Machine Learning on Quantum Computing: From Classical to Quantum

(Week 4 – Session 2)

Weiwen Jiang, Ph.D.

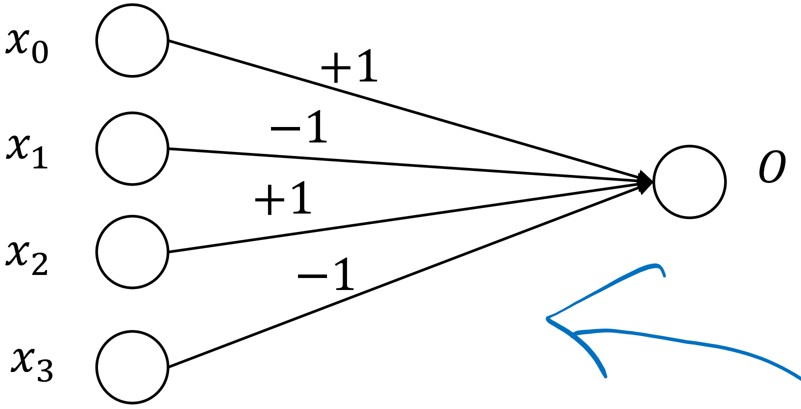
Postdoc Research Associate

Department of Computer Science and Engineering

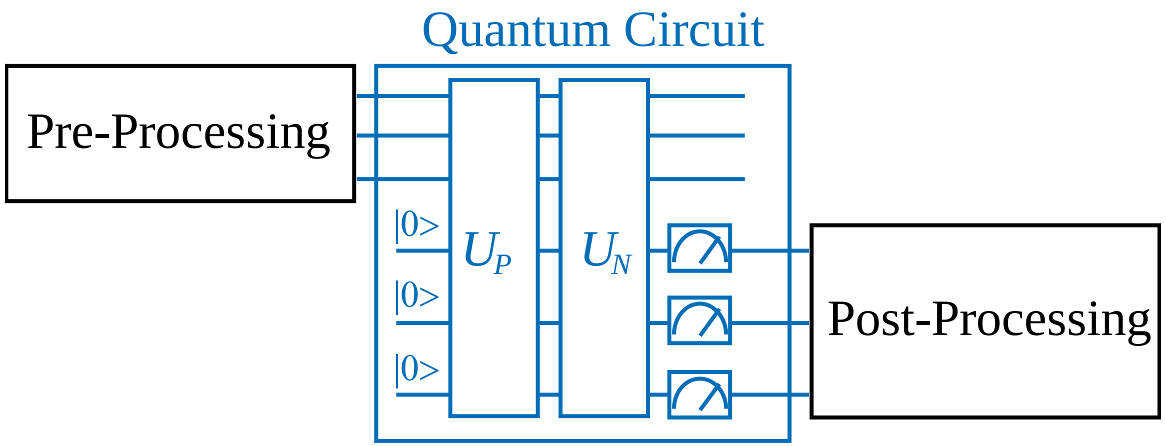
University of Notre Dame

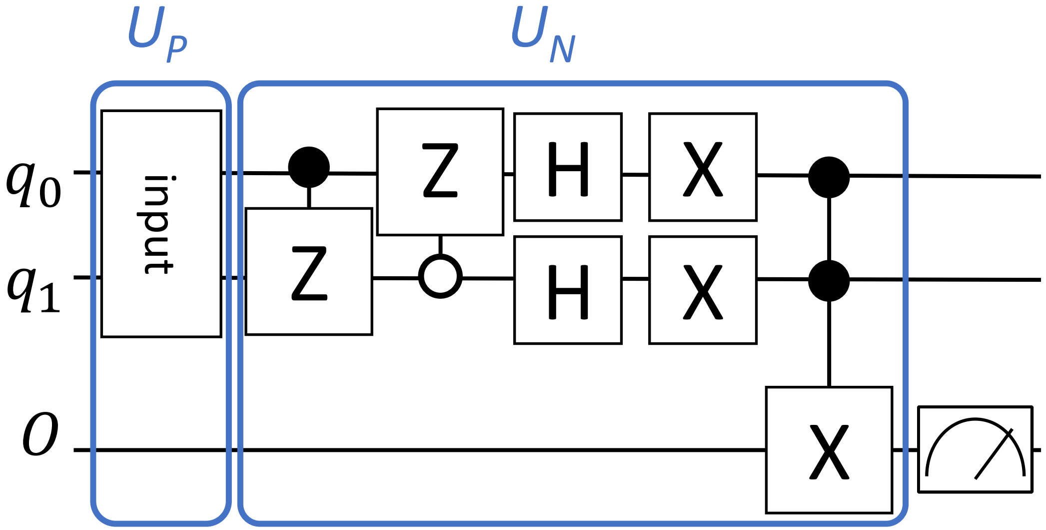
wjiang2@nd.edu | https://wjiang.nd.edu

Review of Previous Session --- Goal 1: Implementing Perceptron Correctly!



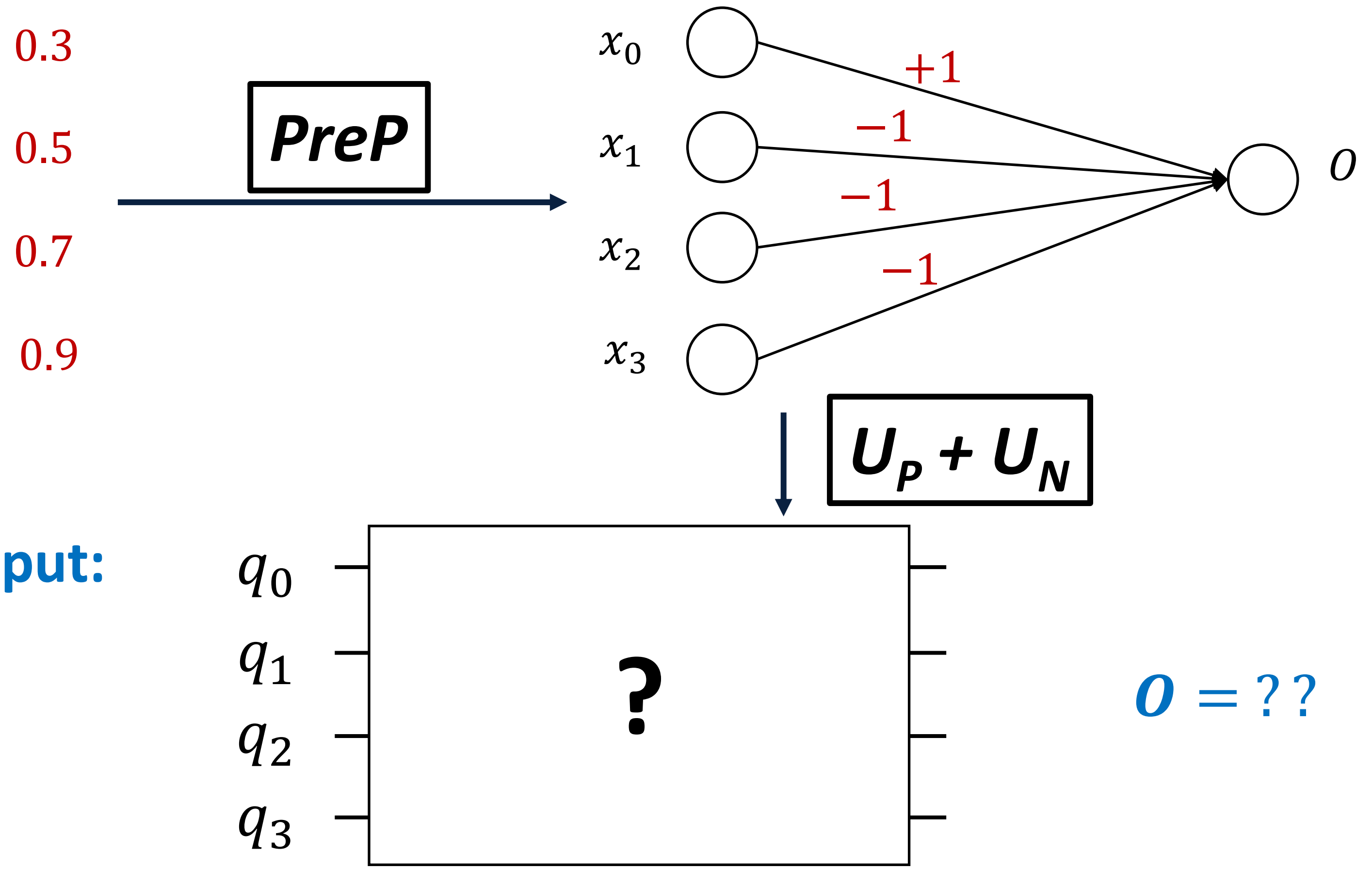
$$O = \frac{1}{4}\left(\sum_{i\in[0,4)} I_i \times W_i\right)^2$$





Have a Try on $PreP + U_P + U_N + M + PostP$!

Given inputs and weights

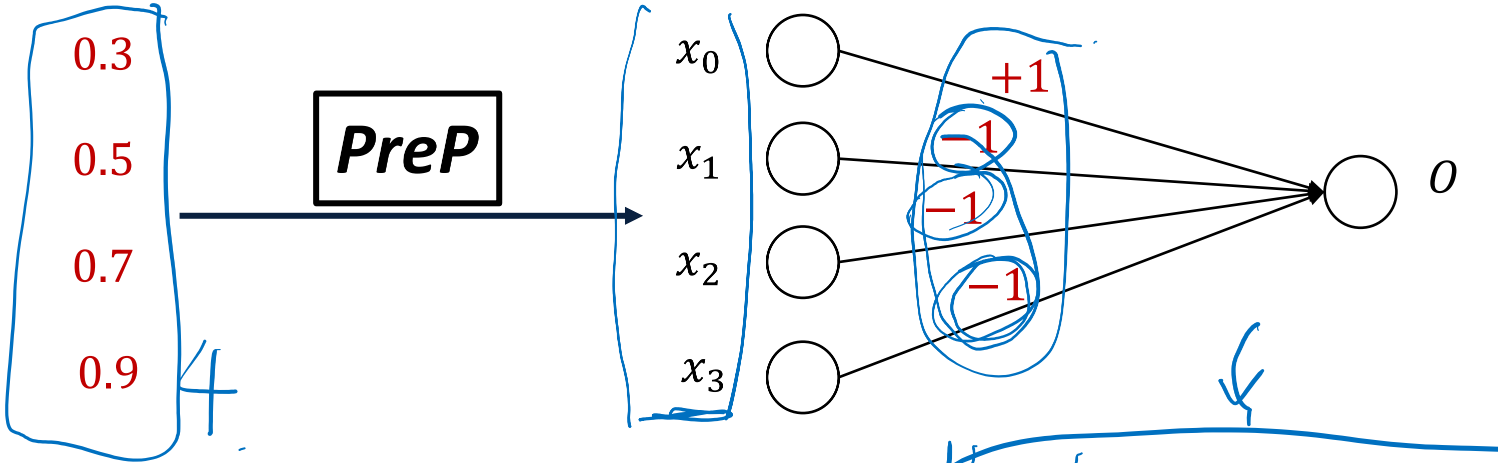


Output:

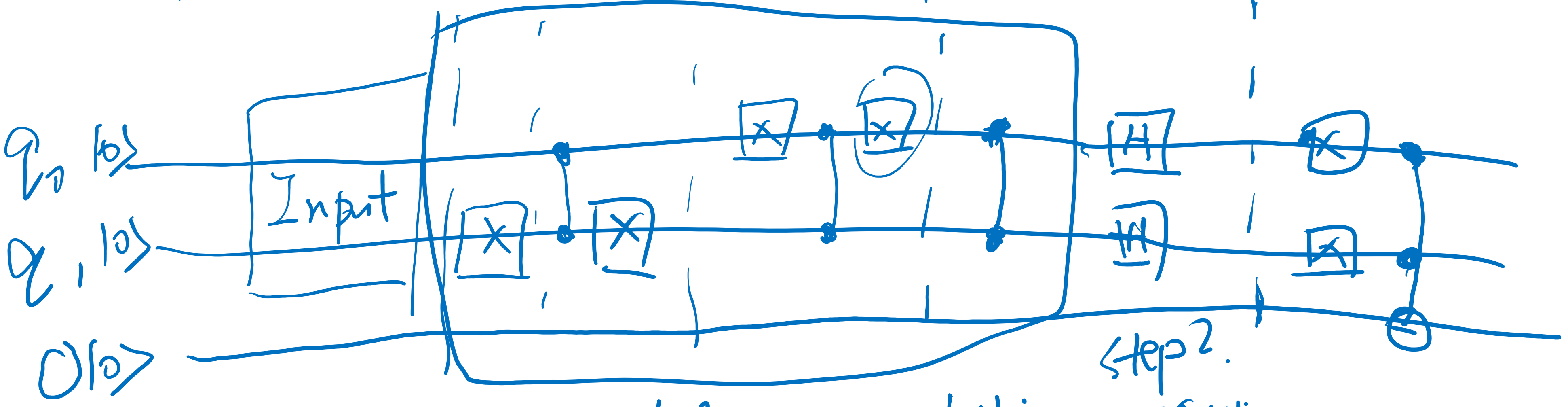
$O = ??$

Have a Try on $PreP + U_P + U_N + M$ + PostP !

