Machine Learning on Quantum Computing: From Classical to Quantum

(Week 4 – Session 1)

Weiwen Jiang, Ph.D.

Postdoc Research Associate

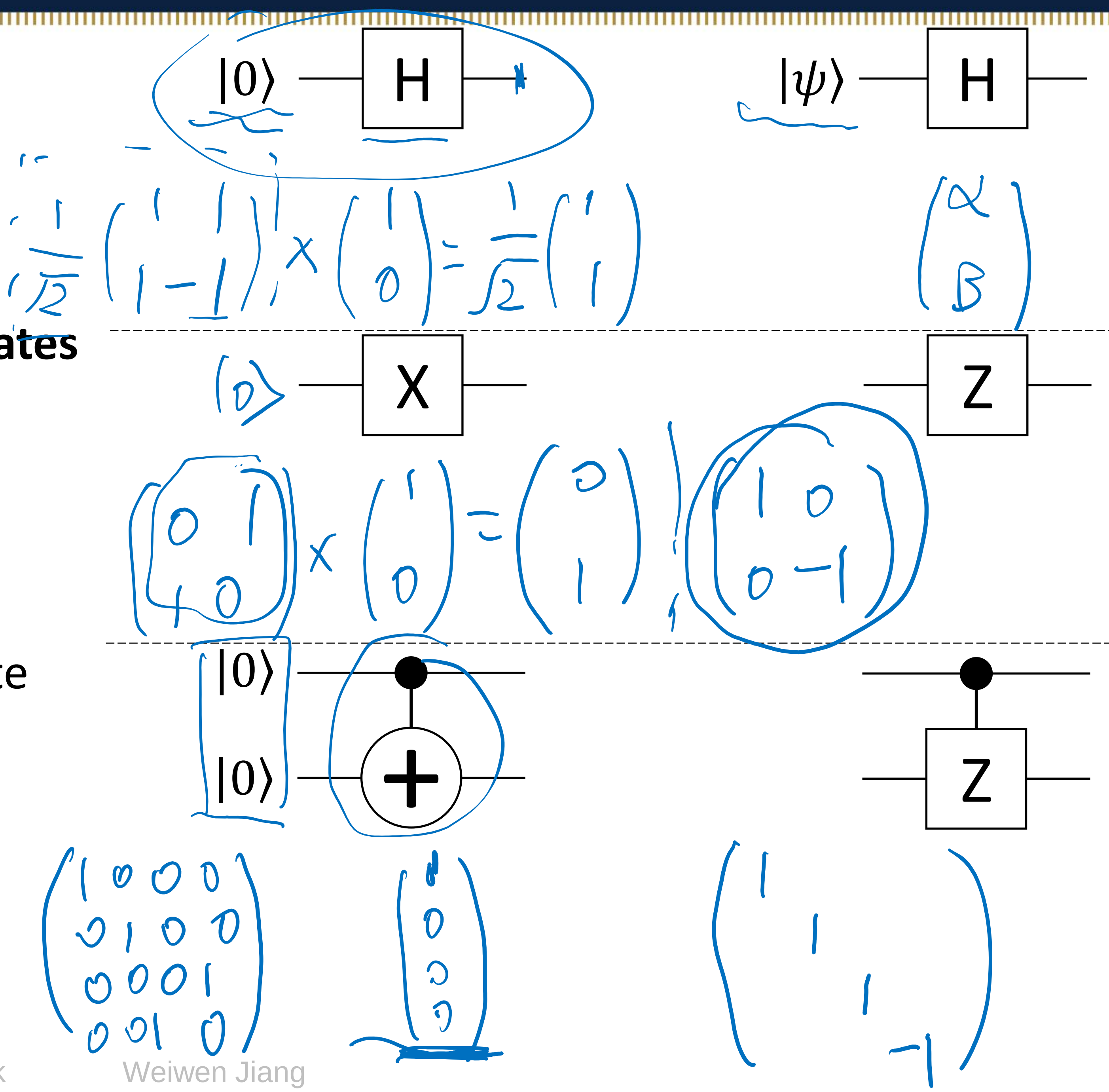
Department of Computer Science and Engineering

University of Notre Dame

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

Review of Previous Sessions

- Single-Qubit Gates
  - **Hadamard gate: H Gate**
  - **Pauli operators: X, Y, Z Gates**
  - General gate: U Gate
- Multi-Qubit Gates
  - **Controlled-Pauli gates**
  - Controlled-Hadamard gate
  - Controlled-Phase gates
  - SWAP gate
  - **Toffoli gate or CCNOT**
  - Fredkin gate or CSWAP



Organization of Quantum Machine Learning Sessions

- **Background and Motivation** *[w4s1]*
  - What is machine learning and neural network
  - Why using quantum computer
  - Our goals

MNIST

- **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

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

What is Machine Learning?

UNIVERSITY OF NOTRE DAME

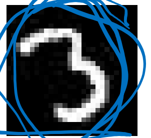
Supervised Learning

Example: Classification

Training

Given: Labeled data as training dataset

(x_i, y_i): x_i training data, y_i: label

$x_i =$  $\quad y_i = 3$

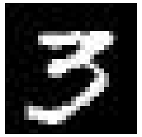
Output: A learned function $\boldsymbol{f}$ from X to Y

$$\boldsymbol{f}: x \mapsto y$$

Inference/Execution

Given: Unseen data/test dataset

A learned function $\boldsymbol{f}$

Do: $\boldsymbol{f}($  $) = 3$ ✓

8 x

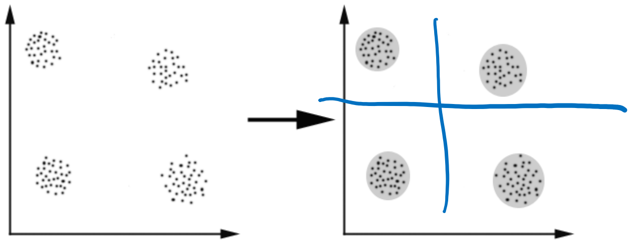
Unsupervised Learning

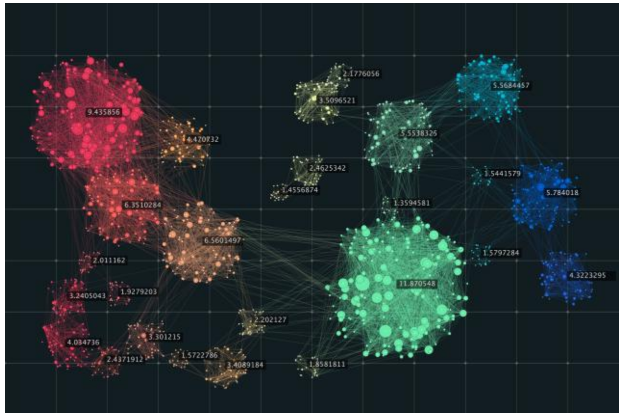
Example: Clustering

Given: Unlabeled data

(x_i)

Goal: discover the “natural groupings” present in the data



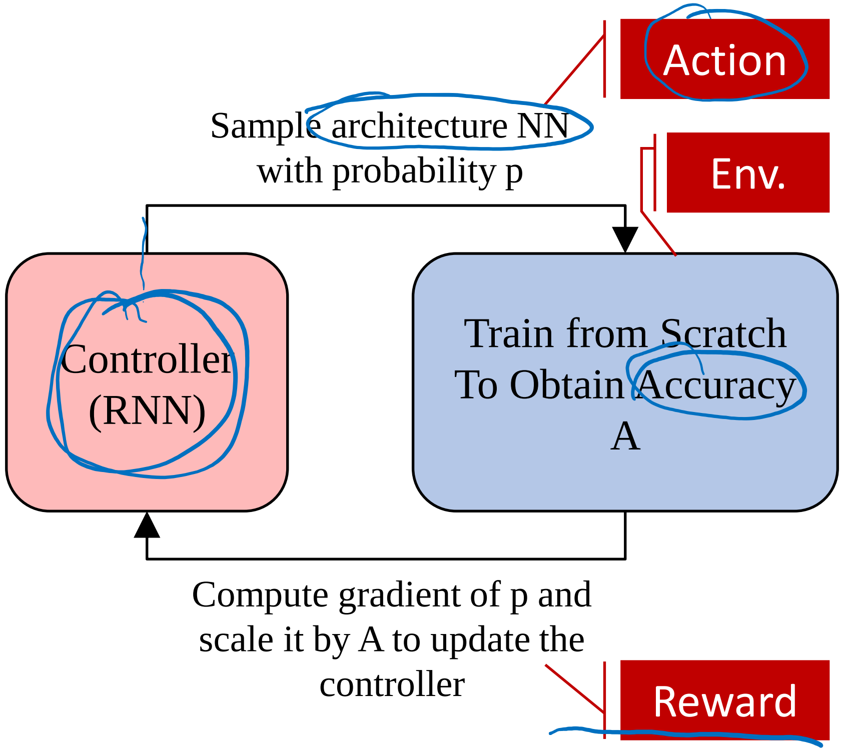


Reinforcement Learning

Example: Neural Architecture Search

Given: An environment that can give us reward based on our action

Goal: Maximize the expected rewards



What is Machine Learning? --- Our Focus

UNIVERSITY OF NOTRE DAME

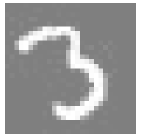
Supervised Learning

Example: Classification

Training

Given: Labeled data as training dataset

(x_i, y_i): x_i training data, y_i: label

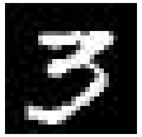
$x_i =$  $y_i = 3$

Output: A learned function f from X to Y

$$f: x \mapsto y$$

Inference/Execution

Given: Unseen data test dataset
A learned function f

Do: $f($  $) = 3$

Unsupervised Learning

Example: Clustering

Given: Unlabeled data

$$(x_i)$$

Goal: discover the "natural groupings" present in the data

Reinforcement Learning

Example: Neural Architecture Search

Given: An environment that can give us reward based on our action

Goal: Maximize the expected rewards

What is Neural Network?

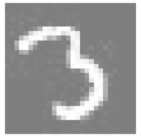
Supervised Learning

Example: Classification

Training

Given: Labeled data as training dataset

(x_i, y_i): x_i training data, y_i: label

$x_i =$  $\quad y_i = 3$

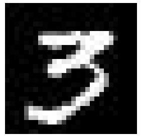
Output: A learned function f from X to Y

$$f: x \mapsto y$$

Inference/Execution

Given: Unseen data test dataset

A learned function f

Do: $f($  $) = 3$

An unknown classification function: $\boldsymbol{g}$

$$\boldsymbol{y = g(x);\ s.t.\ y_i = g(x_i)}$$

Learn a function $\boldsymbol{f}$ with parameters $\boldsymbol{\theta, b}$ to approximate $\boldsymbol{g}$:

$$\boldsymbol{\hat{y} = f(x, \theta, b)}$$

Training is to minimize the loss function by adjusting parameters $\boldsymbol{\theta, b}$

$$\boldsymbol{min:\ \mathcal{L}(f) = \sum_i (f(x_i, \theta, b) - y_i)}$$

Perceptron model, where $\boldsymbol{\sigma}$ is a non-linear function:

$$\boldsymbol{\hat{y} = \sigma(\theta x + b)}$$

Feedforward neural network:

$$\boldsymbol{l_1 = \sigma_1(\theta_1 x + b_1)}$$
$$\boldsymbol{l_2 = \sigma_2(\theta_2 l_1 + b_2)}$$
$$\dots \dots$$
$$\boldsymbol{l_n = \sigma_n(\theta_n l_{n-1} + b_n)}$$
$$\boldsymbol{\hat{y} = classifier(l_n)}$$

What is Neural Network?

