A Co-Design Framework of Neural Networks and Quantum Circuits Towards Quantum Advantage

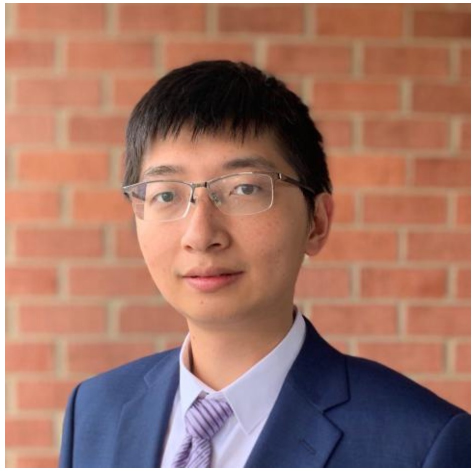
Weiwen Jiang, Ph.D.
Postdoc Research Associate
Department of Computer Science and Engineering
University of Notre Dame
wjiang2@nd.edu | https://wjiang.nd.edu
(Speaker)

Jinjun Xiong, Ph.D.
Program Director,
Cognitive Computing Systems Research
Thomas J. Watson Research Center
jinjun@us.ibm.com

Yiyu Shi, Ph.D.
Associate Professor
Department of Computer Science and Engineering
University of Notre Dame
yshi4@nd.edu

Pre-print: https://arxiv.org/abs/2006.14815
Project website: https://wjiang.nd.edu/categories/qf/

Speaker Information



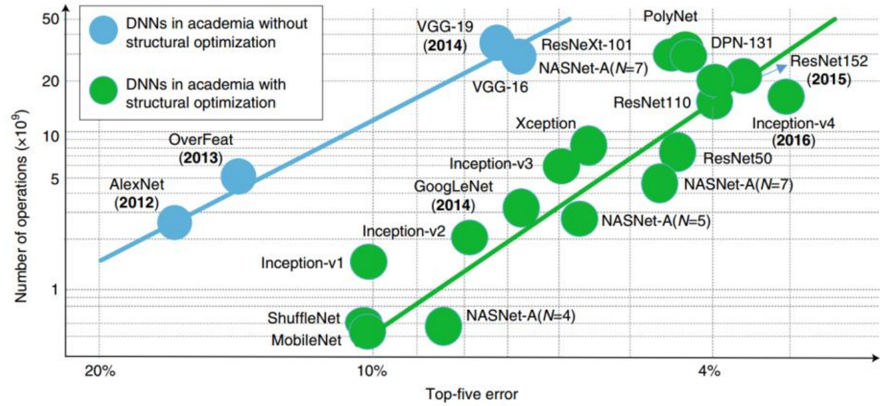
(visit my website)

Weiwen Jiang

- Postdoctoral Associate at the University of Notre Dame
- Research Directions:
  - Co-Design Neural Networks and Quantum Circuits
    - The first Co-Design Framework: QuantumFlow
    - Support from IBM Q Network
  - Co-Design of Neural Networks and Hardware Accelerators (FPGA, ASIC, CiM, etc.)
    - **Best Paper Nominations in DAC 2019, CODES+ISSS 2019, ASP-DAC 2020**
    - Funding support from NSF, Edgecortix, Facebook
  - Co-Design of General Applications and Embedded Systems
    - **Best Paper Award in ICCD 2017**
- **I am currently on Job Market for a Faculty Position**

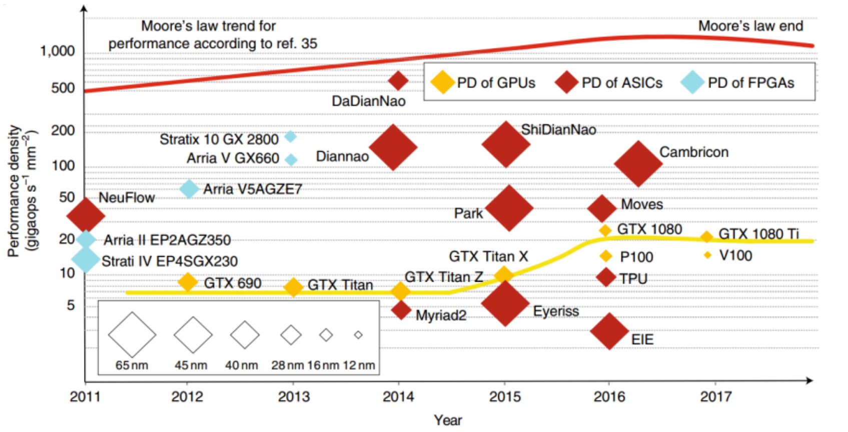
Why Neural Network on Quantum Computer?