Given inputs and weights



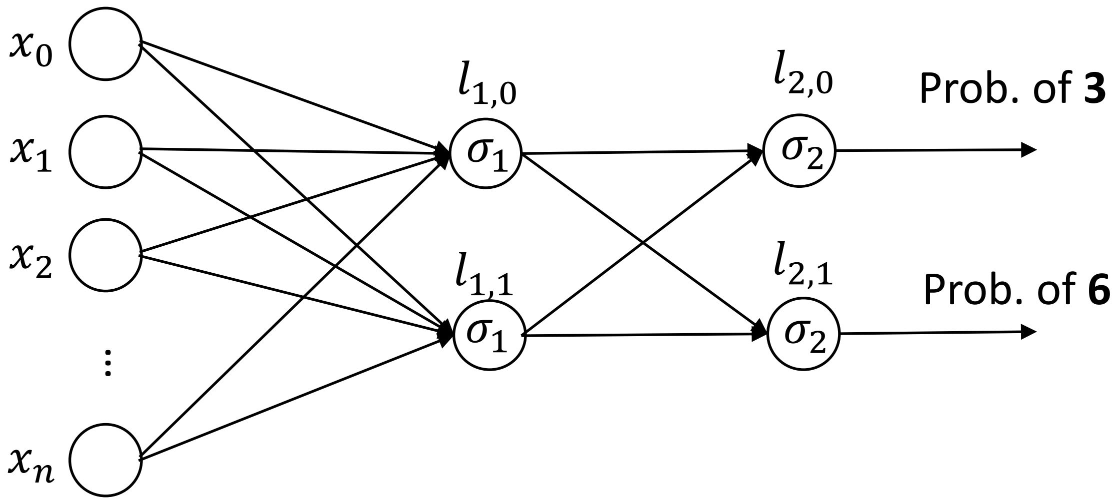
Qiskit Tutorial

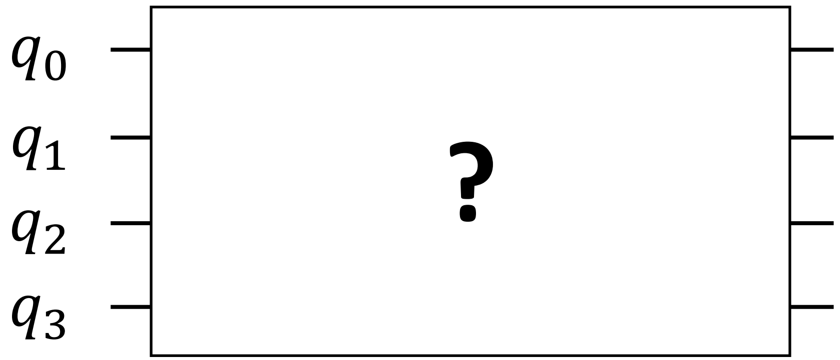


Step 1: U_N. $x_i \cdot w_i$

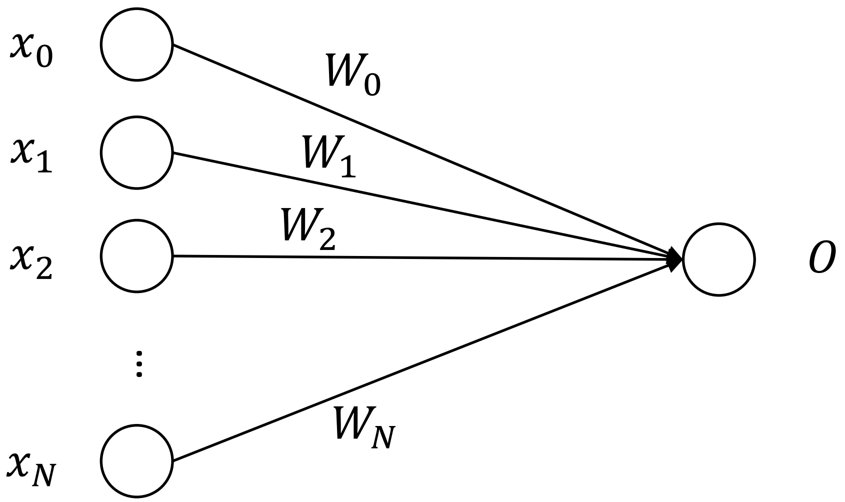
Step 2. $\frac{\sum x_i w_i}{\sqrt{||x||}}$

Goal 2: Implementing Feedforward Net w/ Non-Linearity, w/o Measurement!





Goal 3: Implementing Perceptron to Quantum Efficiently!



$$O = \delta\left(\sum_{i\in[0,N)} x_i \times W_i\right)$$

where δ is a quadratic non-linear function

Neural Computation with input size of 2^N on classical computer

Operation: Multiplication: $\boldsymbol{O(N)}$

Accumulation: $\boldsymbol{O(N)}$

Neural Computation with input size of 2^N on quantum computer

Quantum Gates: $\boldsymbol{O(ploylogN)}$, say $\boldsymbol{O(log^2 n)}$?

Organization of Quantum Machine Learning Sessions

- **Background and Motivation** *[w4s1]*
  - What is machine learning
  - Why using quantum computer
  - Our goals
- **General Framework and Case Study[2] (Tutorial on GitHub[3])** *[w4s1- w4s2]*
  - Implementing neural network accelerators: from classical to quantum
  - A case study on MNIST dataset
- **Optimization towards Quantum Advantage[1] (Nature Communications)** *[w4s2]*
  - The existing challenges
  - The proposed co-design framework: QuantumFlow

G1

G2, G3

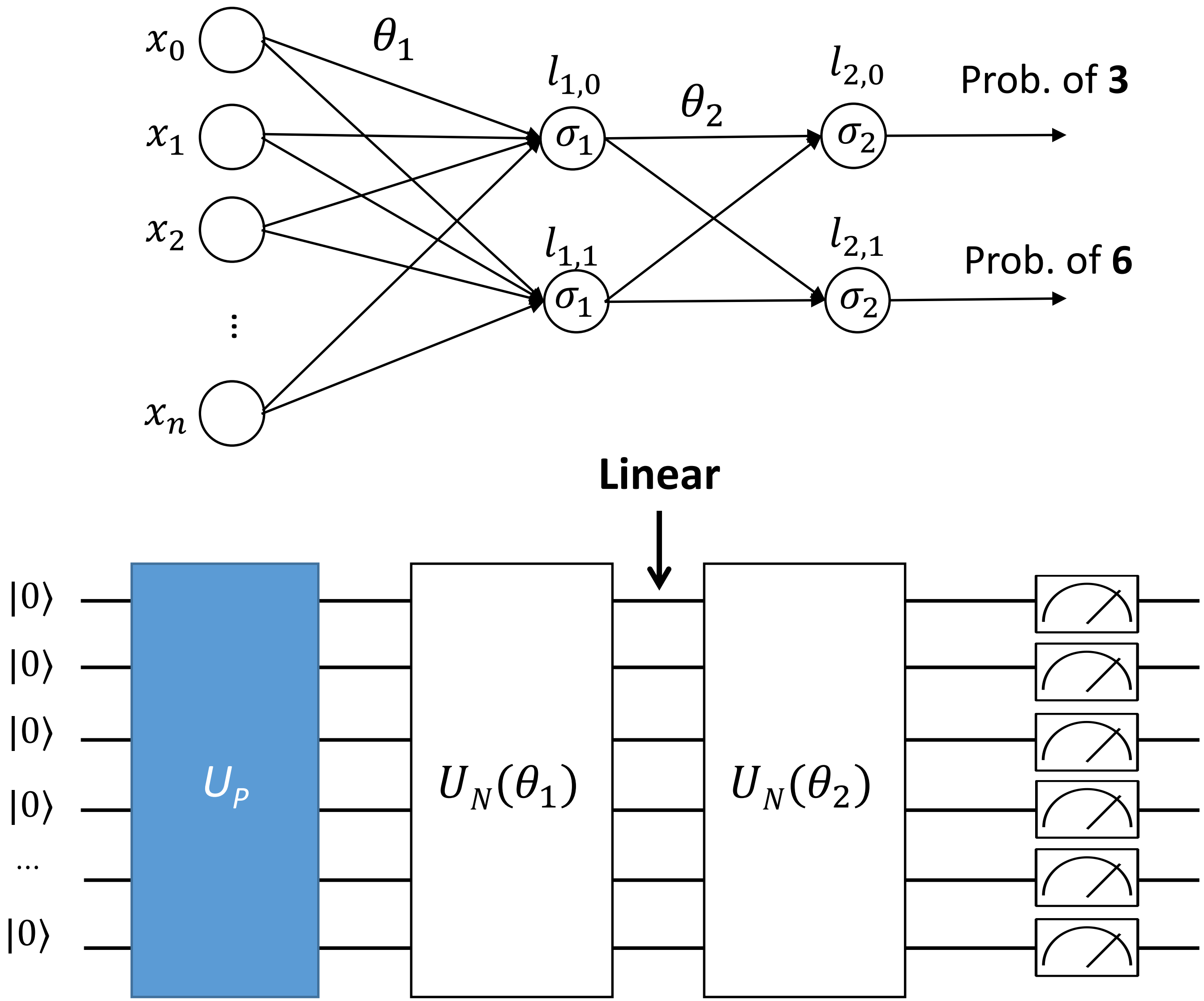
References:

[1] W. Jiang, et al. A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage, Nature Communications

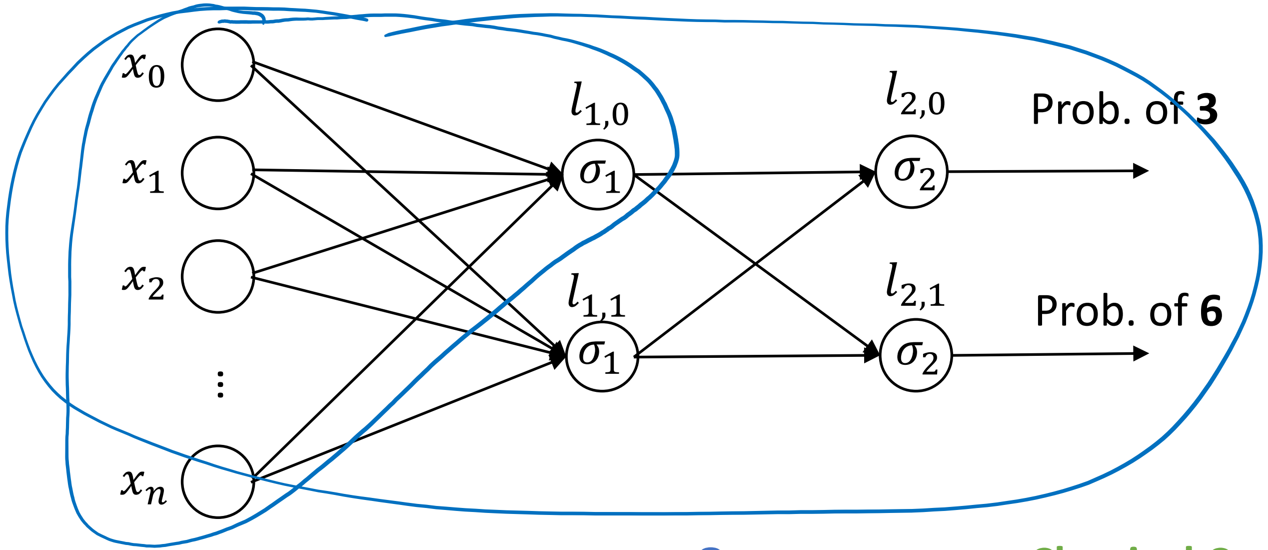
[2] W. Jiang, et al. When Machine Learning Meets Quantum Computers: A Case Study, ASP-DAC'21

[3] W. Jiang, Github Tutorial on Implementing Machine Learning to Quantum Computer using IBM Qiskit

Challenge 1: Non-linearity is Needed, But Difficult in Quantum Circuit



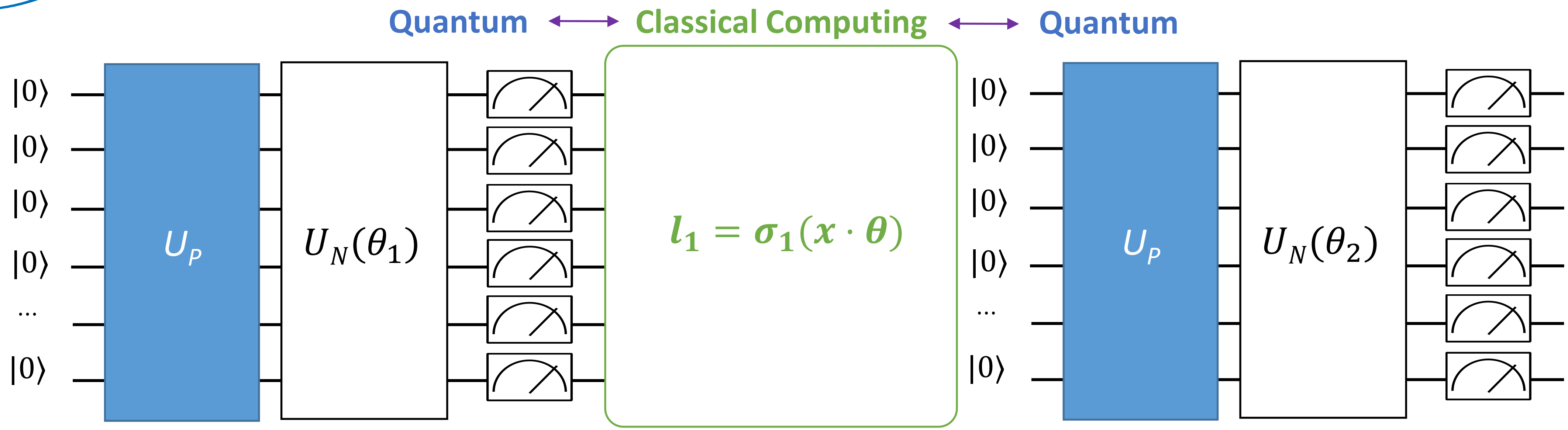
Challenge 2: Quantum-Classical Interface is Expensive