Supervised Learning

Example: Classification

Training

Given: Labeled data as training dataset

(x_i, y_i): x_i training data, y_i: label

$x_i =$ $y_i = 3$

Output: A learned function f from X to Y

$f: x \mapsto y$

$1 \times n$

$(W_{1,0}\ W_{1,1}\ \ldots\ W_{1,n})$
$(W_{2,0}\ W_{2,1}\ \ldots)$

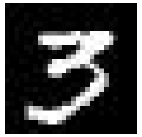
$(x_0,\ x_1,\ \ldots,\ x_n)$ = (0 ...)

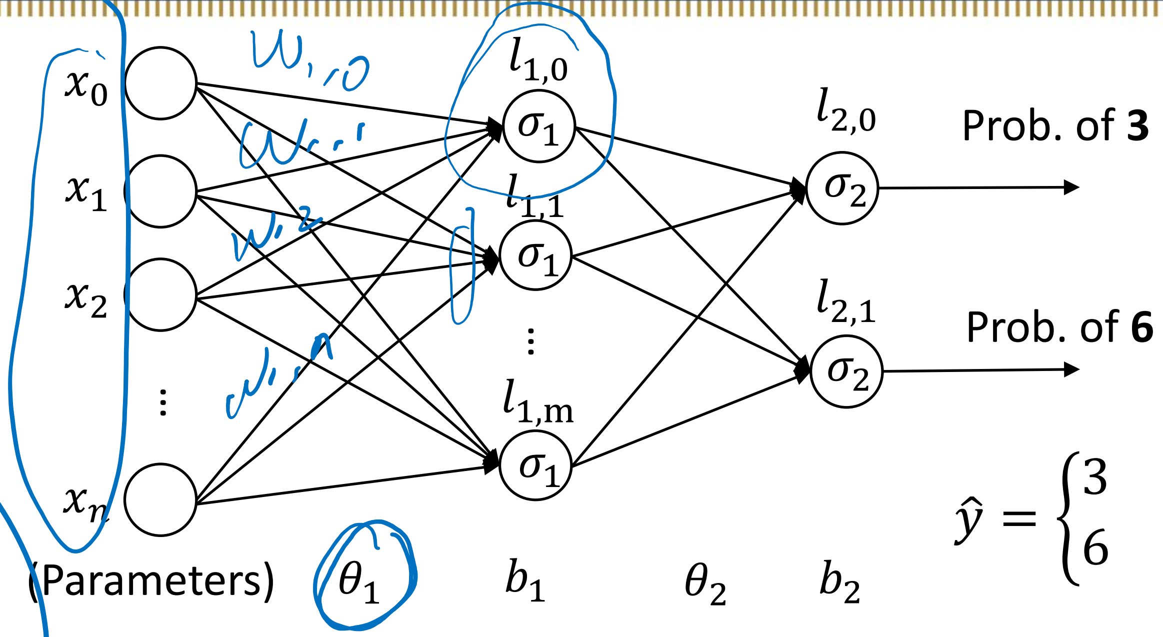
$m \times n$ $n \times 1$ $m \times 1$

$(\theta \cdot x + b)^2$

Inference/Execution

Given: Unseen data test dataset
A learned function f

Do: $f($  $) = 3$



$$\hat{y} = \begin{cases} 3 & l_{2,0} > l_{2,1} \\ 6 & l_{2,0} \le l_{2,1} \end{cases}$$

Example of feedforward neural network for $n = 2$

Perceptron model, where $\boldsymbol{\sigma}$ is a non-linear function:

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{\theta x} + \boldsymbol{b})$$

Feedforward neural network:

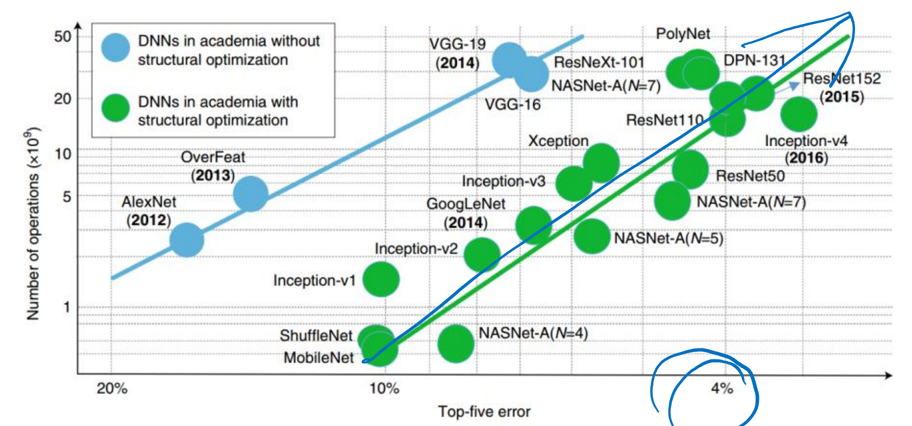
$$\boldsymbol{l_1} = \boldsymbol{\sigma_1}(\boldsymbol{\theta_1 x} + \boldsymbol{b_1})$$
$$\boldsymbol{l_2} = \boldsymbol{\sigma_2}(\boldsymbol{\theta_2 l_1} + \boldsymbol{b_2})$$
$$\ldots\ldots$$
$$\boldsymbol{l_n} = \boldsymbol{\sigma_n}(\boldsymbol{\theta_n l_{n-1}} + \boldsymbol{b_n})$$
$$\hat{\boldsymbol{y}} = \boldsymbol{classifier}(\boldsymbol{l_n})$$

Why Using Quantum Computer for Machine Learning?

UNIVERSITY OF NOTRE DAME

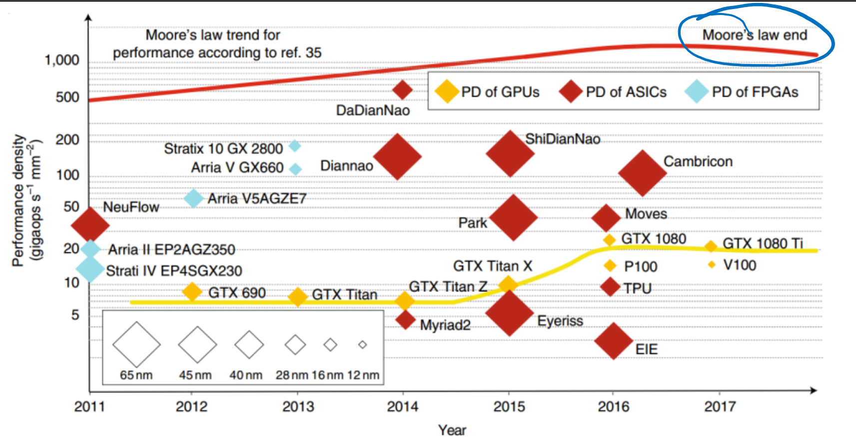
- Imbalanced “demand and supply” of NN on classical computing
- The growing power of quantum computing
- Linear algebra is central for both quantum computing and machine learning

