QuantumFlow: Co-Design Neural Network and Quantum Circuit towards Quantum Advantage

Weiwen Jiang, Ph.D.

Assistant Professor

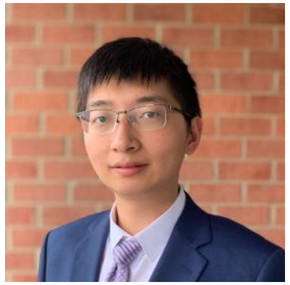
Electrical and Computer Engineering

George Mason University

wjiang8@gmu.edu

https://jqub.ece.gmu.edu

Speaker



Weiwen Jiang
Assistant Professor
Electrical and Computer Engineering (ECE)
George Mason University
Room3247, Nguyen Engineering Building
wjiang8@gmu.edu
(703)-993-5083
https://jqub.ece.gmu.edu/

- **Education Background**
  - **Chongqing University (2013-2019)**
  - **University of Pittsburgh (2017-2019)**
  - **University of Notre Dame (2019-2021)**
- **Research Interests**
  - **HW/SW Co-Design**
  - **Quantum Machine Learning**

First HW/SW Co-Design Framework using NAS

HW/SW Co-Design Framework
FNAS
[DAC'19*]
[TCAD'20*]

Application

- **Medical Imaging**: NAS for Medical Image Seg. [MICCAI'20]; 3D Cardiac MRI Seg. [ICCAD'20]
- **NLP (Transformer)**: FPGA [ICCD'20]; Mobile [DAC'21]; GPU [GLSVLSI'21]
- **Graph-Based**: Social Net [GLSVLSI'21]; Drug Discovery [ICCAD'21]

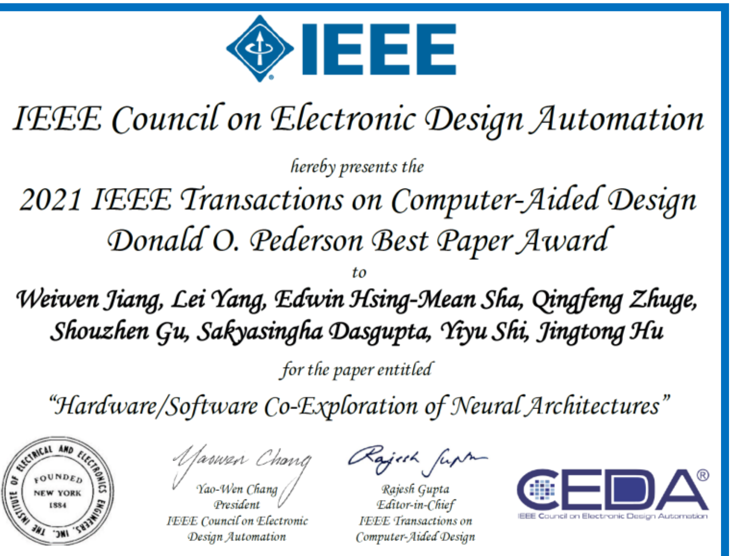
Algorithm

- **NAS Acc.**: HotNAS [CODES+ISSS'20]
- **Model Compression**: NAS for Quan. [ICCAD'19]; Compre.-Compilation [IJCAI'21]
- **Secure Infernece**: NASS [ECAI'20]; BUNET [MICCAI'20]

Hardware

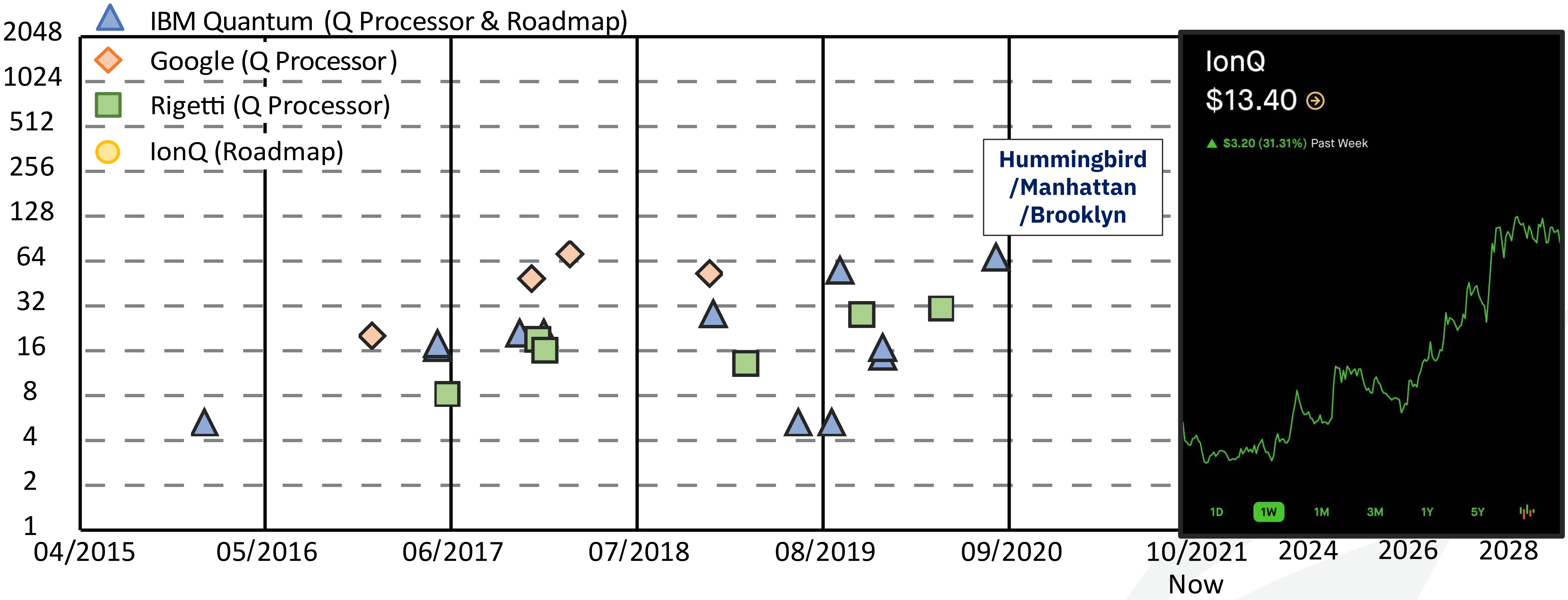
- **FPGA**: XFER [CODES+ISSS'19*]
- **ASIC**: NANDS [ASP-DAC'20*]; ASICNAS [DAC'20]
- **Computing-in-Memory**: Device-Circuit-Arch. [IEEE TC'20]

Best Paper Award:


IEEE
IEEE Council on Electronic Design Automation
hereby presents the
2021 IEEE Transactions on Computer-Aided Design
Donald O. Pederson Best Paper Award
to
Weiwen Jiang, Lei Yang, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Shouzhen Gu, Sakyasingha Dasgupta, Yiyu Shi, Jingtong Hu
for the paper entitled
"Hardware/Software Co-Exploration of Neural Architectures"
Yao-Wen Chang, President, IEEE Council on Electronic Design Automation
Rajesh Gupta, Editor-in-Chief, IEEE Transactions on Computer-Aided Design
CEDA

Best Paper Nominations: 58 Design Automation Conference; Embedded Systems Week; Asia South Pacific Design Automation Conference

Consistently Increasing Qubits in Quantum Computers



The Power of Quantum Computers: Qubit

Classical Bit

$$X = 0 \ \textbf{\textit{or}} \ 1$$

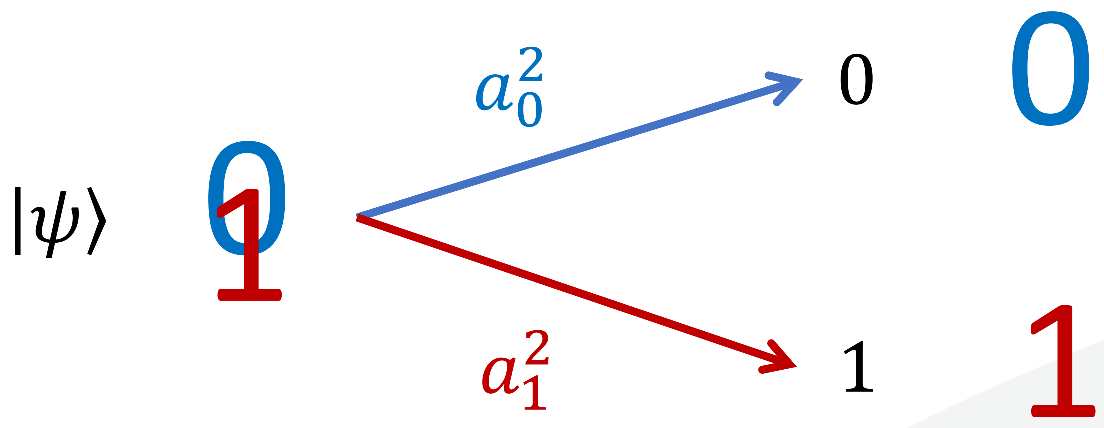
Quantum Bit (Qubit)

$$|\psi\rangle = |0\rangle \ \textbf{\textit{and}} \ |1\rangle$$

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

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

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

2 Classical Bits

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

n bits for 1 value

$\boldsymbol{x \in [0, 2^n - 1]}$

---

2 Qubits

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

n bits for 2^n values

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

Qubits: $\boldsymbol{q_0, q_1}$

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

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

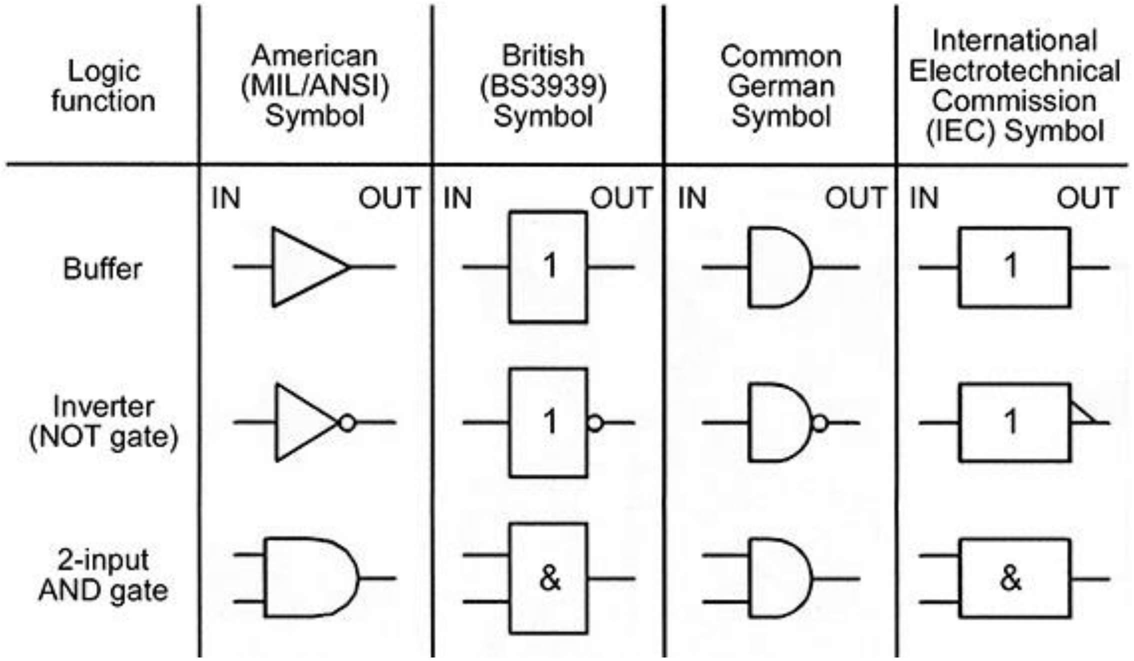
$$\boldsymbol{|q_0, q_1\rangle = |q_0\rangle \otimes |q_1\rangle}$$

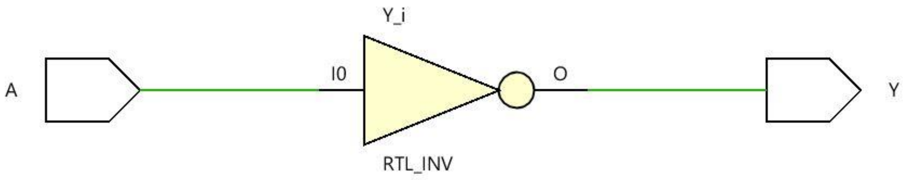
$$\boldsymbol{= c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle}$$

$$\boldsymbol{|q_0, q_1\rangle = |q_0\rangle \otimes |q_1\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}}$$

$$\boldsymbol{= \begin{pmatrix} a_0 \times \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \\ a_1 \times \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix} = \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix}}$$

Computation: Logic Gates vs. Quantum Logic Gates

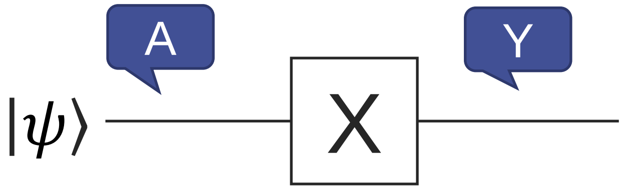




$$Y = \bar{A}$$

A	Y
0	1
1	0

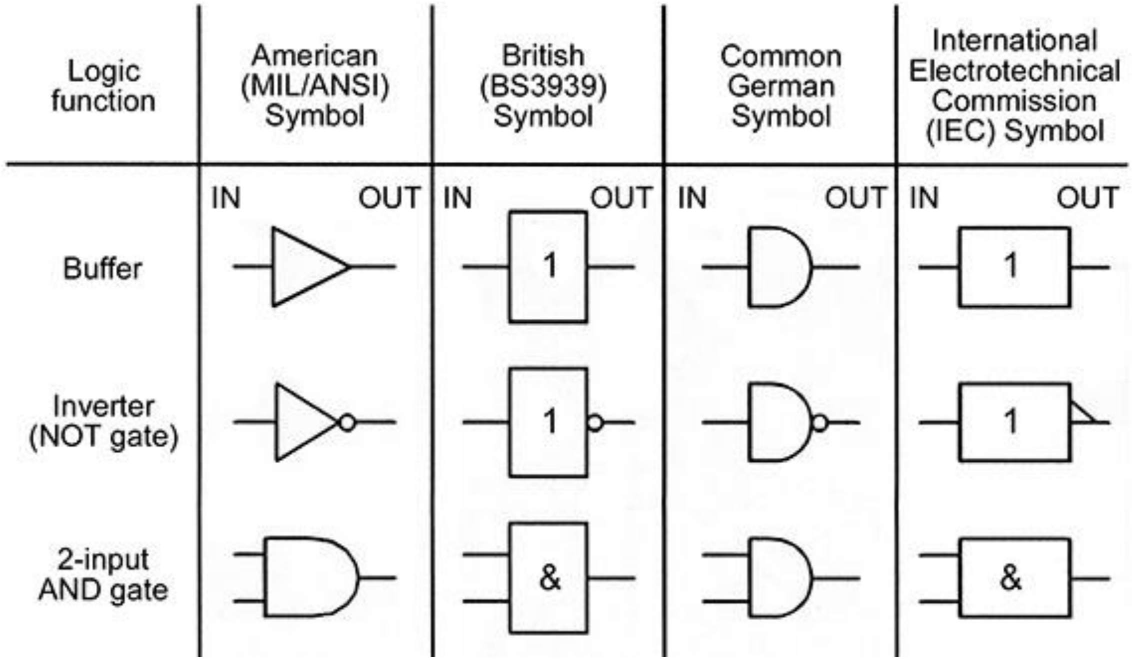
Operator	Gate(s)		Matrix
Pauli-X (X)	X	⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

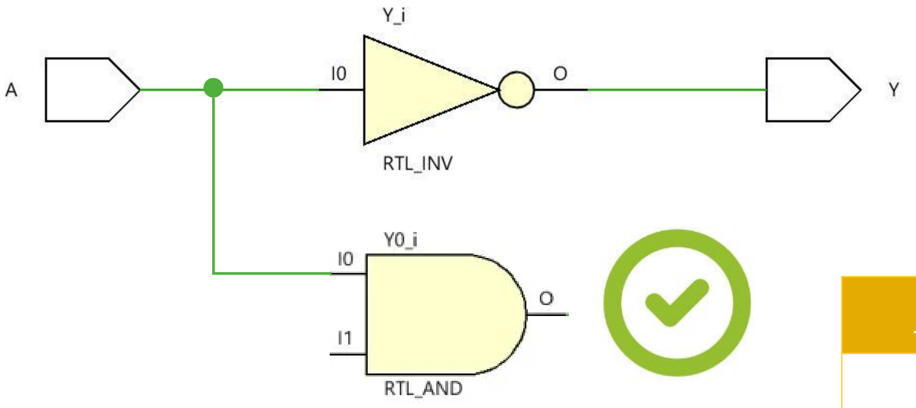


$$|Y\rangle = X \times |A\rangle$$

$$\text{Y} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \text{X} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \text{A} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

Computation: Logic Gates vs. Quantum Logic Gates

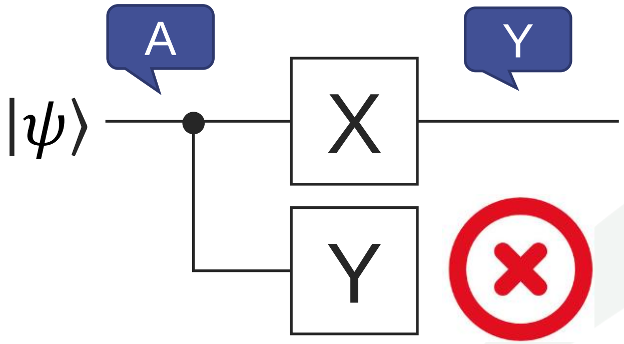




$$Y = \bar{A}$$

A	Y
0	1
1	0

Operator	Gate(s)		Matrix
Pauli-X (X)	X	⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

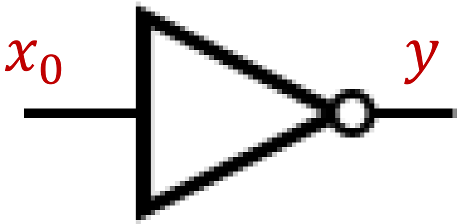


$$|Y\rangle = X \times |A\rangle$$

$$\text{Y} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \text{X} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \text{A} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

Single-Qubit Gates and Superposition

Single-bit Gate

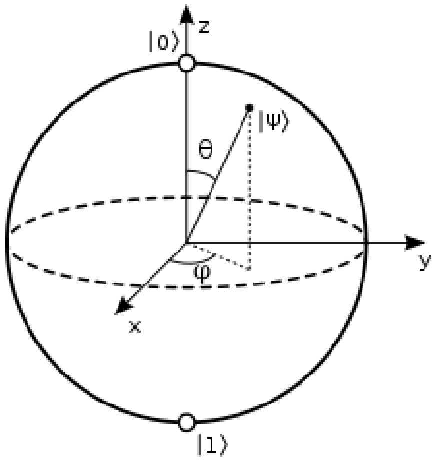


Not Gate

x_0	y
0	1
1	0

Single-Qubit Gates

- **Pauli operators: X, Y, Z Gates**
- **Hadamard gate: H Gate**
- General gate: U Gate



$|0\rangle$ — X — $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$|0\rangle$ — H — $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