Ref [1]

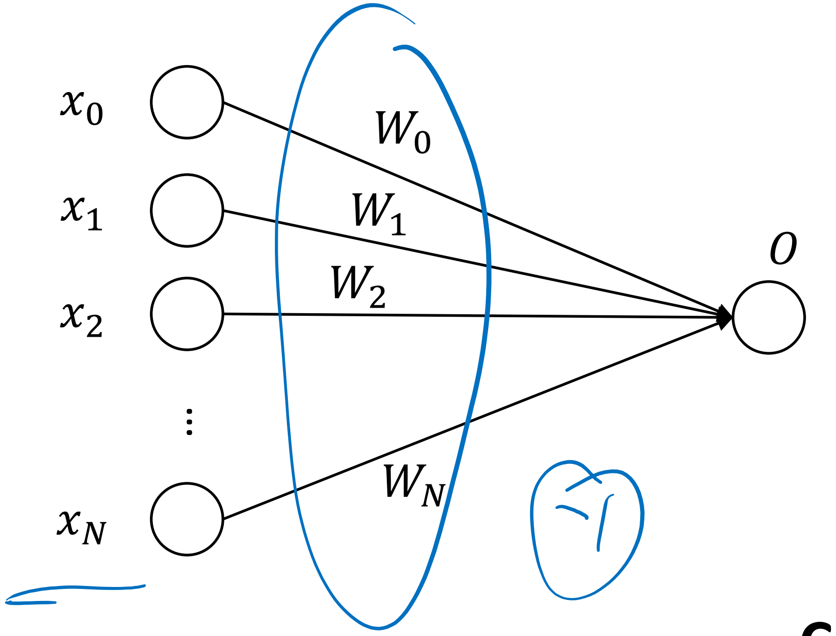
Table 2 Complexity of each step in hybrid quantum-classical computing for deep neural network with U-LYR.

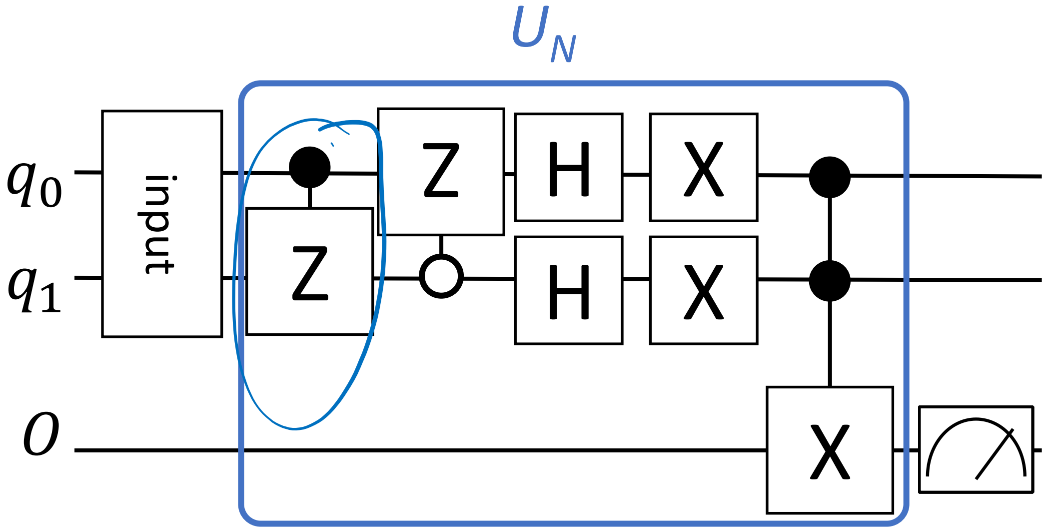
Complexity	State-preparation	Computation	Measurement
Depth (T)	$O(d \cdot \sqrt{n})$	$O(d \cdot \log^2 n)$	$O(d)$
Qubits (S)	$O(n)$	$O(n \cdot \log n)$	$O(n \cdot \log n)$
Cost (TS)	$O(d \cdot n^{\frac{3}{2}})$	$O(d \cdot n \cdot \log^3 n)$	$O(d \cdot n \cdot \log n)$
Total (TS)	$O(d \cdot n^{\frac{3}{2}})$ dominate		



[1] W. Jiang, et al. A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage, Nature Communications

Challenge 3: High Complexity in the Previous Design





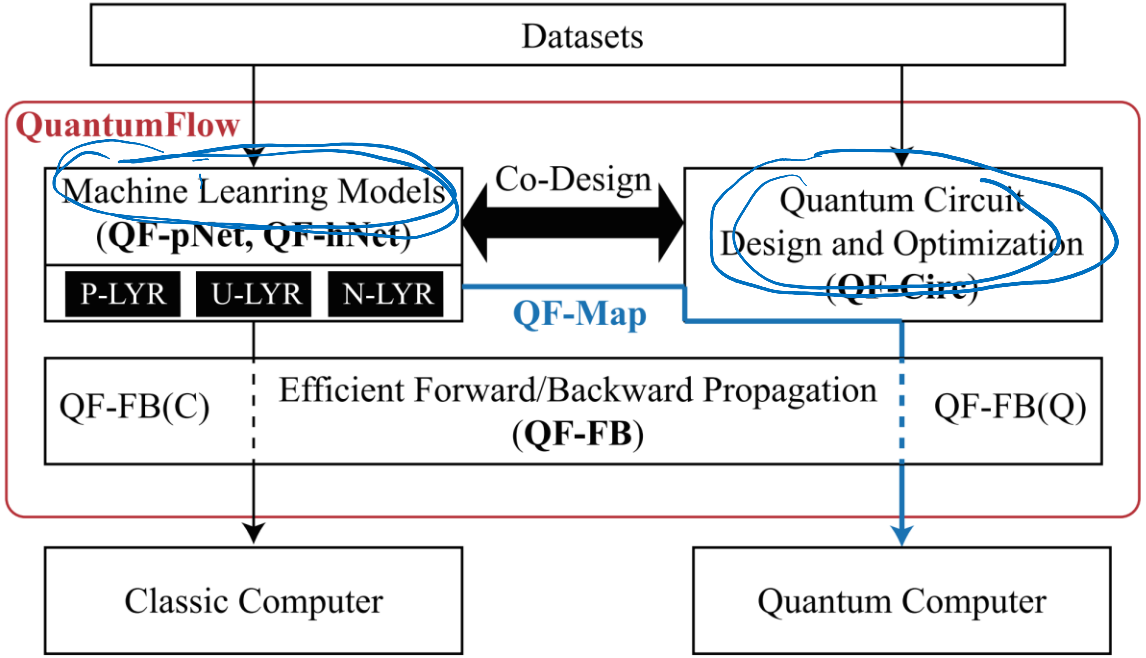
Cost Complexity O(N)

Classical Computing		
	No Parallelism	Full Parallelism
Time (T)	O(N)	O(1)
Space (S)	O(1)	O(N)
Cost (TS)	O(N)	O(N)

Quantum Computing		
	Previous Design	Optimization
Circuit Depth (T)	O(N)	??? O(polylog
Qubits (S)	O($\log N$)	~~O(N)~~ O(log N)
Cost (TS)	O($N \cdot \log N$)	**target** O(ploylog N)

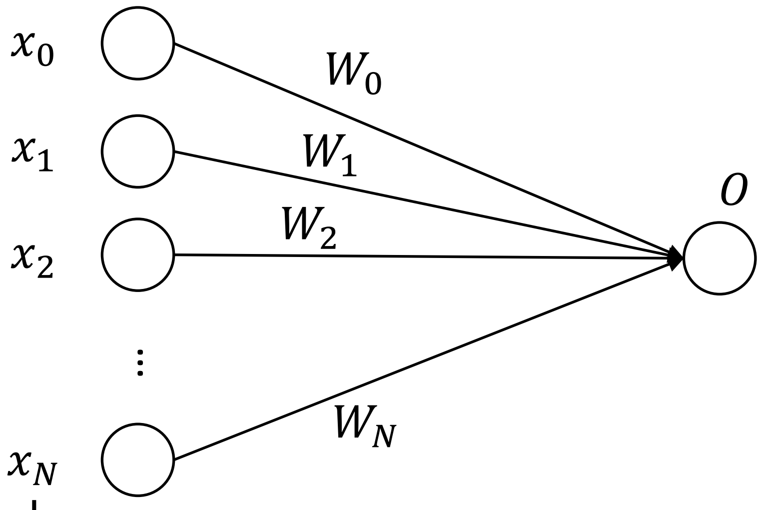
Co-Design Framework

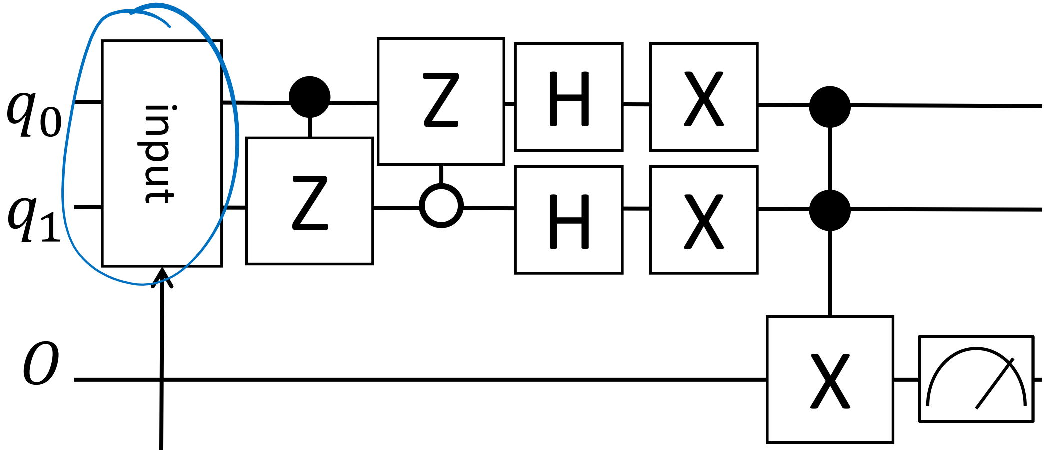
QuantumFlow



[1] W. Jiang, et al. A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage, Nature Communications

Design Direction 1: NN → Quantum Circuit





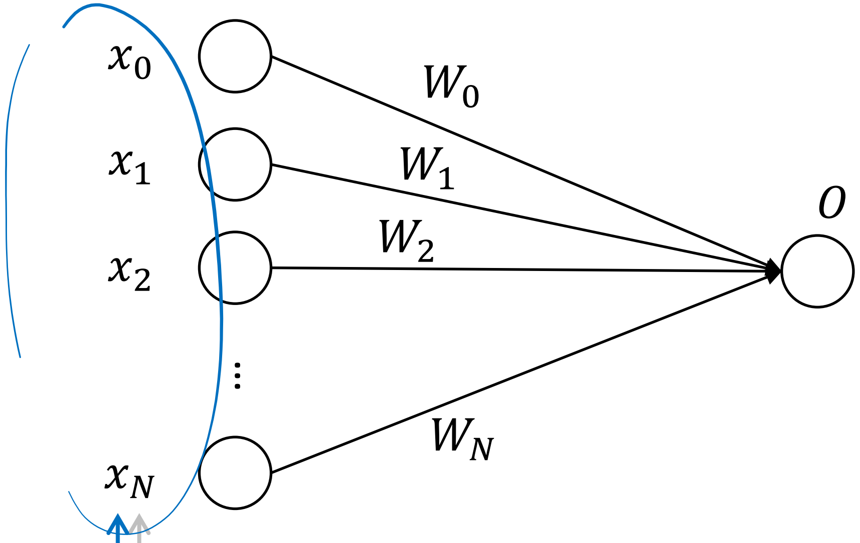
Digital Numbers → Amplitude Encoding

Converting to Unitary Matrix

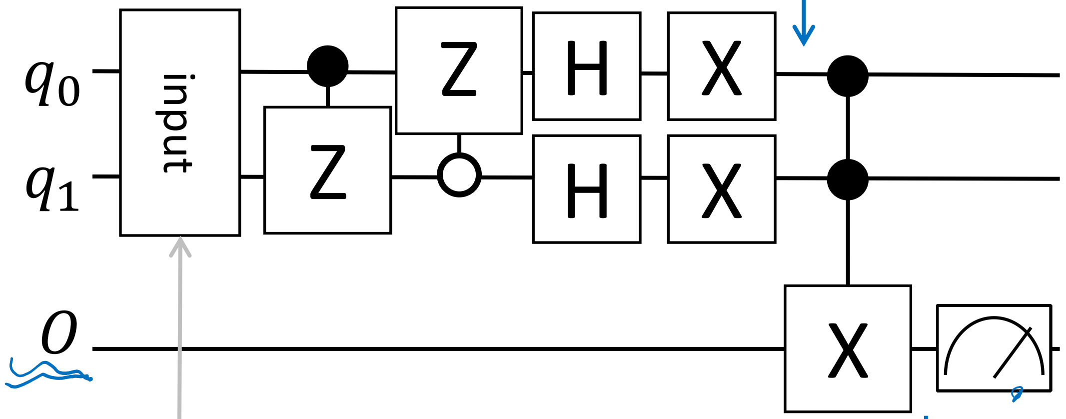
N numbers are encoded to $\log N$ qubits

Design Direction 2: Quantum Circuit → NN

U_θ.