NN on Classical Computer: Computation & Storage Demand > Supply



Number of DNN operations increases exponentially

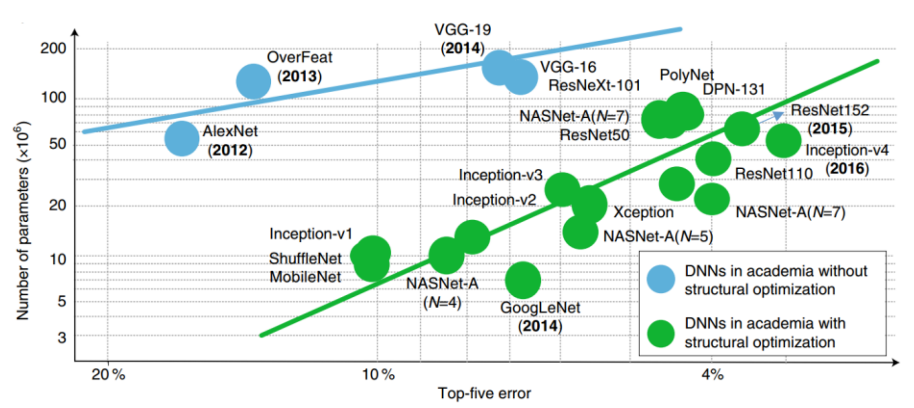
Computation Gap



Performance density almost stops increasing

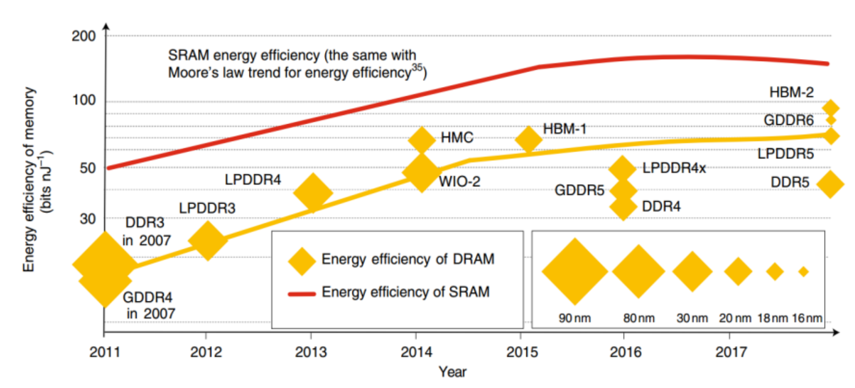
Neural Network Size

Traditional Hardware Capability



Number of DNN parameters increases exponentially

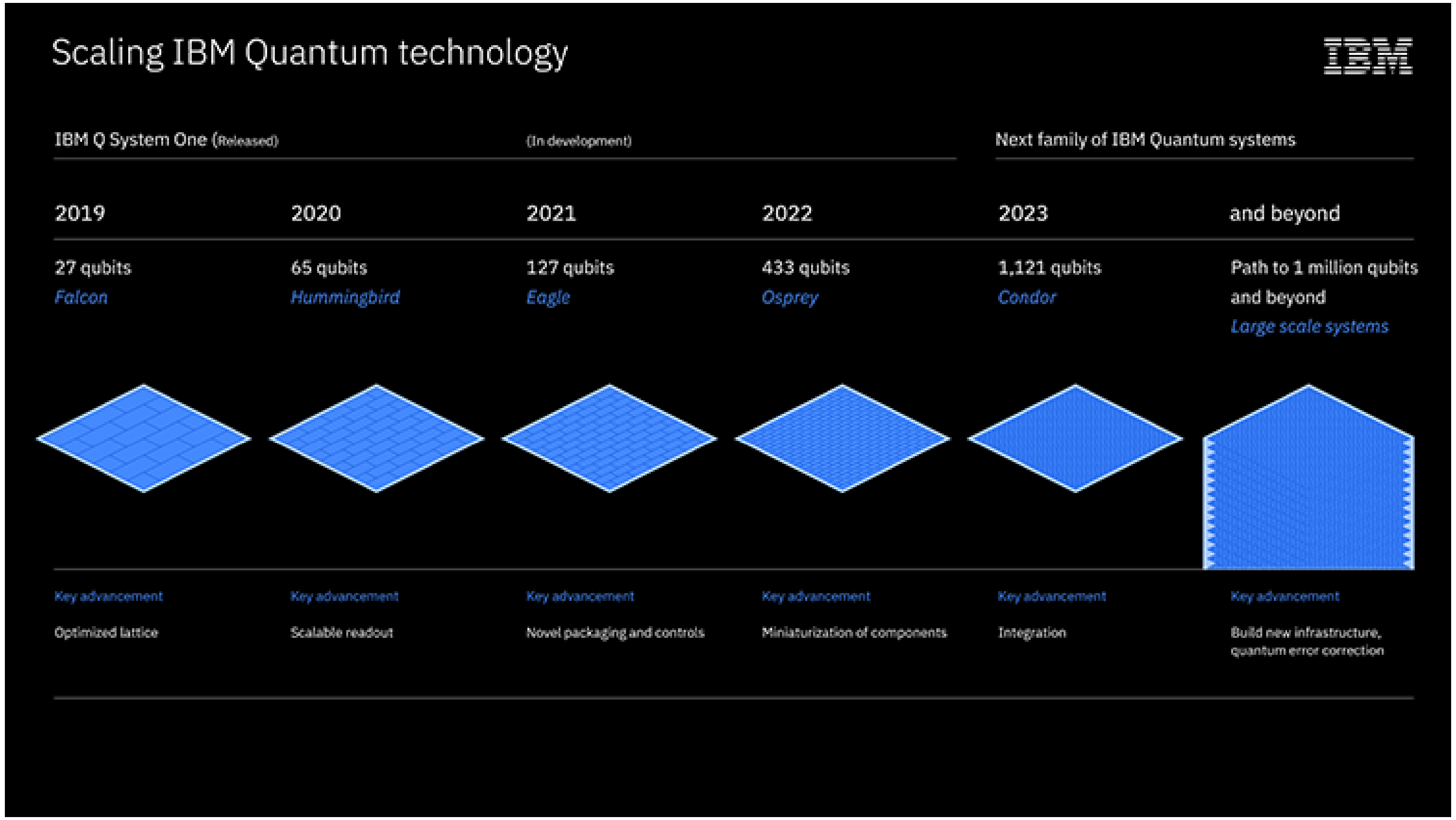
Storage Gap

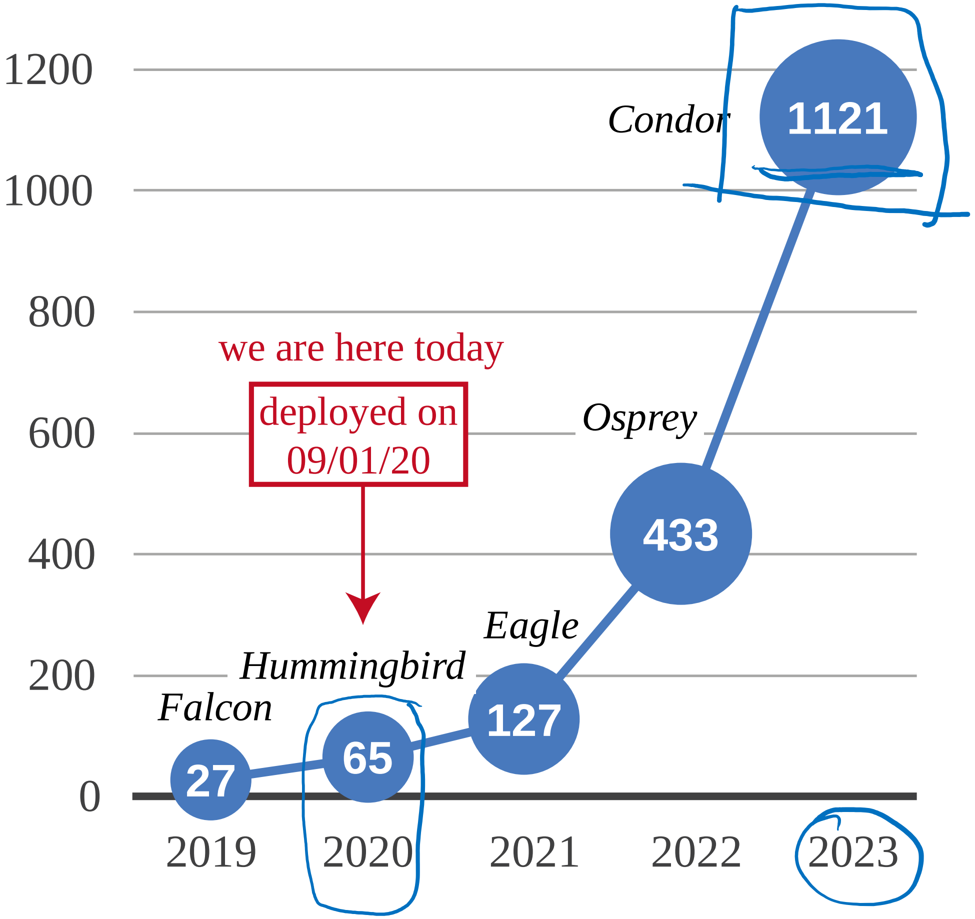


Energy efficiency of memory stops increasing

[ref] Xu, X., et al. 2018. Scaling for edge inference of deep neural networks. *Nature Electronics*, *1*(4), pp.216-222.

Consistently Increasing Qbits in Quantum Computers





The Power of Quantum Computers: Qubit

UNIVERSITY OF NOTRE DAME

Classical Bit

$$X = 0 \text{ or } 1$$

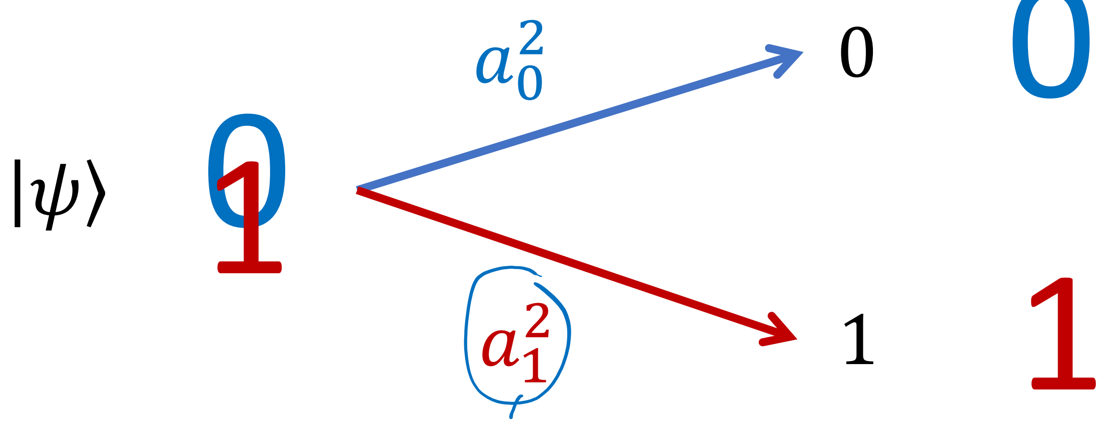
Quantum Bit (Qubit)

$$|\psi\rangle = |0\rangle \text{ and } |1\rangle$$

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle$$

$$\text{s.t. } a_0^2 + a_1^2 = 100\%$$

Reading out Information from Qubit (Measurement)



Probability

Non–Deterministic Computing

$$a_0^2 + a_1^2 = 100\%$$

$$40\% + 60\% = 100\%$$

The Power of Quantum Computers: Qubits

UNIVERSITY OF NOTRE DAME

- 115GB data
- 3×10^{10} numbers
- 35 qubits

2 Classical Bits

00 **or** 01 **or** 10 **or** 11

for 1 value

2 Qubits

$c_{00}|00\rangle$ **and** $c_{01}|01\rangle$ **and** $c_{10}|10\rangle$ **and** $c_{11}|11\rangle$

for 2^n values

$\boldsymbol{a_{00}, a_{01}, a_{10}, a_{11}}$

Qubits: $\boldsymbol{q_0, q_1}$

$$|q_0\rangle = a_0|0\rangle + a_1|1\rangle$$

$$|q_1\rangle = b_0|0\rangle + b_1|1\rangle$$

$$\boldsymbol{|q_0, q_1\rangle = |q_0\rangle \otimes |q_1\rangle}$$

$$\boldsymbol{= c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle}$$

- $|00\rangle$: Both q_0 and q_1 are in state $|0\rangle$
- c_{00}^2: Probability of both q_0 and q_1 are in state $|0\rangle$
- $c_{00}^2 = a_0^2 \times b_0^2$
- $c_{00} = \sqrt{a_0^2 \times b_0^2} = a_0 \times b_0$

Linear Algebra is also Central for Quantum Computing

UNIVERSITY OF NOTRE DAME

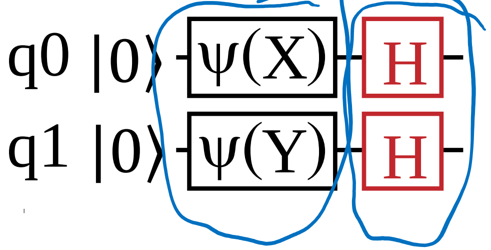
Matrix multiplication on classical computer using 16bit number

$$A_{N,N} \times B_{N,1} = C_{N,1}$$

Operation: Multiplication: $\boldsymbol{M \times M}$

Accumulation: $\boldsymbol{M \times (M-1)}$

Special matrix multiplication on quantum computer



Operation: **logM** Hadamard (H) Gates

$$A_{N,N} \times B_{N,1} = \frac{1}{2} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{bmatrix} = \begin{bmatrix} d_{00} \\ d_{01} \\ d_{10} \\ d_{11} \end{bmatrix}$$

$$|q_0, q_1\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$$

$$\rightarrow \begin{bmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{bmatrix}$$ (vector representation)

$$H \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = A_{N,N}$$

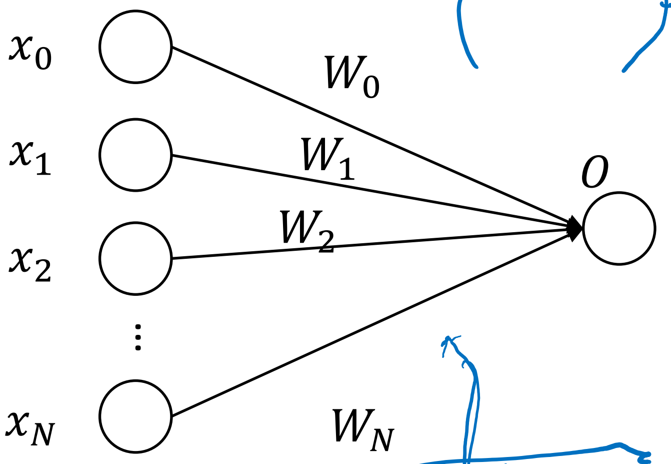
$$H \otimes H|q_0, q_1\rangle$$

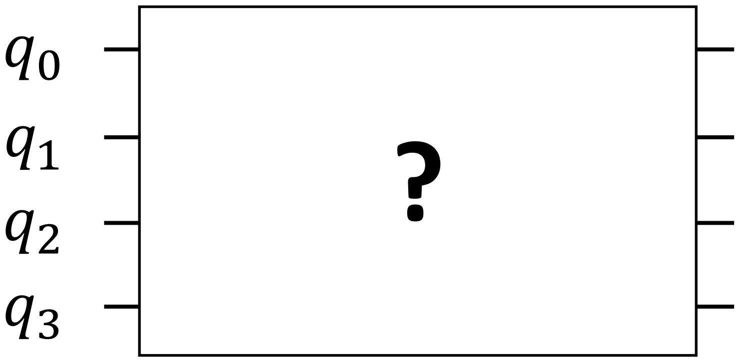
$$= d_{00}|00\rangle + d_{01}|01\rangle + d_{10}|10\rangle + d_{11}|11\rangle$$

Goals

3 Goals to Have an End-to-End Implementation and Quantum Advantages!

Goal 1: Correctly Implement!