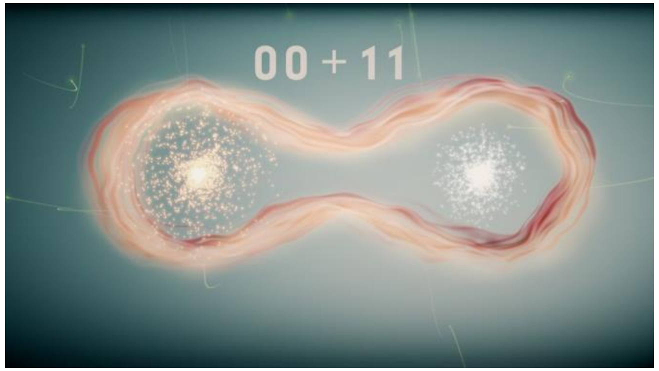
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

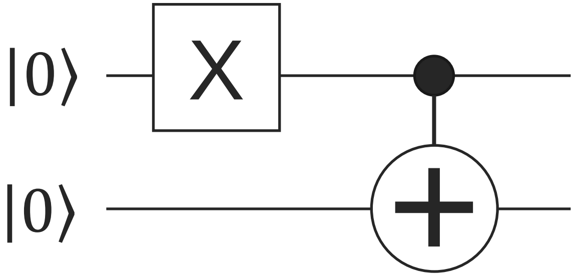
$$|0\rangle \rightarrow |1\rangle$$

Multi-Qubit Gates and Entanglement

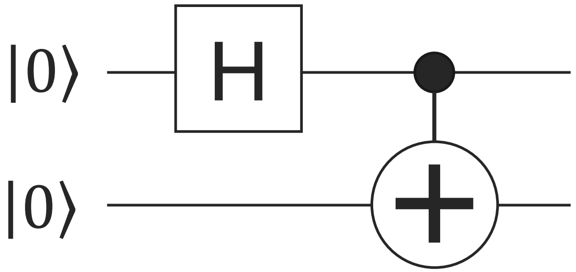
- Multi-Qubit Gates
  - **Controlled-Pauli gates**
  - **Toffoli gate or CCNOT**
  - ……

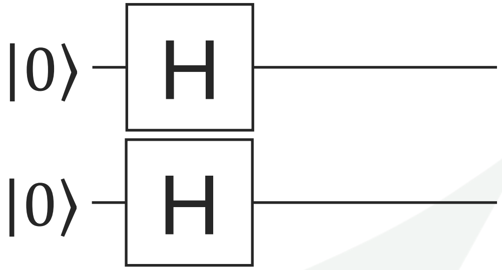


$$\frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\begin{matrix}|00\rangle\\|01\rangle\\|10\rangle\\|11\rangle\end{matrix}$$



$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$$





$$|10\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad |11\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$CNOT \times (H \otimes I) \times |00\rangle = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \times \frac{1}{\sqrt{2}}\begin{bmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{bmatrix}$$

$$\times |00\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1&0&1&0\\0&1&0&1\\0&1&0&-1\\1&0&-1&0\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}\begin{matrix}\mathbf{|00\rangle}\\|01\rangle\\|10\rangle\\\mathbf{|11\rangle}\end{matrix}$$

$$CNOT \times |10\rangle = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix} \times \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Hands-On Tutorial (1)
Basic Quantum Gates

Outline

- Background
- Co-Design: from Classical to Quantum
- QuantumFlow
  - Motivation
  - General Framework for Quantum-Based Neural Network Accelerator
  - Co-Design toward Quantum Advantage
- Recent works and conclusion

Co-Design

Given:

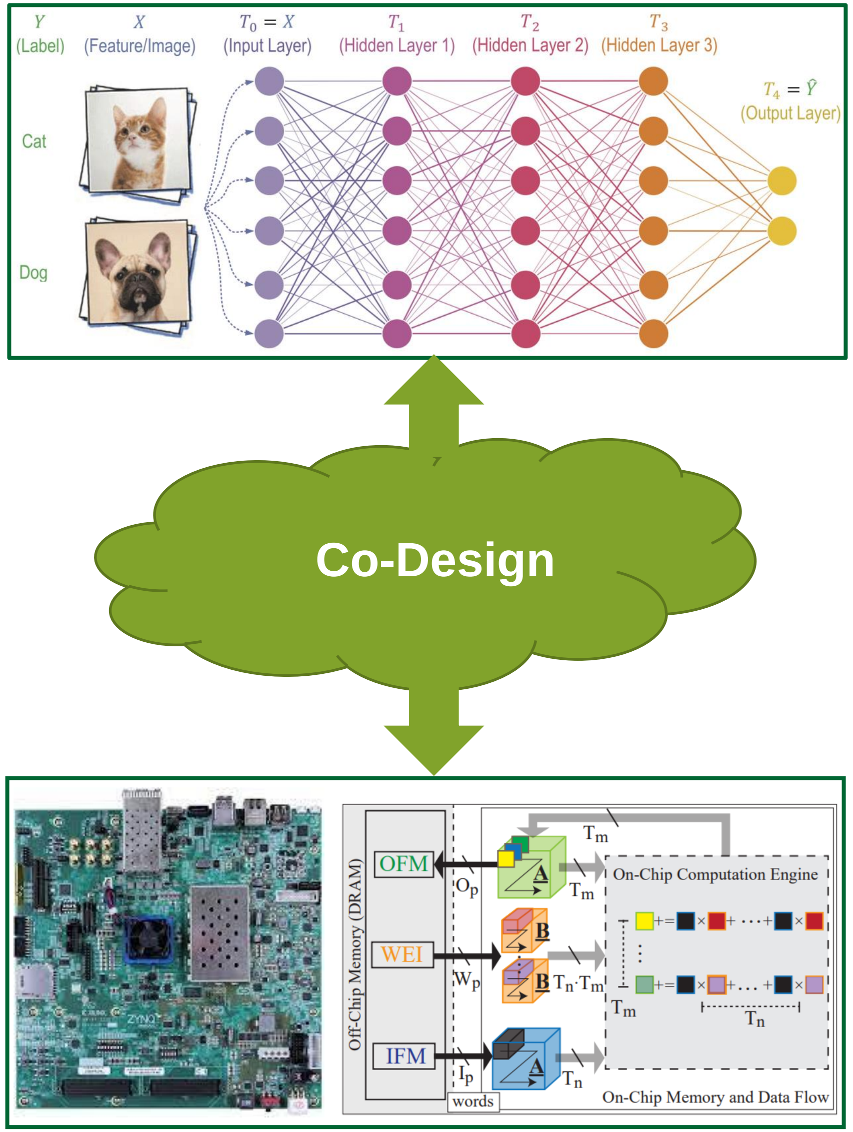
- Dataset (e.g., ImageNet)
- ML Task (e.g., classification)
- HW (e.g., FPGA spec.)

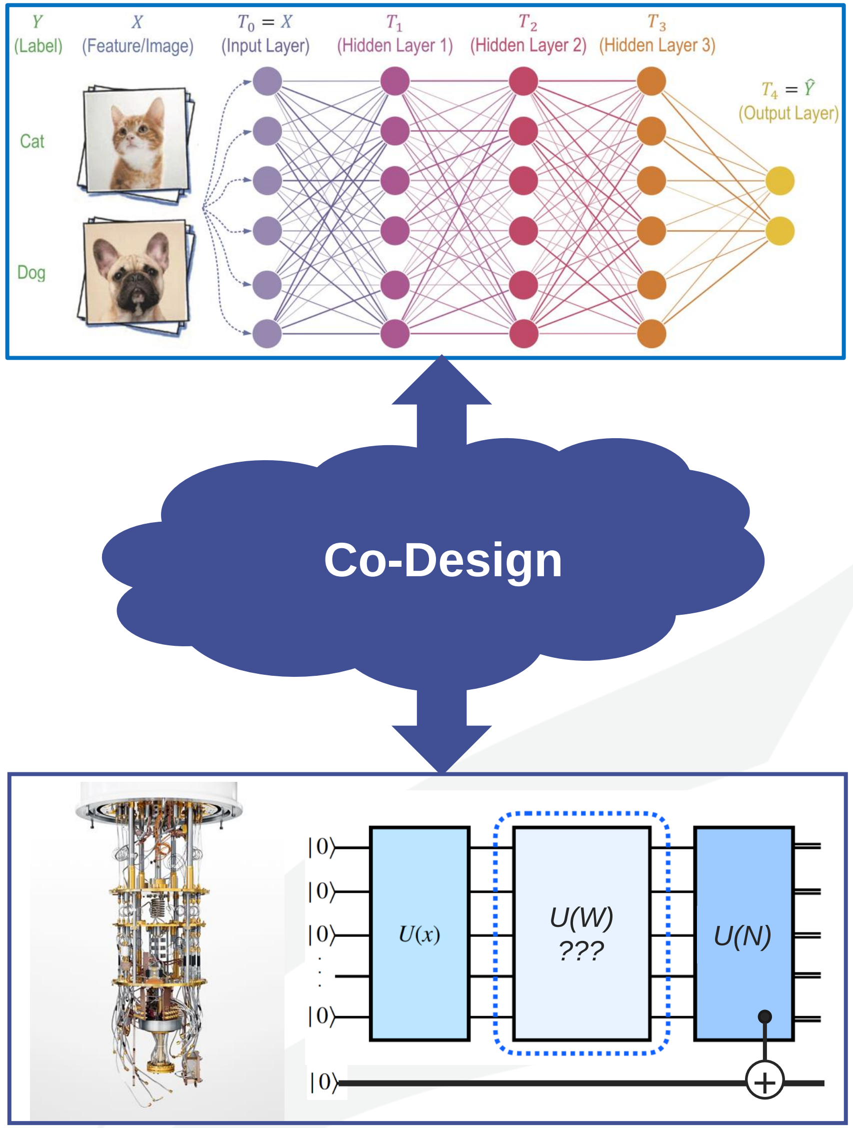
Do:

- Neural network design
- FPGA design

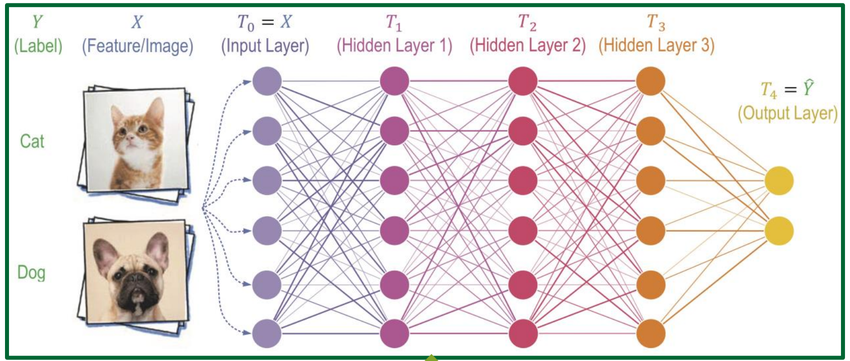
Objective:

- Accuracy
- Latency
- Energy
- …

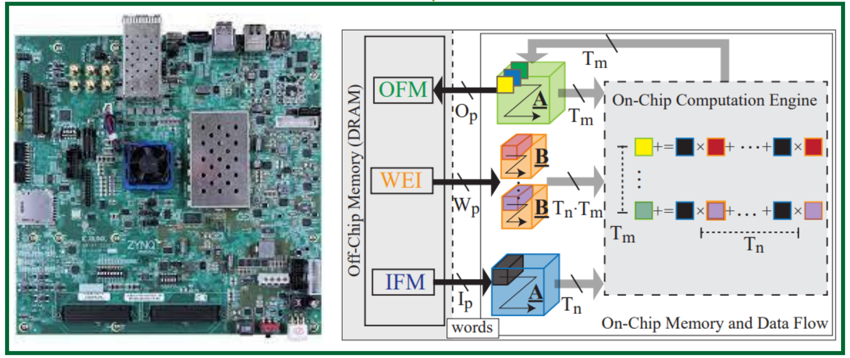




My Previous Background: Co-Design of Neural “Architectures”



Co-Design



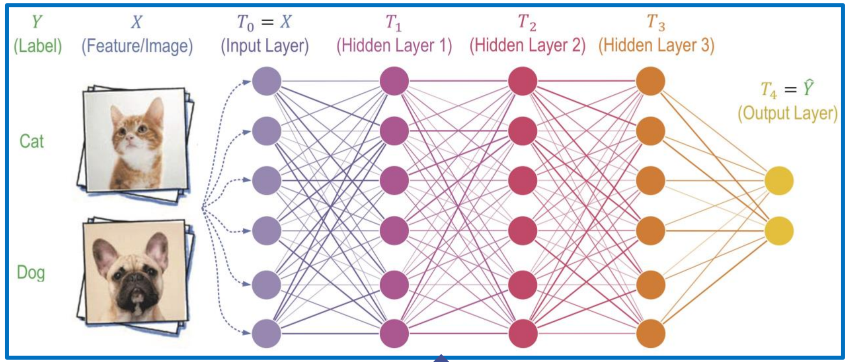
- **What is the best Neural Network Architecture for FPGAs**
- **Model optimization (pruning and quantization)?**
- **Library**

Co-Design Framework (e.g., Our FNAS)

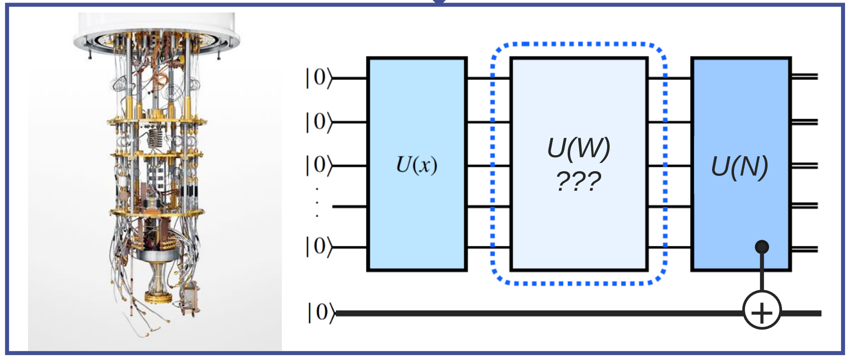
Network exploration	**NAS (Google)**
Programming library	**DNNBuilder (UIUC)**
Place & Route	**DNN on FPGA (UCLA)**

- **Mapping and scheduling?**
- **What is the best FPGA Architecture for neural networks**

Current Works: Co-Design of Neural Networks and Quantum Circuit



Co-Design

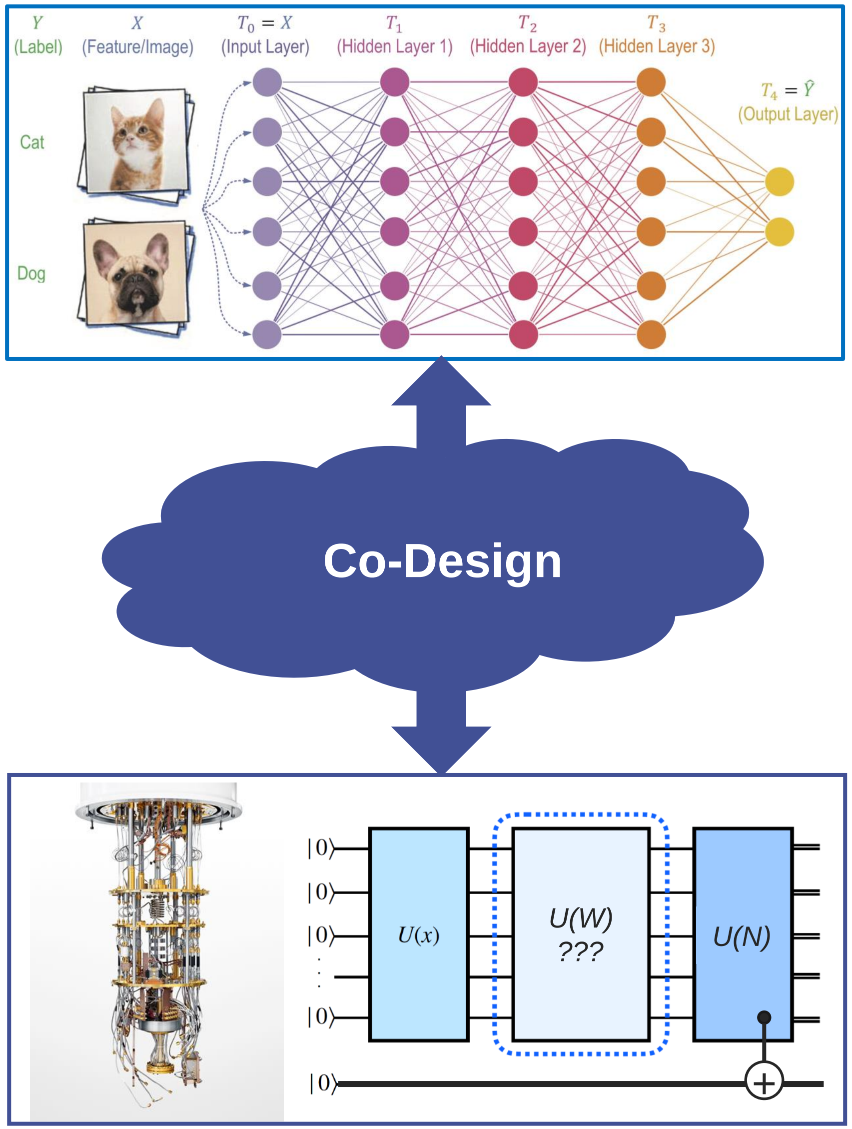


- **What is the best Neural Network Architecture for QC?**
- ……
- **Library**
- ……
- **What is the best QC design for neural networks**

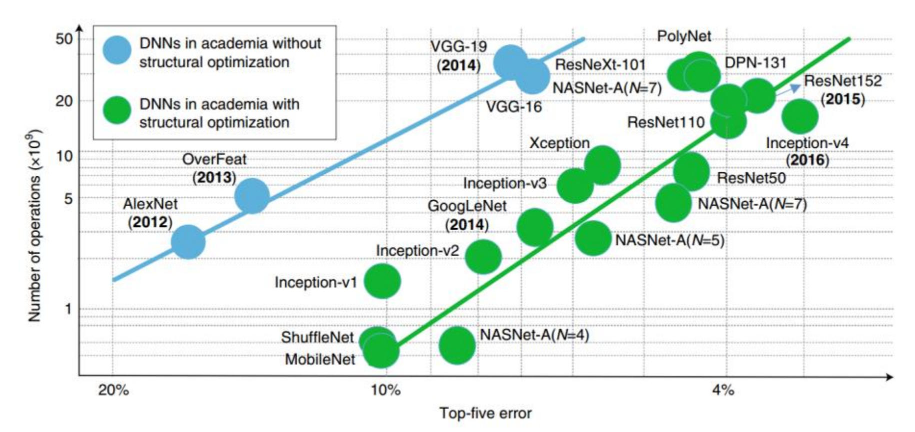
Co-Design Framework
QuantumFlow

Network exploration	QF-Mixer
Programming library	QFNN
Logic-physical Compile	QF-RobustNN

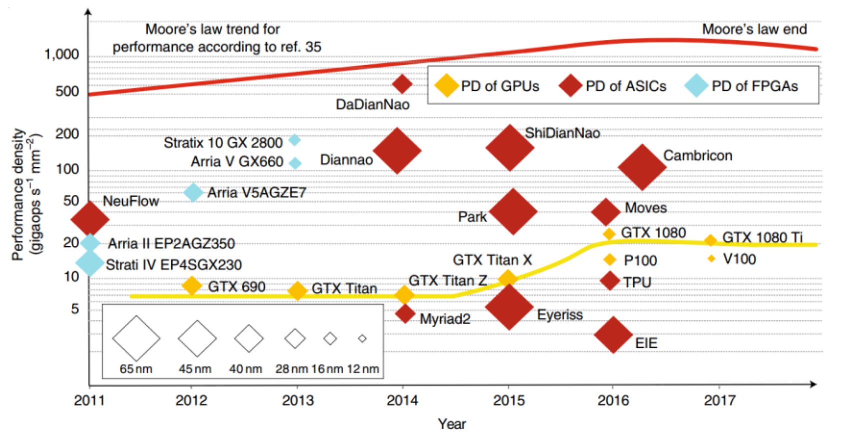
Co-Design of NN Systems on Quantum Computer



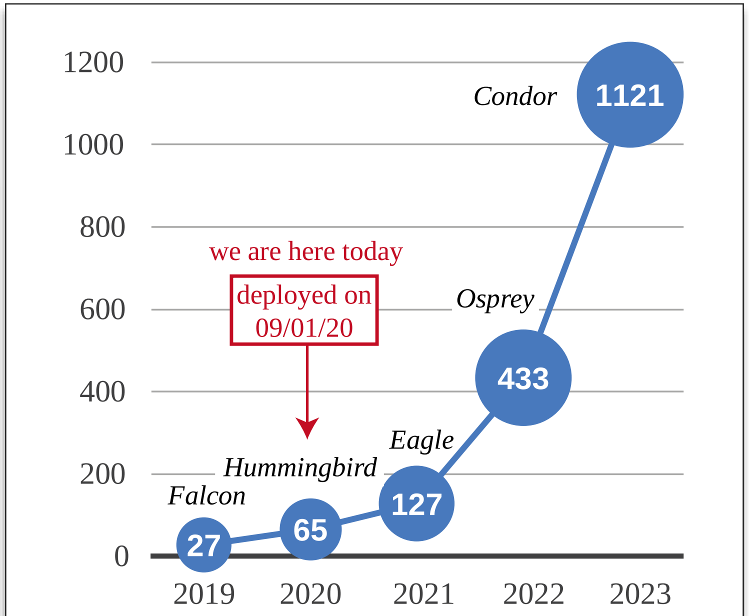
Motivation and Challenges



Deep neural network grows exponentially



Perf. of classical computing stops increasing



Quantum computer grows exponentially

Fundamental questions:

- Can we implement Neural Network on Quantum Computers?
- Can we achieve benefits in doing so?