Amplitude of $|00\rangle$ is $\boldsymbol{n}$



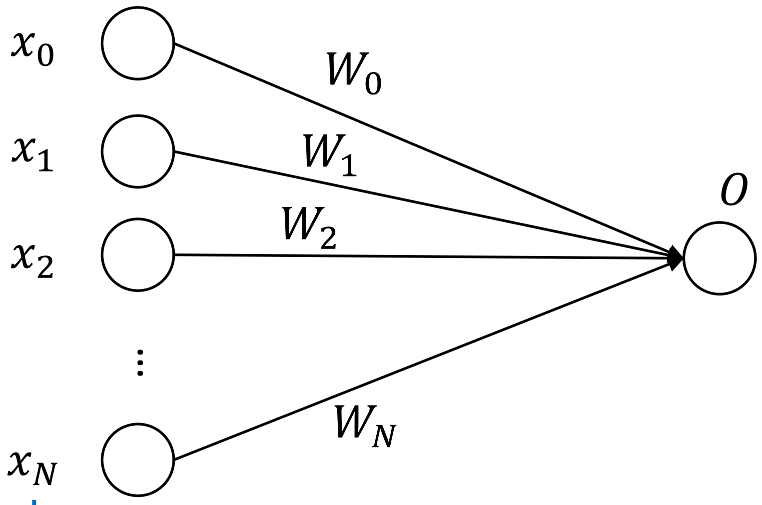
Converting to Unitary Matrix

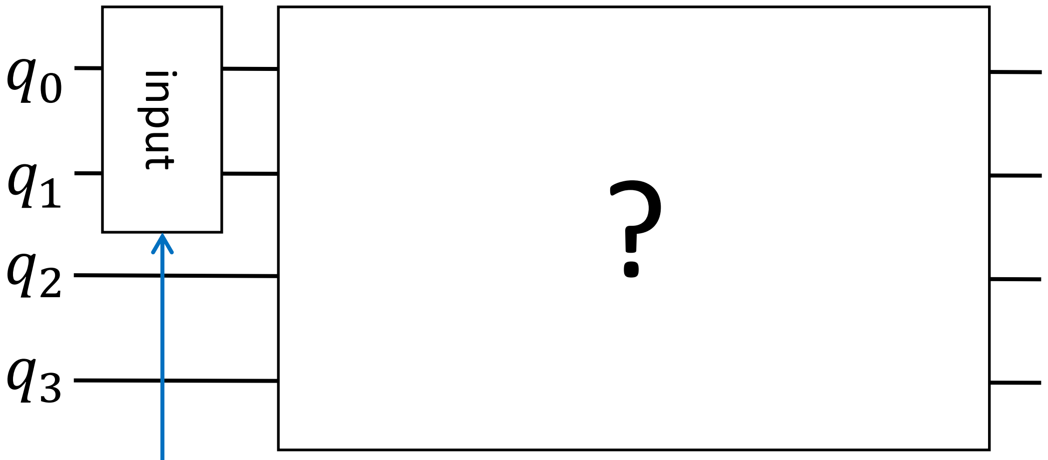
Digital Numbers → Amplitude Encoding

N numbers are encoded to $\log N$ qubits

Random Variable ← Probability of $O = |1\rangle$ is $\boldsymbol{n}^2$

Design Direction 3: NN → Quantum Circuit





Random Variable → **Rotation Gate, say Ry** → Angle Encoding

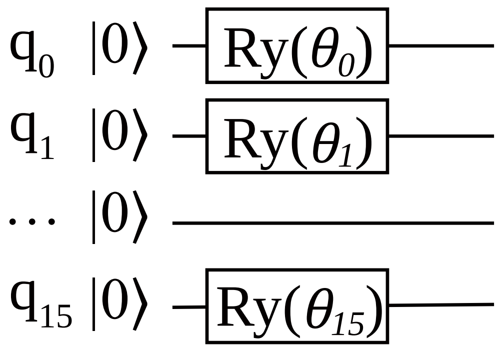
N **numbers are encoded to** ***N*** **qubits**

0.0	0.9	0.0	0.0
0.0	0.1	0.0	0.0
0.0	1.0	0.5	0.5
0.0	0.0	0.0	0.0

→

r.v.	-1	+1
x_0	0.0	1.0
x_1	0.9	0.1
…	…	
x_{15}	0.0	1.0

→



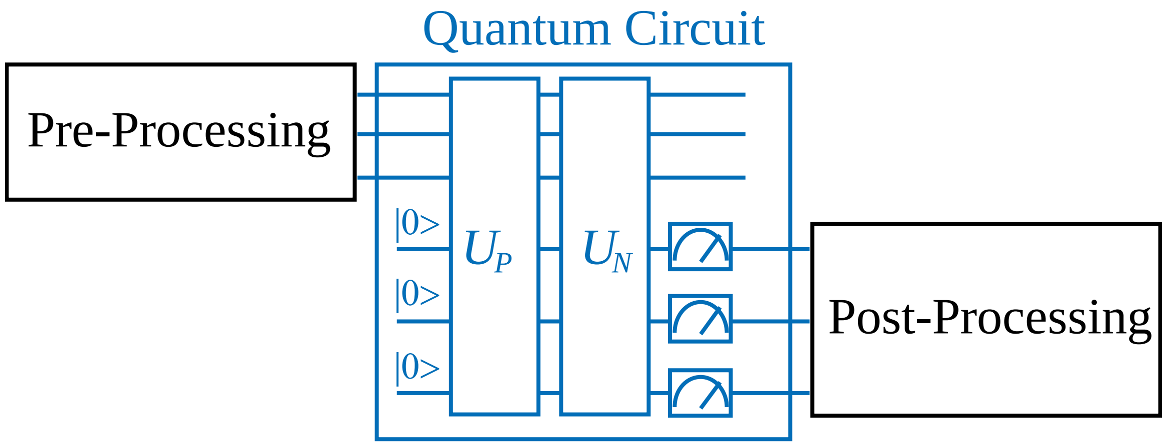
Amp.

2^N → N qubit.

→ 2^N qubit

$1 \leftrightarrow 1$

Still Apply Our Framework to Design Quantum Circuit



(1) Data Pre-Processing (*PreP*)

(2) HW/Quantum Acceleration

(2.1) rvU_p Quantum-State-Preparation

(2.2) rvU_N Quantum Neural Computation

(2.3) M Measurement

(3) Data Post-Processing (*PostP*)

rvU_P --- Data Encoding / Quantum State Preparation

- **Given:** A vector of input data, ranging from [0,1] <u>(do scaling in *PreP* if range out of [0,1])</u>
- **Do:** Applying rotation gate to encode each data to one qubits
- **Output:** A quantum circuit, where the probability of each qubit to be $|1\rangle$ is the same as the corresponding input data

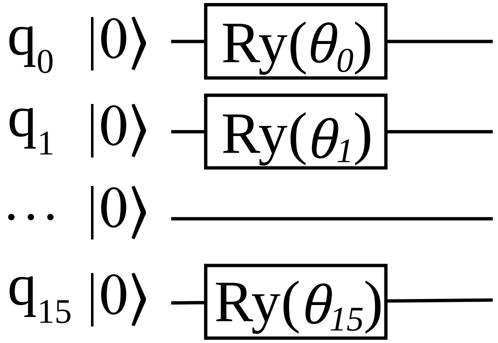
0.0	0.9	0.0	0.0
0.0	0.1	0.0	0.0
0.0	1.0	0.5	0.5
0.0	0.0	0.0	0.0

→

$|r\rangle$

r.v.	-1	+1
x_0	0.0	1.0
x_1	0.9	0.1
…	…	
x_{15}	0.0	1.0

→



Determination of θ_i:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + (\cos\phi + i\cdot\sin\phi)\cdot\sin\frac{\theta}{2}|1\rangle$$

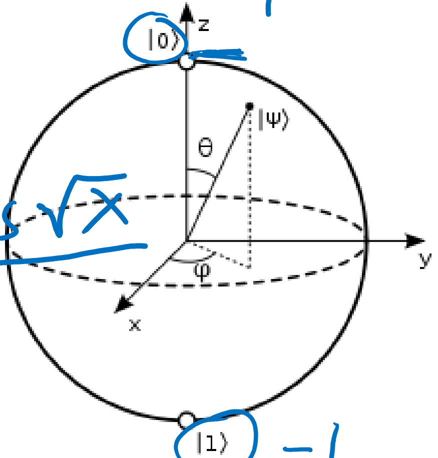
$$\theta_i = 2 \times \arcsin(\sqrt{x_i})$$

$\sin\frac{\theta}{2} = \sqrt{x}$

$\theta/2 = \arcsin\sqrt{x}$

$\theta = 2 * \arcsin\sqrt{x}$

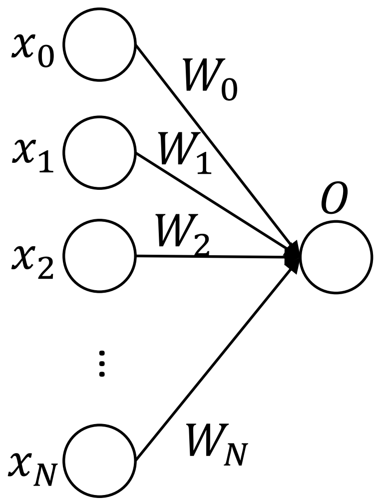
+1



-1

[1] W. Jiang, et al. [A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage](), Nature Communications

rvU_N --- Neural Computation



- **Given:** (1) A circuit with encoded input data $\boldsymbol{x}$; (2) the trained binary weights $\boldsymbol{w}$ for one neural computation, which will be associated to each data.
- **Do:** Place quantum gates on qubits, such that it performs $\frac{(\boldsymbol{x}*\boldsymbol{w})^2}{\|\boldsymbol{x}\|^2}$, where $\boldsymbol{x}$ are random variables

r.v.

Target: $O = \left[\frac{\sum_i (x_i \times w_i)}{\|\boldsymbol{x}\|}\right]^2$

- **Assumption 1:** Parameters/weights (W_0 --- W_N) are binary weight, either +1 or -1
- **Assumption 2:** The weight $W_0 = +1$, otherwise we can use $-\boldsymbol{w}$ (quadratic func.)

Step 1: $m_i = x_i \times w_i$

Step 2: $n = \left[\frac{\sum_i (m_i)}{\|\boldsymbol{x}\|}\right]$

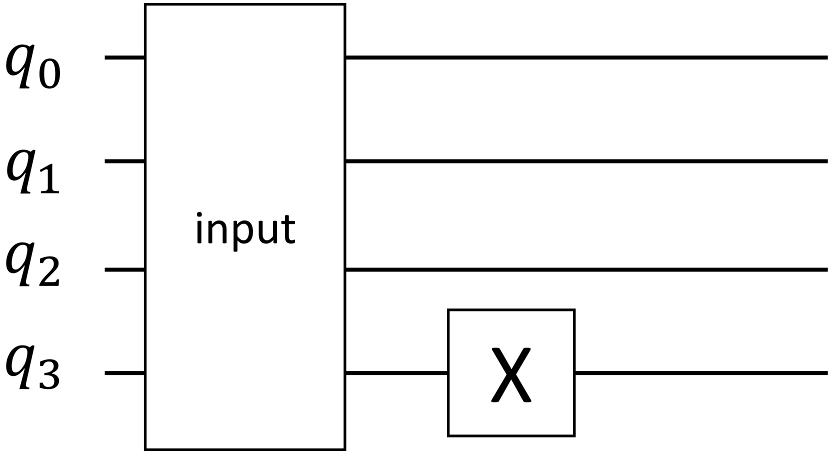
Step 3: $O = n^2$

rvU_N --- Neural Computation: Step 1

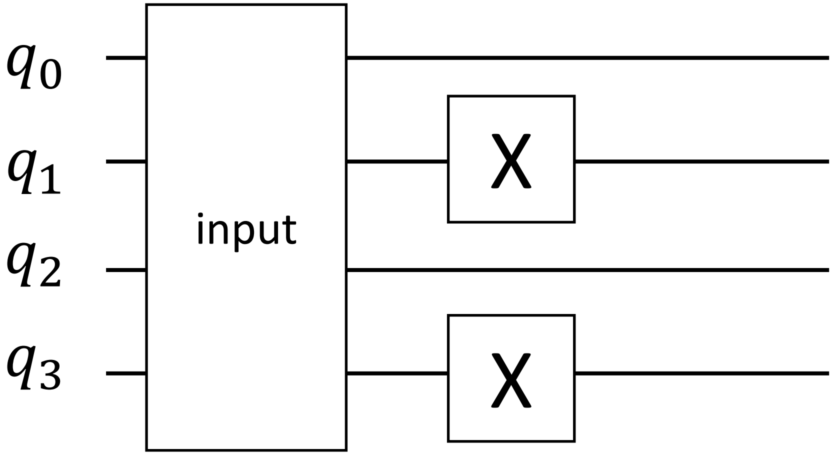
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 4 qubits

$$\boldsymbol{w} = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$



$$\boldsymbol{w} = \begin{bmatrix} +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}$$



	-1	$+1$
X_3	p	q.