Goal 2: Efficiently Implement!

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

where δ is a quadratic function

Classical Computing:

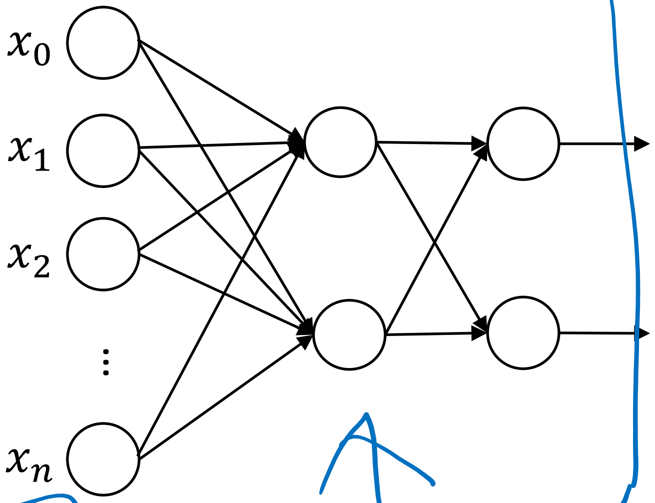
Complexity of $\boldsymbol{O(N)}$

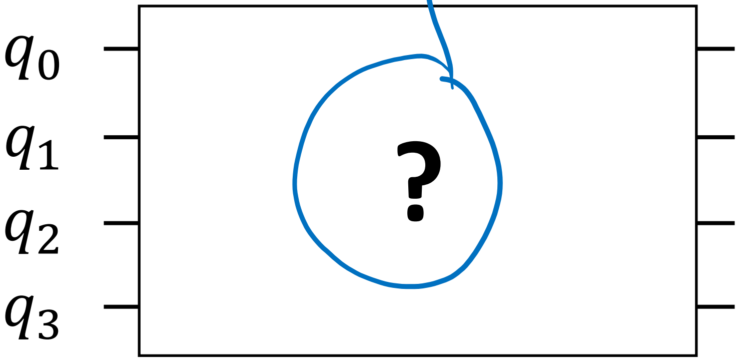
Quantum Computing:

Can we reduce complexity to

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

Goal 3: Scale-Up!





Organization of Quantum Machine Learning Sessions

- **Background and Motivation** *[w4s1]*
  - What is machine learning and neural network
  - 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:

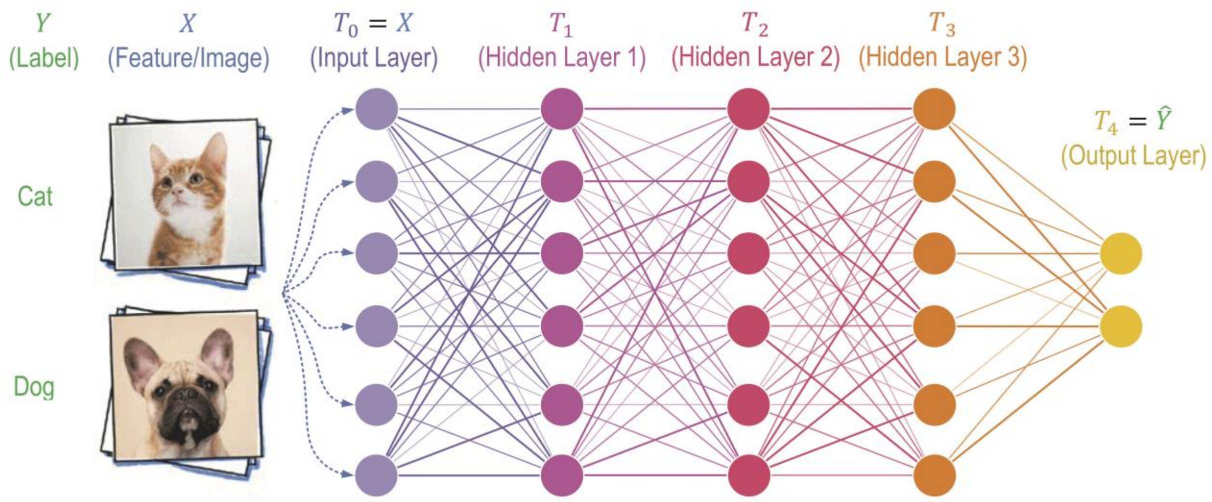
[1] W. Jiang, et al. A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage, Nature Communications

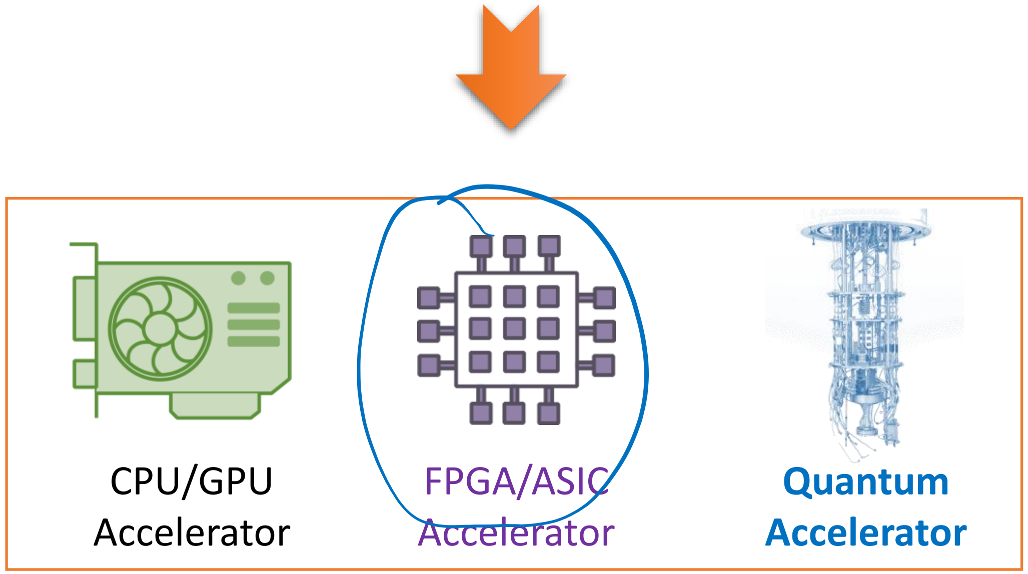
[2] W. Jiang, et al. When Machine Learning Meets Quantum Computers: A Case Study, ASP-DAC'21

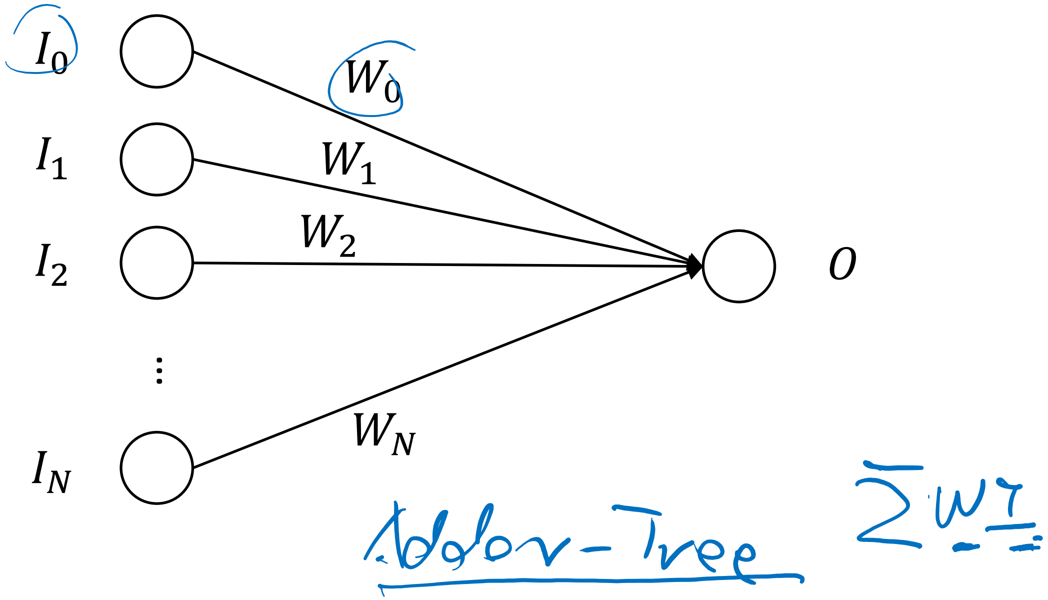
[3] W. Jiang, Github Tutorial on Implementing Machine Learning to Quantum Computer using IBM Qiskit

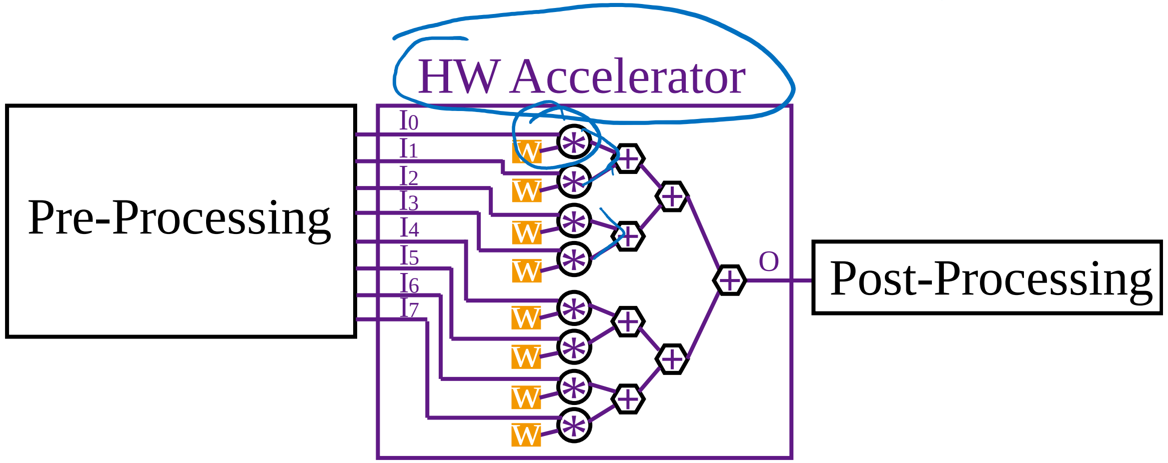
Neural Network Accelerator Design on Classical Hardware