Further questions:

- What is the best neural network architecture for quantum acceleration?
- What is the problem for near-term quantum computing, i.e., in NISQ era?

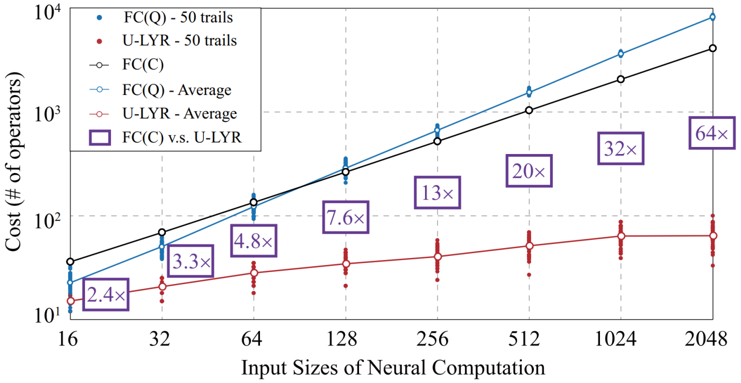
Motivation and Challenges

Fundamental questions:

- Can we implement Neural Network on Quantum Computers?
- Can we achieve benefits in doing so?

QuantumFlow

$$O(N) \rightarrow O(log^3 N)$$



Paper Published at: nature COMMUNICATIONS

Invited Contribution and Tutorial Talks at:

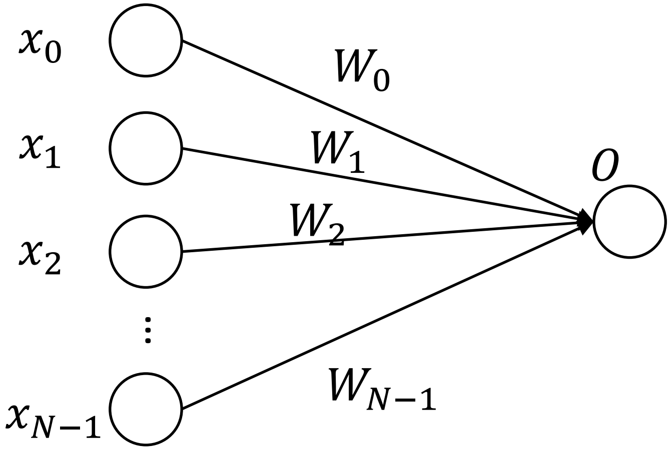
IBM Quantum Summit September 15-17, 2020

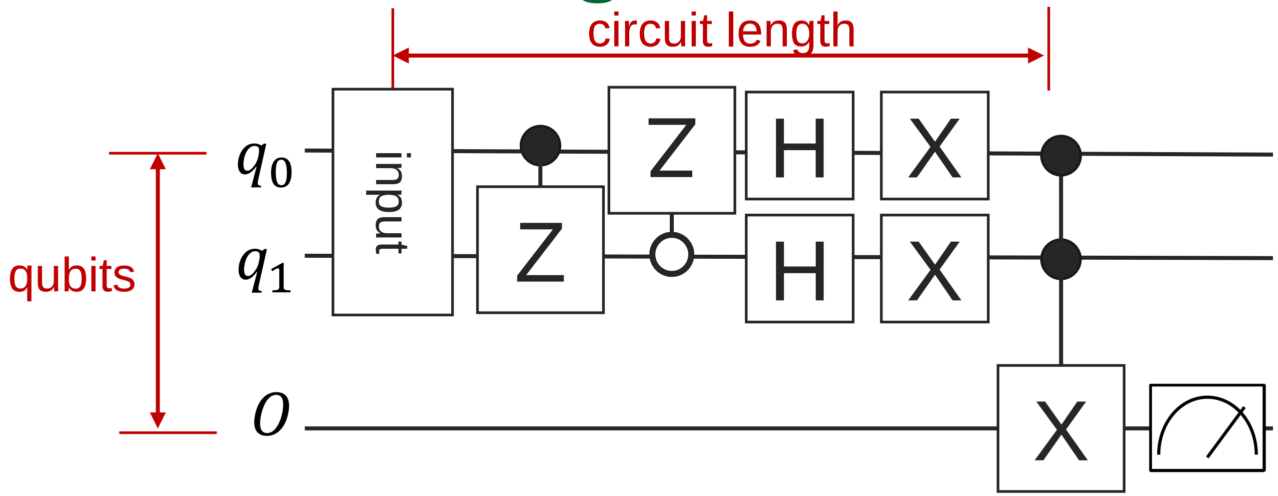
IEEE QUANTUM WEEK — IEEE International Conference on Quantum Computing and Engineering — QCE21

EMBEDDED SYSTEMS WEEK — OCTOBER 10-15, 2021 | VIRTUAL CONFERENCE

IEEE/ACM 2021 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN — ICCAD 40th Edition

What's the complexity? Quantum Advantage?





- **Classical computer with 1 MAC**

  $Time$: $O(N)$

  $Space$ *(Comp. Res.)*: $O(1)$

  $\boldsymbol{Time \times Space}$: $\boldsymbol{O(N)}$

- **Classical computer with N MAC**

  $Time$: $O(1)$

  $Space$ *(Comp. Res.)*: $O(N)$

  $\boldsymbol{Time \times Space}$: $\boldsymbol{O(N)}$

- **Time-Space Complexity in Quantum computer**

  $Time$: Circuit Length

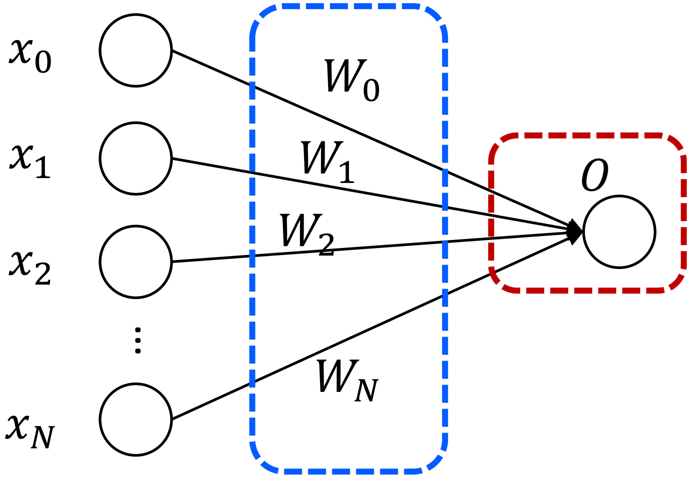
  $Space$ *(Comp. Res.)*: Qubits

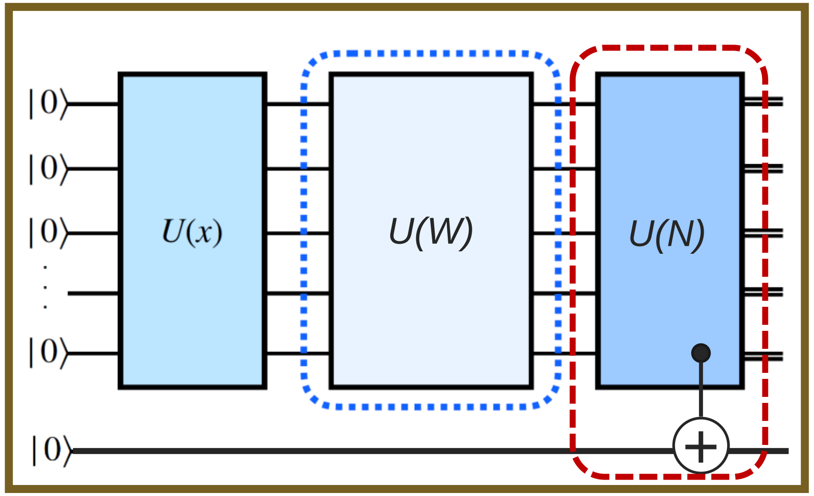
  $\boldsymbol{Time \times Space\ (T - S)}$: $\boldsymbol{Qubits \times Circuit\ Length}$

- **Given that $\boldsymbol{T - S}$ complexity on classical computer is $\boldsymbol{O(N)}$, Quantum Advantage is achieved if $\boldsymbol{T - S}$ complexity on Quantum can be $\boldsymbol{O(ploylogN)}$ or lower. ------- Exponential Speedup!**

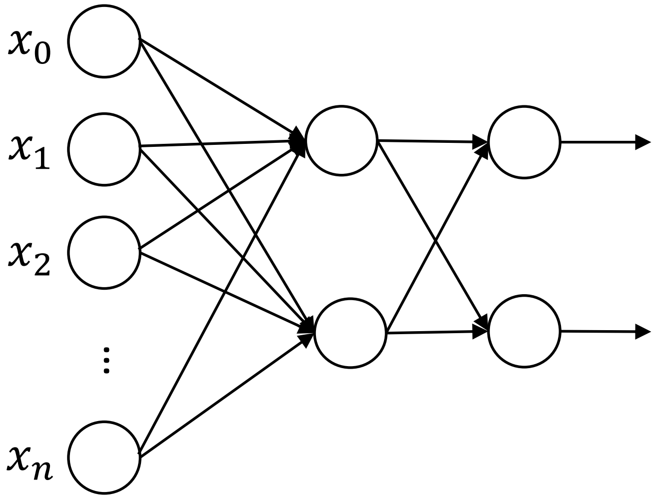
What's the Goals?

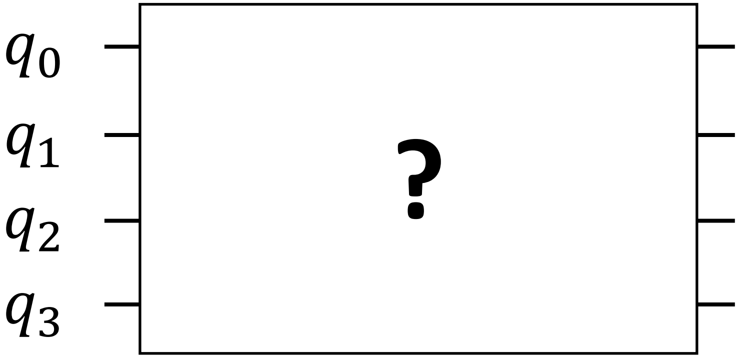
Goal 1: Correctly Implement!





Goal 2: Scale-Up!





Goal 3: 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:

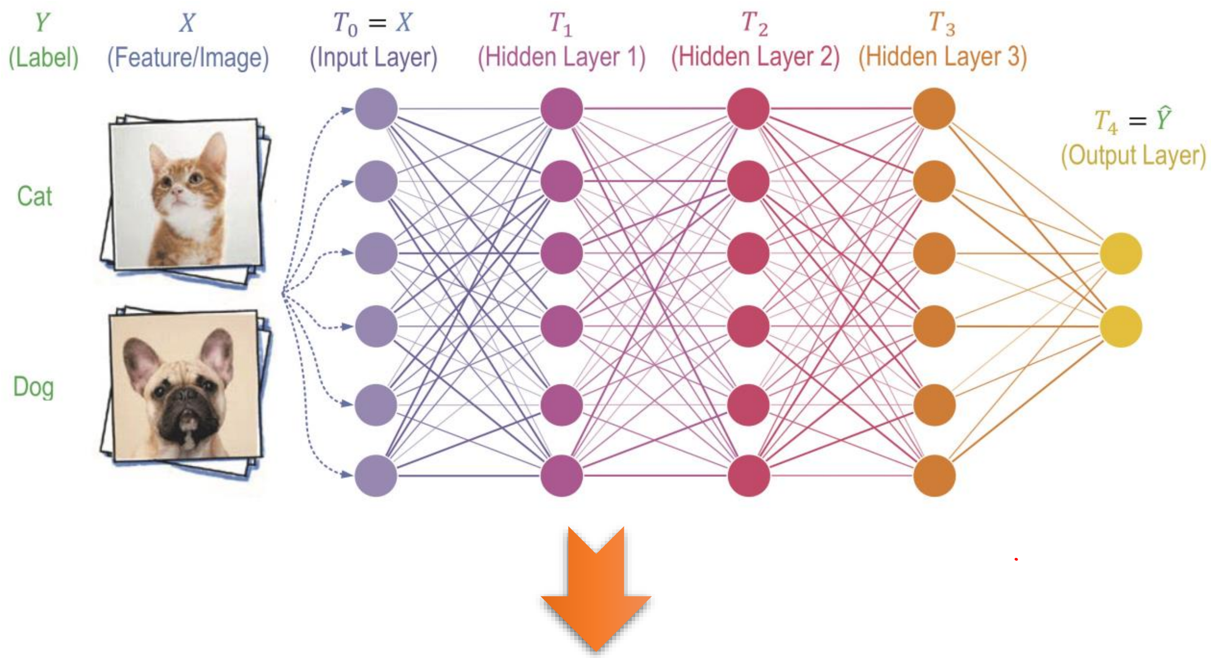
Can we reduce complexity to

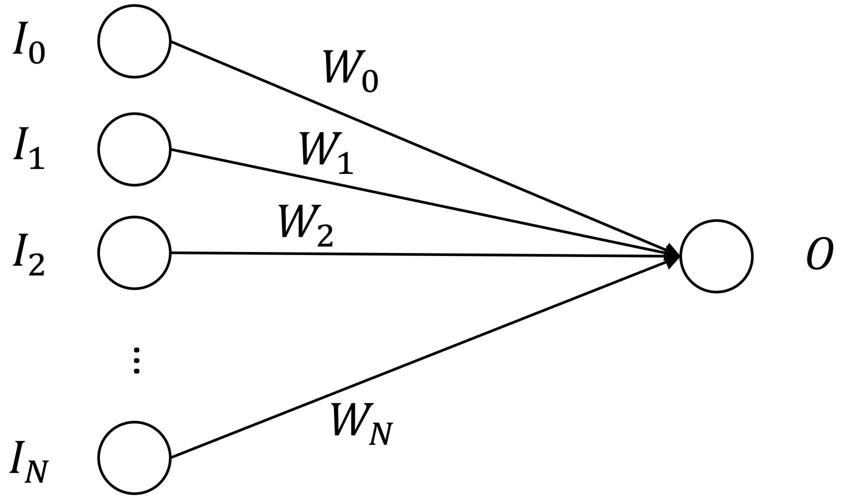
$\boldsymbol{O(ploylogN)}$, say $\boldsymbol{O(log^2N)}$?