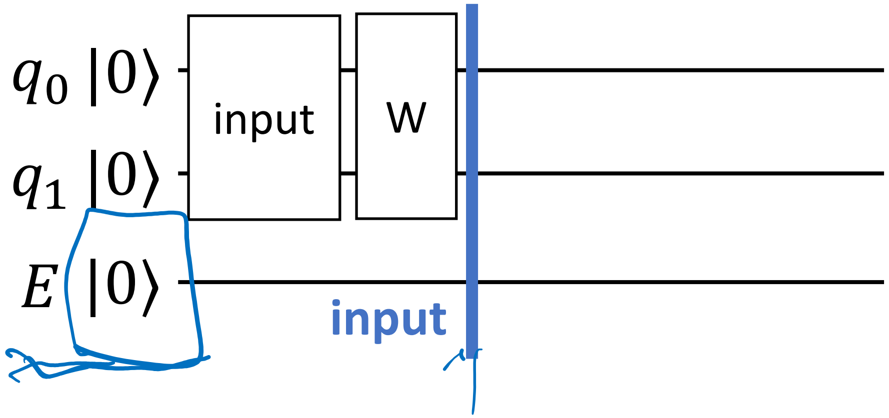
$X_3 \times -1$

	$+1$	-1
$-X_3$	p	q.

rvU_N --- Neural Computation: Step 2

Step 2: $n = \left[\frac{\sum_i (m_i)}{\|x\|}\right]$

EX: 2 input data on 2 qubits



r.v.	-1 (|1⟩)	+1 (|0⟩)
m_0	p_0	q_0
m_1	p_1	q_1

r.v.	-1	0	+1
n	p_0p_1	$p_0q_1 + p_1q_0$	q_0q_1

m_0	m_1	$\frac{m_0+m_1}{2}$
-1	-1	-1
-1	+1	0
+1	-1	0
+1	+1	+1

Input

$\sqrt{q_1q_0}$	$\|00\rangle$
$\sqrt{q_1p_0}$	$\|01\rangle$
$\sqrt{p_1q_0}$	$\|10\rangle$
$\sqrt{p_1p_0}$	$\|11\rangle$

Input

$\sqrt{q_1q_0}$	$\|000\rangle$
$\sqrt{q_1p_0}$	$\|001\rangle$
$\sqrt{p_1q_0}$	$\|010\rangle$
$\sqrt{p_1p_0}$	$\|011\rangle$
0	$\|100\rangle$
0	$\|101\rangle$
0	$\|110\rangle$
0	$\|111\rangle$

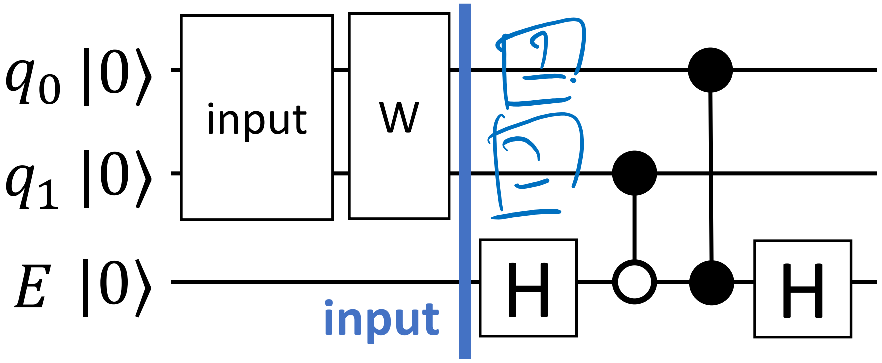
rvU_N --- Neural Computation: Step 2

Step 2: $n = \left[\frac{\sum_i (m_i)}{\|x\|}\right]$

EX: 2 input data on 2 qubits

r.v.	-1 ($\|1\rangle$)	+1 ($\|0\rangle$)
m_0	p_0	q_0
m_1	p_1	q_1

r.v.	-1	0	+1
n	p_0p_1	$p_0q_1 + p_1q_0$	q_0q_1



$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes I_4$

$\begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix}$

Input / **IIH**

Input		IIH	
$\sqrt{q_1q_0}$	$\|000\rangle$	$\sqrt{q_1q_0}/\sqrt{2}$	$\|000\rangle$
$\sqrt{q_1p_0}$	$\|001\rangle$	$\sqrt{q_1p_0}/\sqrt{2}$	$\|001\rangle$
$\sqrt{p_1q_0}$	$\|010\rangle$		$\|010\rangle$
$\sqrt{p_1p_0}$	$\|011\rangle$		$\|011\rangle$
0	$\|100\rangle$	$\sqrt{q_1q_0}/\sqrt{2}$	$\|100\rangle$
0	$\|101\rangle$		$\|101\rangle$
0	$\|110\rangle$		$\|110\rangle$
0	$\|111\rangle$		$\|111\rangle$

Input / **IIH**

Input		IIH	
$\sqrt{q_1q_0}$	$\|000\rangle$	$\sqrt{q_1q_0}/\sqrt{2}$	$\|000\rangle$
0	$\|100\rangle$	$\sqrt{q_1q_0}/\sqrt{2}$	$\|100\rangle$
$\sqrt{q_1p_0}$	$\|001\rangle$	$\sqrt{q_1p_0}/\sqrt{2}$	$\|001\rangle$
0	$\|101\rangle$	$\sqrt{q_1p_0}/\sqrt{2}$	$\|101\rangle$
$\sqrt{p_1q_0}$	$\|010\rangle$	$\sqrt{p_1q_0}/\sqrt{2}$	$\|010\rangle$
0	$\|110\rangle$	$\sqrt{p_1q_0}/\sqrt{2}$	$\|110\rangle$
$\sqrt{p_1p_0}$	$\|011\rangle$	$\sqrt{p_1p_0}/\sqrt{2}$	$\|011\rangle$
0	$\|111\rangle$	$\sqrt{p_1p_0}/\sqrt{2}$	$\|111\rangle$

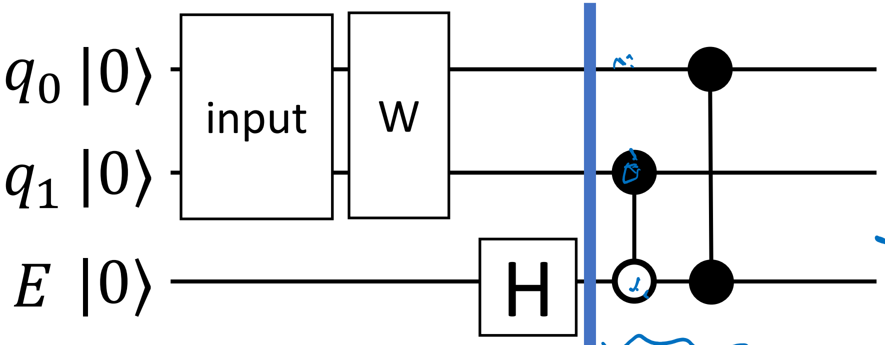
rvU_N --- Neural Computation: Step 2

Step 2: $n = \left[\frac{\sum_i (m_i)}{\|x\|}\right]$

r.v.	-1 ($\|1\rangle$)	+1 ($\|0\rangle$)
m_0	p_0	q_0
m_1	p_1	q_1

r.v.	-1	0	+1
n	$p_0 p_1$	$p_0 q_1 + p_1 q_0$	$q_0 q_1$

EX: 2 input data on 2 qubits



After ave.

IIH

$+\sqrt{q_1 q_0}/\sqrt{2}$	$\|000\rangle$
$+\sqrt{q_1 q_0}/\sqrt{2}$	$\|100\rangle$
$+\sqrt{q_1 p_0}/\sqrt{2}$	$\|001\rangle$
$-\sqrt{q_1 p_0}/\sqrt{2}$	$\|101\rangle$
$-\sqrt{p_1 q_0}/\sqrt{2}$	$\|010\rangle$
$+\sqrt{p_1 q_0}/\sqrt{2}$	$\|110\rangle$
$-\sqrt{p_1 p_0}/\sqrt{2}$	$\|011\rangle$
$-\sqrt{p_1 p_0}/\sqrt{2}$	$\|111\rangle$

m_0	m_1
0	0
+1	+1
-1	+1

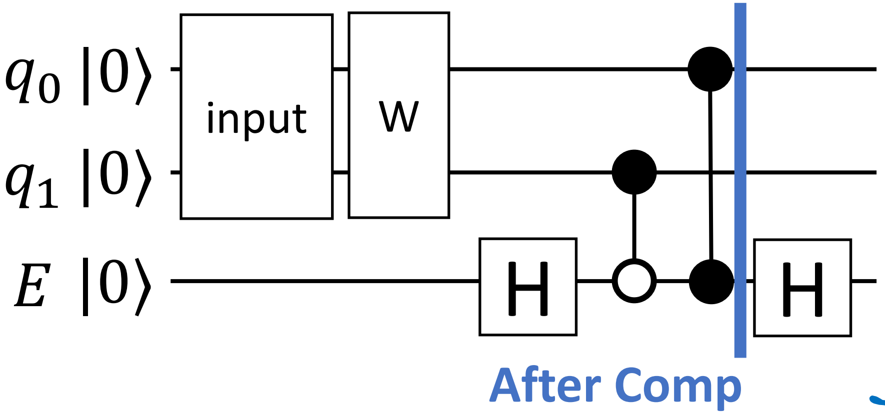
rvU_N --- Neural Computation: Step 2

Step 2: $n = \left[\frac{\sum_i (m_i)}{\|x\|}\right]$

r.v.	-1 ($\|1\rangle$)	+1 ($\|0\rangle$)
m_0	p_0	q_0
m_1	p_1	q_1

r.v.	-1	0	+1
n	p_0p_1	$p_0q_1 + p_1q_0$	q_0q_1

EX: 2 input data on 2 qubits



$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

After Comp

$\sqrt{q_1q_0}/\sqrt{2}$	$\|000\rangle$
$\sqrt{q_1q_0}/\sqrt{2}$	$\|100\rangle$
$+\sqrt{q_1p_0}/\sqrt{2}$	$\|001\rangle$
$-\sqrt{q_1p_0}/\sqrt{2}$	$\|101\rangle$
$-\sqrt{p_1q_0}/\sqrt{2}$	$\|010\rangle$
$+\sqrt{p_1q_0}/\sqrt{2}$	$\|110\rangle$
$-\sqrt{p_1p_0}/\sqrt{2}$	$\|011\rangle$
$-\sqrt{p_1p_0}/\sqrt{2}$	$\|111\rangle$

IIH

$\sqrt{q_1q_0}$	$\|000\rangle$
---	$\|100\rangle$
0	$\|001\rangle$
---	$\|101\rangle$
0	$\|010\rangle$
---	$\|110\rangle$
$-\sqrt{p_1p_0}$	$\|011\rangle$
---	$\|111\rangle$

rvU_N --- Neural Computation: Step 3

Classical:

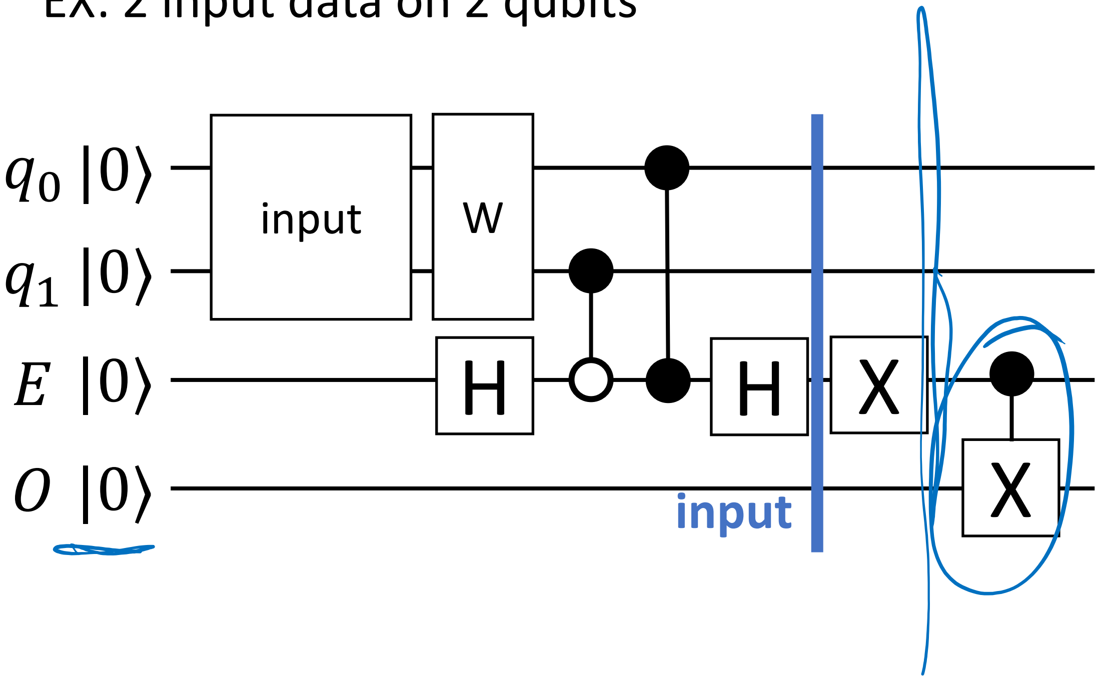
$$E(O) = E(n^2)$$
$$= 0 \times (p_0 q_1 + p_1 q_0) + 1 \times (q_0 q_1 + p_0 p_1)$$