UNIVERSITY OF NOTRE DAME

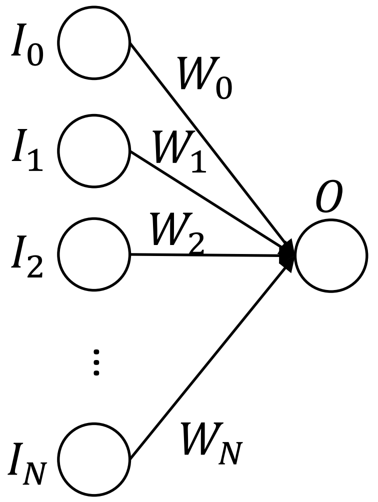


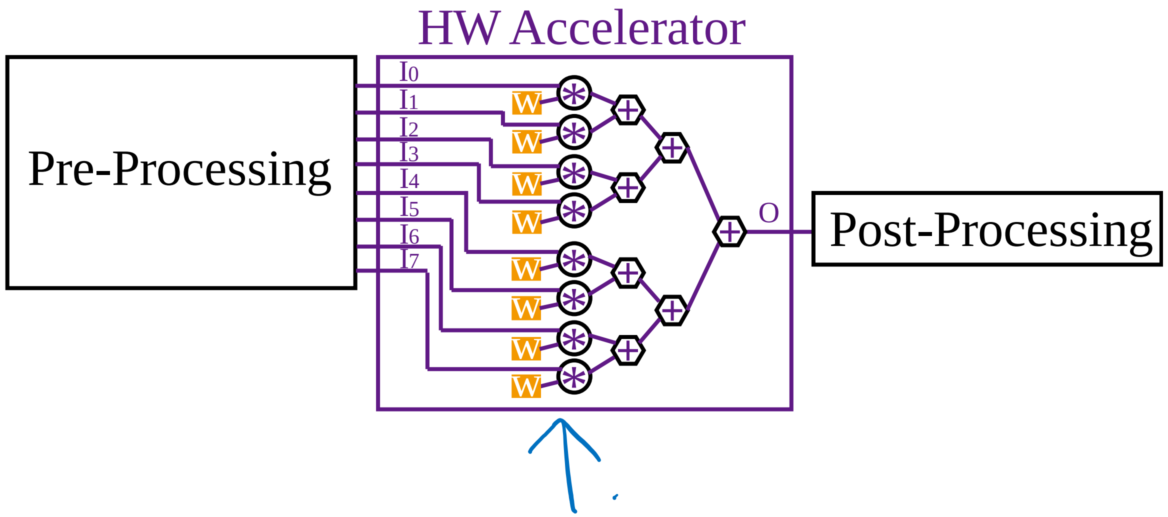


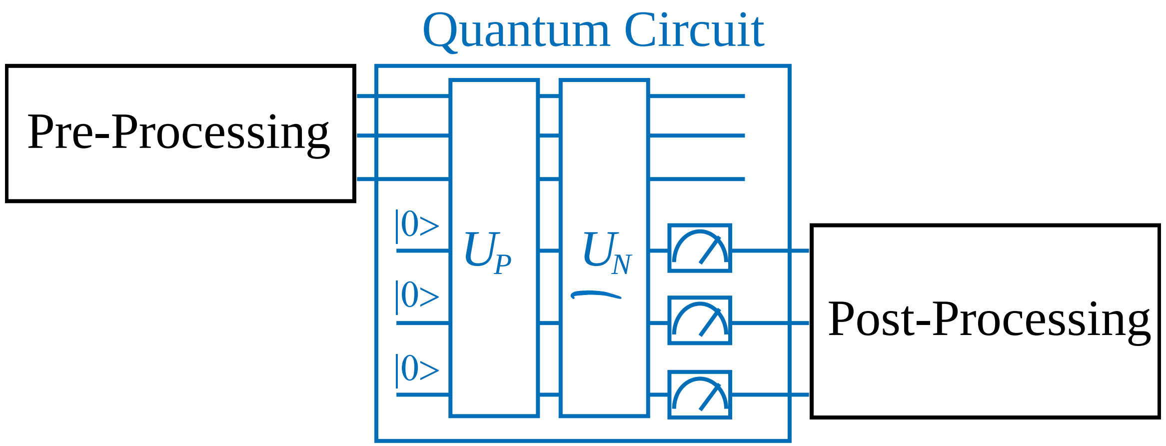




Neural Network Accelerator Design from Classical to Quantum Computing







(1) data pre-P PreP

(2) HW Arc

(3) data postP

(2)

(2.1) Quantum-State-preps U_P

(2.2) Neural comp. U_N

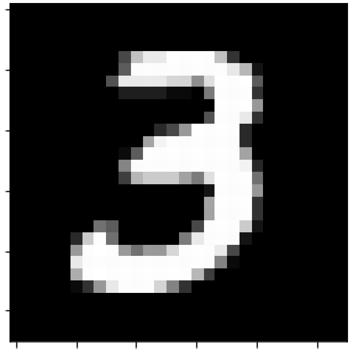
(2.3) Measurement M.

PreP + U_P + U_N + M + PostP

PreP + U_P + U_N + *M* + *PostP* : Data Pre-Processing

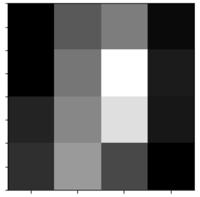
$2^4 = 16$

- **Given:** (1) 28×28 image, (2) the number of qubits to encode data (say Q=4 qubits in the example)
- **Do:** (1) downsampling from 28×28 to $2^Q = 16 = 4 \times 4$; (2) converting data to be the state vector in a unitary matrix
- **Output:** A unitary matrix, $M_{16\times16}$



Step 1: Downsampling

From 28×28 to 4×4



$$\begin{bmatrix} 0.0039 & 0.2118 & 0.2941 & 0.0275 \\ 0.0039 & 0.2784 & 0.5961 & 0.0667 \\ 0.0863 & 0.3176 & 0.5216 & 0.0588 \\ 0.1137 & 0.3608 & 0.1725 & 0.0039 \end{bmatrix}$$

$\sum A^2 = 1$

$$\begin{bmatrix} 0.0039 & 0.2118 & 0.2941 & 0.0275 \\ 0.0039 & 0.2784 & 0.5961 & 0.0667 \\ 0.0863 & 0.3176 & 0.5216 & 0.0588 \\ 0.1137 & 0.3608 & 0.1725 & 0.0039 \end{bmatrix}$$

Step 2: Formulate Unitary Matrix

Applying SVD method
(See Listing 1 in Tutorial Paper)

Unitary matrix: $M_{16\times16}$

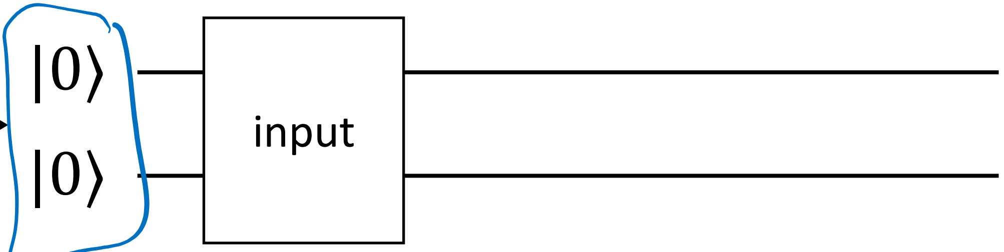
[Tutorial Paper] W. Jiang, et al. When Machine Learning Meets Quantum Computers: A Case Study, ASP-DAC'21

PreP + U_P + U_N + *M* + *PostP* --- Data Encoding / Quantum State Preparation

- **Given:** The unitary matrix provided by *PreP,* $M_{16\times16}$
- **Do:** Quantum-State-Preparation, encoding data to qubits
- **Verification:** Check the amplitude of states are consistent with the data in the unitary matrix, $M_{16\times16}$

Let's use a 2-qubit system as an example to encode a matrix $M_{4\times4}$

$$\begin{bmatrix} 0.3 & 0.5 \\ 0.7 & 0.9 \end{bmatrix} \xrightarrow{PreP} \overset{\textbf{data\_matrix}}{\begin{bmatrix} \mathbf{0.2343} & X & X & X \\ \mathbf{0.3904} & X & X & X \\ \mathbf{0.5466} & X & X & X \\ \mathbf{0.7028} & X & X & X \end{bmatrix}} \xrightarrow{U_P}$$



State Transition:

data_matrix $|00\rangle$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.23 \\ 0.39 \\ 0.54 \\ 0.70 \end{pmatrix}$$

IBM Qiskit Implementation:

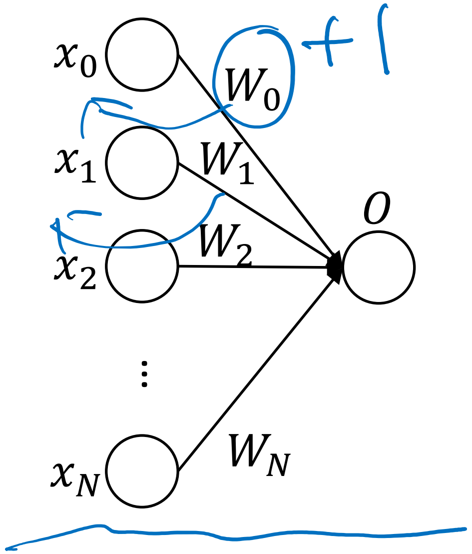
inp = QuantumRegister(~~4~~ 2, "in_qubit")
circ = QuantumCircuit(inp)
iniG = **UnitaryGate(data_matrix, label="input")**
circ.append(iniG, inp[0:~~4~~])

Initization

Tutorial 1: $PreP + U_P + U_N + M + PostP$

https://github.com/weiwenjiang/QML_tutorial/blob/main/Tutorial_1_DataPreparation.ipynb

PreP + U_P + U_N + M + PostP --- 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 the qubits, such that it performs $\frac{(\boldsymbol{x} * \boldsymbol{w})^2}{\|\boldsymbol{x}\|}$.
- **Verification:** Whether the output data of quantum circuit and the output computed using torch on classical computer are the same.

binary

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

- **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: $n = \left[\frac{\sum_i (m_i)}{\sqrt{\|x\|}}\right]$