NN on Classical Computer: Computation & Storage Demand > Supply



Number of DNN operations increases exponentially

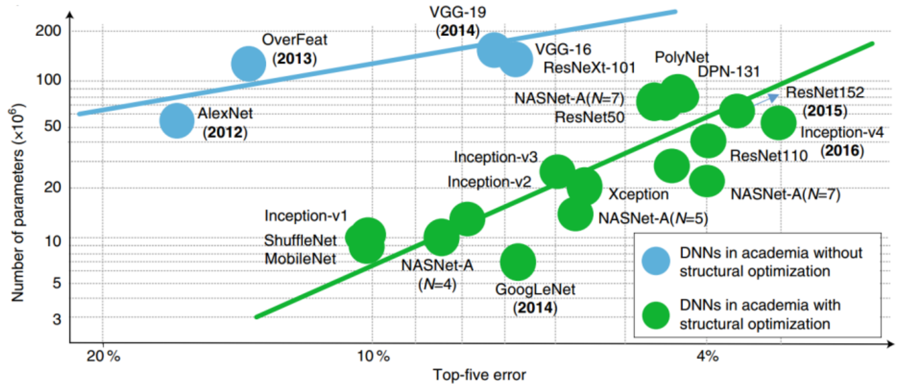
Computation Gap



Performance density almost stops increasing

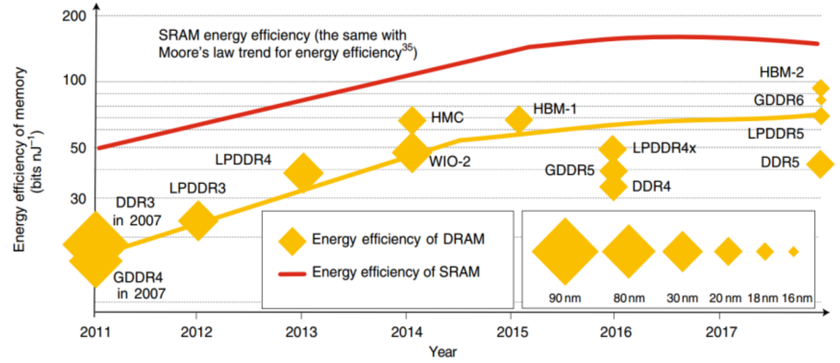
Neural Network Size

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.

The Power of Quantum Computers: Qbit

Classical Bit

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

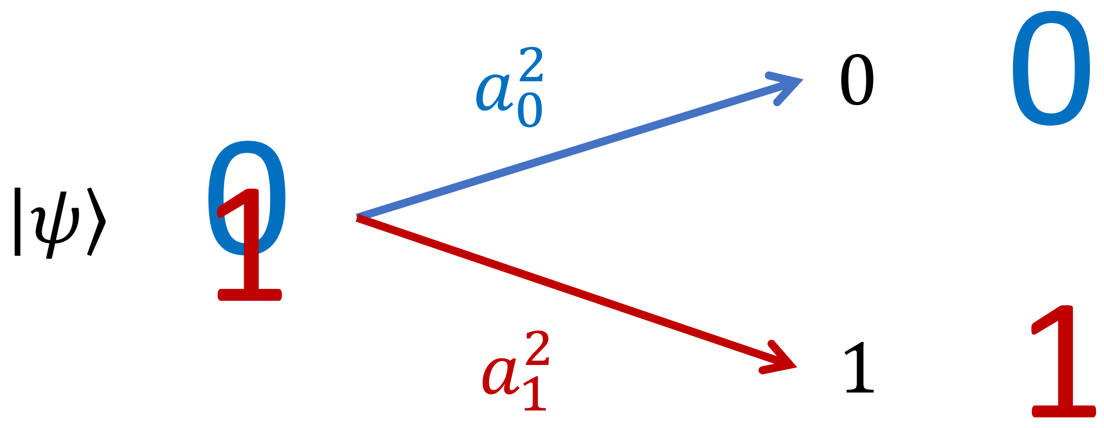
Quantum Bit (Qbit)

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

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

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

Reading out Information from Qbit (Measurement)



Probability

Non–Deterministic Computing

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

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

The Power of Quantum Computers: Qbits

2 Classical Bits

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

n bits for 1 value

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

2 Qbits

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

n bits for 2^n values

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

Qbits: $\boldsymbol{q_0, q_1}$

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

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

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

- $|00\rangle$: 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 $|0\rangle$
- $c_{00}^2 = a_0^2 \times b_0^2$; $c_{00} = \sqrt{(a_0^2 \times b_0^2)}$

The Power of Quantum Computers: Logic Gate

UNIVERSITY OF NOTRE DAME

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

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

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

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

$$H \otimes H|q_0, q_1\rangle$$
$$= d_{00}|00\rangle + d_{01}|01\rangle + d_{10}|10\rangle + d_{11}|11\rangle$$