Step 3: $O = n^2$

r.v.	-1	0	+1
n	$p_0 p_1$	$p_0 q_1 + p_1 q_0$	$q_0 q_1$

r.v.	0	+1
n^2	$p_0 q_1 + p_1 q_0$	$q_0 q_1 + p_0 p_1$

EX: 2 input data on 2 qubits



Input

$\sqrt{q_1 q_0}$	$\vert 000\rangle$
---	$\vert 100\rangle$
0	$\vert 001\rangle$
---	$\vert 101\rangle$
0	$\vert 010\rangle$
---	$\vert 110\rangle$
$-\sqrt{p_1 p_0}$	$\vert 011\rangle$
---	$\vert 111\rangle$

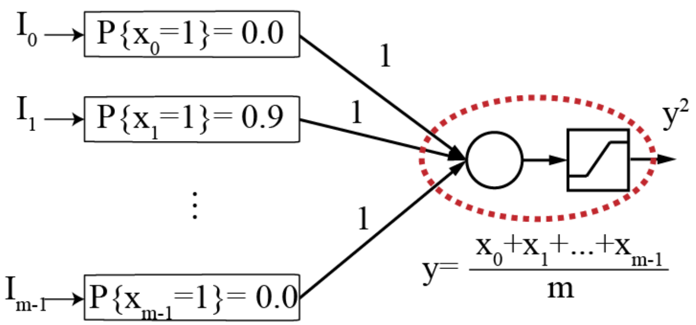
IIX

---	$\vert 000\rangle$
$\sqrt{q_1 q_0}$	$\vert 100\rangle$
---	$\vert 001\rangle$
0	$\vert 101\rangle$
---	$\vert 010\rangle$
0	$\vert 110\rangle$
---	$\vert 011\rangle$
$-\sqrt{p_1 p_0}$	$\vert 111\rangle$

Quantum:

$$P(E = |1\rangle)$$
$$= \sqrt{q_1 q_0}^2 + (-\sqrt{p_1 p_0})^2$$
$$= q_0 q_1 + p_0 p_1$$

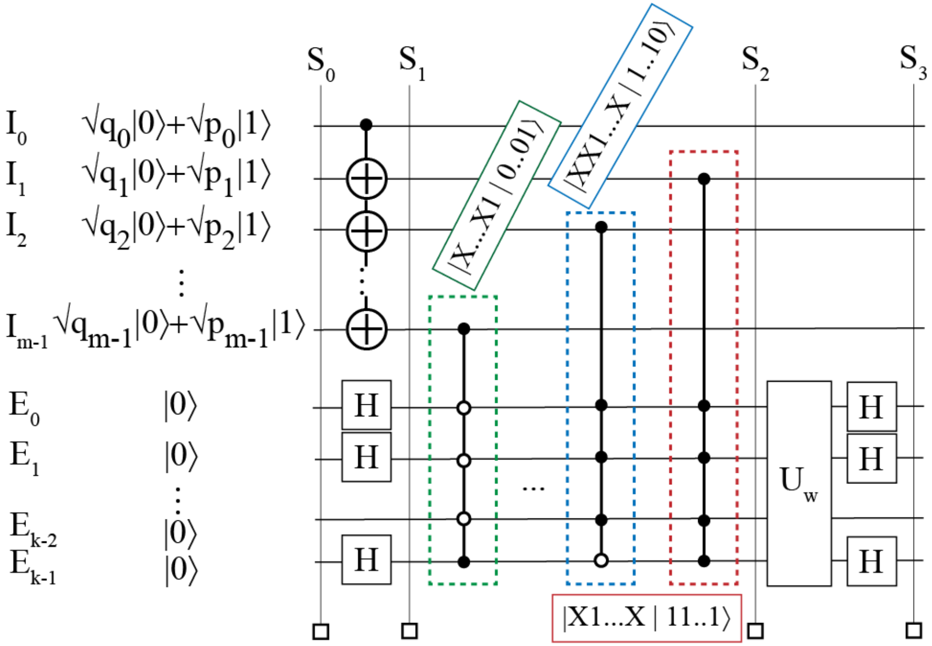
rvU_N --- Neural Computation



	$\|1\rangle$	$\|0\rangle$
	-1	+1
x_0	p_0	q_0
x_1	p_1	q_1
⋮		
x_{m-1}	p_{m-1}	q_{m-1}

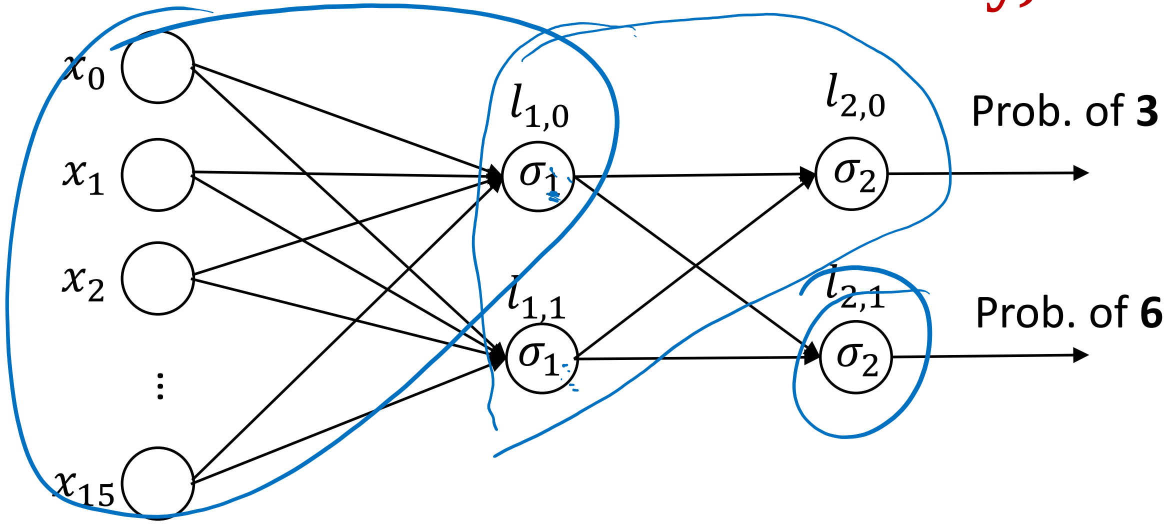
y	-1	$\frac{-m+2}{m}$	… 0 …	$\frac{m-2}{m}$	1
	$\prod p_i$	$p_{m-1}\ldots p_1q_0$ + $p_{m-1}\ldots q_1p_0$ + ⋮ + $q_{m-1}\ldots p_1p_0$	…	$q_{m-1}\ldots q_1p_0$ + $q_{m-1}\ldots p_1q_0$ + ⋮ + $p_{m-1}\ldots q_1q_0$	$\prod q_i$

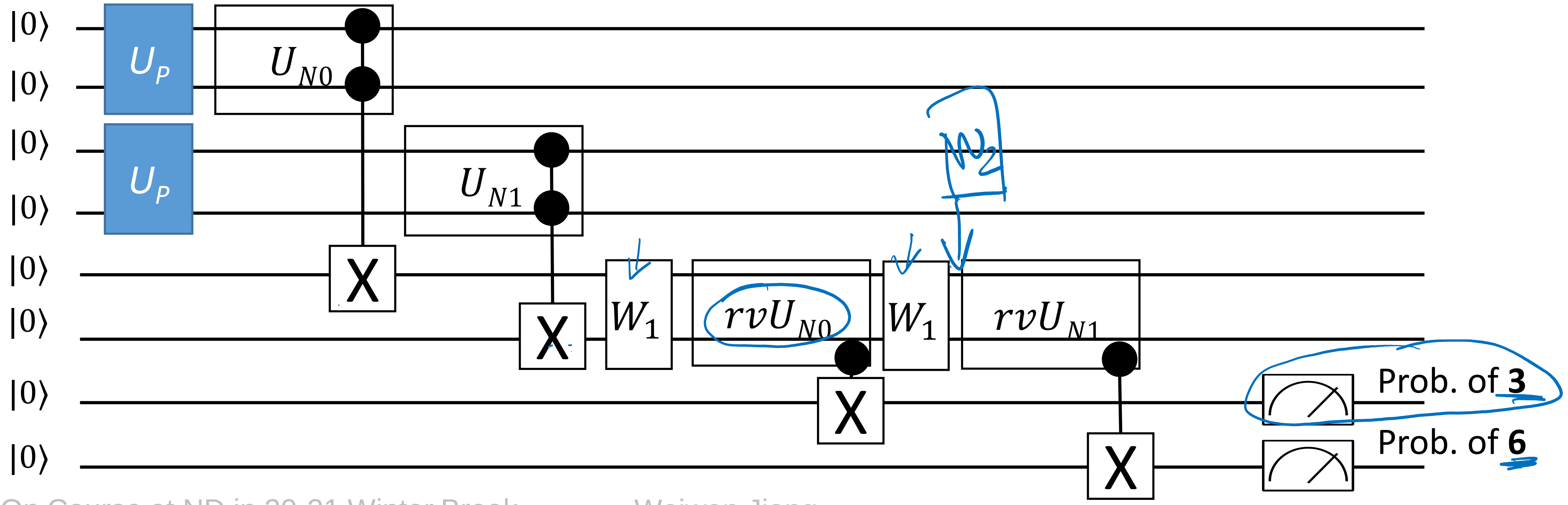
y^2	0 …	$(\frac{m-2}{m})^2$	1
		$p_{m-1}\ldots p_1q_0$ + $p_{m-1}\ldots q_1p_0$ + ⋮ + $q_{m-1}\ldots p_1p_0$ + $q_{m-1}\ldots q_1p_0$ + $q_{m-1}\ldots p_1q_0$ + ⋮ + $p_{m-1}\ldots q_1q_0$	$\prod q_i$ + $\prod p_i$



m-k Encoder States	Amplitude			
	S_0	S_1	S_2	S_3
$\|00\ldots0\rangle\otimes\|0..0\rangle$ $\|00\ldots0\rangle\otimes\|0..1\rangle$ … $\|00\ldots0\rangle\otimes\|1..1\rangle$	$\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ 0 … 0	$\frac{1}{2^{k/2}}$ $\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ $\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ … $\sqrt{q_{m-1}q_{k-1}\ldots q_0}$	$\frac{1}{2^{k/2}}$ $\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ $\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ … $\sqrt{q_{m-1}q_{m-2}\ldots q_0}$	$\sqrt{q_{m-1}q_{m-2}\ldots q_0}$ xxxxxxxxxx … xxxxxxxxxx
$\|00\ldots1\rangle\otimes\|0..0\rangle$ $\|00\ldots1\rangle\otimes\|0..1\rangle$ … $\|00\ldots1\rangle\otimes\|1..1\rangle$	$\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ 0 … 0	$\frac{1}{2^{k/2}}$ $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ … $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$	$\frac{1}{2^{k/2}}$ $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ $-\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ … $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$	(m-2)/m $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ xxxxxxxxxx … xxxxxxxxxx
…	…	…	…	…
$\|11\ldots1\rangle\otimes\|0..0\rangle$ $\|11\ldots1\rangle\otimes\|0..1\rangle$ … $\|11\ldots1\rangle\otimes\|1..1\rangle$	$\sqrt{p_{m-1}p_{m-2}\ldots p_0}$ 0 … 0	$\frac{1}{2^{k/2}}$ $\sqrt{p_{m-1}q_{m-2}\ldots q_0}$ $\sqrt{p_{m-1}q_{m-2}\ldots q_0}$ … $\sqrt{p_{m-1}q_{m-2}\ldots q_0}$	$\frac{1}{2^{k/2}}$ $\sqrt{p_{m-1}q_{m-2}\ldots q_0}$ $-\sqrt{p_{m-1}q_{m-2}\ldots q_0}$ … $-\sqrt{p_{m-1}q_{m-2}\ldots q_0}$	(2-m)/m $\sqrt{q_{m-1}q_{m-2}\ldots p_0}$ xxxxxxxxxx … xxxxxxxxxx

Implementing Feedforward Net w/ Non-Linearity, w/o Measurement!