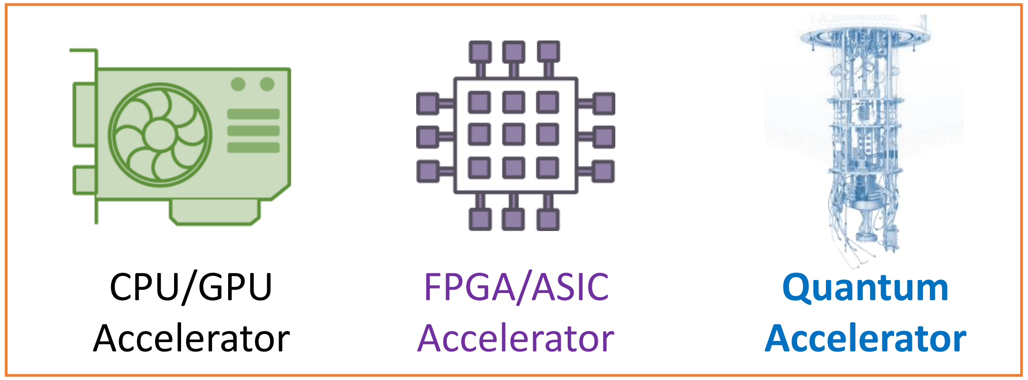
Outline – QuantumFlow

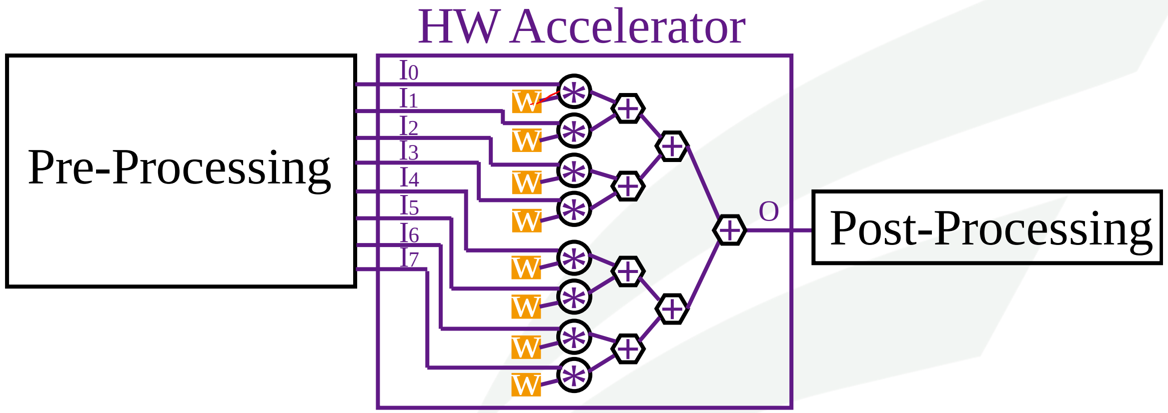
- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

Neural Network Accelerator Design on Classical Hardware

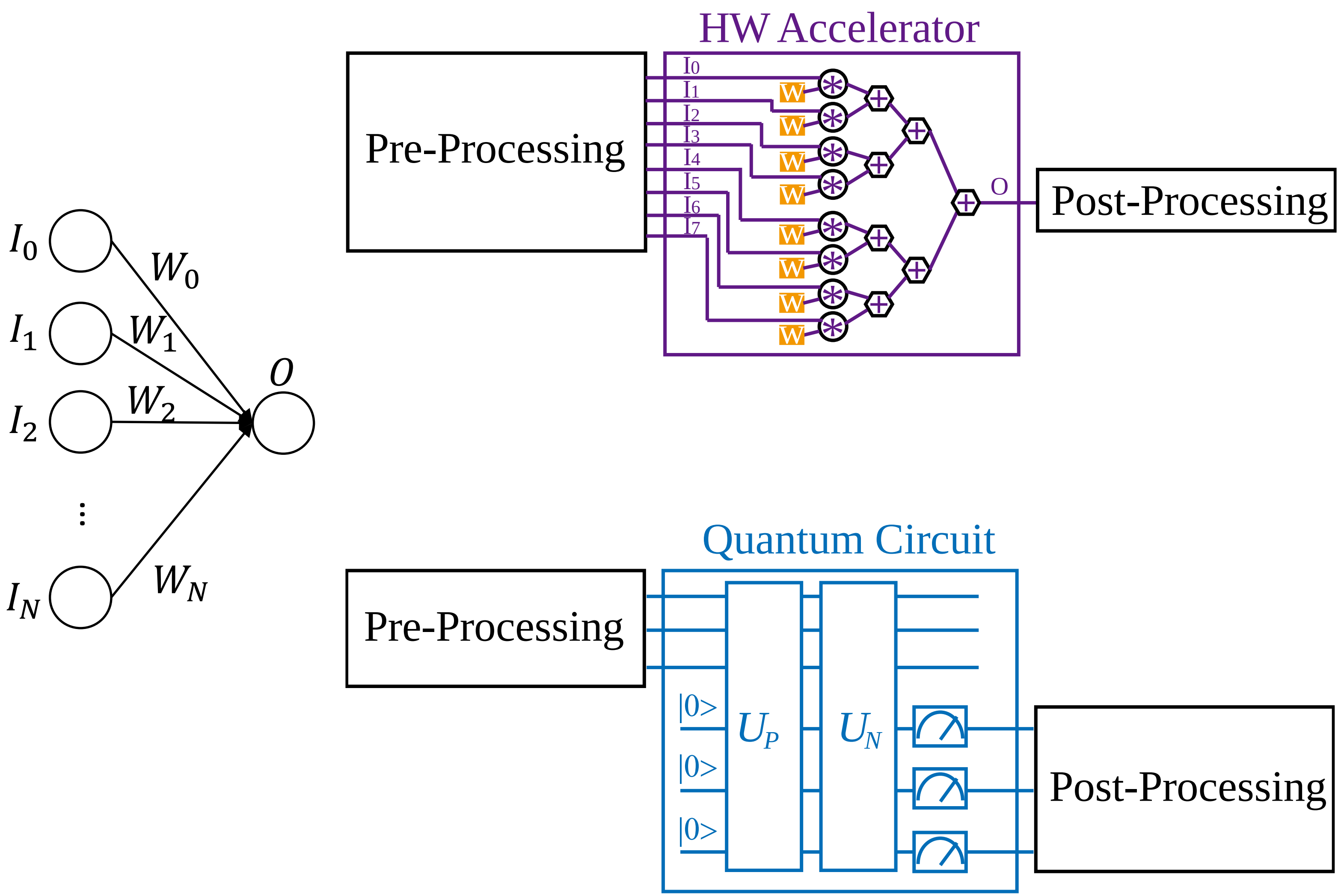








Neural Network Accelerator Design from Classical to Quantum Computing



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

(2) HW Acceleration

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

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

(2) HW/Quantum Acceleration

(2.1) U_p Quantum-State-Preparation

(2.2) U_N Quantum Neural Computation

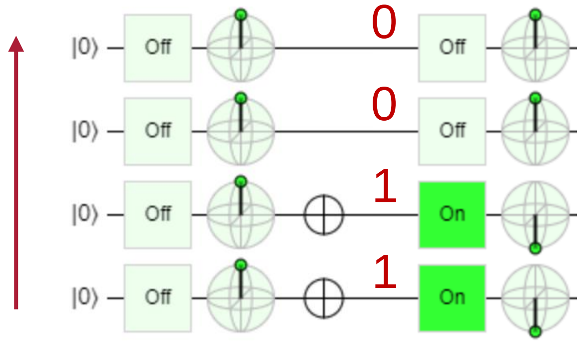
(2.3) *M* Measurement

(3) Data Post-Processing (PostP)

$$PreP + U_P + U_N + M + PostP$$

What Data Can Be Encoded to Quantum Computers, and how?

- **Can we encode an arbitrary number into quantum computer? Is it efficient?**
- **Yes / No**



Yes, we can!

Number: 12

Number: 1.5

Number: 0.75

No, because it uses too many qubits!

This encoding is similar to classical bits, where each qubit is regarded as a binary number!

1-to-N mapping! (Boolean Function)

What Data Can Be Encoded to Quantum Computers, and how?

- Can we encode an arbitrary number into quantum computer? Is it efficient?
  - Yes / No

Yes, we can!

Number: 12

Number: 1.5

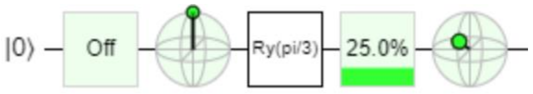
Number: 0.75

No, because it uses too many qubits!

This encoding is similar to classical bits, where each qubit is regarded as a binary number!

1-to-N mapping! (Boolean Function)

- Can we take use of superposition of qubits to encode data? Is this solution perfect?
  - Yes / No



Yes, we can!

0.25 in one qubit

No, (1) data needs in the range of [0,1]!
(2) same complexity *O(1)* as classical

1-to-1 mapping! (Angle Encoding)

What Data Can Be Encoded to Quantum Computers, and how?

- Can we encode an arbitrary number into quantum computer? Is it efficient?
  - Yes / No

Yes, we can!

Number: 12

Number: 1.5

Number: 0.75

No, because it uses too many qubits!

This encoding is similar to classical bits, where each qubit is regarded as a binary number!

1-to-N mapping! (Boolean Function)

- Can we take use of superposition of qubits to encode data? Is this solution perfect?
  - Yes / No

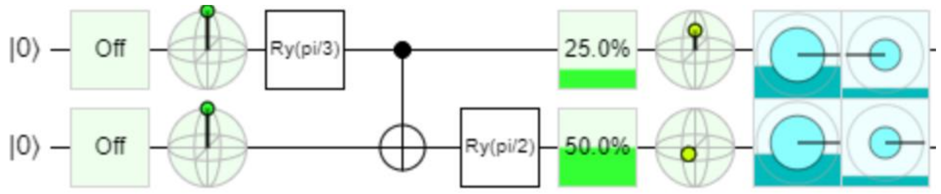
Yes, we can!

0.25 in one qubit

No, (1) data needs in the range of [0,1]!
(2) same complexity *O(1)* as classical

1-to-1 mapping! (Angle Encoding)

- **Can we take use of entanglement of qubits to encode data? Is this solution perfect?**
  - **Yes / No**



Yes, 4 # in 2 qubits

$$\begin{bmatrix} +0.61 & -0.35 \\ +0.61 & +0.35 \end{bmatrix}$$

No, (1) sum of the square of data need to be 1
(2) may have high cost to encode data

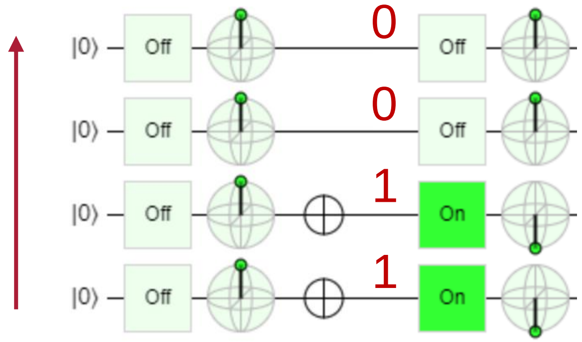
N-to-logN mapping! (Amplitude Encoding)

Encoding: 1-to-N *v.s.* 1-to-1 *v.s.* N-to-*log*N

Data Encoding	# of Qubit (C v.s. Q)	Data Limitation	Encoding Complexity
1-to-N	O(N) vs. O(N^2)	**Almost No!**	**Low**
1-to-1	O(N) vs. O(N)	[0,+1]	**Low**
N-to-logN	**O(N) vs. O(*log*N)**	[-1,+1] and $\sum x^2 = 1$	**High**

What Data Can Be Encoded to Quantum Computers, and how?

- **Can we encode an arbitrary number into quantum computer? Is it efficient?**
- **Yes / No**



Yes, we can!

Number: 12

Number: 1.5

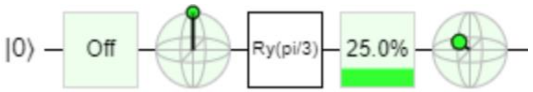
Number: 0.75

No, because it uses too many qubits!

This encoding is similar to classical bits, where each qubit is regarded as a binary number!

1-to-N mapping! (Boolean Function)

- **Can we take use of superposition of qubits to encode data? Is this solution perfect?**
- **Yes / No**



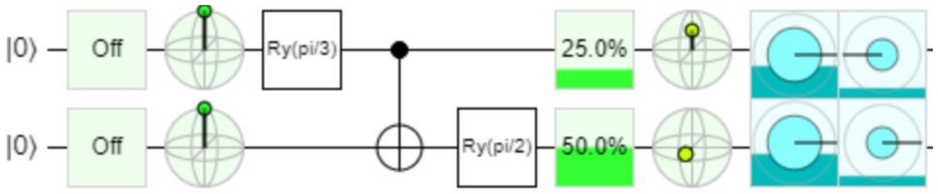
Yes, we can!

0.25 in one qubit

No, (1) data needs in the range of [0,1]!
(2) same complexity *O(1)* as classical

1-to-1 mapping! (Angle Encoding)

- **Can we take use of entanglement of qubits to encode data? Is this solution perfect?**
- **Yes / No**



Yes, 4 # in 2 qubits

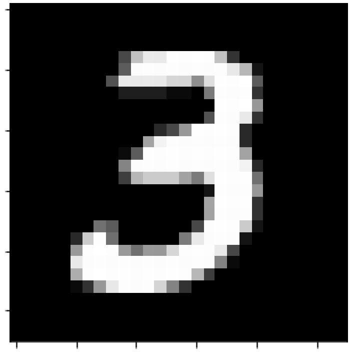
$$\begin{bmatrix} +0.61 & -0.35 \\ +0.61 & +0.35 \end{bmatrix}$$

No, (1) sum of the square of data need to be 1
(2) may have high cost to encode data

n-to-logn mapping! (Amplitude Encoding)

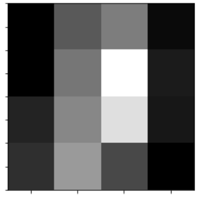
PreP + U_P + U_N + *M* + *PostP* : Data Pre-Processing

- **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



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

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

Step 2: Formulate Unitary Matrix

Applying SVD method

(See Listing 1 in ASP-DAC SS Paper)

Unitary matrix: $M_{16\times16}$

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

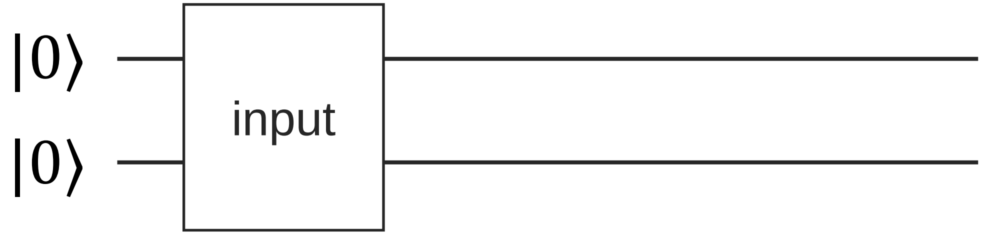
PreP + U_P + U_N + *M* + *PostP* --- Data Encoding / Quantum State Preparation

- **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 $M_{4\times4}$

data_matrix

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



State Transition:

data_matrix $|00\rangle$

IBM Qiskit Implementation:

```
inp = QuantumRegister(4, ”in_qubit”)
circ = QuantumCircuit(inp)
iniG = UnitaryGate(data_matrix, label=”input”)
circ.append(iniG, inp[0:4])
```

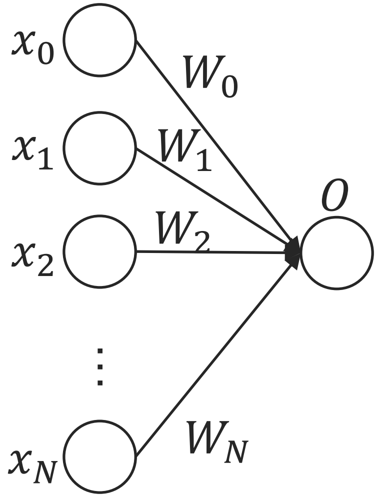
Hands-On Tutorial (1)

PreP + U_P

Outline – QuantumFlow

- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

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.

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

$$\boldsymbol{x} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad \begin{matrix} w_0 = 1 \\ w_1 = 1 \\ w_2 = 1 \\ w_3 = -1 \end{matrix}$$

$\rightarrow$ $m_3 = -1 \times a_3 = -a_3$

Output = **U** × **Input**

a_0	$\vert 00\rangle$
a_1	$\vert 01\rangle$
a_2	$\vert 10\rangle$
$m_3 = -a_3$	$\vert 11\rangle$

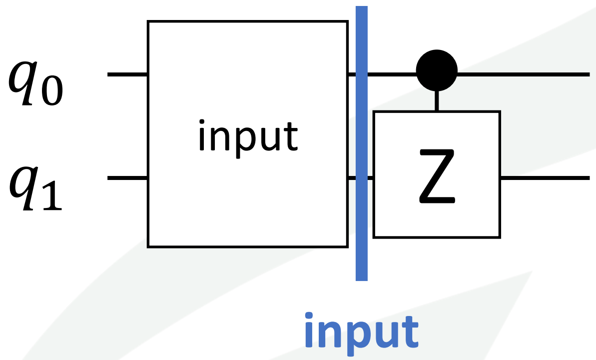
=

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

×

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

Quantum Circuit

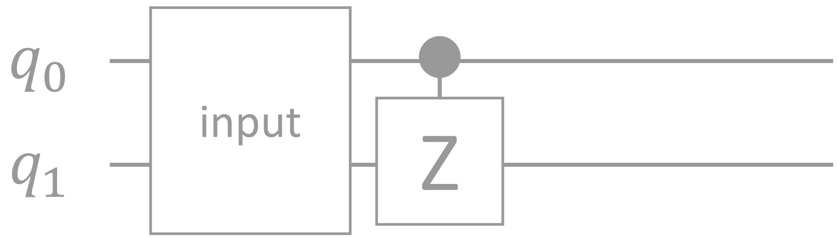


PreP + U_P + U_N + M + PostP --- Neural Computation: Step 1

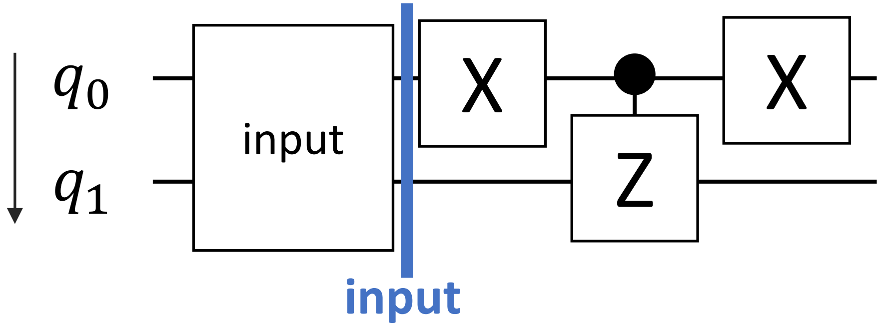
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

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



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



Output = **U** × **Input**

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

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

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

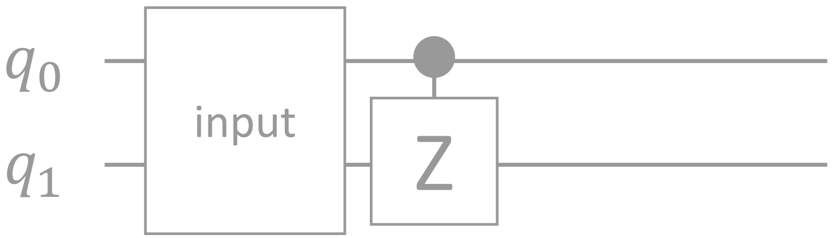
$$\boldsymbol{U = (X \otimes I) \times CZ \times (X \otimes I)}$$

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

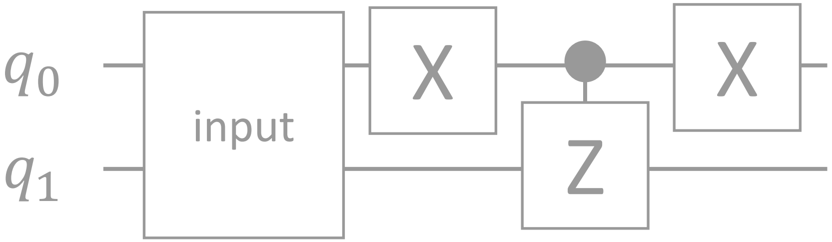
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

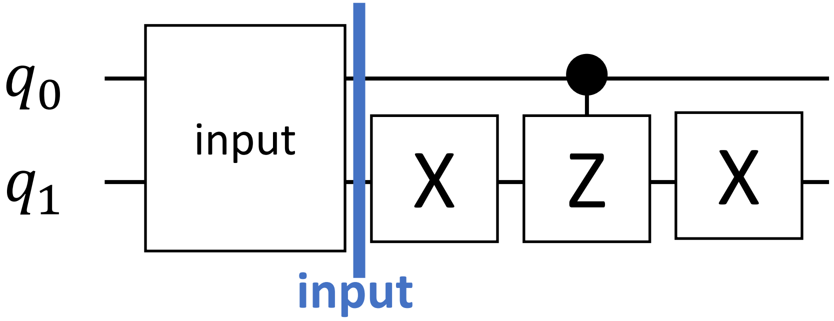
$w = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$



