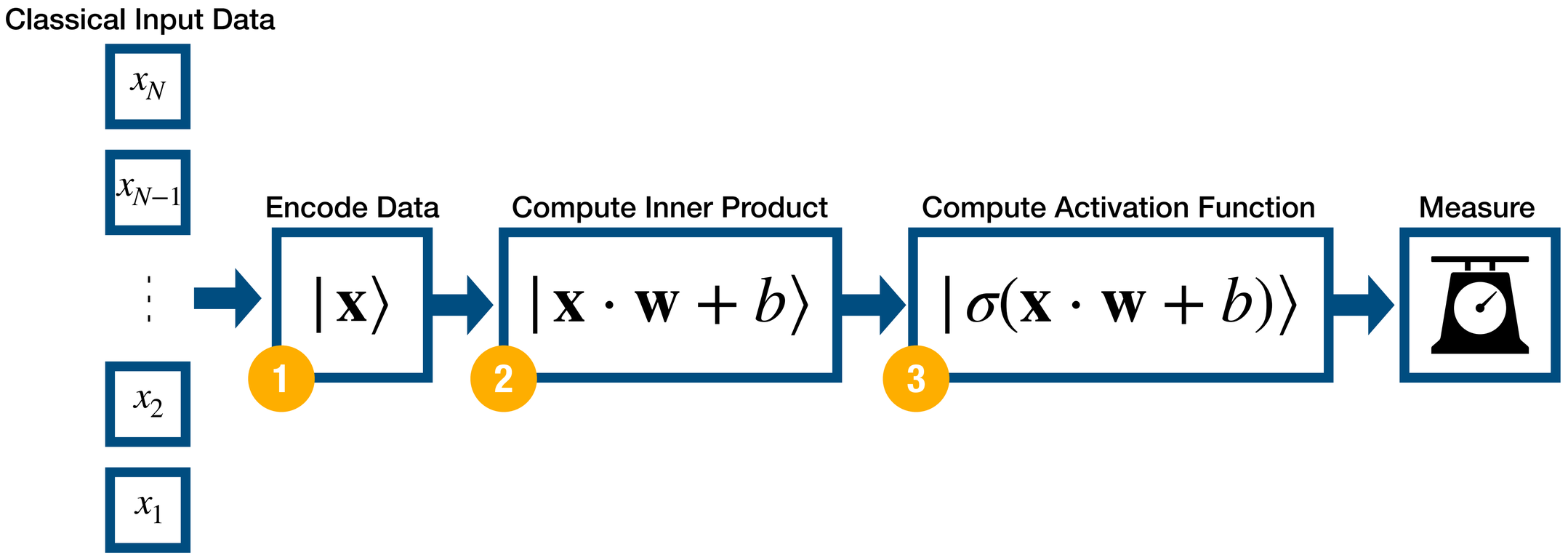
- QASM code is then sent to the desired IBMQ backend to be executed
- Each execution is a “shot” and returns the probability of every possible qubit state



Circuit Design

Circuit Design

(High Level) Design Flow



Circuit Design

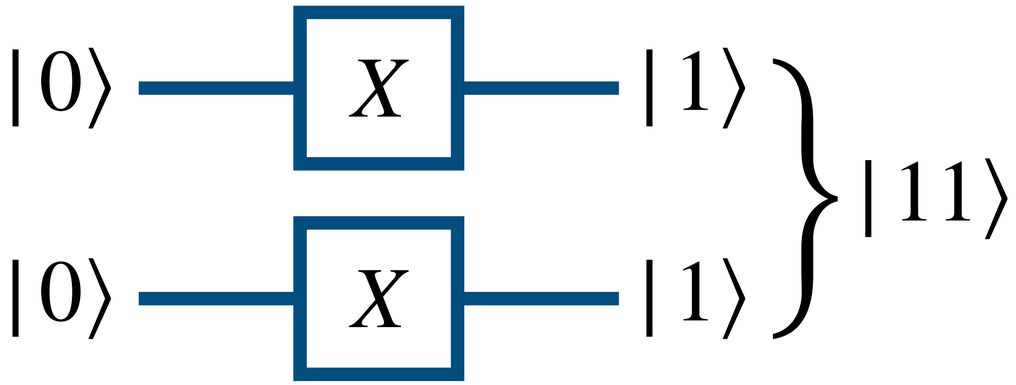
1 Encode Data

- Two main methods of encoding data: basis encoding and amplitude encoding
- Basis encoding encodes the binary form of the data: $\text{bin}(0.75) \equiv 0.11 \mapsto |11\rangle$
- Amplitude encoding encodes the data into the amplitudes of the qubit(s):