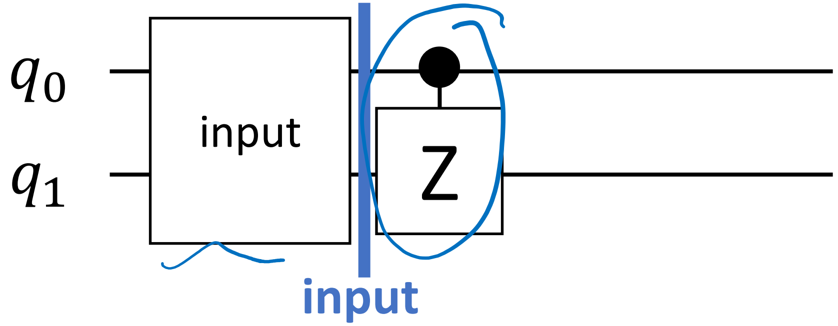
Step 3: $O = n^2$

$PreP + U_P + U_N + M + PostP$ --- Neural Computation: Step 1

Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

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



CZ

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

Input

a_0	$\|00\rangle$
a_1	$\|01\rangle$
a_2	$\|10\rangle$
a_3	$\|11\rangle$

Output

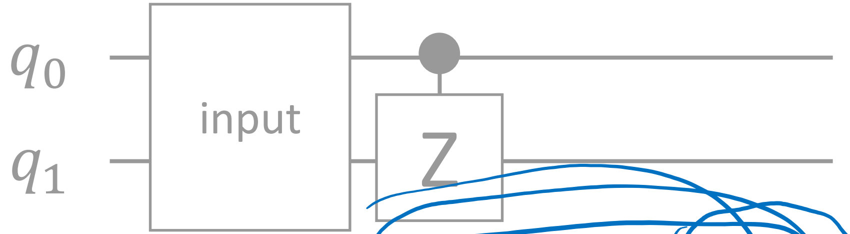
a_0	$\|00\rangle$
a_1	$\|01\rangle$
a_2	$\|10\rangle$
$-a_3$	$\|11\rangle$

$PreP + U_P + U_N + M + PostP$ --- Neural Computation: Step 1

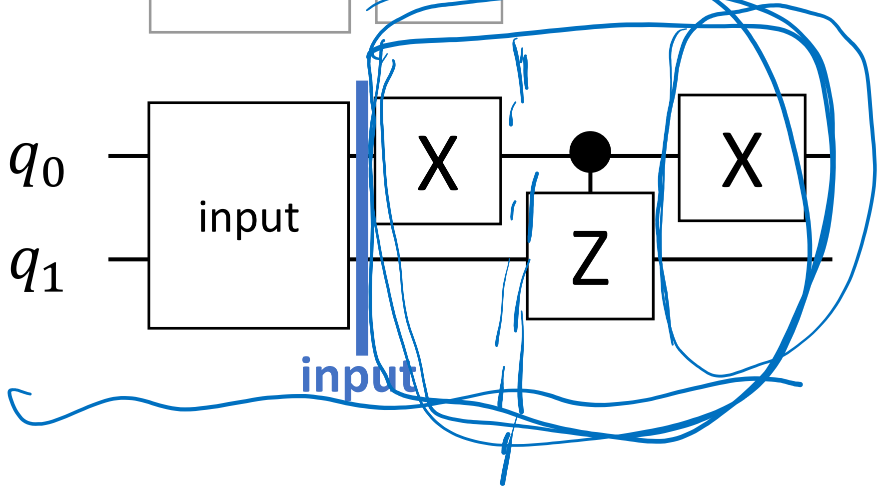
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

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



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



Input	
a_0	$\|00\rangle$
a_1	$\|01\rangle$
a_2	$\|10\rangle$
a_3	$\|11\rangle$

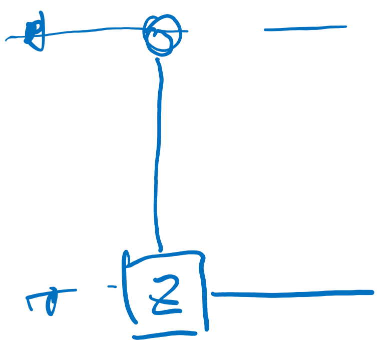
XI

a_1	$\|00\rangle$
a_0	$\|01\rangle$
a_3	$\|10\rangle$
$-a_2$	$\|11\rangle$

XI

Output	
a_0	$\|00\rangle$
a_1	$\|01\rangle$
$-a_2$	$\|10\rangle$
a_3	$\|11\rangle$

CZ



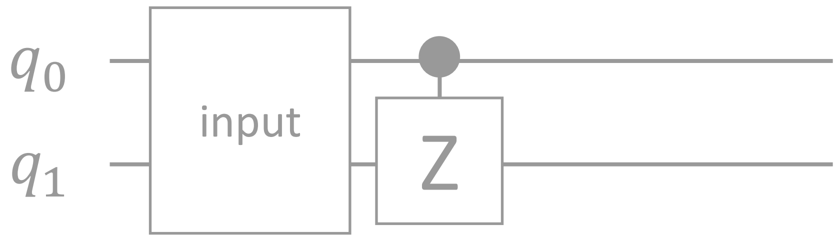
$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & 1 \end{pmatrix}$$

$PreP + U_P + U_N + M + PostP$ --- Neural Computation: Step 1

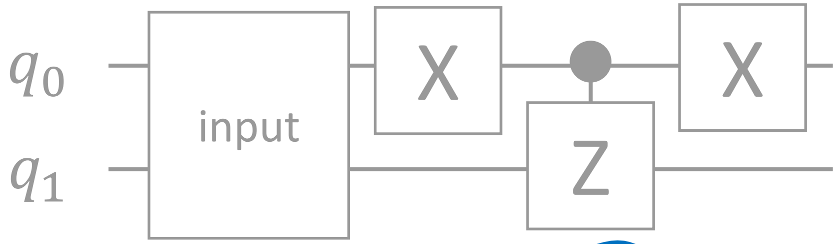
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

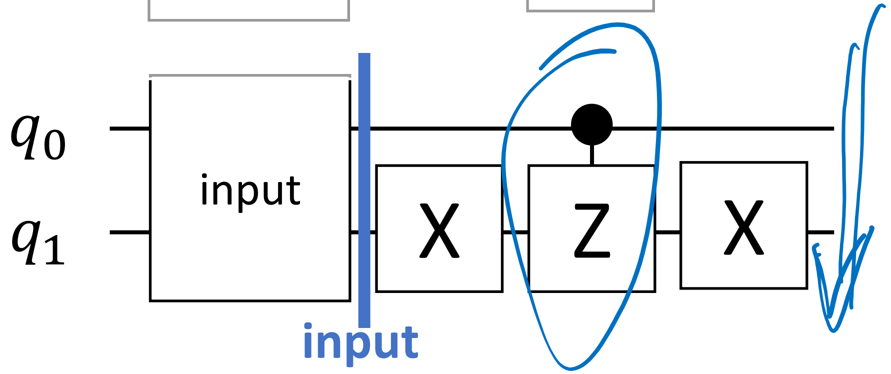
$$w = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$



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



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



Input

a_0	$\|00\rangle$
a_1	$\|01\rangle$
a_2	$\|10\rangle$
a_3	$\|11\rangle$

XI

a_2	$\|00\rangle$
a_3	$\|01\rangle$
a_0	$\|10\rangle$
$\boldsymbol{a_1}$	$\boldsymbol{\|11\rangle}$

CZ

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} a_2 \\ a_3 \\ a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} a_2 \\ a_3 \\ a_0 \\ -a_1 \end{bmatrix}$$

XI Output

a_0	$\|00\rangle$
$\boldsymbol{-a_1}$	$\boldsymbol{\|01\rangle}$
a_2	$\|10\rangle$
a_3	$\|11\rangle$

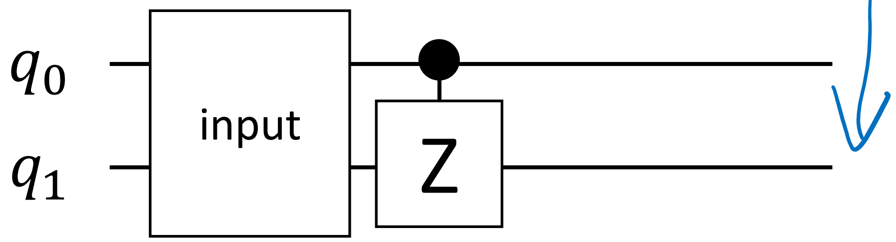
PreP + U_P + U_N + M + PostP --- Neural Computation: Step 1

Step 1: $m_i = x_i \times w_i$

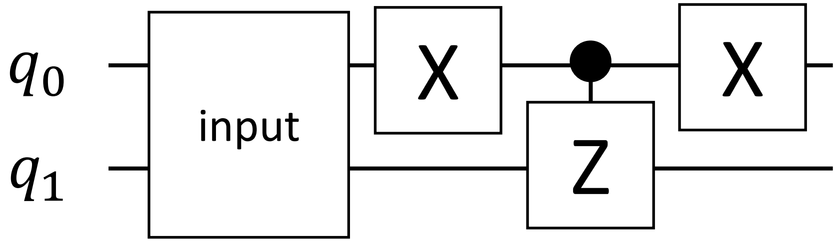
EX: 4 input data on 2 qubits

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

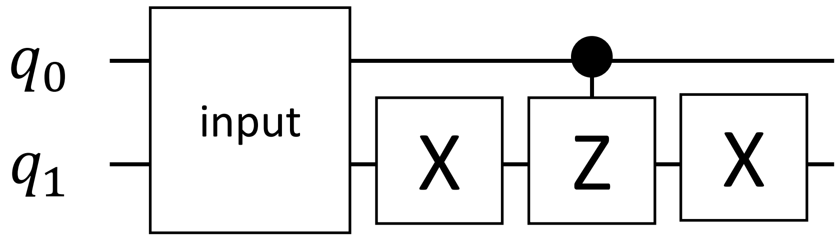
$$\boldsymbol{w} = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$

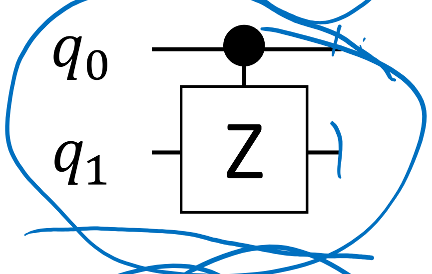


$$\boldsymbol{w} = \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \end{bmatrix}$$

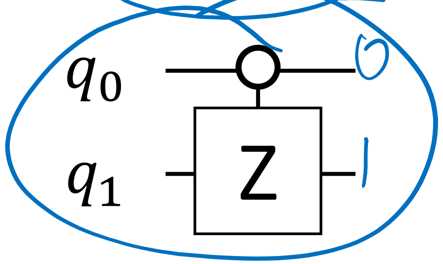


$$\boldsymbol{w} = \begin{bmatrix} +1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

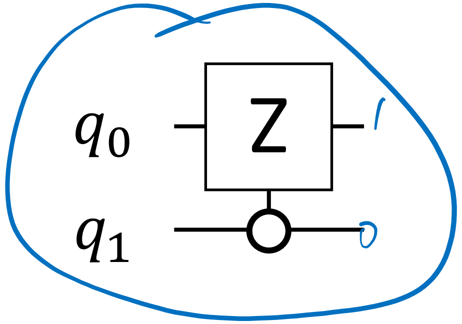




Flip the sign of $|11\rangle$



Flip the sign of $|10\rangle$



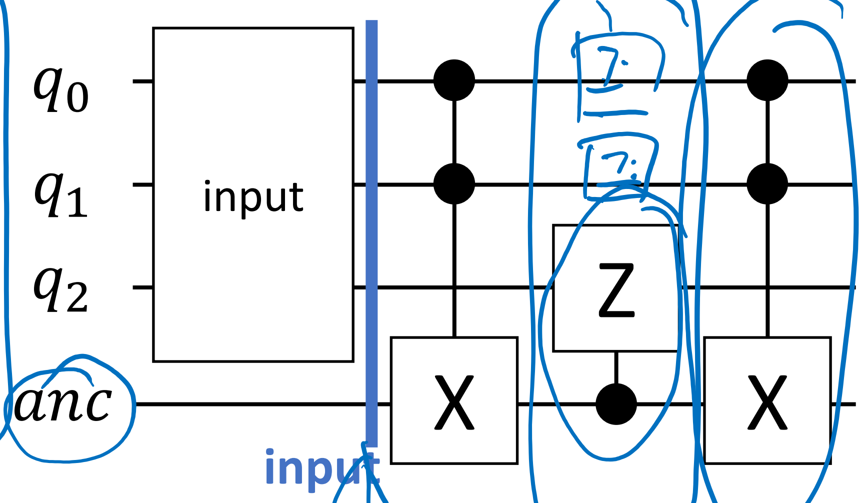
Flip the sign of $|01\rangle$

$PreP + U_P + U_N + M + PostP$ --- Neur… ation

Step 1: $m_i = x_i \times w_i$

EX: 8 input data on 3 qubits

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



IICZ:

$$\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} = \begin{pmatrix} I_4 & & & \\ & I_4 & & \\ & & I_4 & \\ & & & -I_4 \end{pmatrix}$$