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



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



Output = **U** × **Input**

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

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times$$

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

$$U = (I \otimes X) \times CZ \times (I \otimes X)$$

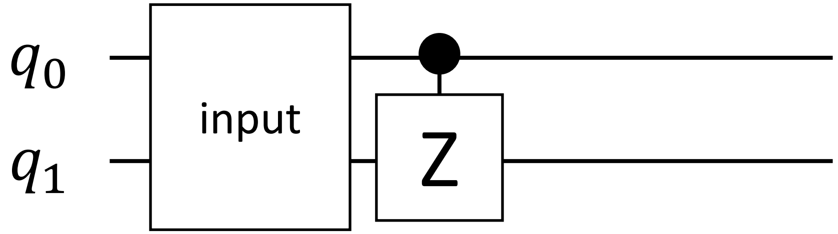
PreP + *U_P* + *U_N* + *M* + *PostP* --- Neural Computation: Step 1

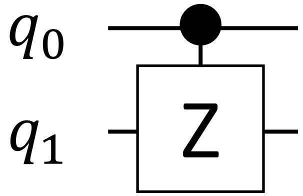
Step 1: $m_i = x_i \times w_i$

EX: 4 input data on 2 qubits

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

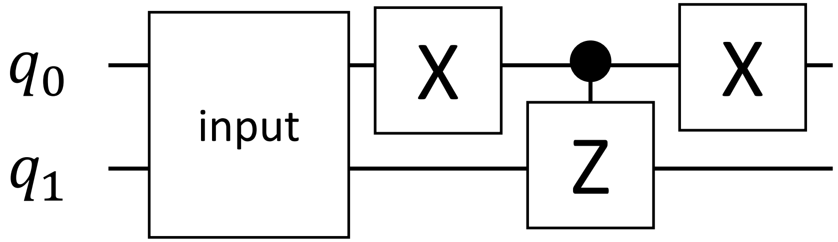
$\boldsymbol{w} = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \end{bmatrix}$

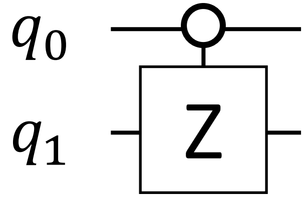




Flip the sign of $|11\rangle$

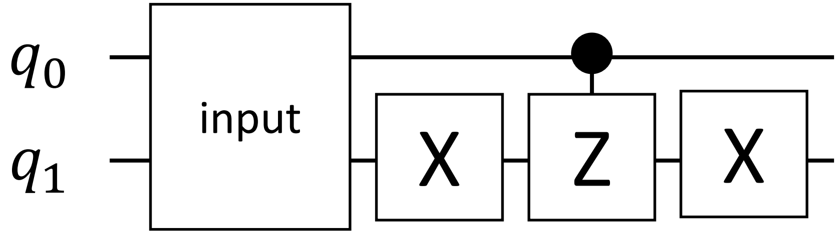
$\boldsymbol{w} = \begin{bmatrix} +1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$

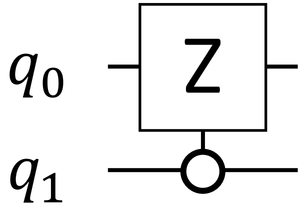




Flip the sign of $|01\rangle$

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





Flip the sign of $|10\rangle$

PreP + U_P + 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

Output = **U** × **Input**

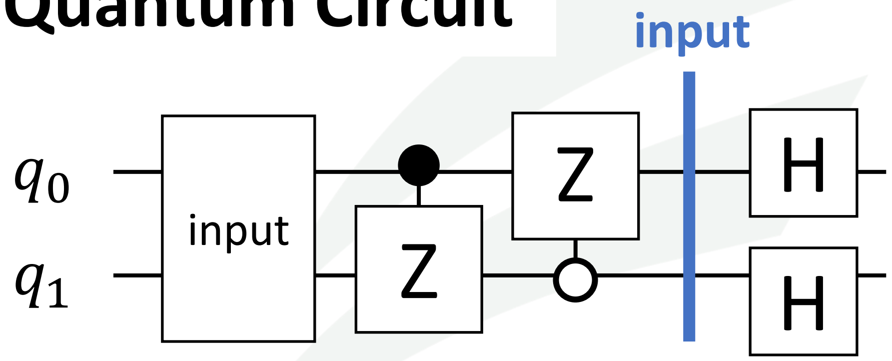
$\sum_i (m_i) / \sqrt{\|x\|}$	$\|00\rangle$
Do not care 1	$\|01\rangle$
Do not care 2	$\|10\rangle$
Do not care 3	$\|11\rangle$

$$= \frac{1}{\sqrt{\|x\|}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \times$$

m_0	$\|00\rangle$
m_1	$\|01\rangle$
m_2	$\|10\rangle$
m_3	$\|11\rangle$

note: $\|x\| = 2^N$

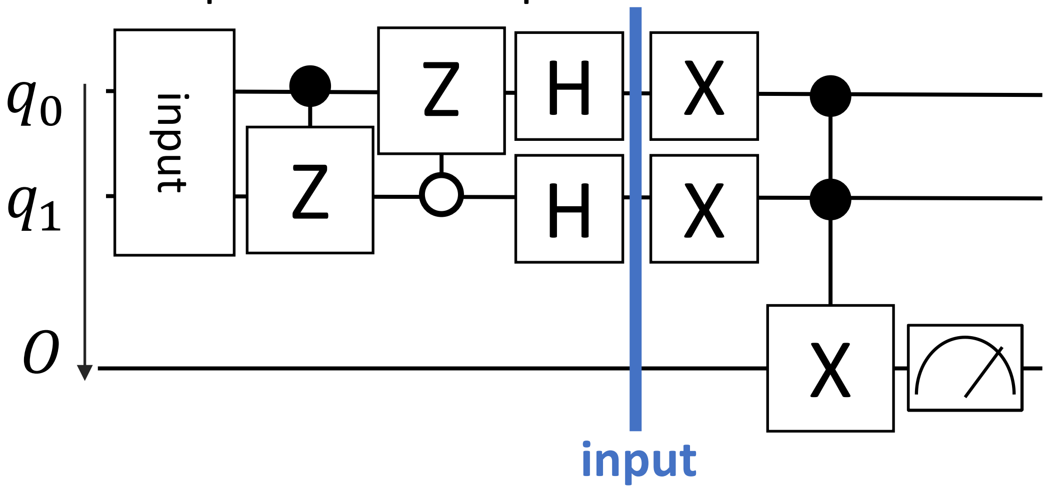
Quantum Circuit



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

$\sum_i (m_i) / \sqrt{\|x\|}$	$\|000\rangle$
0	$\|001\rangle$
Do not care 1	$\|010\rangle$
0	$\|011\rangle$
Do not care 2	$\|100\rangle$
0	$\|101\rangle$
Do not care 3	$\|110\rangle$
0	$\|111\rangle$

$X^{\otimes 2}$

Do not care 3	$\|000\rangle$
0	$\|001\rangle$
Do not care 2	$\|010\rangle$
0	$\|011\rangle$
Do not care 1	$\|100\rangle$
0	$\|101\rangle$
$\sum_i (m_i) / \sqrt{\|x\|}$	$\|110\rangle$
0	$\|111\rangle$

CCX

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

Output

$$P\{O = |1\rangle\} = P\{|001\rangle\} + P\{|011\rangle\} + P\{|101\rangle\} + P\{|111\rangle\} = \left[\frac{\sum_i (m_i)}{\sqrt{\|x\|}}\right]^2$$

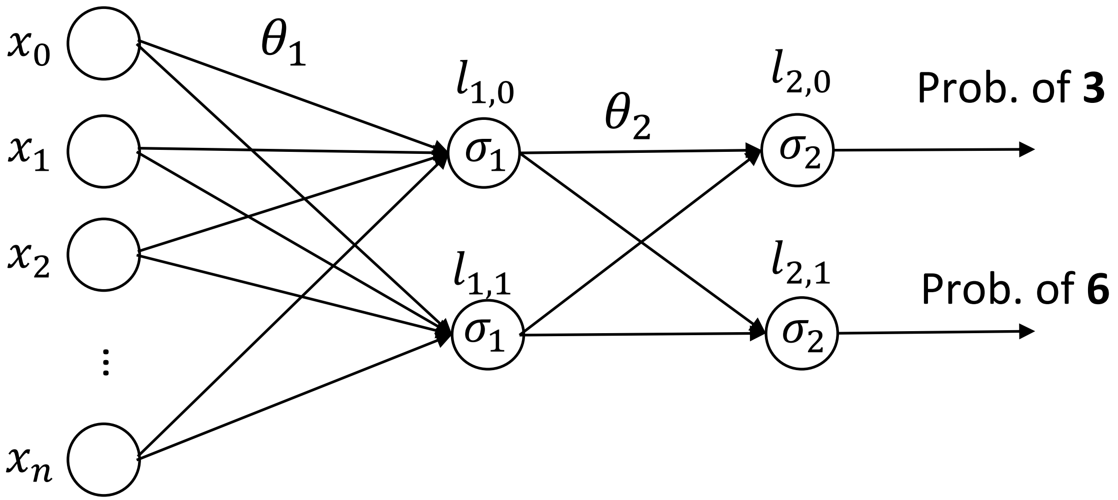
Hands-On Tutorial (2)

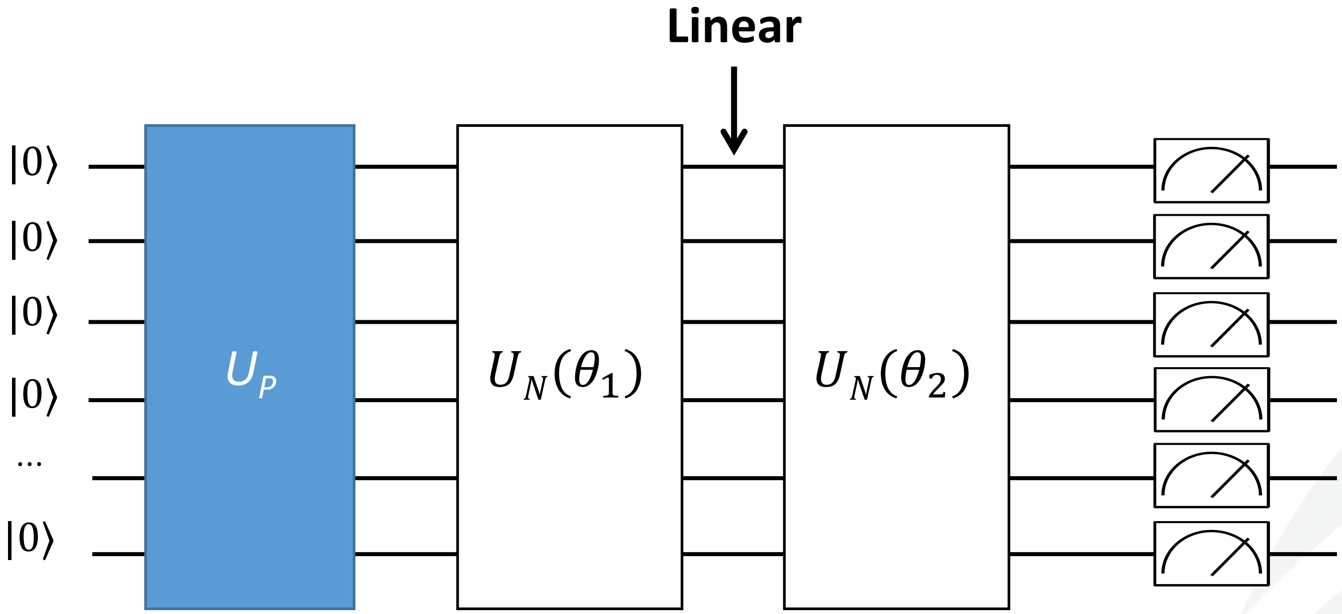
$PreP + U_P + U_N$

Outline – QuantumFlow

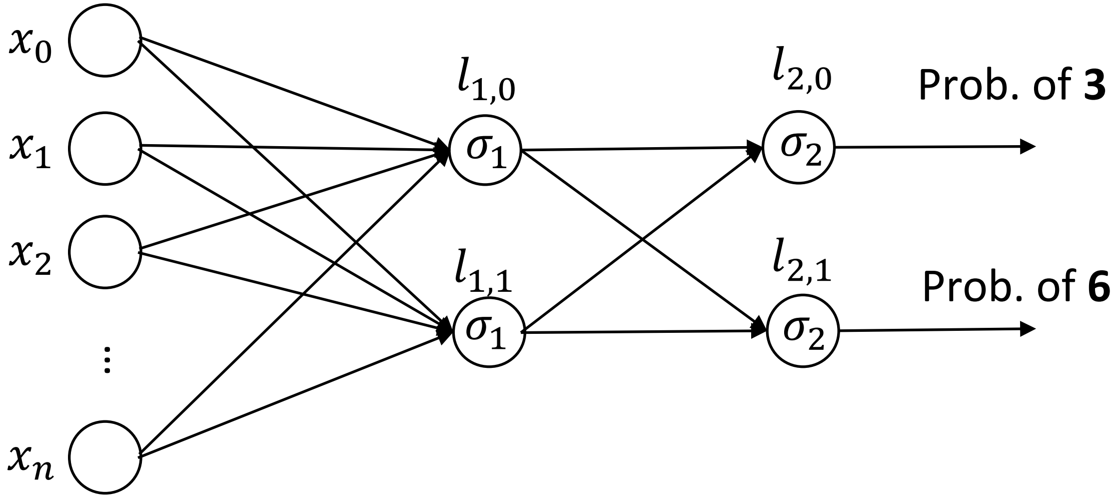
- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

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





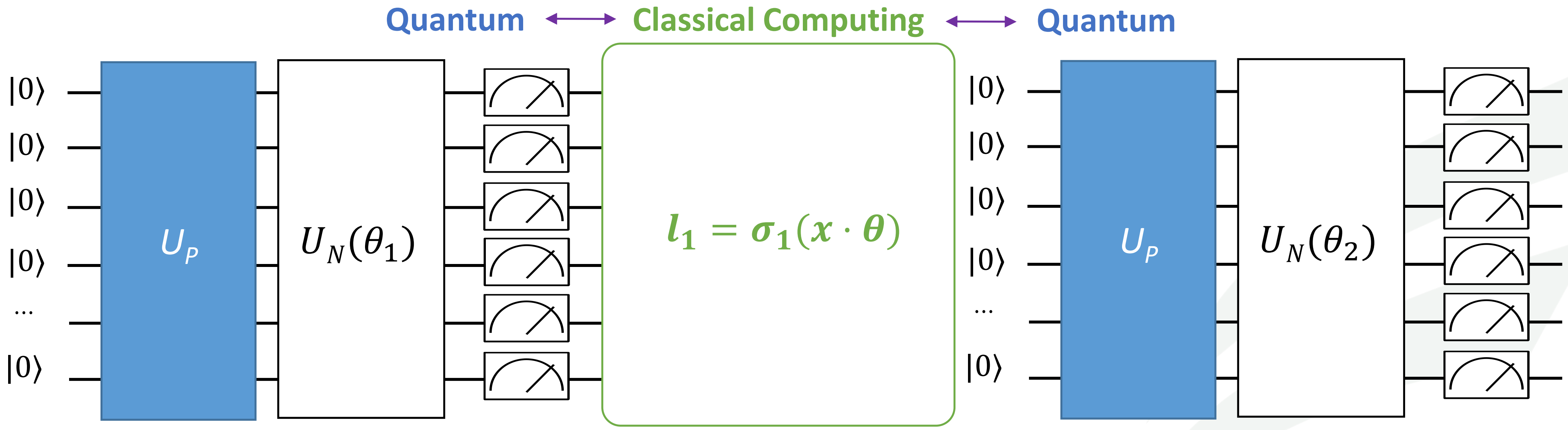
Challenge 2: Quantum-Classical Interface is Expensive



Ref [1]

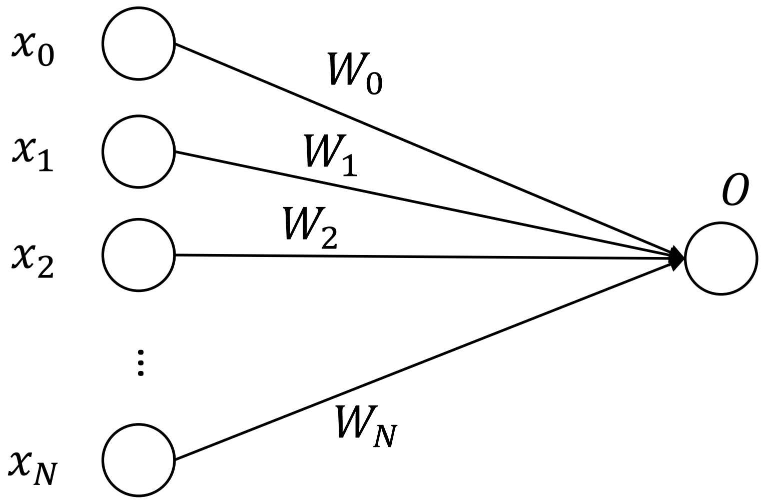
Table 2 Complexity of each step in hybrid quantum-classical computing for deep neural network with U-LYR.

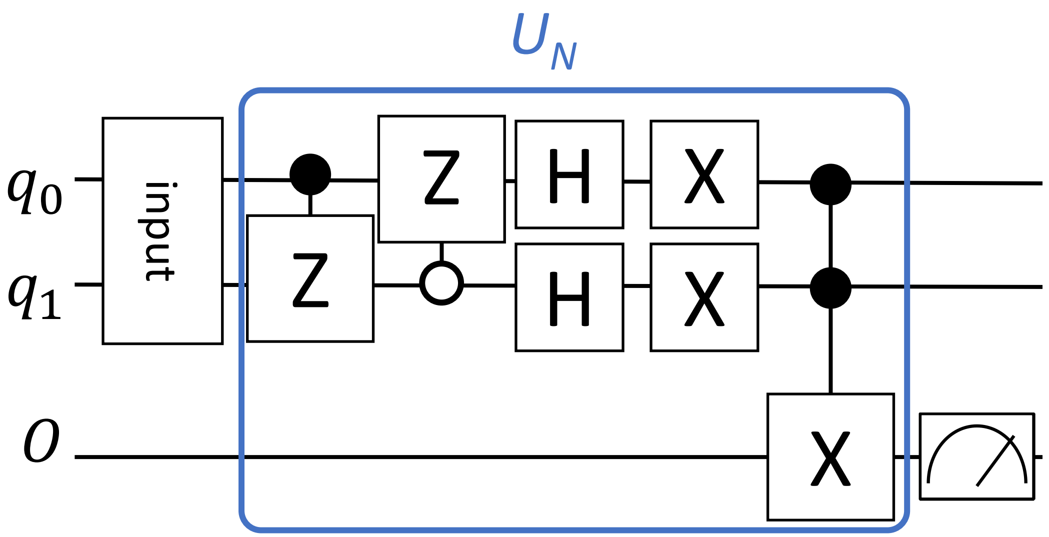
Complexity	State-preparation	Computation	Measurement
Depth (T)	$O(d \cdot \sqrt{n})$	$O(d \cdot \log^2 n)$	$O(d)$
Qubits (S)	$O(n)$	$O(n \cdot \log n)$	$O(n \cdot \log n)$
Cost (TS)	$O(d \cdot n^{\frac{3}{2}})$	$O(d \cdot n \cdot \log^3 n)$	$O(d \cdot n \cdot \log n)$
Total (TS)	$O(d \cdot n^{\frac{3}{2}})$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

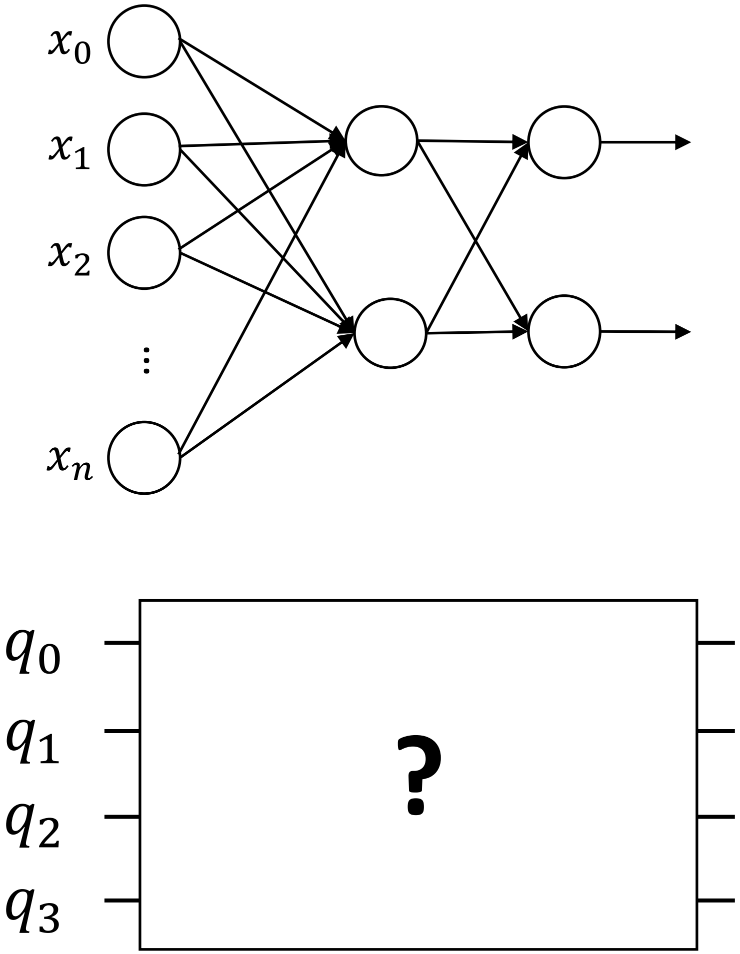
Classical Computing		
	No Parallelism	Full Parallelism
Time (T)	O(N)	O(1)
Space (S)	O(1)	O(N)
Cost (TS)	O(N)	O(N)

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

What's the Goals?