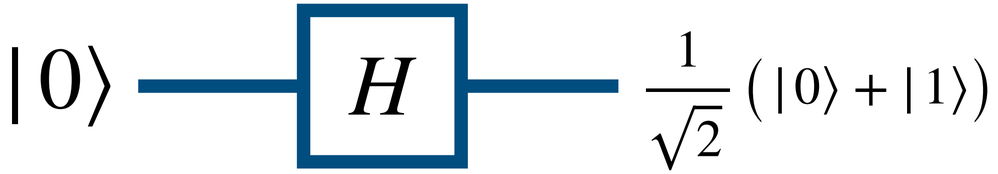
$$\mathbf{x} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \mapsto |\mathbf{x}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)$$

- Each of these methods has their own advantages and disadvantages on NISQ hardware

Basis Encoding



Amplitude Encoding



Circuit Design

1 Encode Data

Basis Encoding	
Advantages	**Disadvantages**
• O(n) gates per n bit number • Extremely low gate depth	• Requires O(n) qubits to represent 'n' bit number • 1-to-1 encoding (not very quantum)

Amplitude Encoding	
Advantages	**Disadvantages**
• Requires N qubits to represent 2**N dimensional data • Takes advantage of quantum superposition	• Extremely high gate depth for large dim data • Impractical on NISQ hardware • Data is normalized prior to encoding

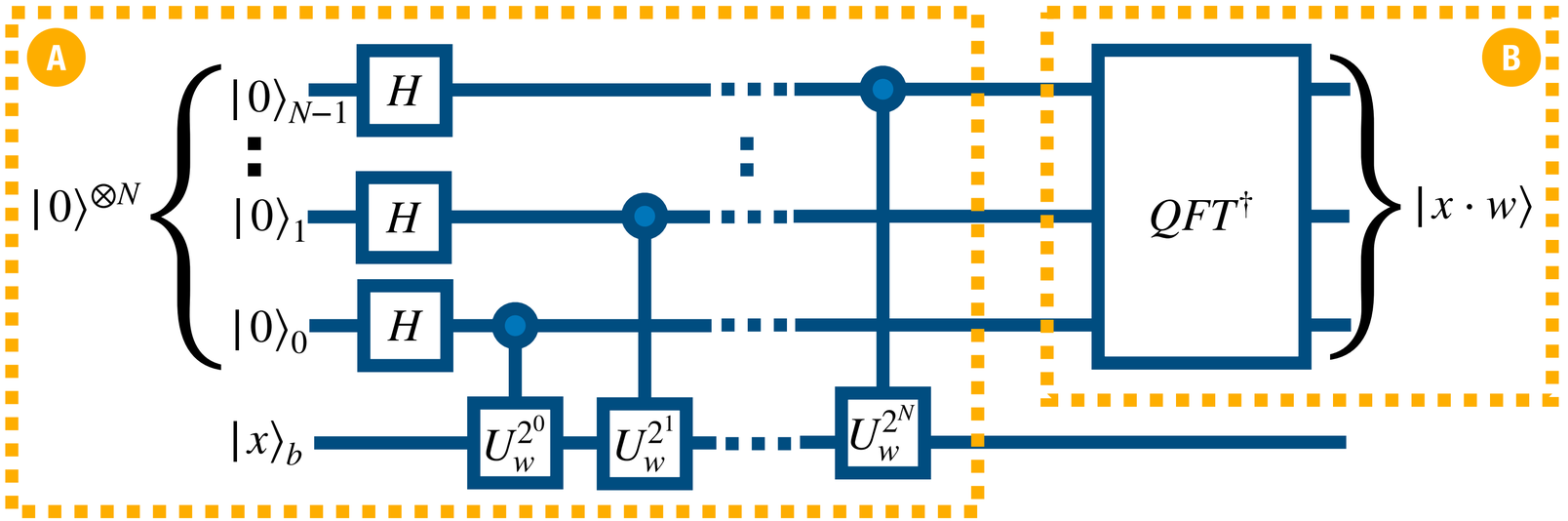
Circuit Design

2 Compute Inner Product

- In order to multiply two numbers (x, w), various gates and algorithms are required along with an ‘ancilla’ register to write the result to,