Matrix multiplication on classical computer using 16bit number

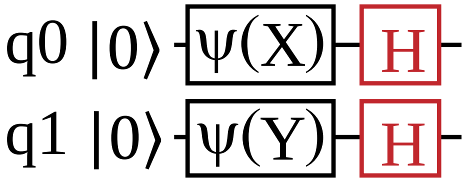
$$A_{N,N} \times B_{N,1} = C_{N,1}$$

Data: $(M \times M + 2 \times M) \times 16bit, \quad M = 2^2$

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

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

Special Matrix multiplication on quantum computer

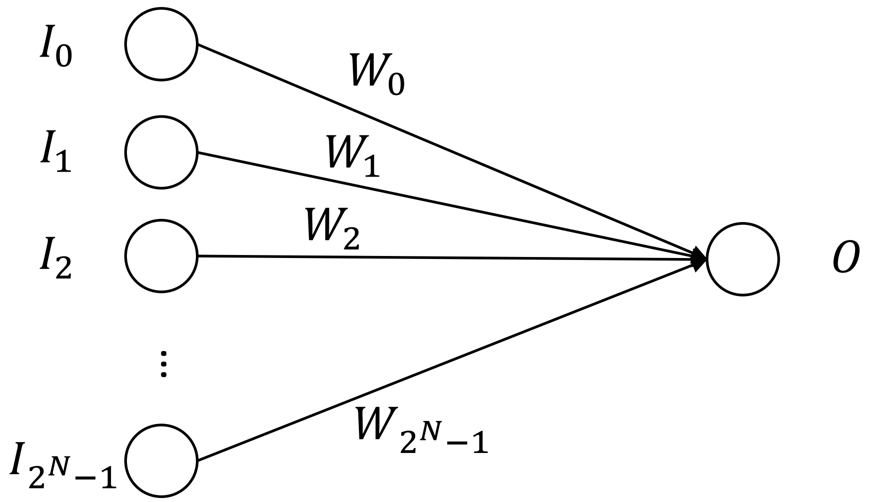


Data: **K** Qbits, **K = logM = 2**

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

Goal: From $O(2^N)$ to $O(poly(N))$ for Neural Computation

UNIVERSITY OF NOTRE DAME



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

where δ is a non-linear function, say quadratic

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

Operation: Multiplication: $O(2^N)$

Accumulation: $O(2^N)$

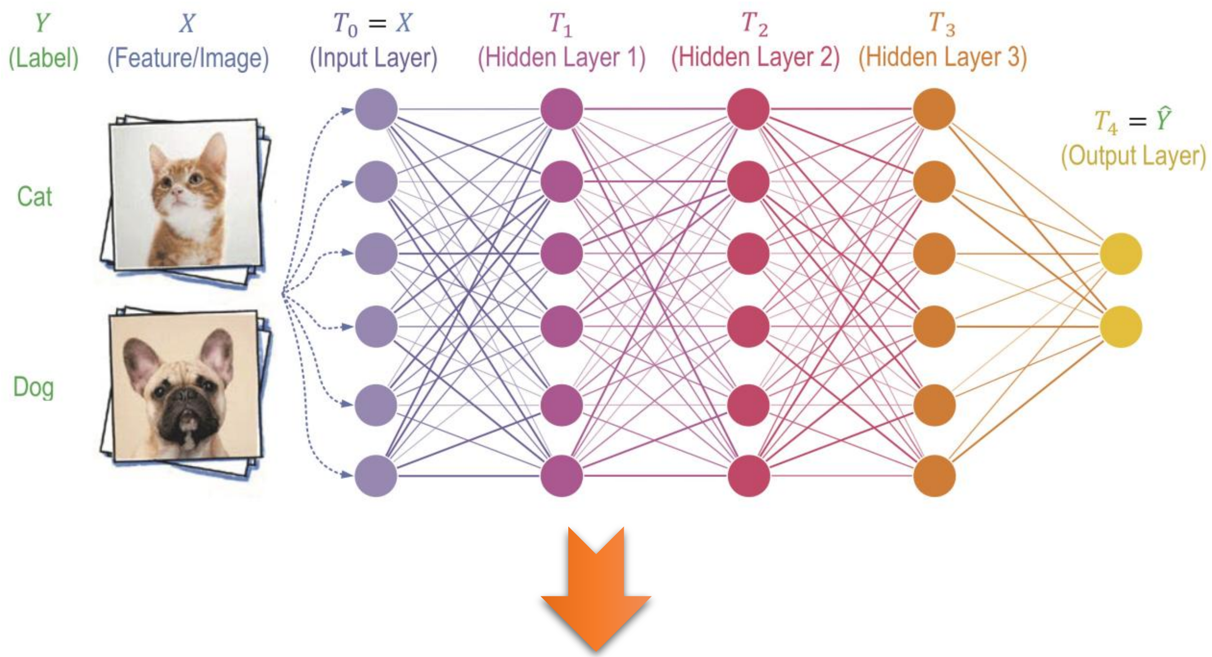
Neural Computation with input size of 2^N on quantum computer