Goal 1: Correctly Implement!

Goal 2: Scale-Up!



Goal 3: 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 $O(N)$

Quantum Computing:

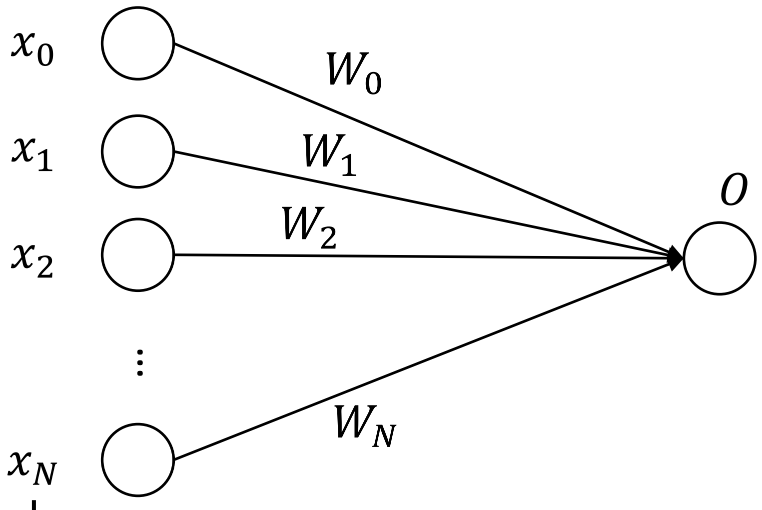
Can we reduce complexity to

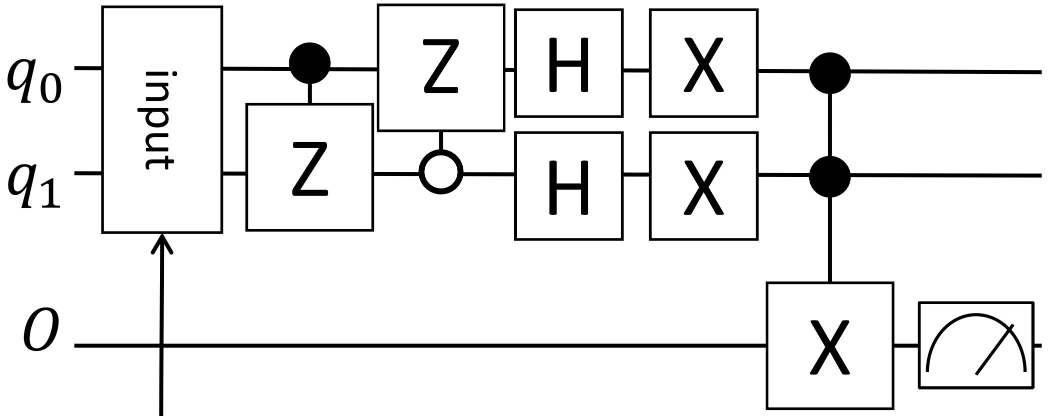
$O(ploylogN)$, say $O(log^2 n)$?

Outline – QuantumFlow

- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

Design Direction 1: NN → Quantum Circuit



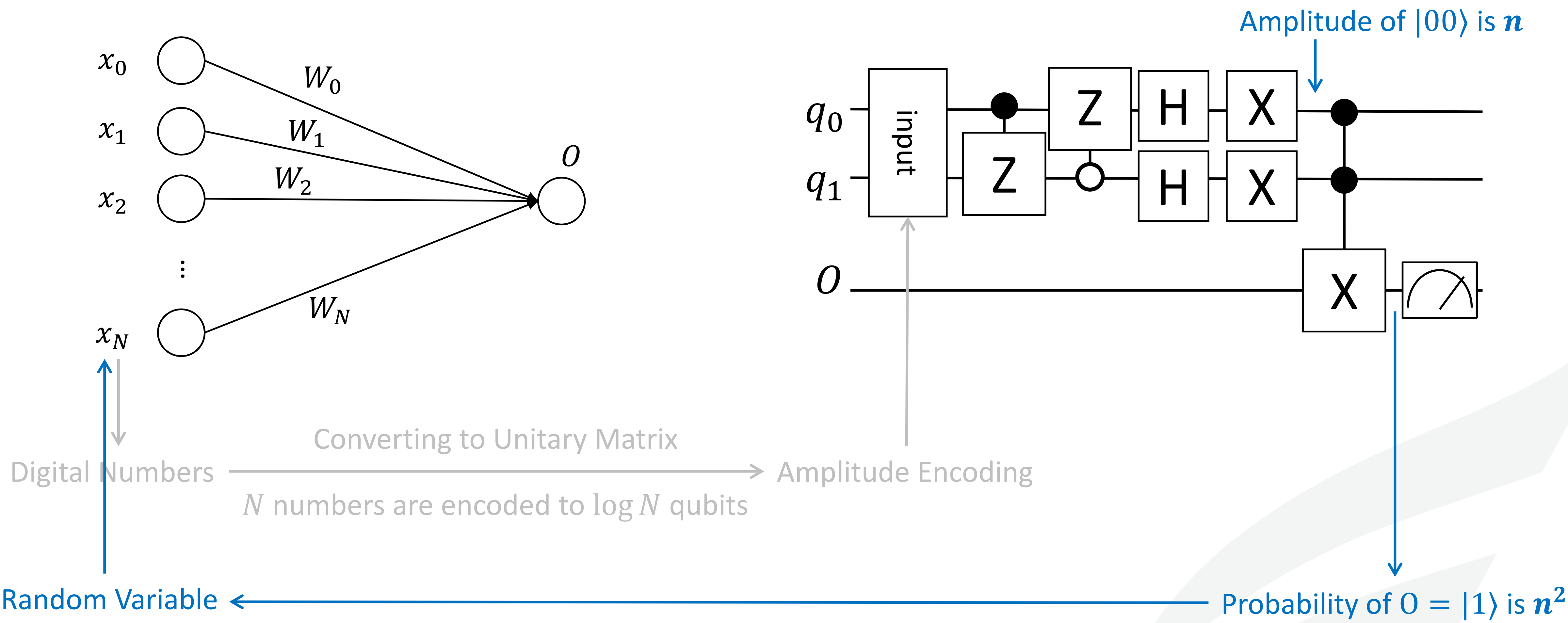


Digital Numbers → Amplitude Encoding

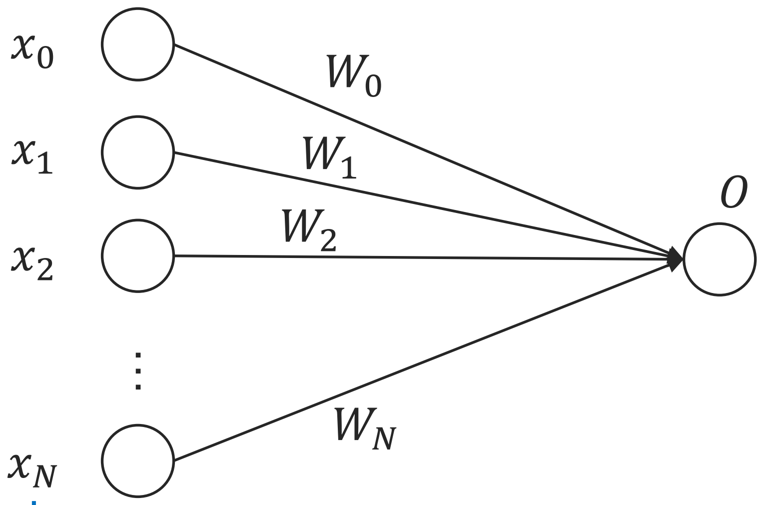
Converting to Unitary Matrix

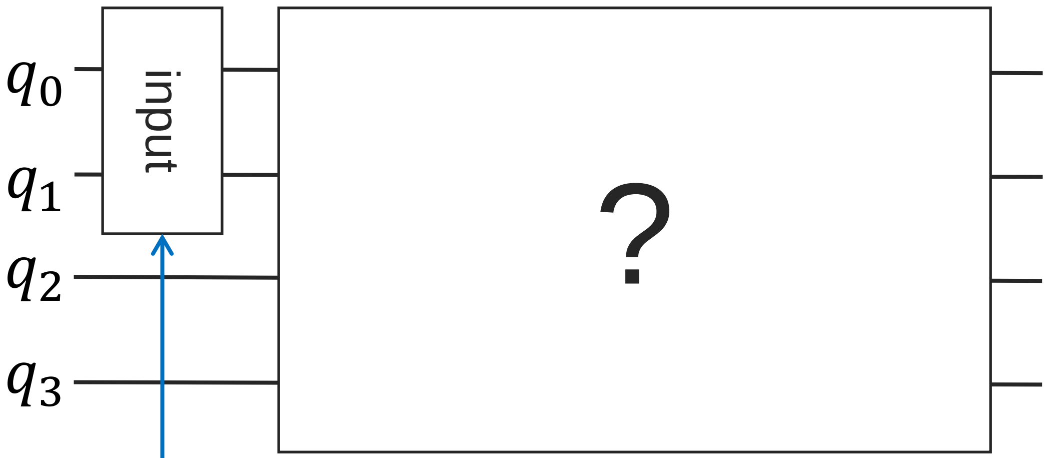
N numbers are encoded to $\log N$ qubits

Design Direction 2: Quantum Circuit → NN



Design Direction 3: NN → Quantum Circuit





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

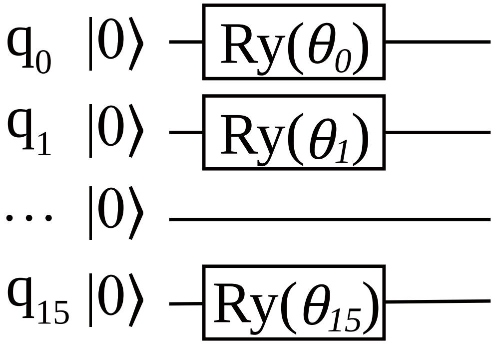
***N* numbers are encoded to *N* qubits**

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

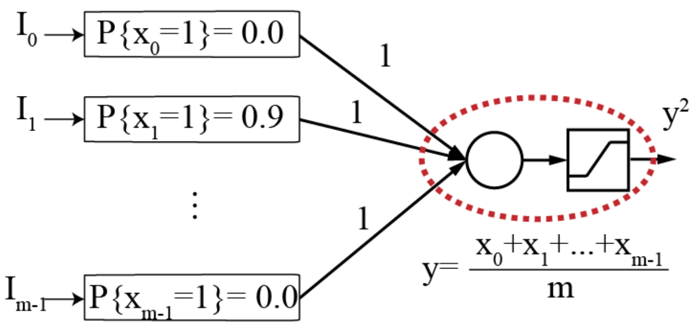
→

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

→



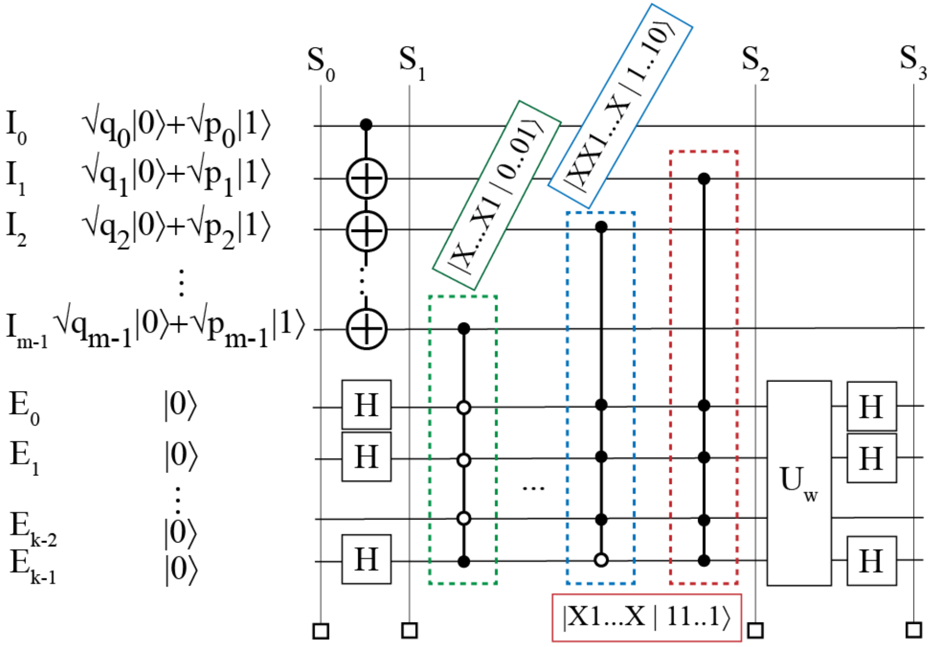
rvU_N --- Neural Computation



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

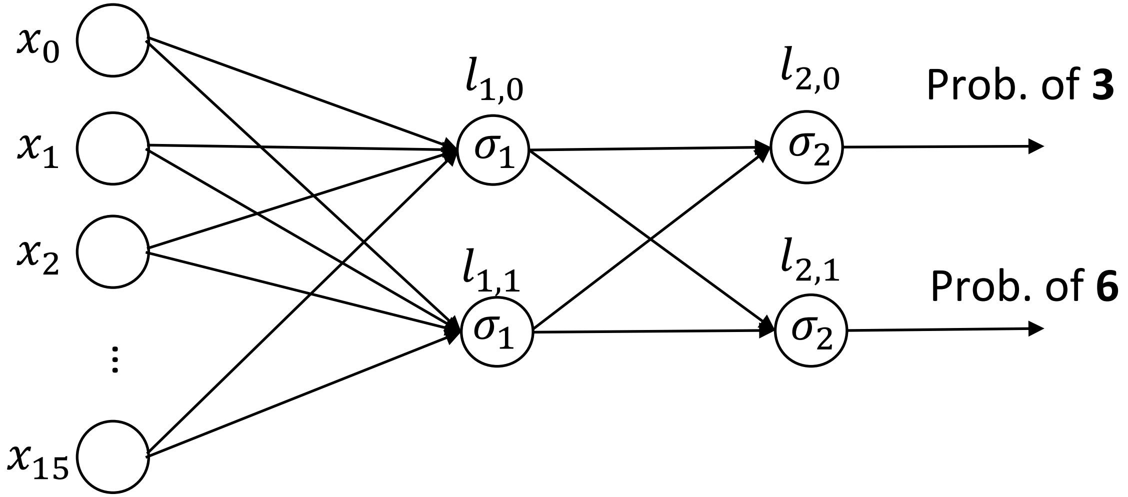
y	-1	$\frac{-m+2}{m}$	… 0 …	$\frac{m-2}{m}$	1
	$\prod p_i$	$p_{m-1}\ldots p_1q_0$ + $p_{m-1}\ldots q_1p_0$ + $\vdots$ + $q_{m-1}\ldots p_1p_0$	…	$q_{m-1}\ldots q_1p_0$ + $q_{m-1}\ldots p_1q_0$ + $\vdots$ + $p_{m-1}\ldots q_1q_0$	$\prod q_i$

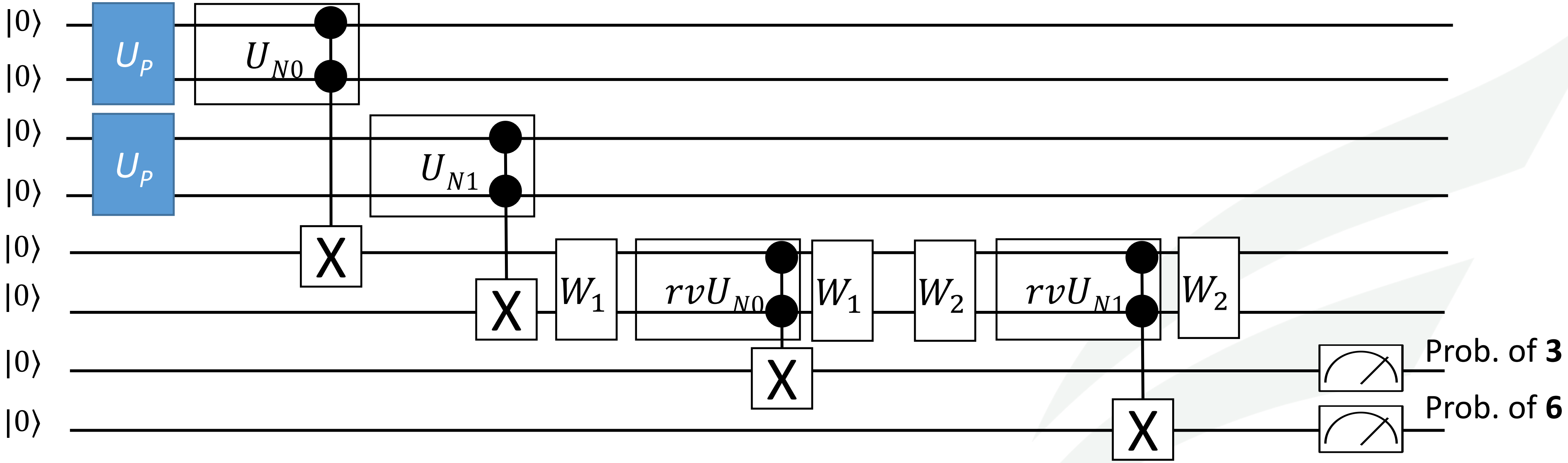
y^2	0 …	$(\frac{m-2}{m})^2$	1
		$p_{m-1}\ldots p_1q_0$ + $p_{m-1}\ldots q_1p_0$ + $\vdots$ + $q_{m-1}\ldots p_1p_0$ + $q_{m-1}\ldots q_1p_0$ + $q_{m-1}\ldots p_1q_0$ + $\vdots$ + $p_{m-1}\ldots q_1q_0$	$\prod q_i$ + $\prod p_i$



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!