A Encodes the multiplication result into the phase of the ‘ancilla’ register

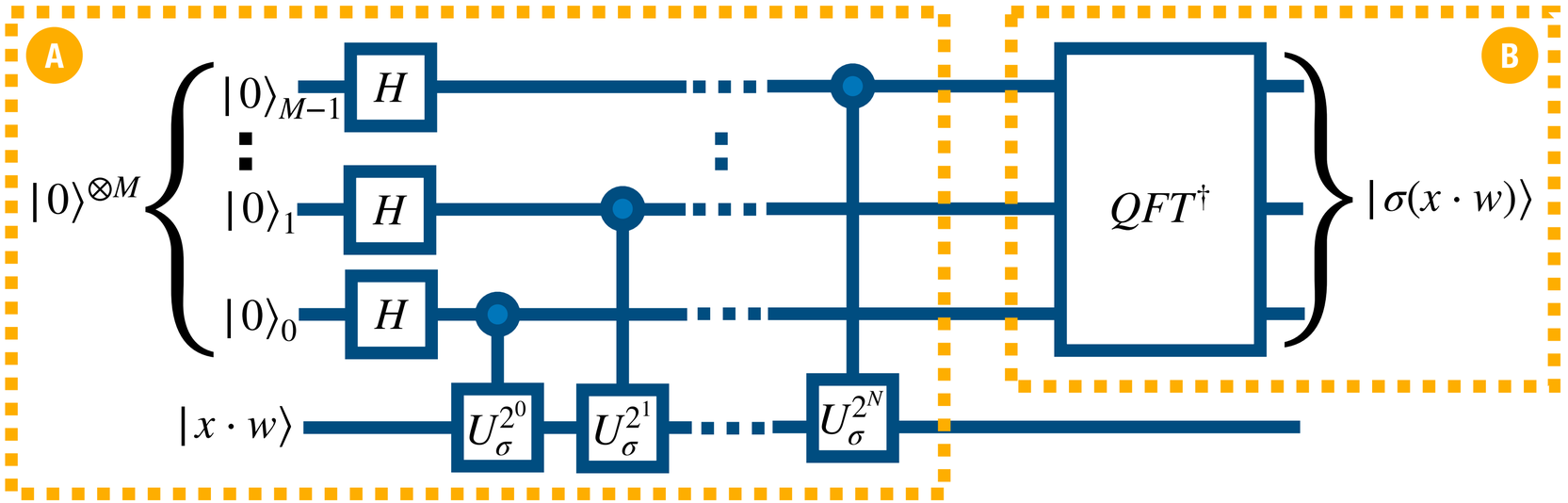
B Using the Inverse Quantum Fourier Transform algorithm, result is read out to N bit accuracy



Circuit Design

3 Compute Activation Function

- We can use the same circuit to find the output of a function
- This time we have a different gate, U_σ, acting on the input qubit(s)
- Function output can be read out to M bit accuracy