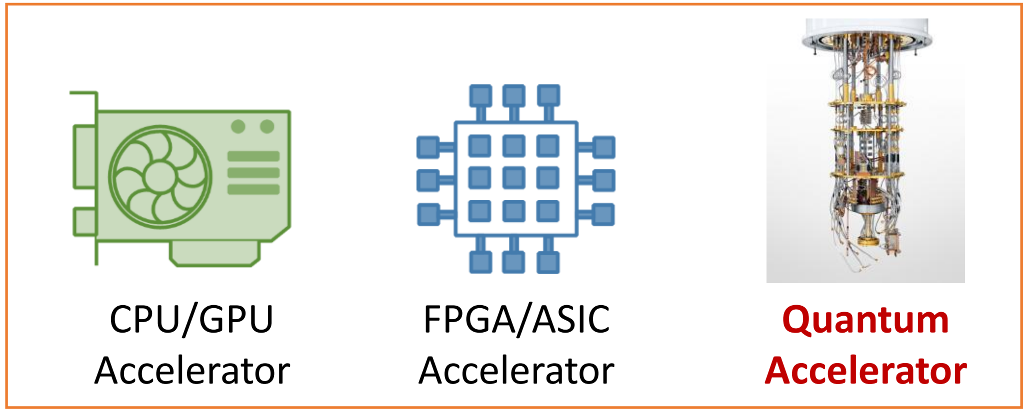
Basic Logical Gates: $O(poly(N))$, say $O(N^2)$?

Quantum Computing has Great Potential to Close the Gaps

But How?