Input

a_0	$\vert 0000\rangle$
a_1	$\vert 0001\rangle$
a_2	$\vert 0010\rangle$
a_3	$\vert 0011\rangle$
a_4	$\vert 0100\rangle$
a_5	$\vert 0101\rangle$
a_6	$\vert 0110\rangle$
a_7	$\vert 0111\rangle$
0	$\vert 1000\rangle$
0	$\vert 1001\rangle$
0	$\vert 1010\rangle$
0	$\vert 1011\rangle$
0	$\vert 1100\rangle$
0	$\vert 1101\rangle$
0	$\vert 1110\rangle$
0	$\vert 1111\rangle$

CCIX

a_0	$\vert 0000\rangle$
a_1	$\vert 0001\rangle$
a_2	$\vert 0010\rangle$
0	$\vert 0011\rangle$
a_4	$\vert 0100\rangle$
a_5	$\vert 0101\rangle$
a_6	$\vert 0110\rangle$
0	$\vert 0111\rangle$
0	$\vert 1000\rangle$
0	$\vert 1001\rangle$
0	$\vert 1010\rangle$
a_3	$\vert 1011\rangle$
-0	$\vert 1100\rangle$
-0	$\vert 1101\rangle$
-0	$\vert 1110\rangle$
$-a_7$	$\vert 1111\rangle$

CCIX

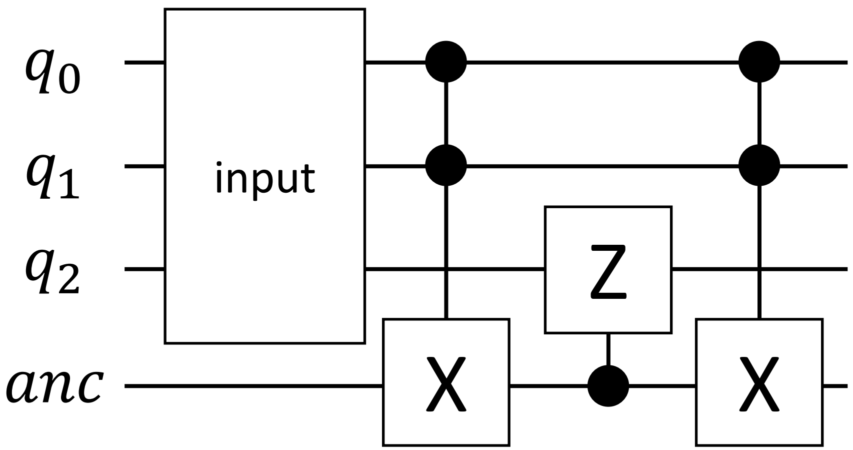
a_0	$\vert 0000\rangle$
a_1	$\vert 0001\rangle$
a_2	$\vert 0010\rangle$
a_3	$\vert 0011\rangle$
a_4	$\vert 0100\rangle$
a_5	$\vert 0101\rangle$
a_6	$\vert 0110\rangle$
$-a_7$	$\vert 0111\rangle$
0	$\vert 1000\rangle$
0	$\vert 1001\rangle$
0	$\vert 1010\rangle$
0	$\vert 1011\rangle$
0	$\vert 1100\rangle$
0	$\vert 1101\rangle$
0	$\vert 1110\rangle$
0	$\vert 1111\rangle$

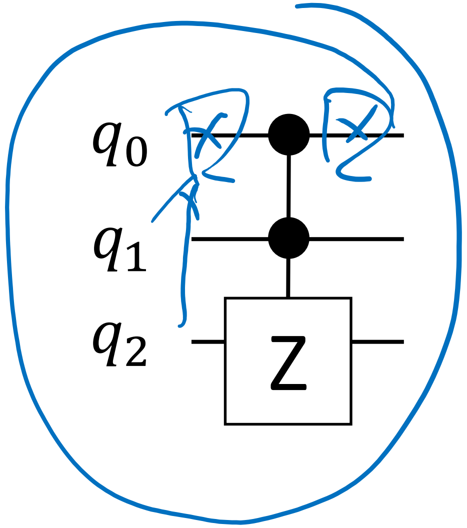
PreP + U_P + U_N + *M* + *PostP* --- Neural Computation: Step 1

Step 1: $m_i = x_i \times w_i$

EX: 8 input data on 3 qubits

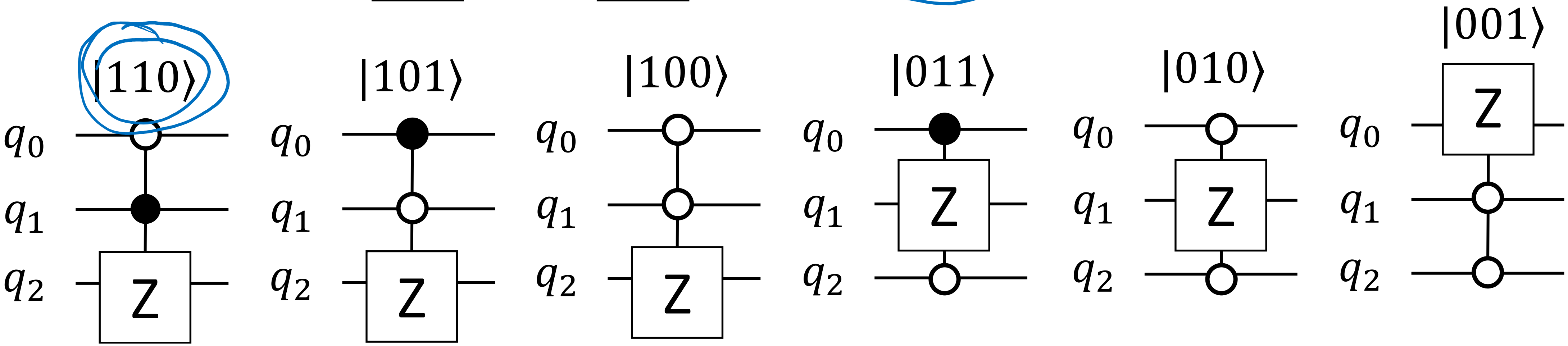
$$\boldsymbol{w} = \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$$





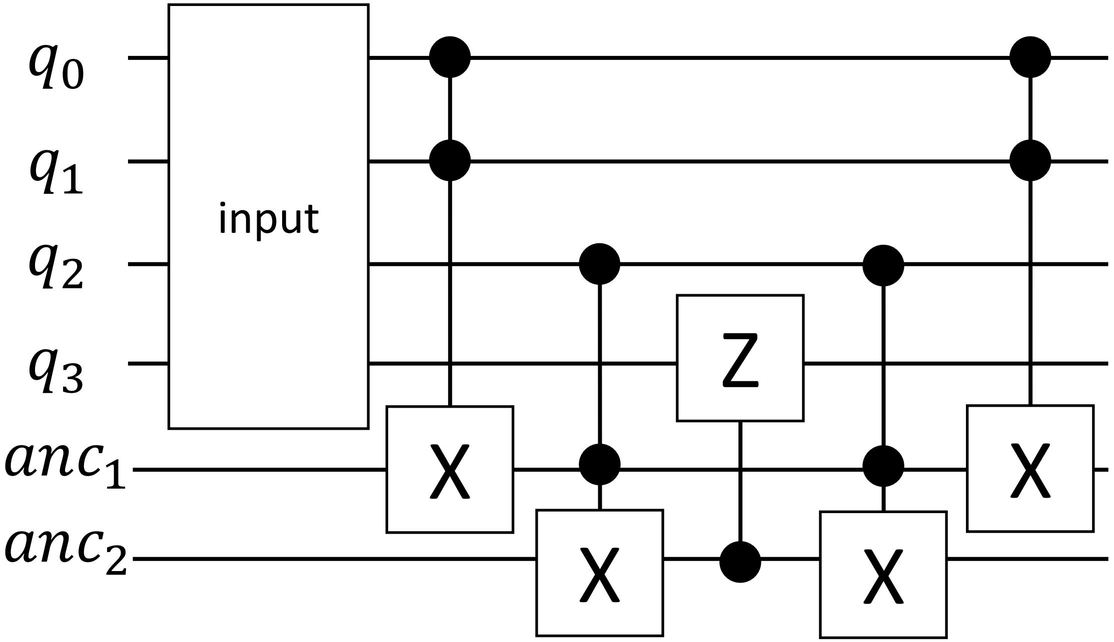
Flip the sign of $|111\rangle$

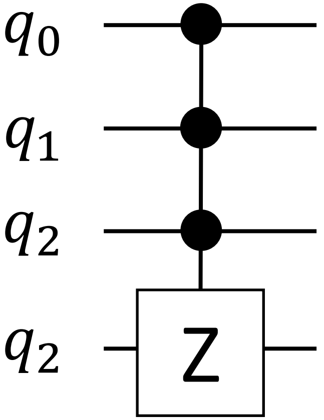
CCZ



Step 1: $m_i = x_i \times w_i$

EX: 16 input data on 4 qubits





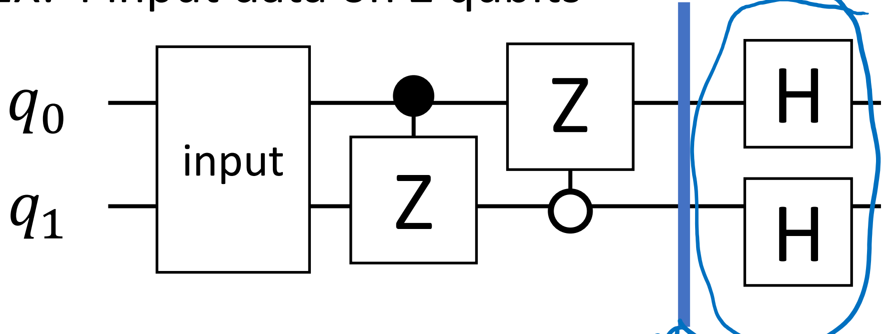
Flip the sign of $|1111\rangle$

CCCZ

PreP + U_P + $\boldsymbol{U_N}$ + *M* + *PostP* --- Neural Computation: Step 2

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

EX: 4 input data on 2 qubits



$H^{\otimes 2}$

$\frac{1}{2}$

Input

a_0	m_0	$\|00\rangle$
$-a_1$	m_1	$\|01\rangle$
a_2	m_2	$\|10\rangle$
$-a_3$	m_3	$\|11\rangle$