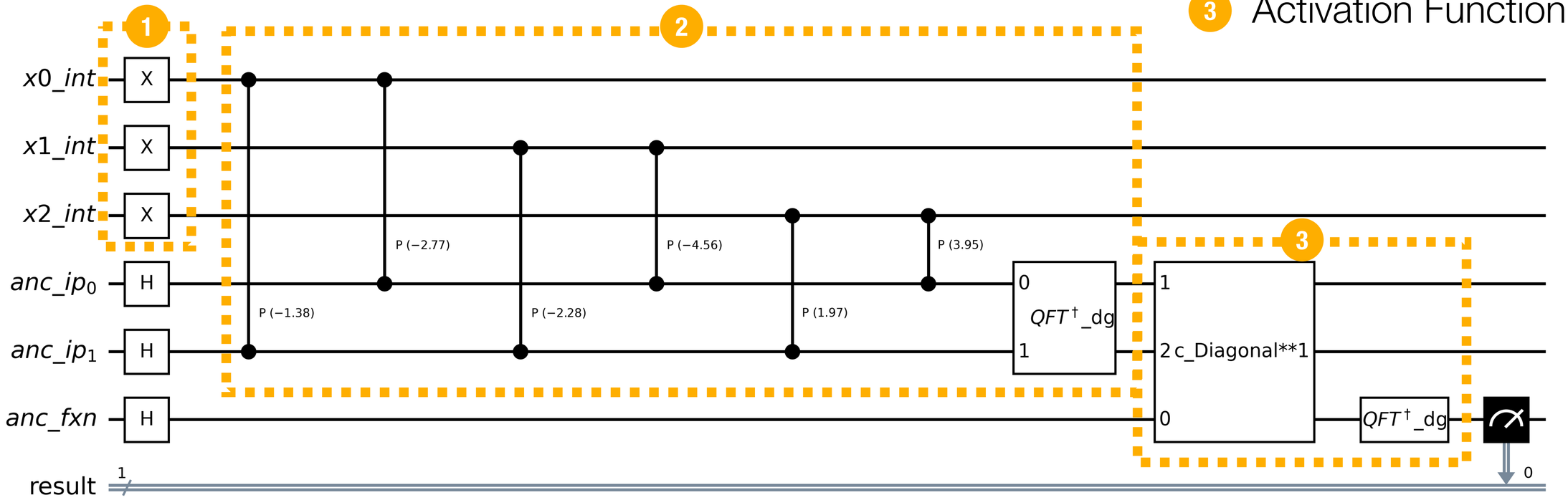
Implementation

Implementation

Scalable Quantum Circuit

1. Encode
2. Inner Product
3. Activation Function

- Here is an example quantum circuit testing $\mathbf{x} = (1,1)$



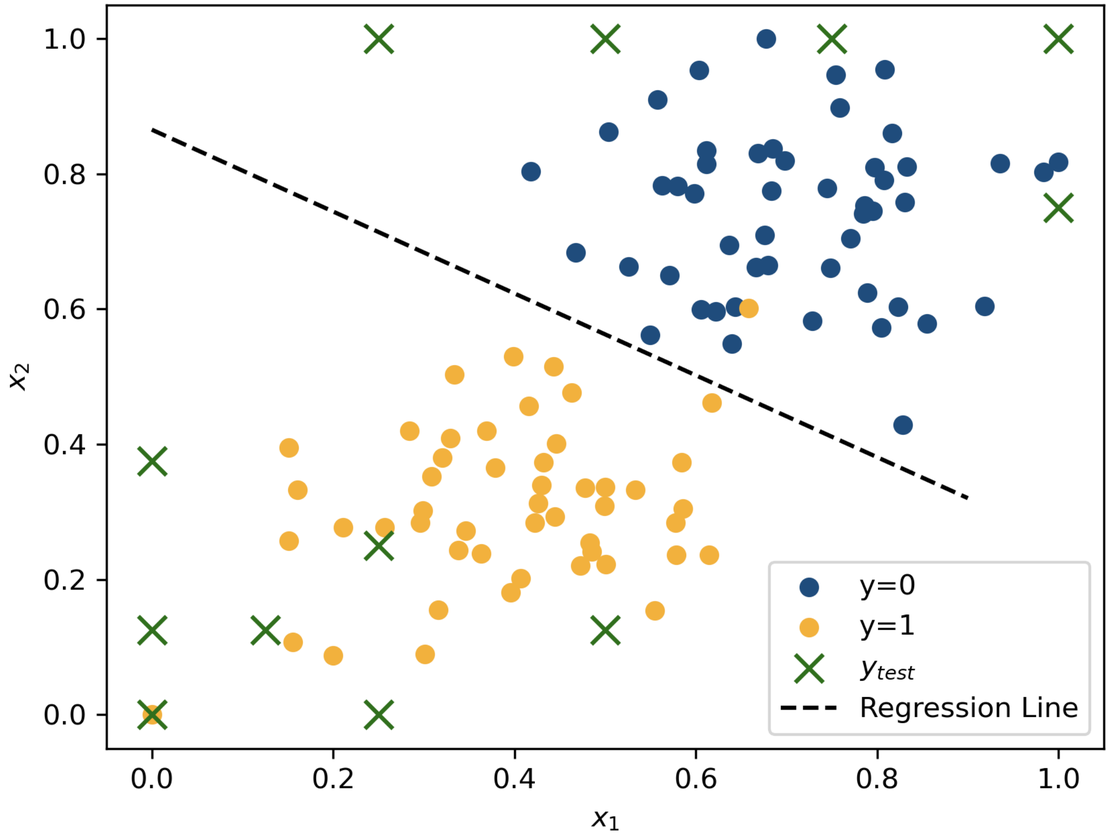
Implementation

Dataset

- A two parameter dataset was used to test the quantum circuit
- Dataset was designed to be clustered around encodable numbers
- Trained parameters were calculated classically to be:

$$\mathbf{w} = (-8.803, \ -14.53), \ b = 12.57$$

- 12 test vectors were chosen



Results

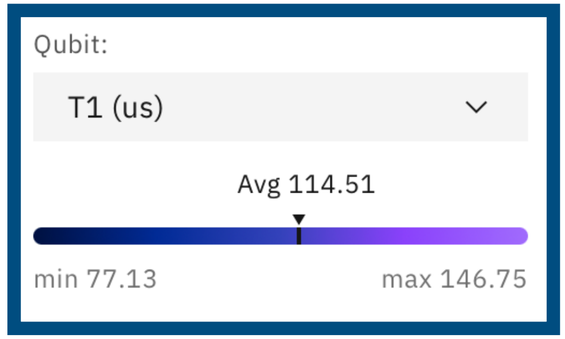
Results

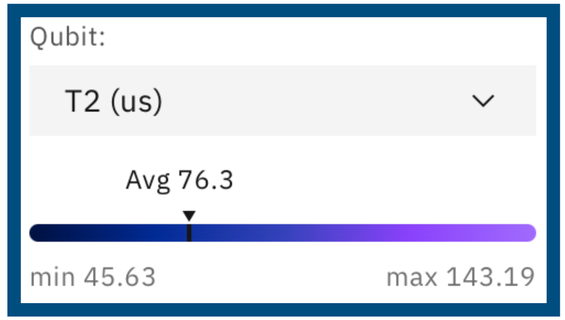
Quantum Metrics

Metric	Result
Max # of Gates	166
Max Gate Depth	110
Max # of Qubits	7
Max # of Readout Bits	1
Max # of Shots	100,000
Quantum Volume	32
CLOPS	2.6K

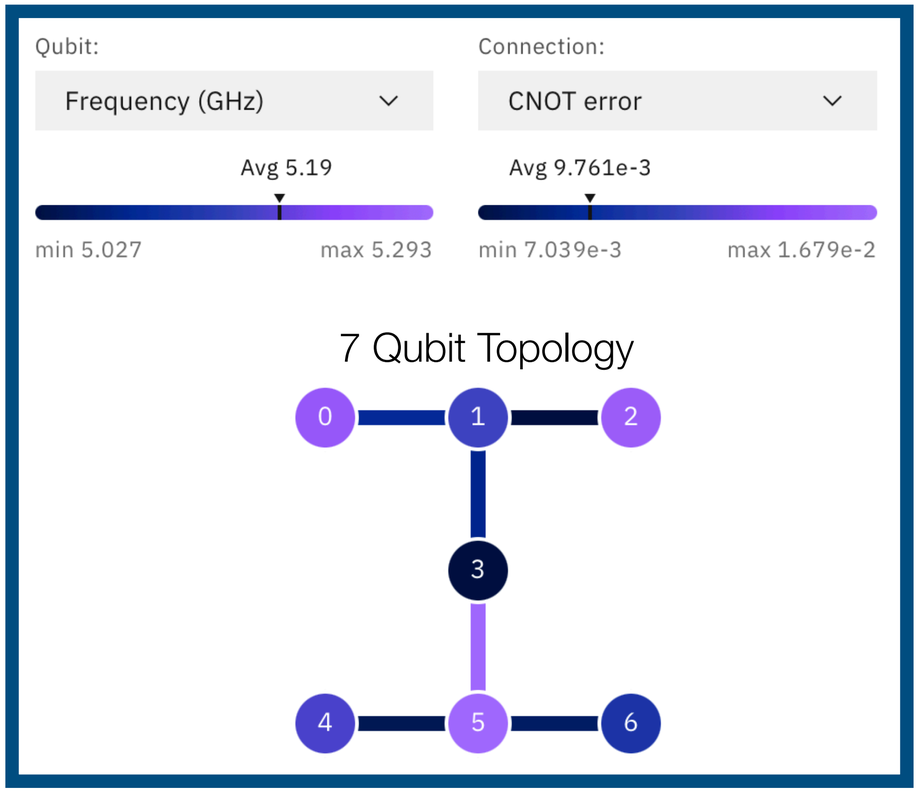
Extracted circuit metrics through Qiskit and QV/CLOPS through IBMQ

“ibm_nairobi” Hardware





* T1: Thermal relaxation time (larger is better)
* T2: Dephasing time (larger is better)