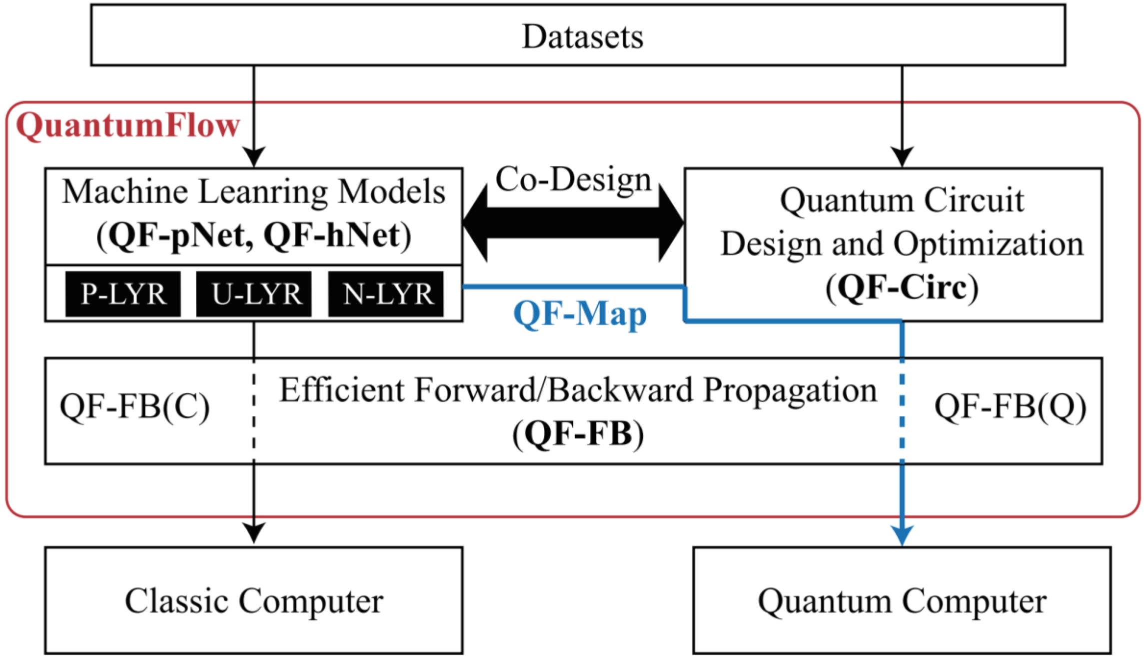
Quantum Computing for Neural Network



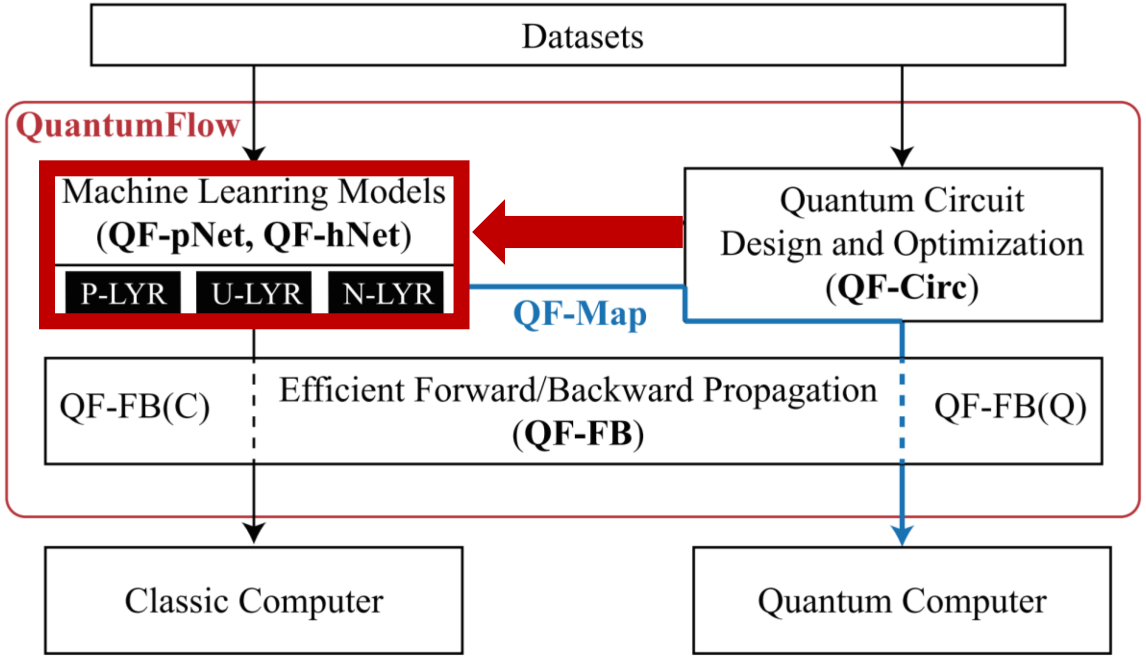


- What is the Quantum Friendly Neural Network?
- How to automatically map NN to QC?
- Can we achieve quantum advantage by implementing NN on QC?

QuantumFlow: A Co-Design Framework



QuantumFlow: Design QC-Aware NN



- **What is the Quantum Friendly NN?**

1. Operating 2^N inputs on N qbits!

   U-LYR: Unitary matrix-based data encoding

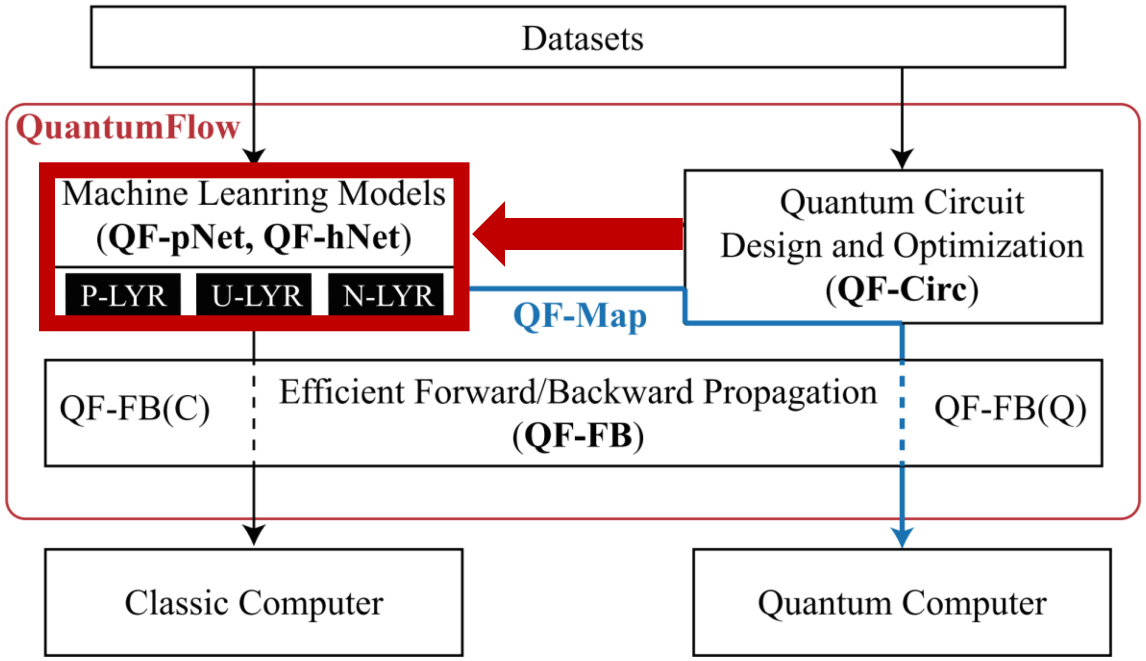
2. Connecting layers without measurement!

   P-LYR: Random variable-based data encoding

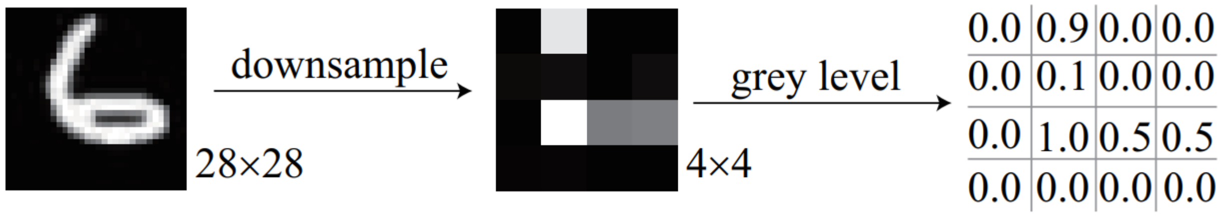
3. Normalizing intermediate results!