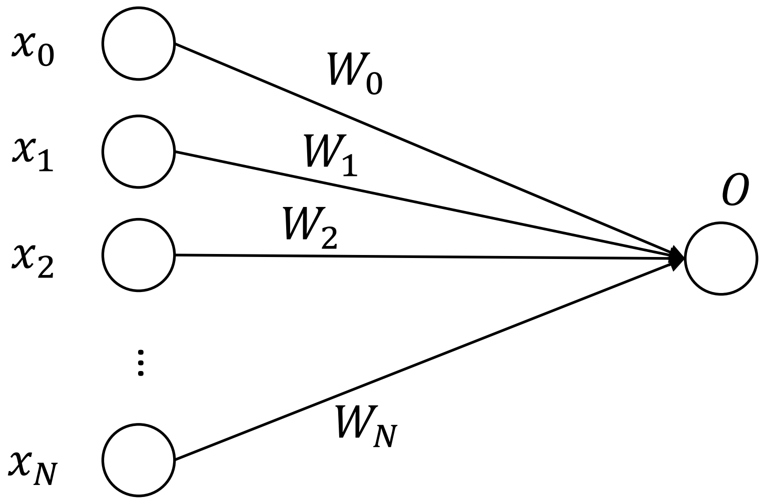
Hands-On Tutorial (3)

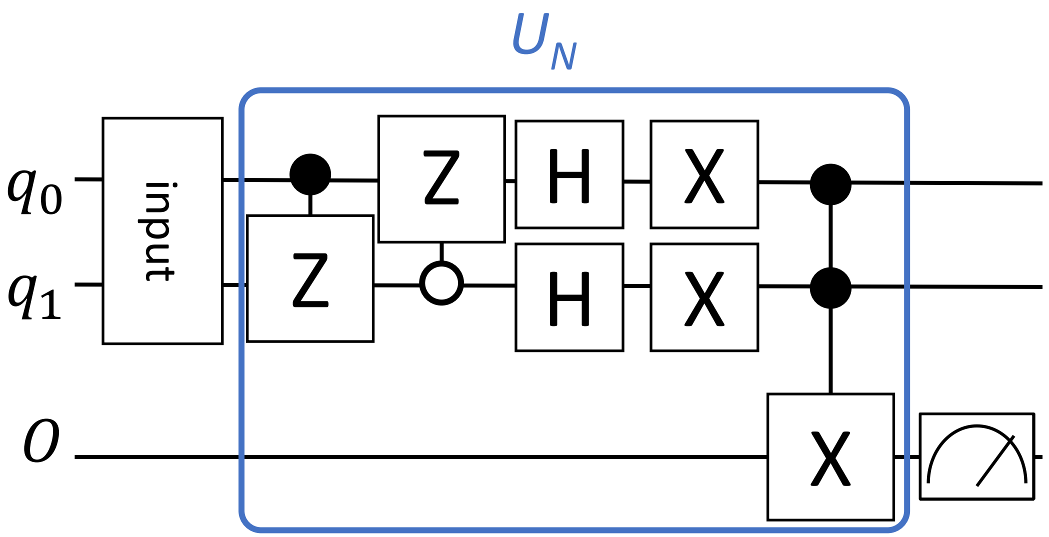
PreP+ U_P+ U_N+ M+ PostP (MNIST)

Outline – QuantumFlow

- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

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)

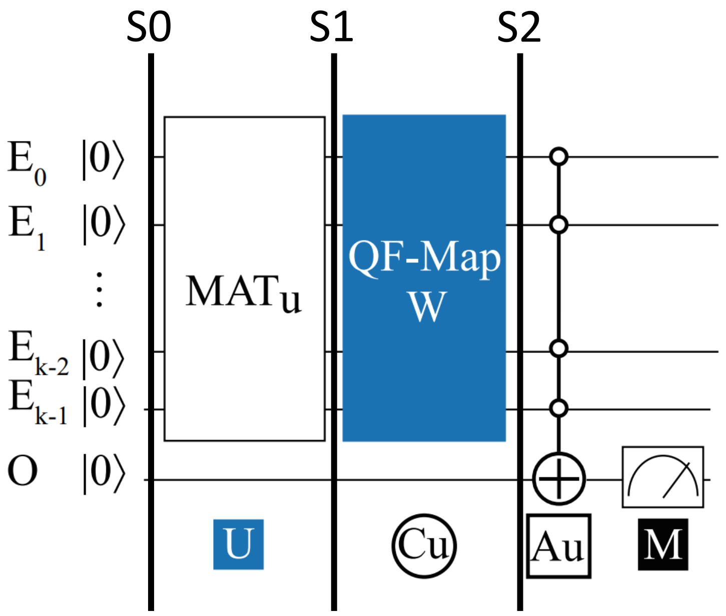
Quantum Computing		
	Previous Design	Optimization
Circuit Depth (T)	O(N)	???
Qubits (S)	O($\log N$)	O($\log N$)
Cost (TS)	O($N \cdot \log N$)	**target** O(ploylog N)

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

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:

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

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

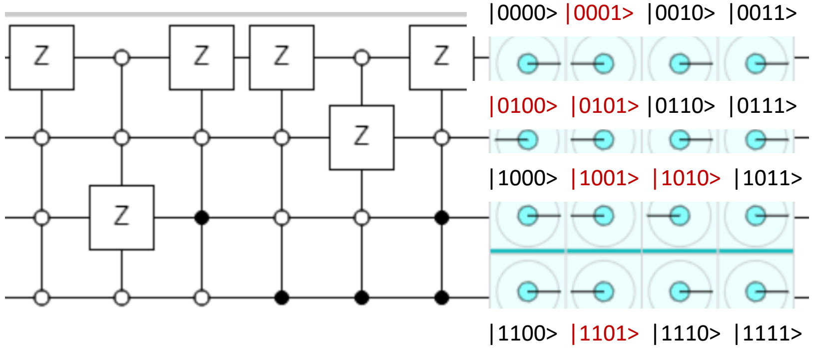
S1 -> S2:

$$W = [+1, -1, +1, +1, -1, -1, +1, +1, +1, -1, -1, +1, +1, -1, +1, +1]^T$$

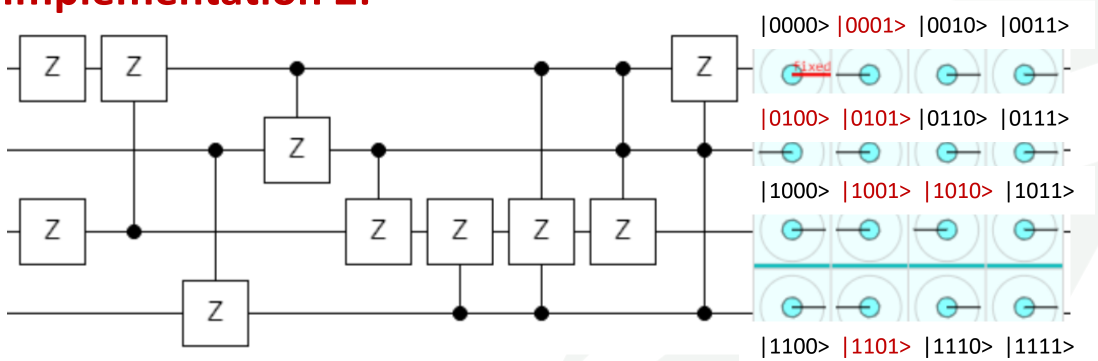
|0000> |0001> |0010> |0011> |0100> |0101> |0110> |0111> |1000> |1001> |1010> |1011> |1100> |1101> |1110> |1111>

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

Implementation 1 (example in Quirk):

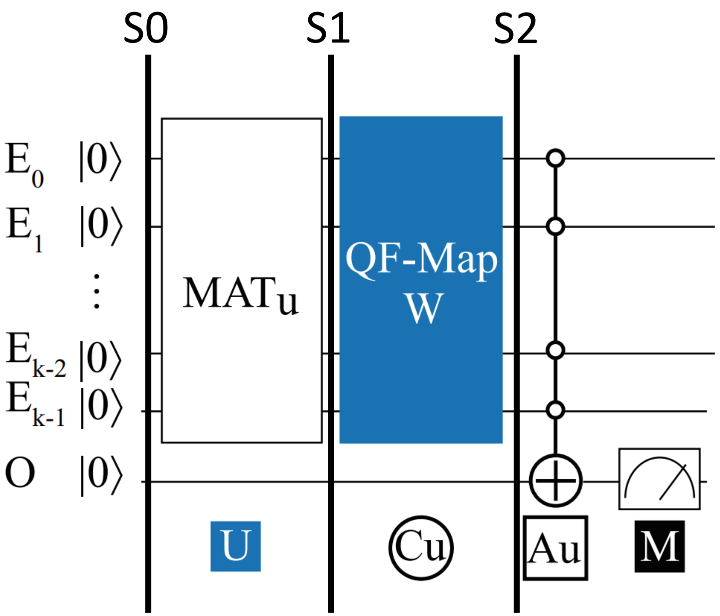


Implementation 2:



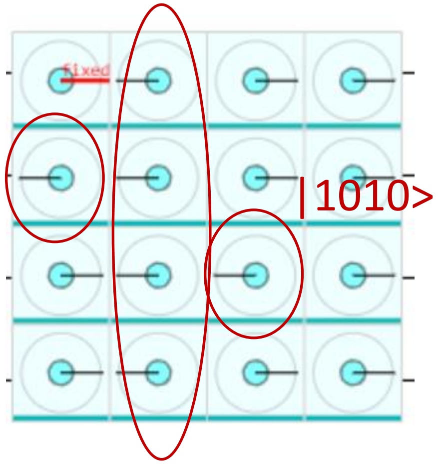
[ref] Tacchino, F., et al., 2019. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1), pp.1-8.

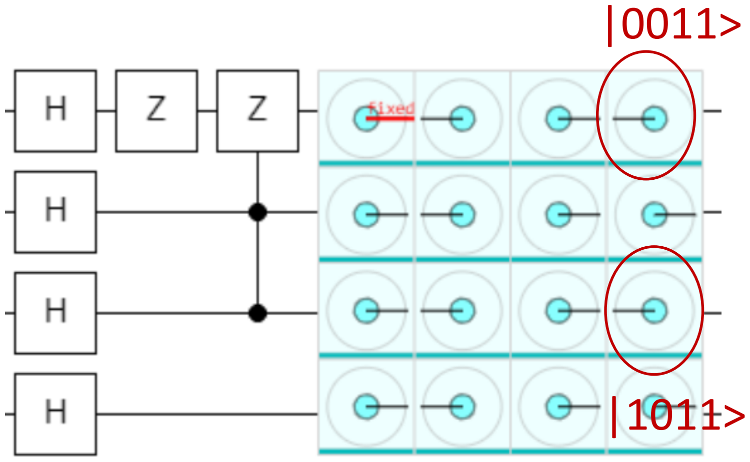
QuantumFlow: Taking NN Property to Design QC



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.





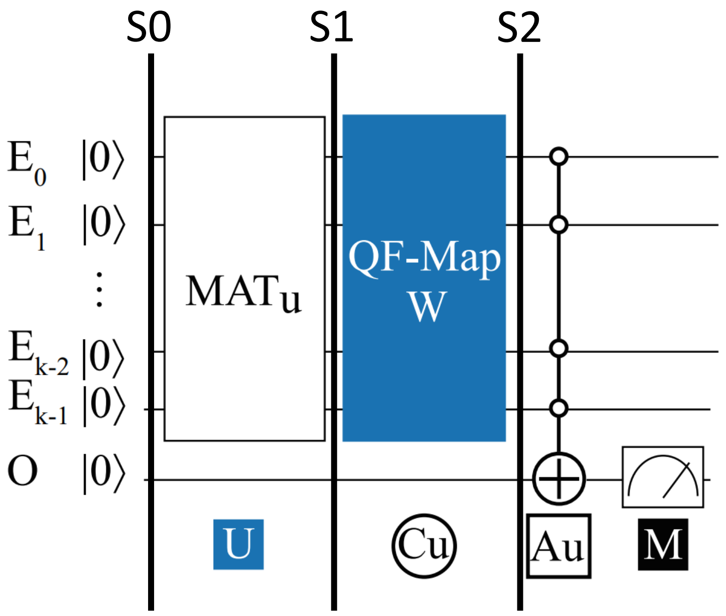
$S1 = [0, 0.59, 0, \mathbf{0}, \mathbf{0}, 0.07, 0, 0, 0.66, \mathbf{0.33}, \mathbf{0.33}, 0, 0, 0, 0]^T$

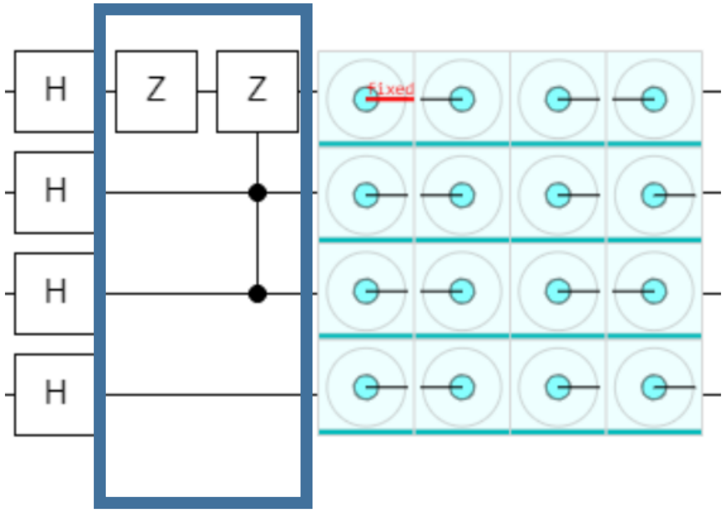
ori + - - +

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





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

$O(k^2)$

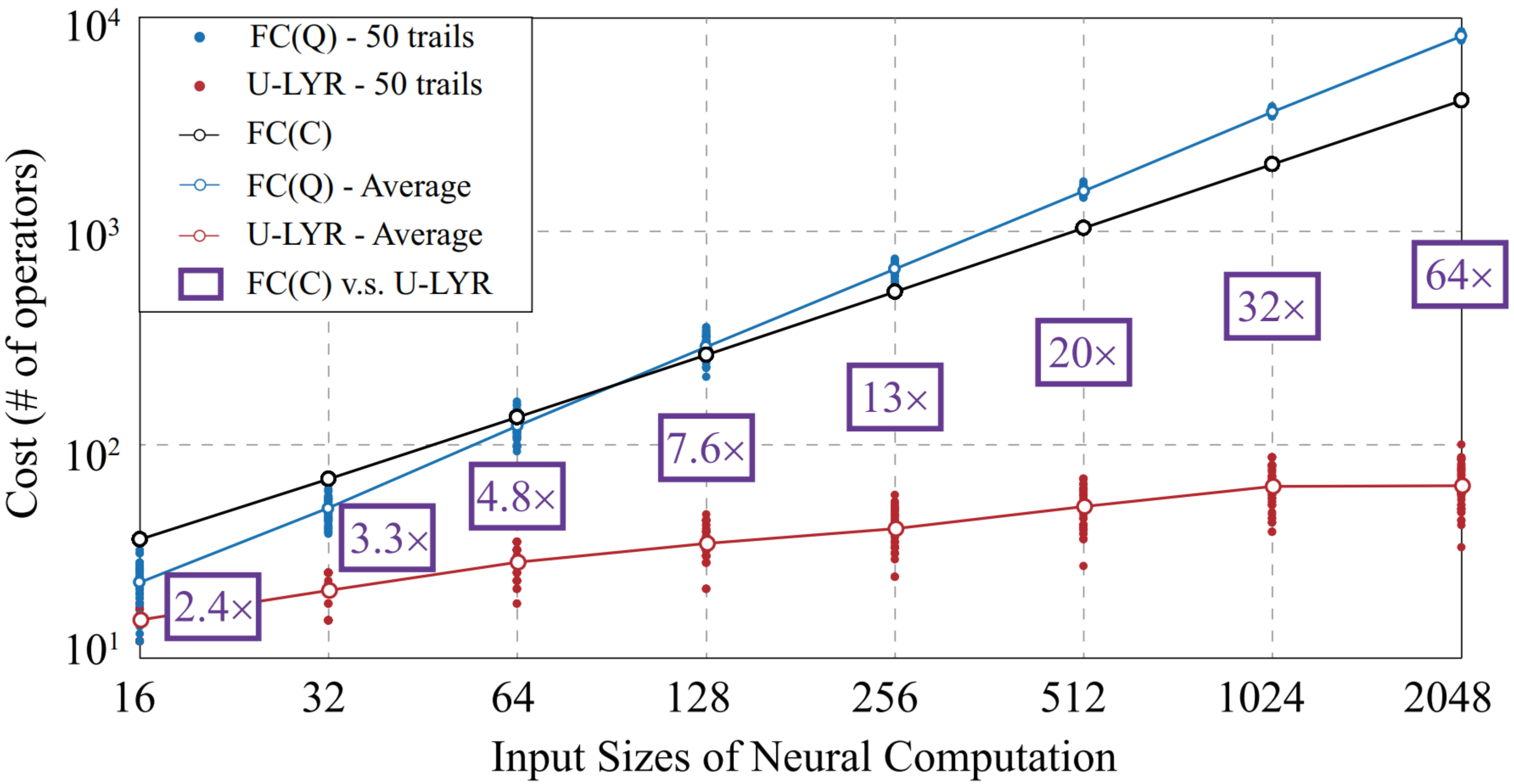
Hands-On Tutorial (4)

PreP + U_P+ Optimized U_N+ M+PostP (MNIST)

Outline – QuantumFlow

- Motivation
- **General Framework for Quantum-Based Neural Network Accelerator**
  - Data Preparation and Encoding
  - *Colab Hands-On (2): From Classical Data to Quantum Data*
  - Quantum Circuit Design
  - *Colab Hands-On (3): A Quantum Neuron*
- **Co-Design toward Quantum Advantage**
  - Challenges?
  - Feedforward Neural Network
  - *Colab Hands-On (4): End-to-End Neural Network on MNIST*
  - Optimization for Quantum Neuron
  - *Colab Hands-On (5): QuantumFlow*
  - Results

QuantumFlow Results



[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%.