=

Output

$\sum_i (m_i)/\sqrt{\|x\|}$	$\|00\rangle$
Do not care	$\|01\rangle$
Do not care	$\|10\rangle$
Do not care	$\|11\rangle$

$\frac{1}{\sqrt{\|x\|}} \sum m_i$

$\|x\| = 2^n$

$\left(\frac{1}{\sqrt{2}}\right)^n = \frac{1}{\sqrt{2^n}}$

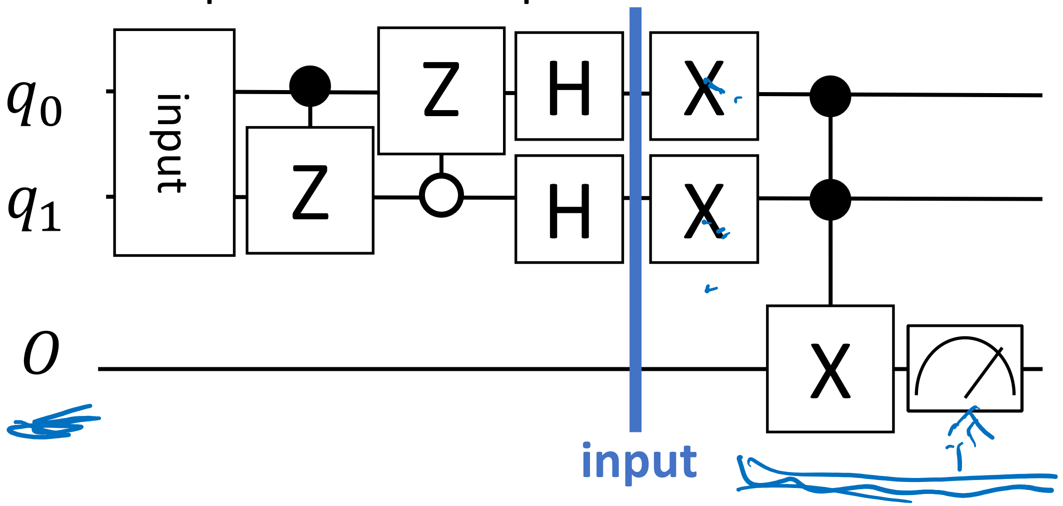
$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$

$= \frac{1}{\sqrt{\|x\|}}$

PreP + U_P + U_N + *M* + *PostP* -- Neural Computation (Step 3) & Measurement

Step 3: $O = n^2$

EX: 4 input data on 2 qubits



Input

$\sum_i (m_i)/\sqrt{\lVert x \rVert}$	$\lvert 000\rangle$
Do not care	$\lvert 001\rangle$
Do not care	$\lvert 010\rangle$
Do not care	$\lvert 011\rangle$
0	$\lvert 100\rangle$
0	$\lvert 101\rangle$
0	$\lvert 110\rangle$
0	$\lvert 111\rangle$

$X^{\otimes 2}$

Do not care	$\lvert 000\rangle$
Do not care	$\lvert 001\rangle$
Do not care	$\lvert 010\rangle$
$\sum_i (m_i)/\sqrt{\lVert x \rVert}$	$\lvert 011\rangle$
0	$\lvert 100\rangle$
0	$\lvert 101\rangle$
0	$\lvert 110\rangle$
0	$\lvert 111\rangle$

CCX

Do not care	$\lvert 000\rangle$
Do not care	$\lvert 001\rangle$
Do not care	$\lvert 010\rangle$
0	$\lvert 011\rangle$
0	$\lvert 100\rangle$
0	$\lvert 101\rangle$
0	$\lvert 110\rangle$
$\sum_i (m_i)/\sqrt{\lVert x \rVert}$	$\lvert 111\rangle$

Output

$$P\{O = \lvert 1\rangle\} = P\{\lvert 100\rangle\} + P\{\lvert 101\rangle\} + P\{\lvert 110\rangle\} + P\{\lvert 111\rangle\} = \left[\frac{\sum_i (m_i)}{\sqrt{\lVert x \rVert}}\right]^2$$

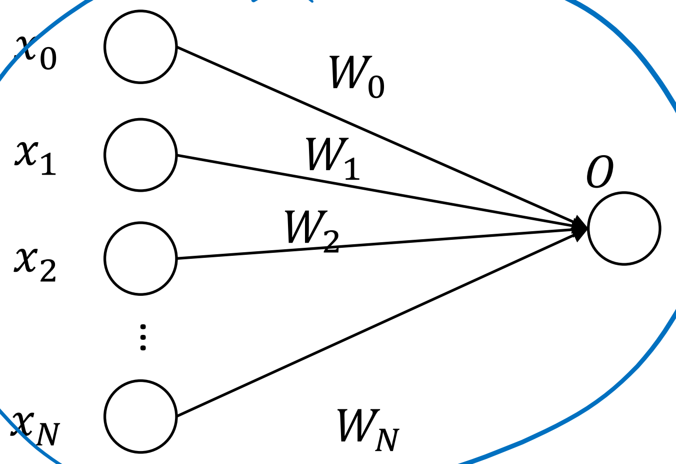
$P\{O = \lvert 1\rangle\} = n^2 + 0^2 + 0^2 + 0^2 = n^2$

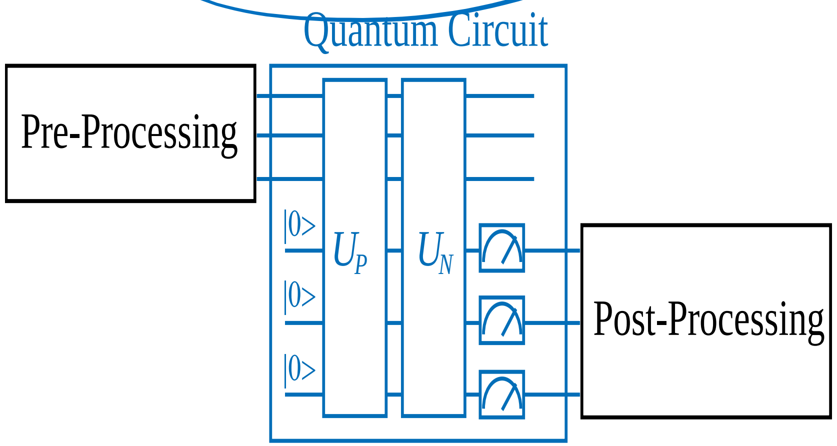
Tutorial 2: $PreP + U_P + U_N + M + PostP$

https://github.com/weiwenjiang/QML_tutorial/blob/main/Tutorial_2_Hidden_NeuralComp.ipynb

Takeaway: A Framework and Detailed Design for Goal 1

Goal 1: Correctly Implement!





Goal 2: Efficiently Implement!

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

where δ is a quadratic function

Classical C

Next Course on Friday 01/08/21

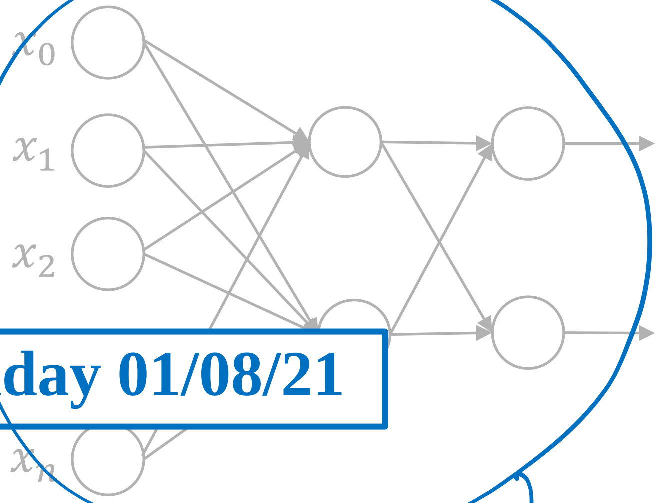
Complexity of $\boldsymbol{O(N)}$

Quantum Computing:

Can we reduce complexity to

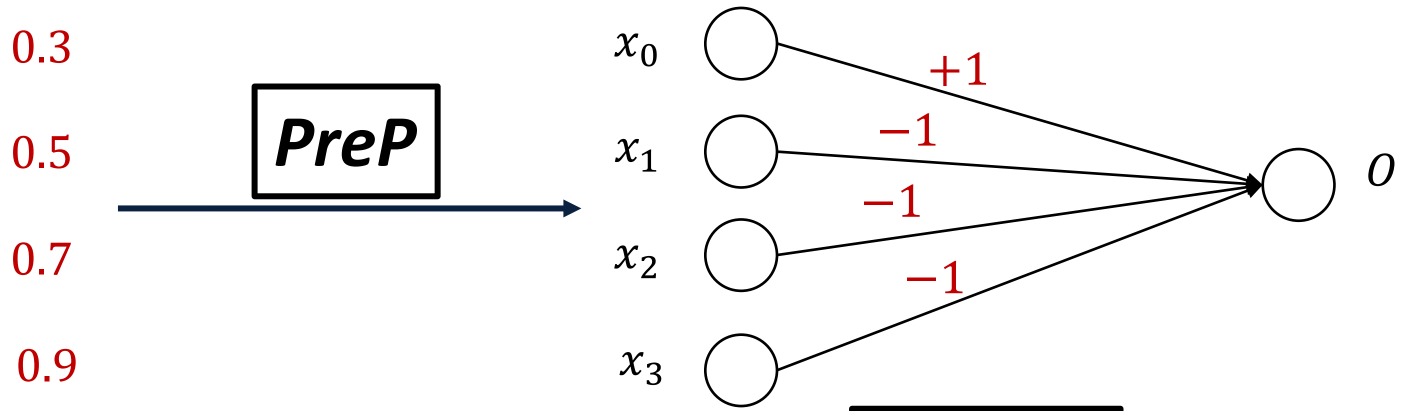
$\boldsymbol{O(ploylogN)}$, say $\boldsymbol{O(log^2 n)}$?

Goal 3: Scale-Up!

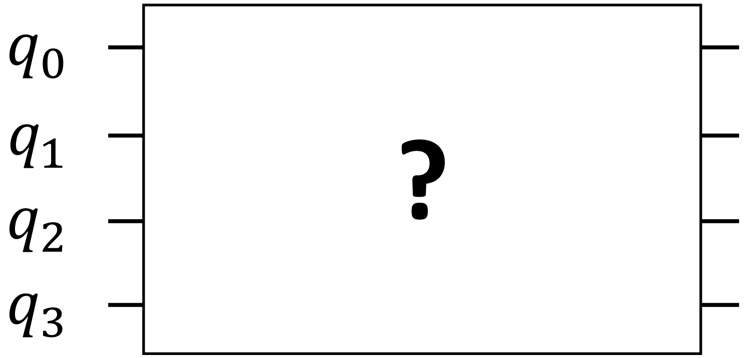


Have a Try on *PreP* + U_P + U_N + *M* + *PostP* !

Given inputs and weights



Output:



$\boldsymbol{O} = ??$

Send your answer to us to check whether it is correct!

wjiang2@nd.edu;
mcoffey1@nd.edu;
cmcdona8@nd.edu;

Thank You!

wjiang2@nd.edu