   N-LYR: Quantum friendly normalization

QuantumFlow: QF-Nets (U-LYR)



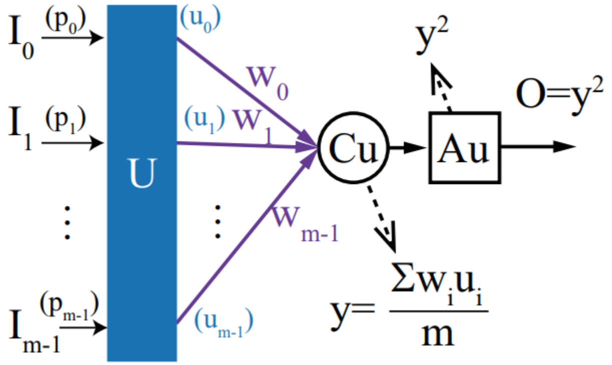
1. Operating 2^N inputs on N qbits!



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

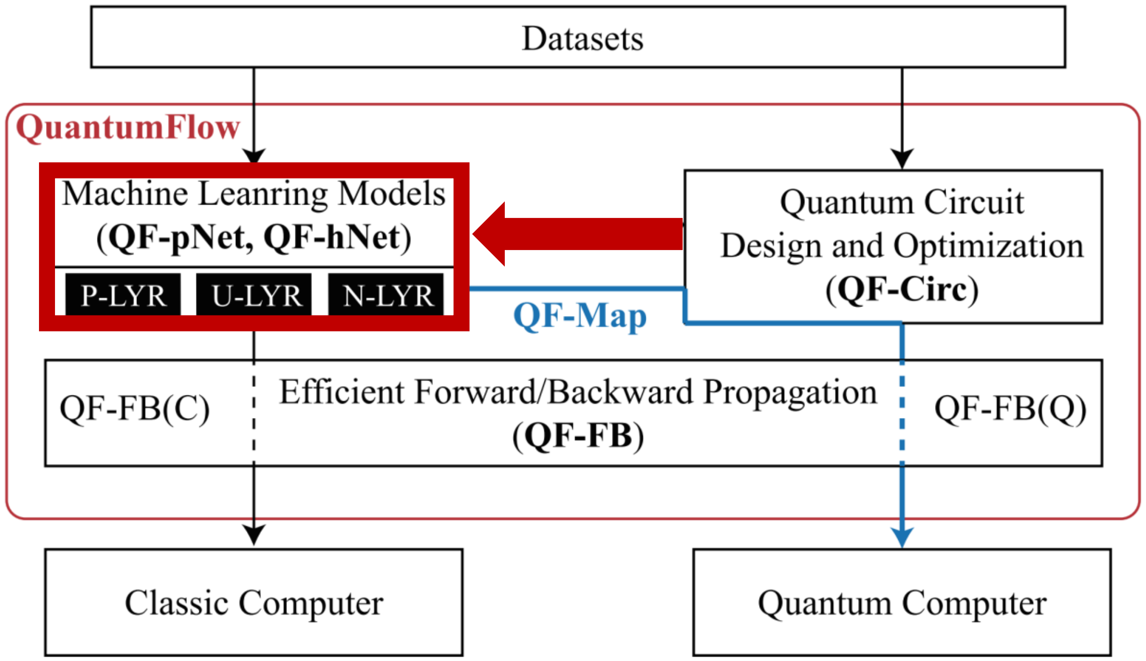
U

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

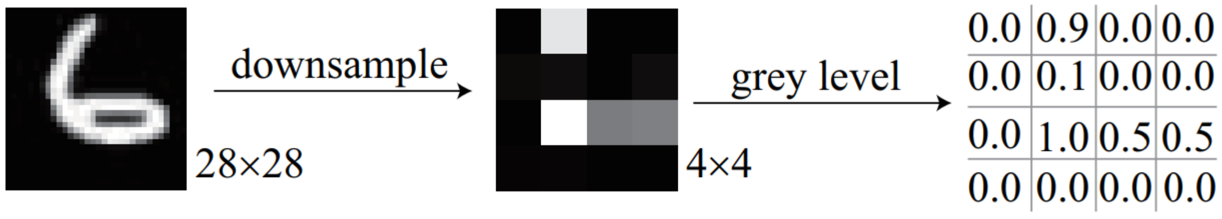


- The **first column** in a unitary matrix
- **16** inputs can be encoded to **16** states of **4 qbits** for computation.

QuantumFlow: QF-Nets (P-LYR)



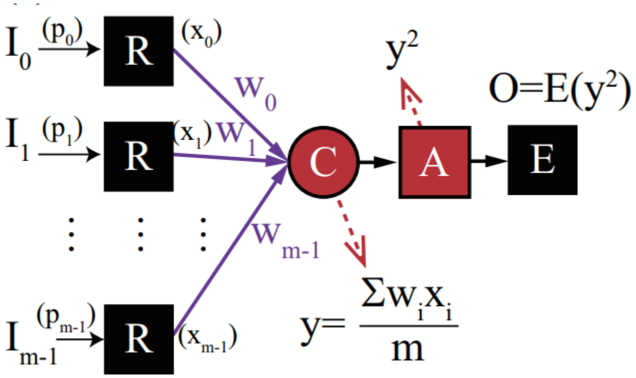
2. Connecting layers without measurement!