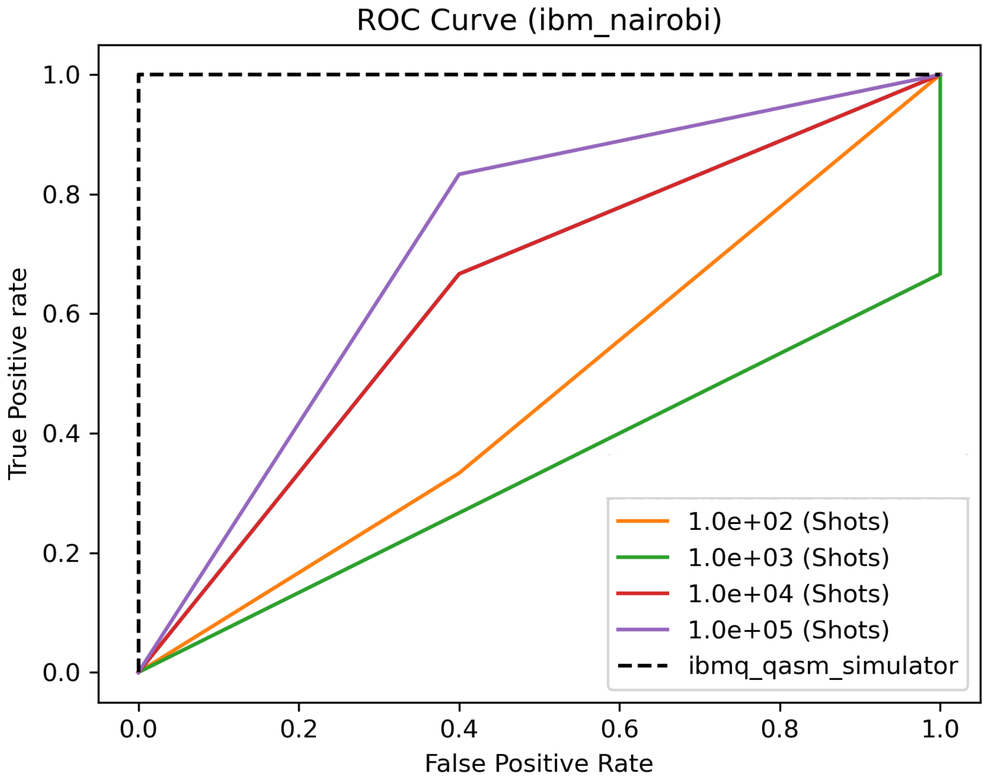
* CNOT Error: benchmark metric that reflects error in entanglement (lower is better)

Results

Classical Metrics

Metric	IBMQ simulator	IBM Nairobi
Precision	1	0.6
Recall	1	0.75
Accuracy	1	0.72
F1	1	0.67
AUC	1	0.7

* These results are from 100,000 shots (maximum allowed on backend)



Discussion

Discussion

Key Takeaways

- QML circuit is a perfect classifier when executed on simulator
- Circuit struggles on NISQ hardware due to high gate depth and noise characteristics
- Circuit needs to be executed for ~1e5 shots @ ~29 ms/shot to provide decent results
- Circuit performs poorly despite strict design considerations to limit NISQ hardware error,
    1. Test vectors were selected to be resource efficient
    2. Uses basis encoding which requires lowest # of quantum resources
    3. Shallow dynamic range (used the minimum bit accuracy needed for dataset computations)

Discussion

Next Steps

- Working with IBM Quantum to get access to backends with higher QV and qubit count to improve results
  - Error mitigation algorithms can be implemented with increased qubit count
  - Circuit depth can be decreased by increasing circuit width
- QML inference circuit that uses amplitude encoding is built and is currently being verified and tested. Should improve test vector dynamic range but will introduce more noise due to increased gate depth

Conclusion

Conclusion

Final Thoughts

- Mapping CML algorithms onto quantum circuits does not prove fruitful for practical ML
- Results from NISQ hardware are unreliable and fail to provide accurate results even with the simplest QML model being instantiated
- Research should focus on,
    1. Increasing the viable gate depth threshold of NISQ hardware
    2. Increasing decoherence times in physical qubits