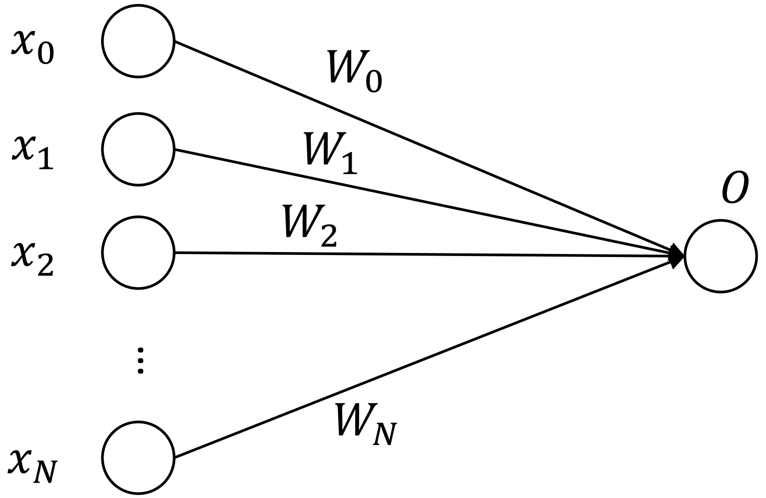


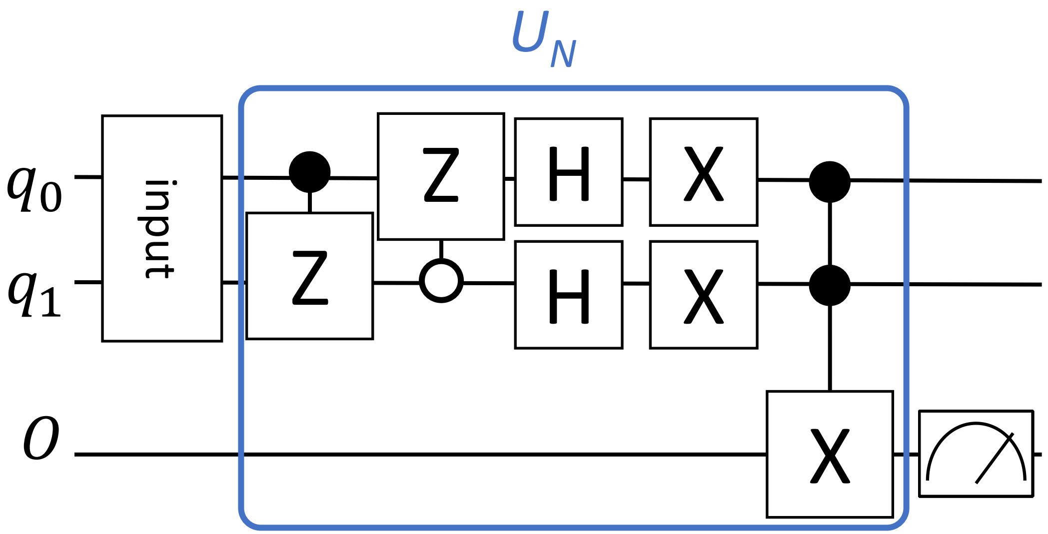


Tutorial 3: $PreP + U_P + U_N + M + PostP$

https://github.com/weiwenjiang/QML_tutorial/blob/main/Tutorial_3_Full_MNIST_Prediction.ipynb

Challenge 3: High Complexity in the Previous Design





Cost Complexity

Classical Computing		
	No Parallelism	Full Parallelism
Time (T)	O(N)	O(1)
Space (S)	O(1)	O(N)
Cost (TS)	O(N)	O(N)

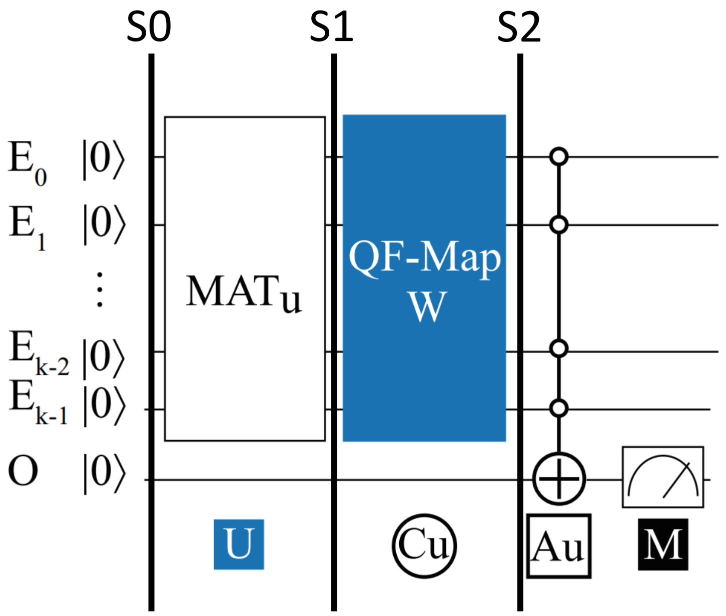
Quantum Computing		
	Previous Design	Optimization
Circuit Depth (T)	O(N)	???
Qubits (S)	O($\log N$)	O(N)
Cost (TS)	O($N \cdot \log N$)	**target** O(ploylog N)

$[0, 0.9, 0, 0, 0, 0.1, 0, 0, 1.0, 0.5, 0.5, 0, 0, 0, 0]^T$

U

$[0, 0.59, 0, 0, 0, 0.07, 0, 0, 0.66, 0.33, 0.33, 0, 0, 0, 0]^T$

QuantumFlow: Taking NN Property to Design QC



S0 -> S1:

$$(v_o; v_{x1}; v_{x2}; \ldots; v_{xn}) \times \begin{pmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix} = (v_0)$$

$$S1 = [0, 0.59, 0, 0, 0, 0.07, 0, 0, 0.66, 0.33, 0.33, 0, 0, 0, 0]^T$$

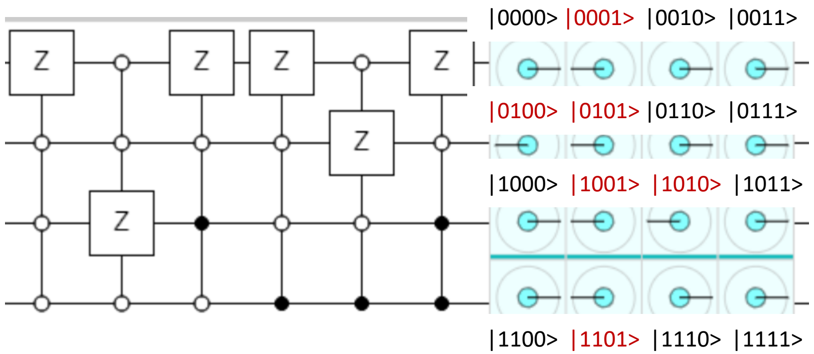
S1 -> S2:

$$\mathrm{W} = [+1, -1, +1, +1, -1, -1, +1, +1, +1, -1, -1, +1, +1, -1, +1, +1]^T$$

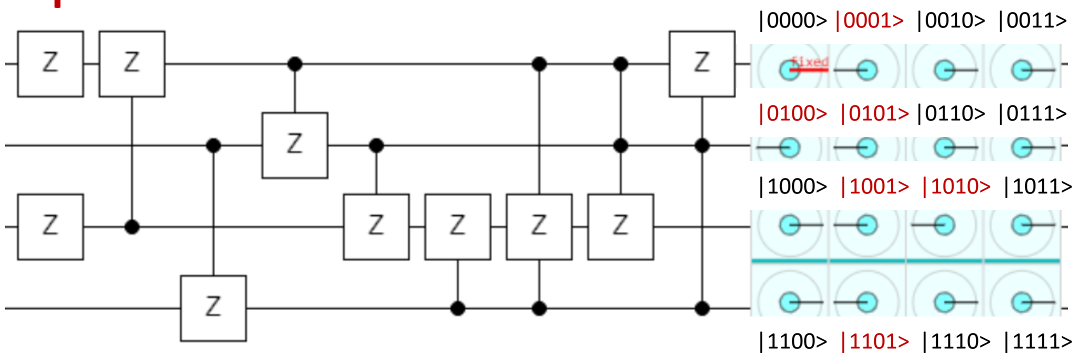
|0000> |0001> |0010> |0011> |0100> |0101> |0110> |0111> |1000> |1001> |1010> |1011> |1100> |1101> |1110> |1111>

$$S2 = [0, -0.59, 0, 0, -0, -0.07, 0, 0, 0, -0.66, -0.33, 0.33, 0, -0, 0, 0]^T$$

Implementation 1 (example in Quirk):

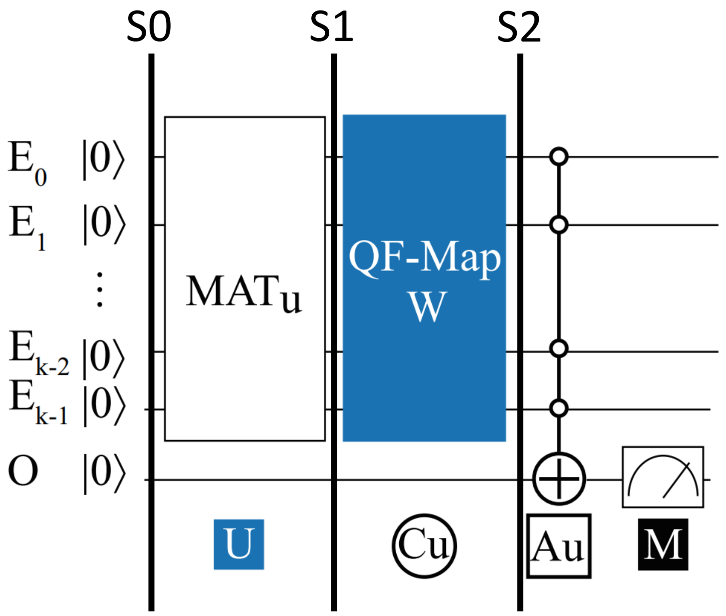


Implementation 2:



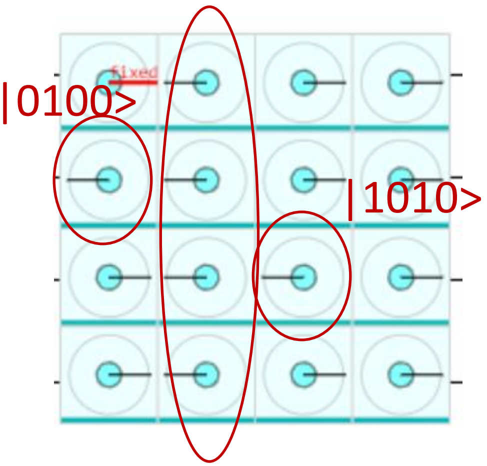
[ref] Tacchino, F., et al., 2019. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, *5*(1), pp.1-8.

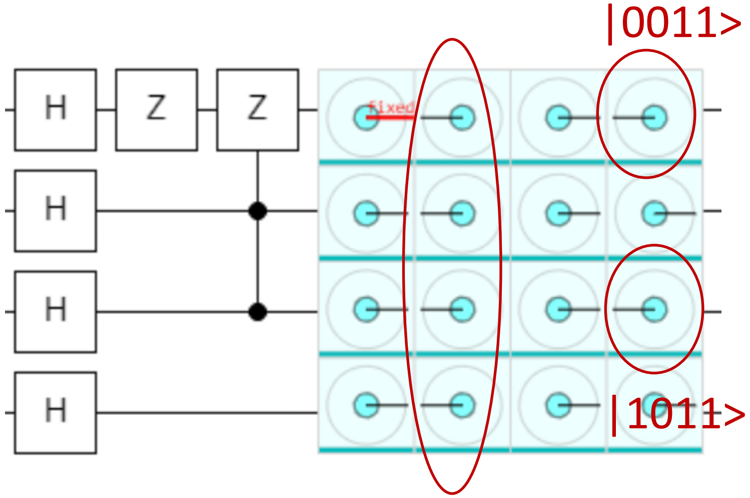
QuantumFlow: Taking NN Property to Design QC



Property from NN

- The **weight order** is not necessary to be fixed, which can be adjusted if the order of inputs are adjusted accordingly
- **Benefit:** No need to require the positions of sign flip are exactly the same with the weights; instead, only need the number of signs are the same.





$$S1 = [0, 0.59, 0, \mathbf{0}, \mathbf{0}, 0.07, 0, 0, 0.66, \mathbf{0.33}, \mathbf{0.33}, 0, 0, 0, 0]^T$$

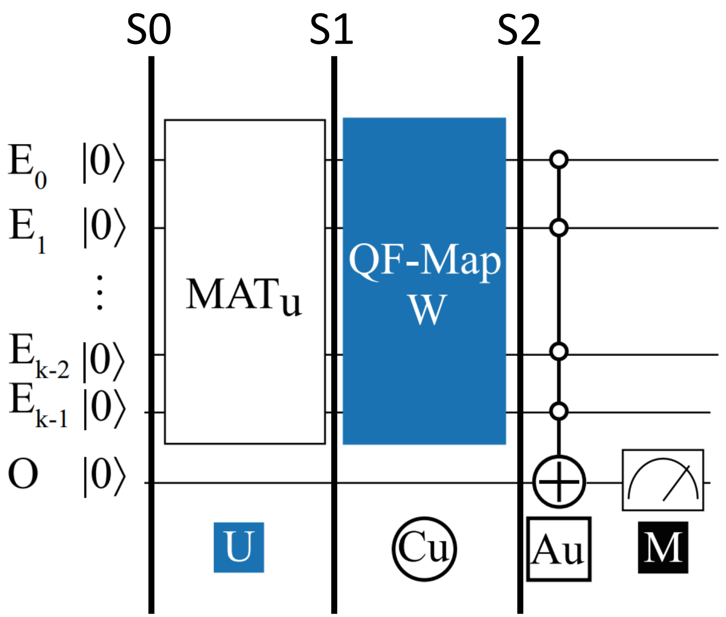
ori + - - +

fin - + + -

$$S1' = [0, 0.59, 0, \mathbf{0.33}, \mathbf{0.33}, 0.07, 0, 0, 0.66, \mathbf{0}, \mathbf{0}, 0, 0, 0, 0]^T$$

QuantumFlow: Taking NN Property to Design QC

$O(2^k)$



Algorithm 4: QF-Map: weight mapping algorithm

Input: (1) An integer $R \in (0, 2^{k-1}]$; (2) number of qbits k;
Output: A set of applied gate G