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

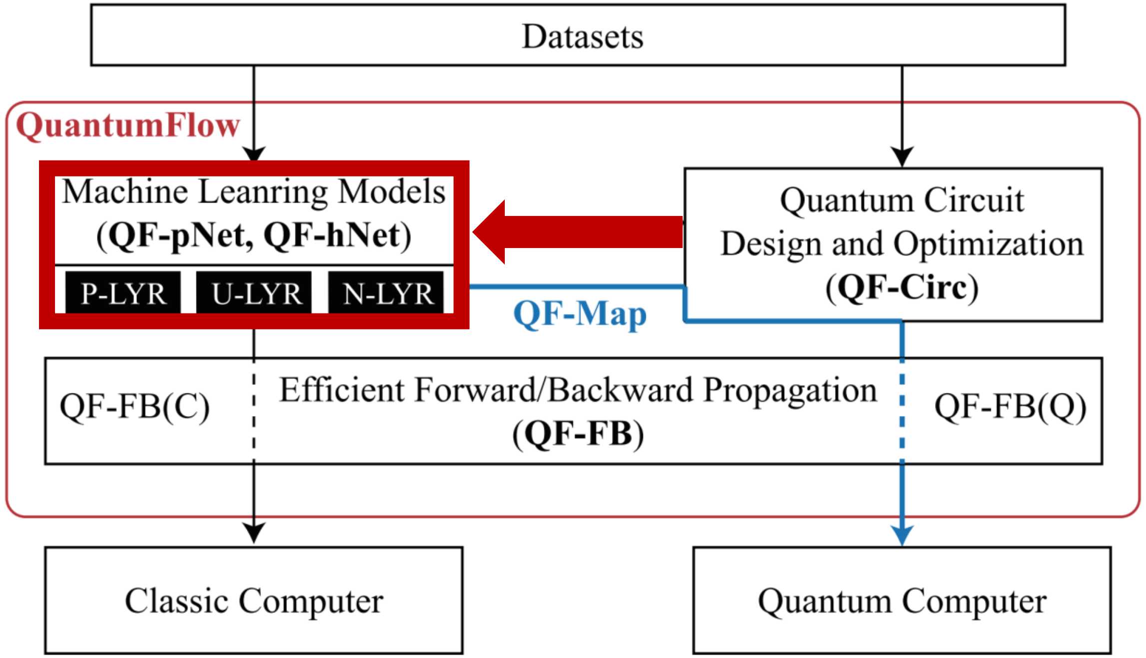
$\xrightarrow{R}$

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

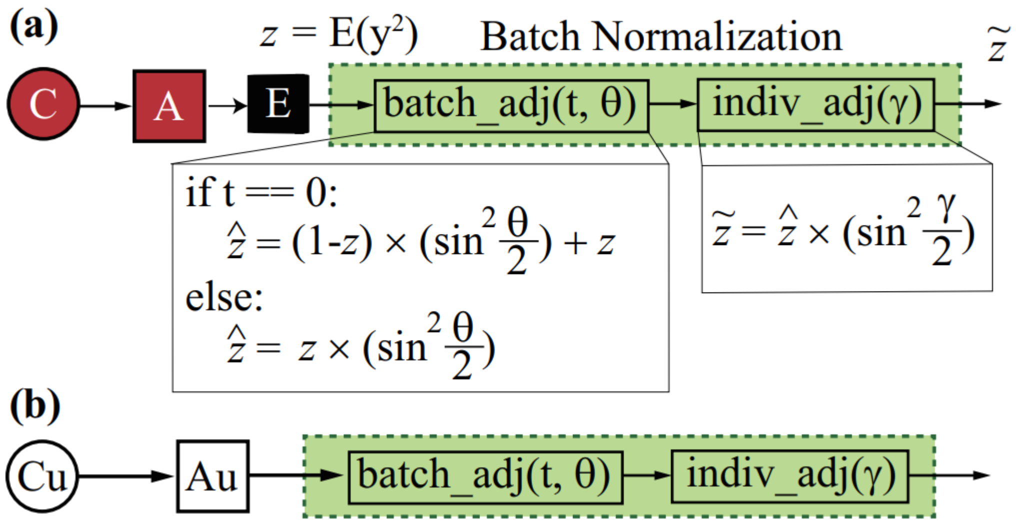


- The output of a U-LYR is stored on a qbits
- Using **random variable** as input can operate outputs without measurement.

QuantumFlow: QF-Nets (N-LYR)