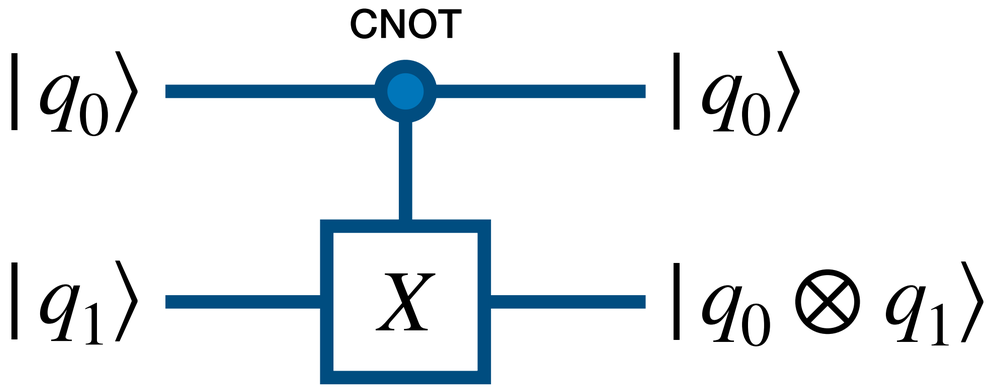
Thank you for listening!

Back-up Slides

Theoretical Background

Entanglement

- Entanglement gates are often referred to as “controlled” gate operations
- Consider the simplest example, the controlled NOT gate CNOT.



$\|q_0\rangle$	$\|q_1\rangle$	$\|q_0\rangle$	$\|q_0 \otimes q_1\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

CNOT Gate Truth Table

Theoretical Background

Confusion Matrix and Measures

- Precision: Ratio of correct positive predictions to the total predicted positives (larger is better)
- Recall: Ratio of correct positive predictions to the total positive examples (larger is better)
- Accuracy: Ratio of correctly predicted examples by the total number of examples (larger is better)
- F1: Score that is a function of Recall and Precision that is often used to compare the performance of two classifiers (larger is better)
- AUC: Area Under (ROC) Curve (larger is better)
- ROC: Shows the performance of a classification model at a ll possible classification thresholds.

Theoretical Background

Quantum Fourier transform (QFT)

- Similar to Discrete Fourier transform, QFT is responsible for transforming a state from the computational basis to the Fourier basis.

- This is done by encoding the state in the computational basis into the phase of the ‘ancilla’ register

- Inverse Quantum Fourier transform is used extensively in this research to read out the phase of ‘ancilla’ registers that contain various computational results.

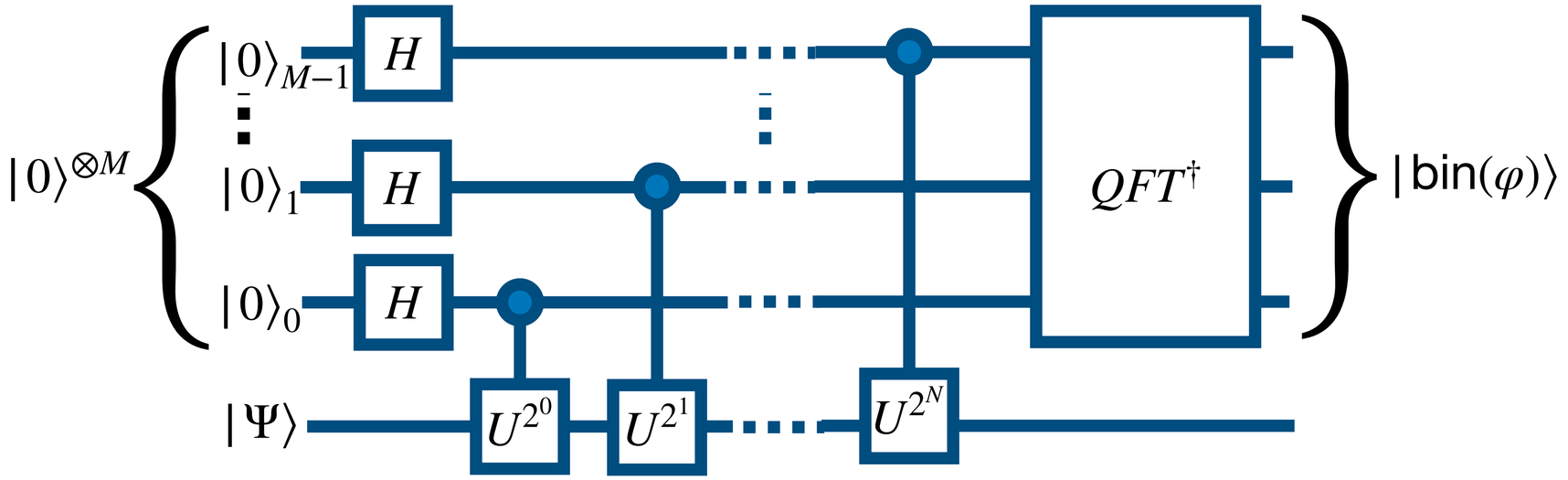
- However, this algorithm is expensive in terms of resource and is primarily where the gate depth and count comes from.