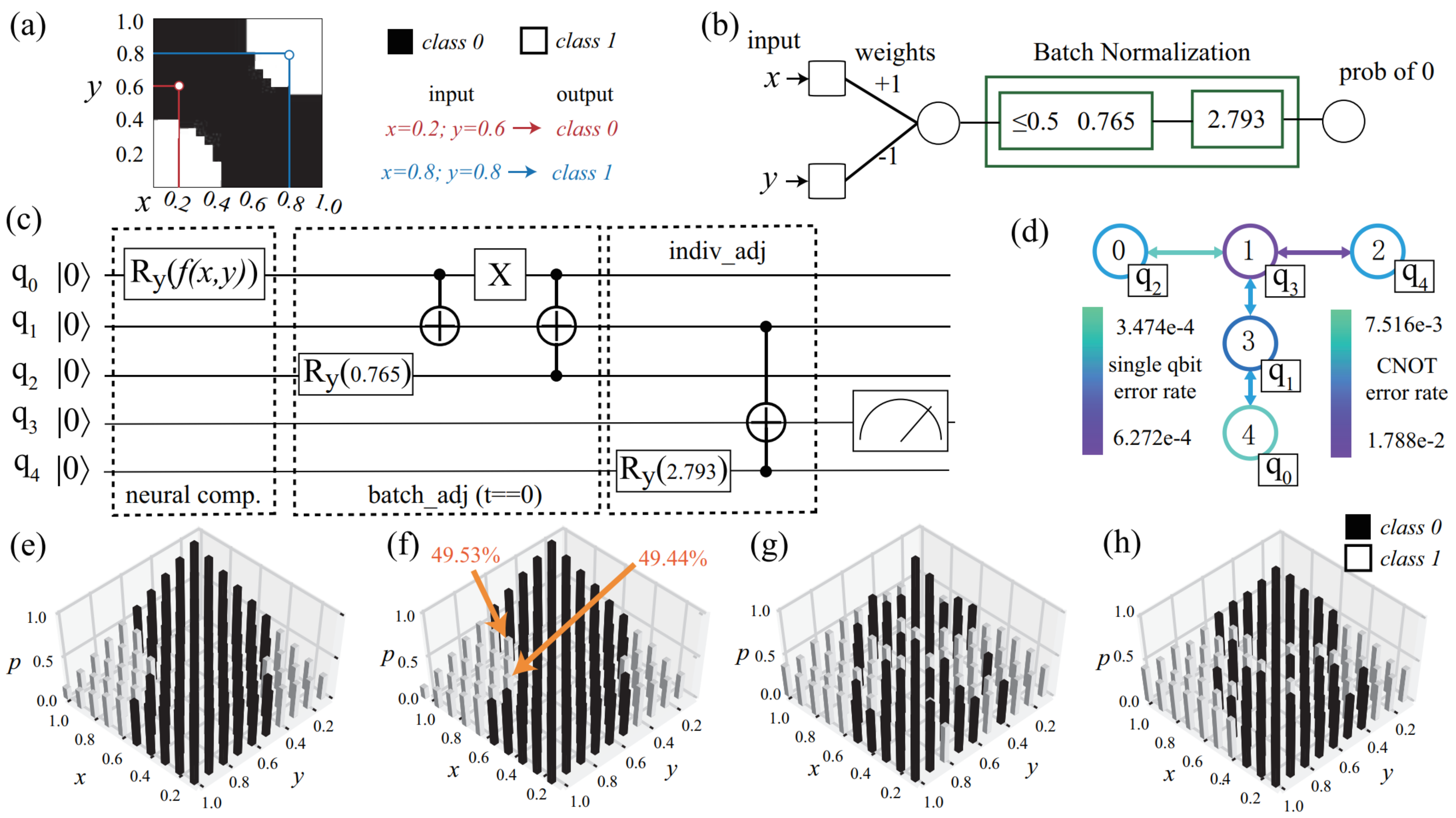
[ref of FFNN] Tacchino, F., et al., 2019. Quantum implementation of an artificial feed-forward neural network. *arXiv preprint arXiv:1912.12486*.

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” (retired) Quantum Processor



Hands-On Tutorial (5)

Comparison

Outline

- Background
- Co-Design: from Classical to Quantum
- QuantumFlow
  - Motivation
  - General Framework for Quantum-Based Neural Network Accelerator
  - Co-Design toward Quantum Advantage
- Recent works and conclusion

Motivation and Challenges

Deep neural network grows exponentially

Perf. of classical computing stops increasing

Quantum computer grows exponentially

Fundamental questions:

- Can we implement Neural Network on Quantum Computers?
- Can we achieve benefits in doing so?

Further questions:

- What is the best neural network architecture for quantum acceleration?
- What is the problem for near-term quantum computing, i.e., in NISQ era?

On-Going Works in Building Quantum NN Co-Design Stack and Next

Current works:

Quatnum NN Co-Design Stack

Co-Design Framework
Quantum Flow

Network exploration — QF-Mixer

VQC

Advantage

- Real-valued weights

Disadvantage

- Linear classifier
- Cannot be extended to multiple nonlinear layers with low cost

Mixing

1+1 > 2

QuantumFlow

Disadvantage

- Binary weights

Advantage

- Easy to be extended to multiple nonlinear layers w/t measurement

TABLE I
EVALUATION OF QNNS WITH DIFFERENT NEURAL ARCHITECTURE

Architecture		MNIST-2†	MNIST-3†	MNIST-4‡	MNIST-5‡	MNIST§
VQC (V×R1)		**97.91%**	90.09%	93.45%	91.35%	52.77%
QuantumFlow		95.63%	91.42%	94.26%	89.53%	69.92%
QF-MixNN	V+U	97.36%	**92.77%**	**94.41%**	**93.85%**	88.46%
	V+U+P	87.45%	82.9%	92.44%	91.56%	**90.62%**
	V+P	91.72%	76.93%	88.43%	85.02%	49.57%

Input resolutions: † 4×4; ‡ 8×8; § 16×16;

Exploration of Quantum Neural Architecture by Mixing Quantum Neuron Designs

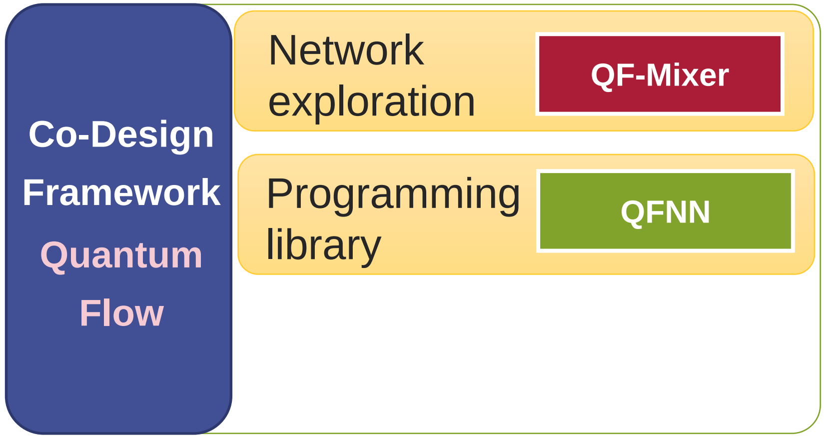
Z. Wang, Z. Liang, S. Zhou, C. Ding, J. Xiong, Y. Shi, **W. Jiang**, *Accepted by IEEE/ACM International Conference On Computer-Aided Design (**ICCAD**), Virtual, 2021.* ***(11/02/2021)***

On-Going Works in Building Quantum NN Co-Design Stack and Next

Current works:

Quatnum NN Co-Design Stack

Qiskit + PyTorch + python Package Index



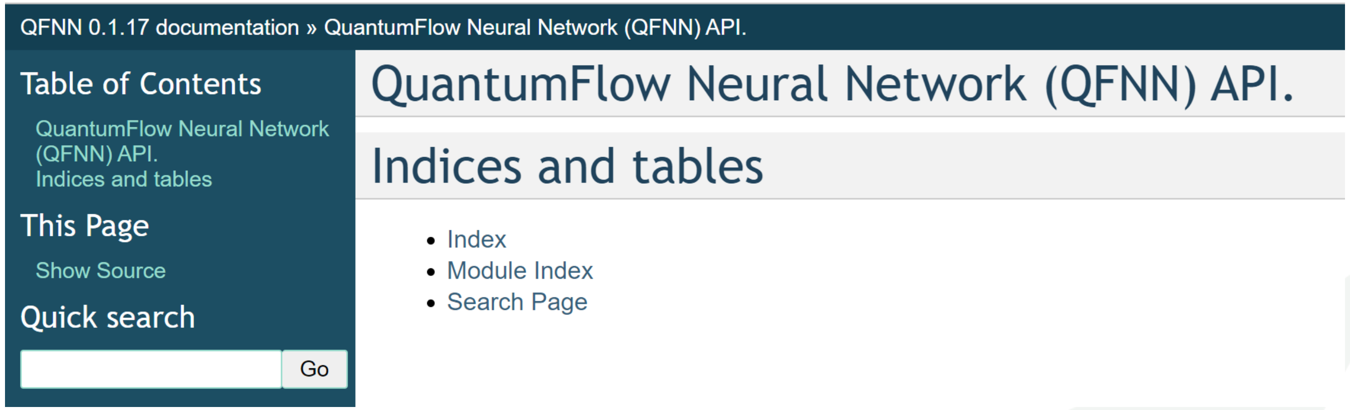

QFNN 0.1.17 documentation » QuantumFlow Neural Network (QFNN) API.

Table of Contents

QuantumFlow Neural Network (QFNN) API.
Indices and tables

This Page

Show Source

Quick search

Go

QuantumFlow Neural Network (QFNN) API.

Indices and tables

- Index
- Module Index
- Search Page

https://jqub.ece.gmu.edu/categories/QF/qfnn/index.html

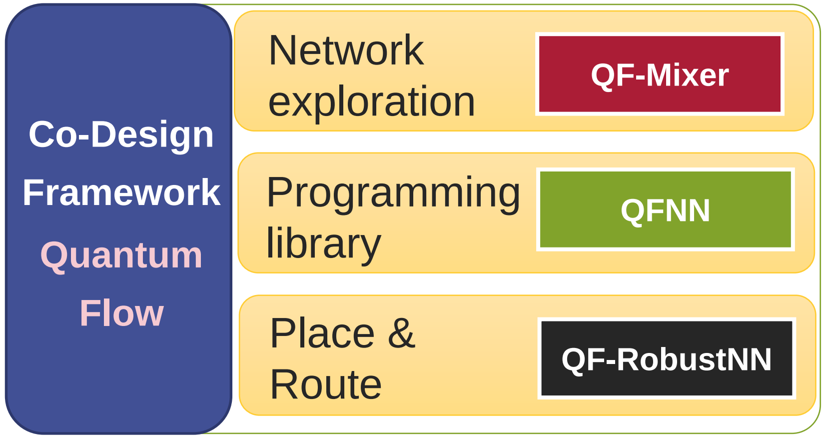
https://github.com/jqub/qfnn

QuantumFlow: An End-to-End Quantum Neural Network Acceleration Framework
Zhirui Hu and **W. Jiang**
*IEEE International Conference on Quantum Computing and Engineering QCE 21 (**QuantumWeek**)*

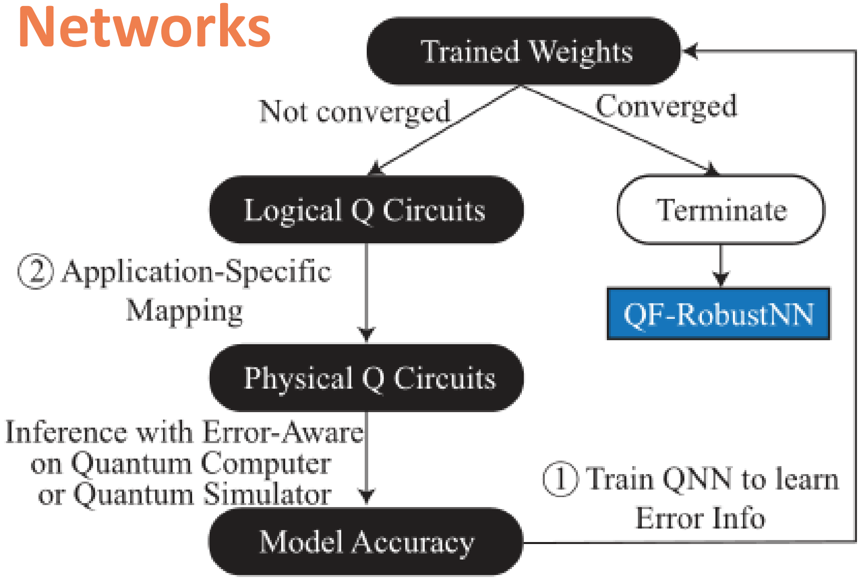
On-Going Works in Building Quantum NN Co-Design Stack and Next

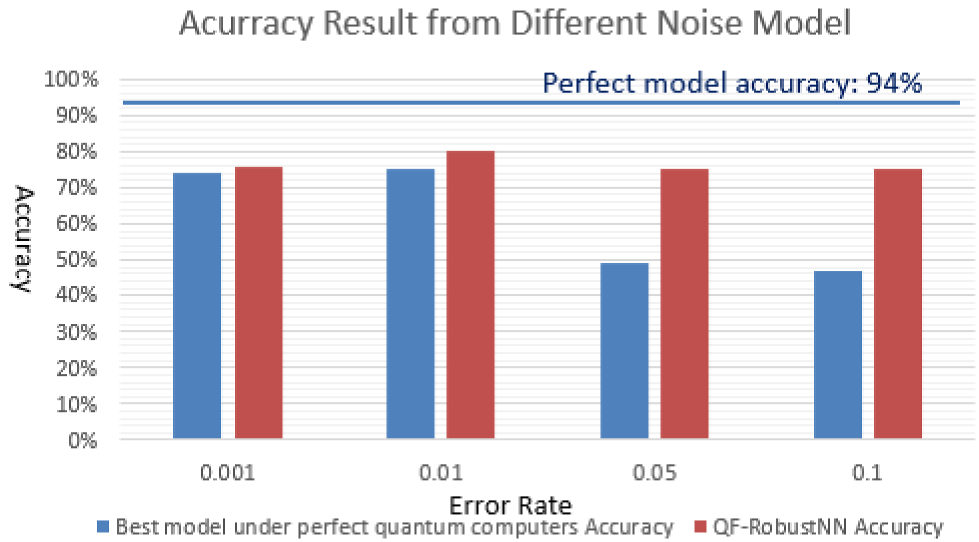
Current works:

Quatnum NN Co-Design Stack



The first noise-aware training for Quantum Neural Networks





Can Noise on Qubits Be Learned in Quantum Neural Network? A Case Study on QuantumFlow

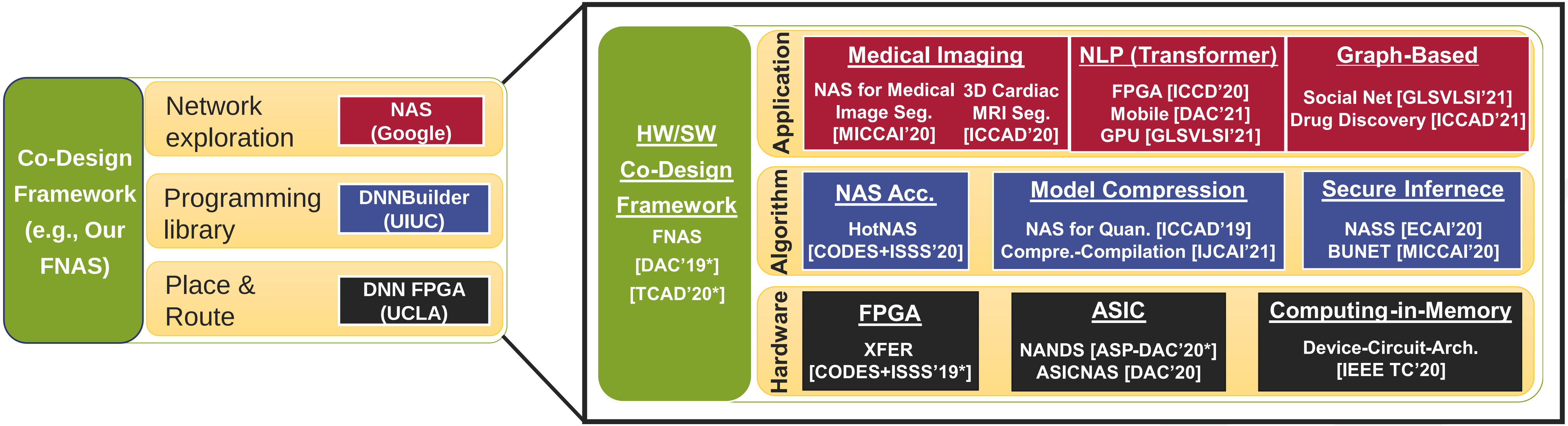
Z. Liang, Z. Wang, J. Yang, L. Yang, J. Xiong, Y. Shi, **W. Jiang**,

*Accepted by IEEE/ACM International Conference On Computer-Aided Design (**ICCAD**), Virtual, 2021.* ***(11/02/2021)***

Development of Co-Design Stack in Classical Computing

Our works:

Co-Design for Automation of Classical Neural Network Systems



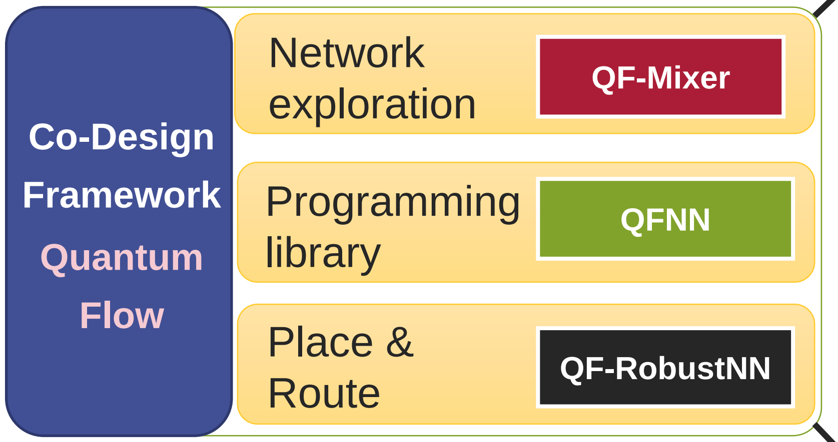
On-Going Works in Building Quantum NN Co-Design Stack and Next

Our future works:

Co-Design for Automation of Quantum Neural Network Systems

Current works:

Quatnum NN Co-Design Stack



Conclusion & Resources

- Quantum computing is promising for accelerating **neural networks**
- **Co-design** can build a better *quantum neural network accelerator*
- Along with the development of quantum computers and quantum neural networks, we will see **real-world applications** in the NISQ Era

https://github.com/JQub/QuantumFlow_Tutorial (Source Code of All Hands-On in Tutorial)

https://github.com/JQub/qfnn (Source Code of QFNN API & Place to post Issues)

https://pypi.org/project/qfnn/ (Package of QFNN on PYPI)

https://libraries.io/pypi/qfnn/ (QFNN on Libraries.io)

https://jqub.ece.gmu.edu (JQub Website)

https://jqub.ece.gmu.edu/categories/QF (News and **slides**)

https://jqub.ece.gmu.edu/categories/QF/qfnn/ (QFNN Documents)

https://www.nature.com/articles/s41467-020-20729-5

https://arxiv.org/pdf/2012.10360.pdf

https://arxiv.org/pdf/2109.03806.pdf

https://arxiv.org/pdf/2109.03430.pdf

GEORGE MASON UNIVERSITY

wjiang8@gmu.edu

George Mason University

4400 University Drive
Fairfax, Virginia 22030

Tel: (703)993-1000