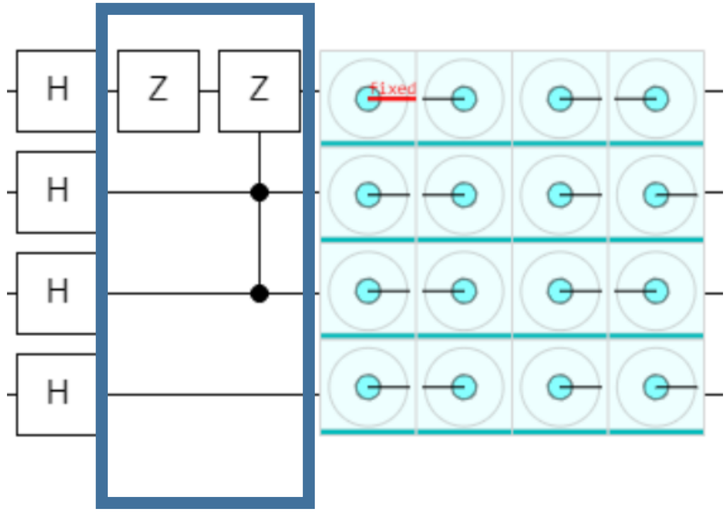
void recursive(G,R,k){
  if ($R < 2^{k-2}$){
    recursive(G,R,$k-1$); // Case 1 in the third step
  }
  else if ($R == 2^{k-1}$){
    $G.append(PG_{2^{k-1}})$; // Case 2 in the third step
    return;
  }else{
    $G.append(PG_{2^{k-1}})$;
    recursive(G,$2^{k-1}-R$,$k-1$); // Case 3 in the third step
  }
}
// Entry of weight mapping algorithm
set main(R,k){
  Initialize empty *set* G;
  recursive(G,R,k);
  return G
}

Used gates and Costs

Gates	Cost
Z	1
CZ	1
C^2Z	3
C^3Z	5
C^4Z	6
…	…
C^kZ	2k-1

Worst case: all gates

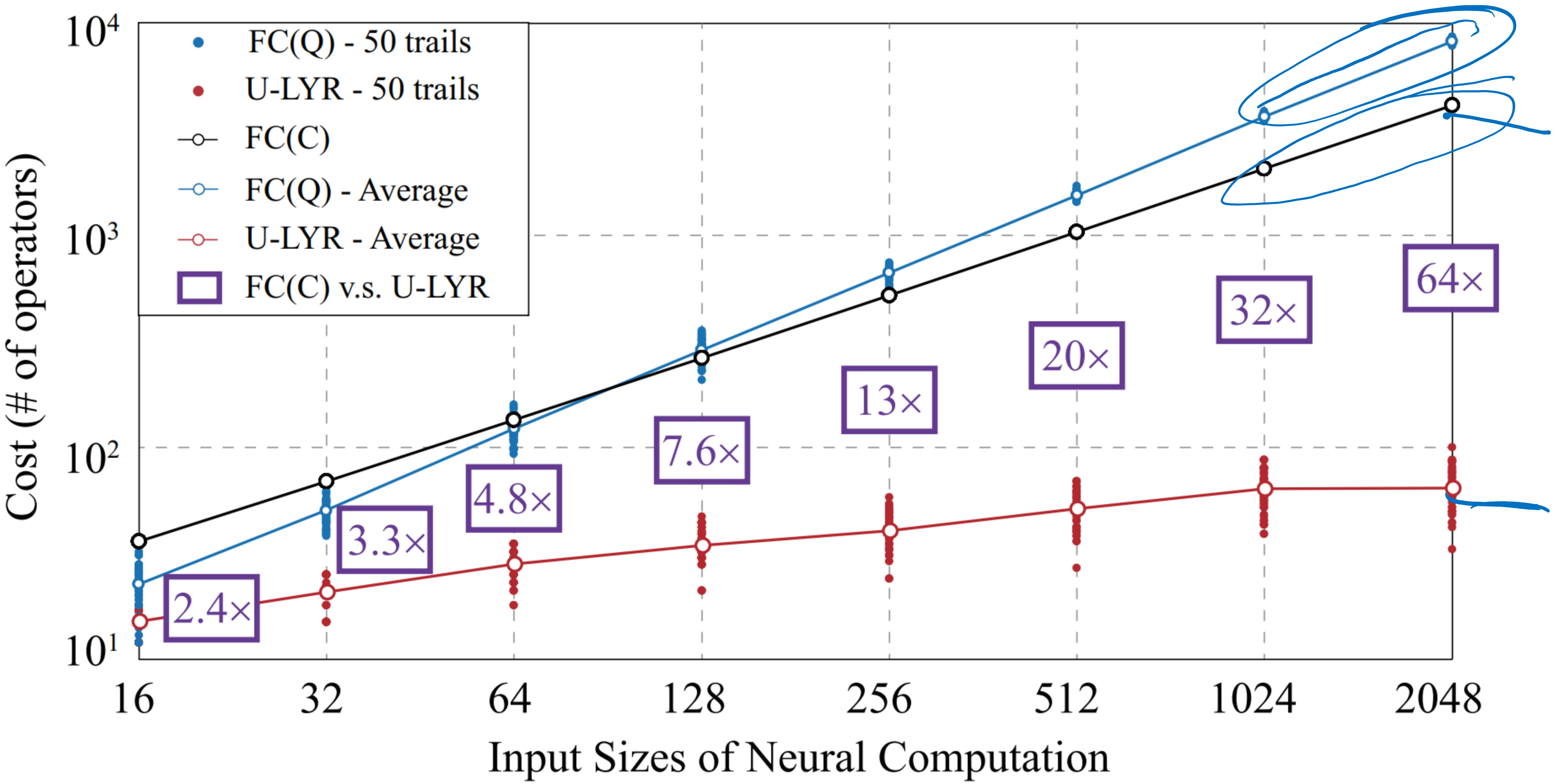
$O(k^2)$



$O(N) \rightarrow O(\log^2 N)$

QuantumFlow Results

U-LYR Achieves Quantum Advantages



[ref] Tacchino, F., et al., 2019. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, *5*(1), pp.1-8.

QuantumFlow Achieves Over 10X Cost Reduction

Dataset	Structure			MLP(C)			FFNN(Q)				QF-hNet(Q)			
	In	L1	L2	L1	L2	Tot.	L1	L2	Tot.	**Red.**	L1	L2	Tot.	**Red.**
{1,5}	16	4	2	132	18	150	80	38	118	**1.27**×	74	38	112	**1.34**×
{3,6}	16	4	2				96	38	134	**1.12**×	58	38	96	**1.56**×
{3,8}	16	4	2				76	34	110	**1.36**×	58	34	92	**1.63**×
{3,9}	16	4	2				98	42	140	**1.07**×	68	42	110	**1.36**×
{0,3,6}	16	8	3	264	51	315	173	175	348	**0.91**×	106	175	281	**1.12**×
{1,3,6}	16	8	3				209	161	370	**0.85**×	139	161	300	**1.05**×
{0,3,6,9}	64	16	4	2064	132	2196	1893	572	2465	**0.89**×	434	572	1006	**2.18**×
{0,1,3,6,9}	64	16	5	2064	165	2229	1809	645	2454	**0.91**×	437	645	1082	**2.06**×
{0,1,2,3,4}	64	16	5				1677	669	2346	**0.95**×	445	669	1114	**2.00**×
{0,1,3,6,9}*	256	8	5	4104	85	4189	5030	251	5281	**0.79**×	135	251	386	**10.85**×

*: Model with 16×16 resolution input for dataset {0,1,3,6,9} to test scalability, whose accuracy is 94.09%, which is higher than 8×8 input with accuracy of 92.62%.

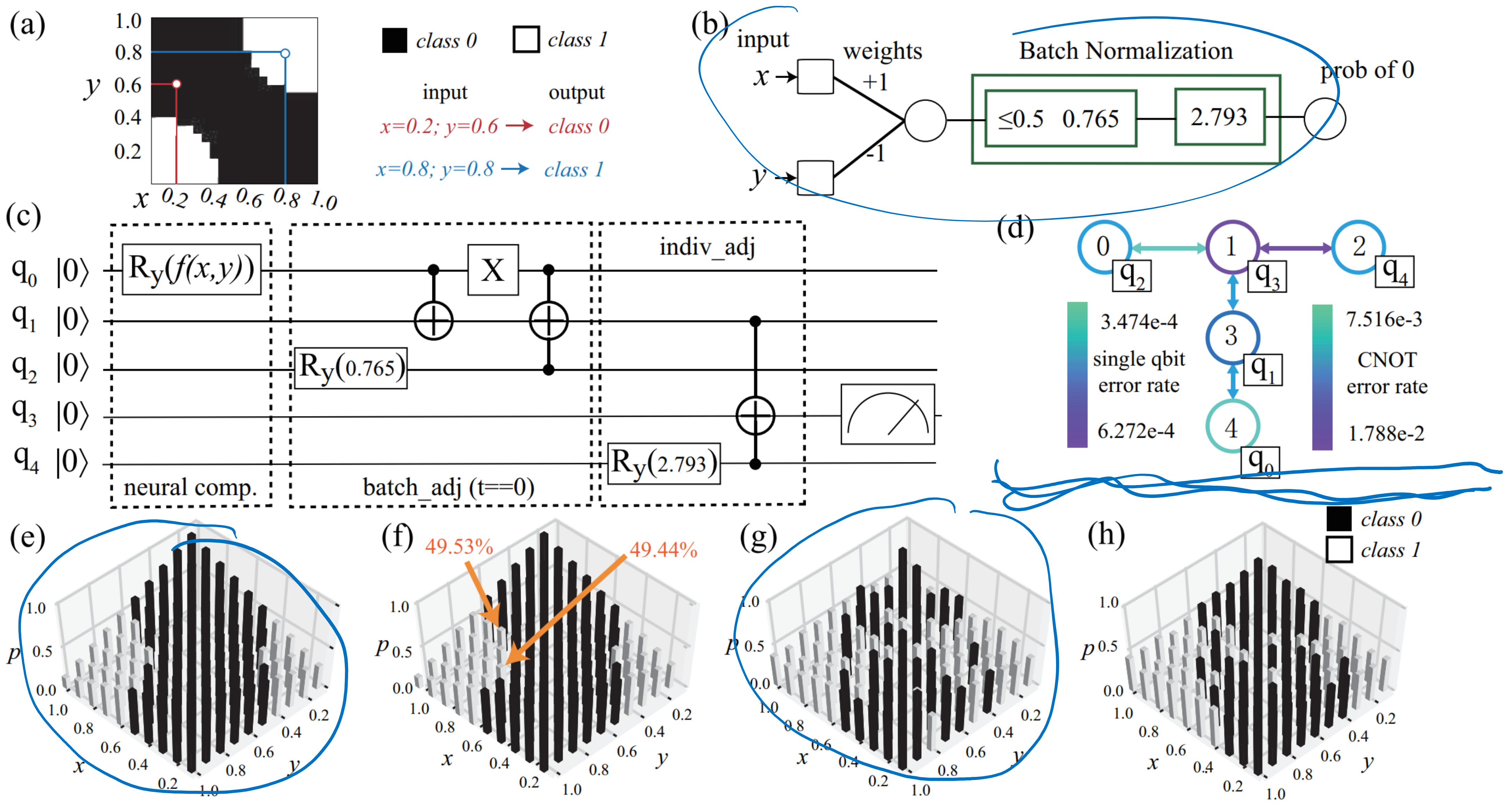
[ref of FFNN] Tacchino, F., et al., 2019. Quantum implementation of an artificial feed-forward neural network. *arXiv preprint arXiv:1912.12486*.

QF-Nets Achieve the Best Accuracy on MNIST

Dataset	w/o BN					w/ BN				
	binMLP(C)	FFNN(Q)	MLP(C)	QF-pNet	QF-hNet	binMLP(C)	FFNN(Q)	MLP(C)	QF-pNet	QF-hNet
1,5	61.47%	61.47%	69.12%	69.12%	90.33%	55.99%	55.99%	85.30%	84.56%	**96.60%**
3,6	72.76%	72.76%	94.21%	91.67%	97.21%	72.76%	72.76%	96.29%	96.39%	**97.66%**
3,8	58.27%	58.27%	82.36%	82.36%	89.77%	58.37%	58.07%	86.74%	86.90%	**87.20%**
3,9	56.71%	56.51%	68.65%	68.30%	95.49%	56.91%	56.71%	80.63%	78.65%	**95.59%**
0,3,6	46.85%	51.63%	49.90%	59.87%	89.65%	50.68%	50.68%	75.37%	78.70%	**90.40%**
1,3,6	60.04%	59.97%	53.69%	53.69%	94.68%	59.59%	59.59%	86.76%	86.50%	**92.30%**
0,3,6,9	72.68%	72.33%	84.28%	87.36%	92.85%	69.95%	68.89%	82.89%	76.78%	**93.63%**
0,1,3,6,9	50.00%	51.10%	49.00%	43.24%	87.96%	60.96%	69.46%	70.19%	71.56%	**92.62%**
0,1,2,3,4	46.96%	50.01%	49.06%	52.95%	83.95%	64.51%	69.66%	71.82%	72.99%	**90.27%**

[ref of FFNN] Tacchino, F., et al., 2019. Quantum implementation of an artificial feed-forward neural network. *arXiv preprint arXiv:1912.12486*.

On Actual IBM “ibmq_essex” Quantum Processor



Thank You!

wjiang2@nd.edu