Theoretical Background

Quantum Phase Estimation

- Given a Unitary operator U, this algorithm estimates the φ in $U|\Psi\rangle = e^{2\pi i\varphi}|\Psi\rangle$

- The ‘ancilla’ register contains M ‘counting’ qubits



Theoretical Background

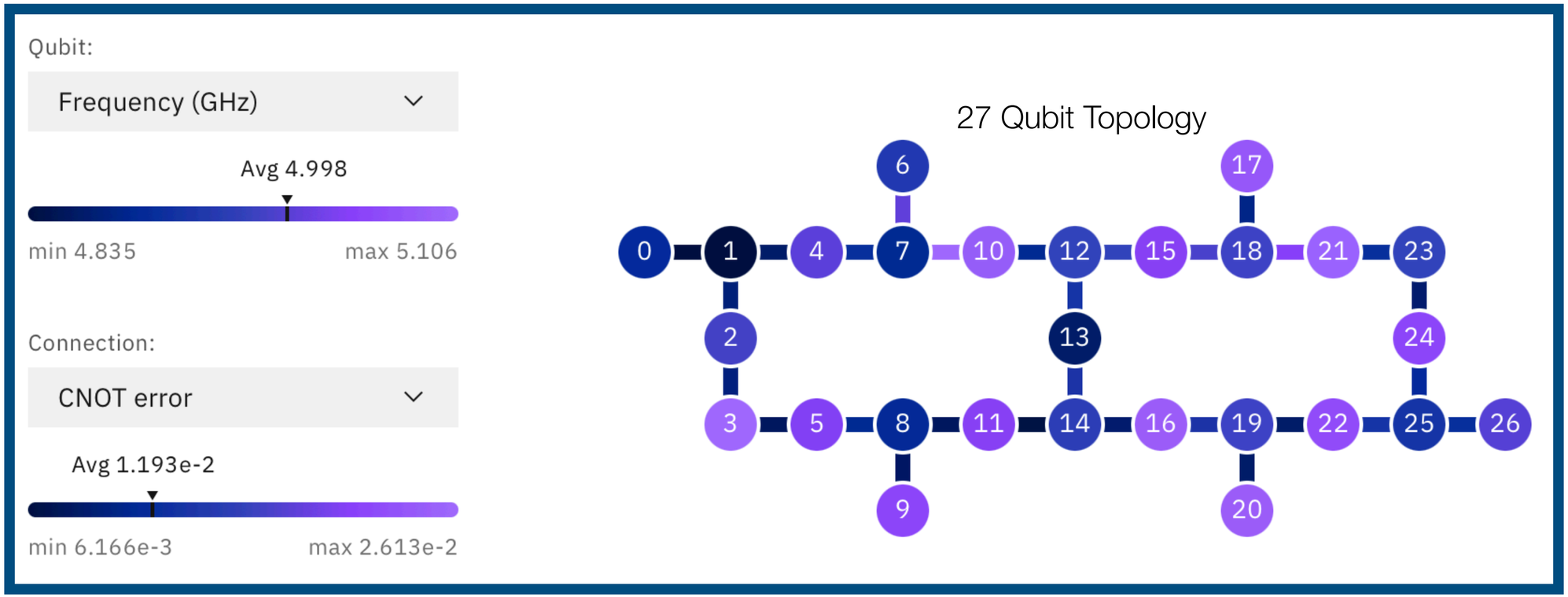
Dynamic Range Limitations

- Using signed integers, 2^N bits can represent 2^{N-1} positive and negative numbers
- Trained weight vector was scaled down by 10 in order to ensure $|\mathbf{x}_{\text{test},i} \cdot \mathbf{w} + b| < 2$ so that only two bits were required for the inner product calculation
- Rescaled back to normal during sigmoid computation
- Since $0 < \sigma(z) < 1 \ \forall z \in (-\infty, \infty)$ we only need one qubit need for sigmoid computation

Results

Quantum Metrics

“ibmq_montreal” Backend



27 Qubit, 128 Quantum Volume, 2K CLOPS

Results

Classical Metrics

Metric	ibmq_simulator	ibmq_montreal
Precision	1	0.67
Recall	1	1
Accuracy	1	0.83
F1	1	0.8
AUC	1	0.83

* Results from 32,000 shots
(maximum number of shots allowed on backend)