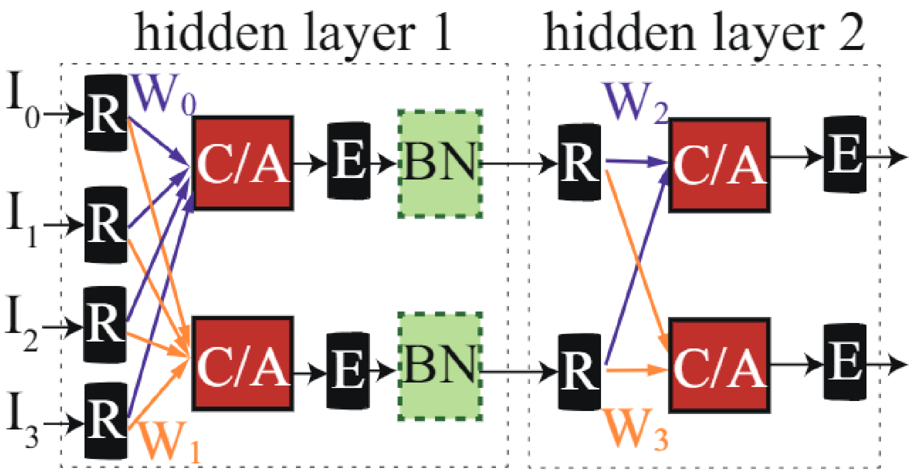


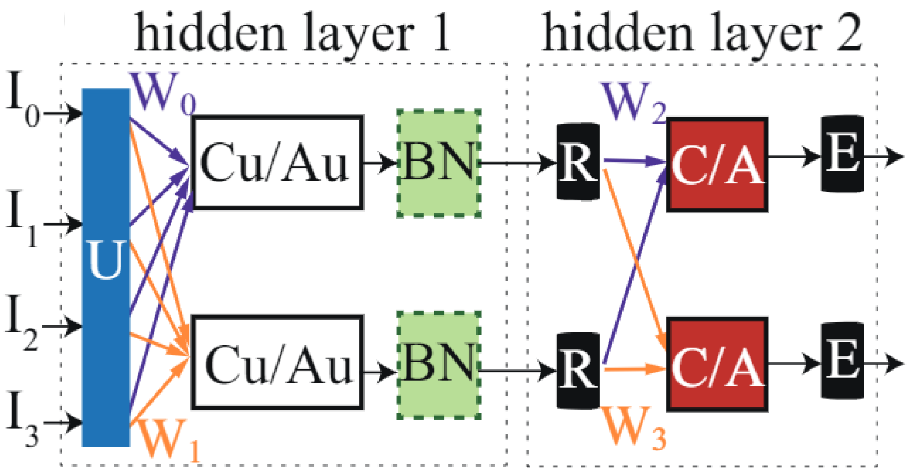
3. Normalize intermediate results!



- **batch_adj:** normalize output qbit in a batch to have probability of 50%
- **Indiv_adj:** adjust the probability of different neurons to make difference for classification

QuantumFlow: QF-pNet and QF-hNet

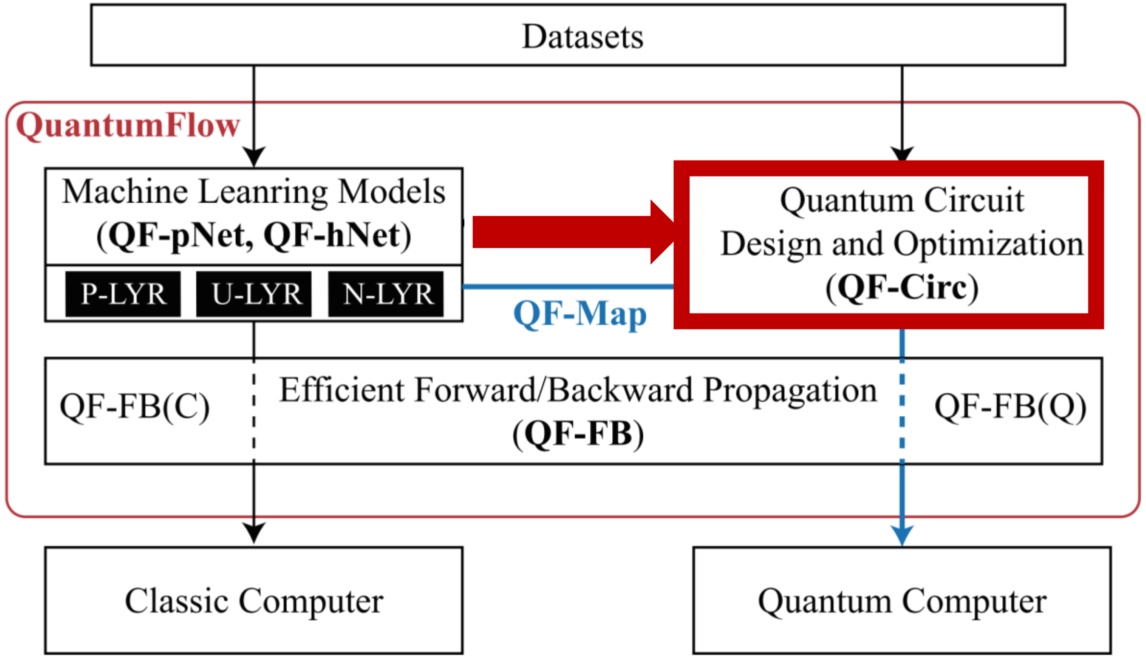




QF-pNet: P-LYR+N-LYR

QF-hNet: U-LYR+P-LYR+N-LYR

QuantumFlow: Taking NN Property to Design QC



- **How to Map NN to QC towards Q-Advantage?**

1. P-LYR and N-LYR (see the paper)

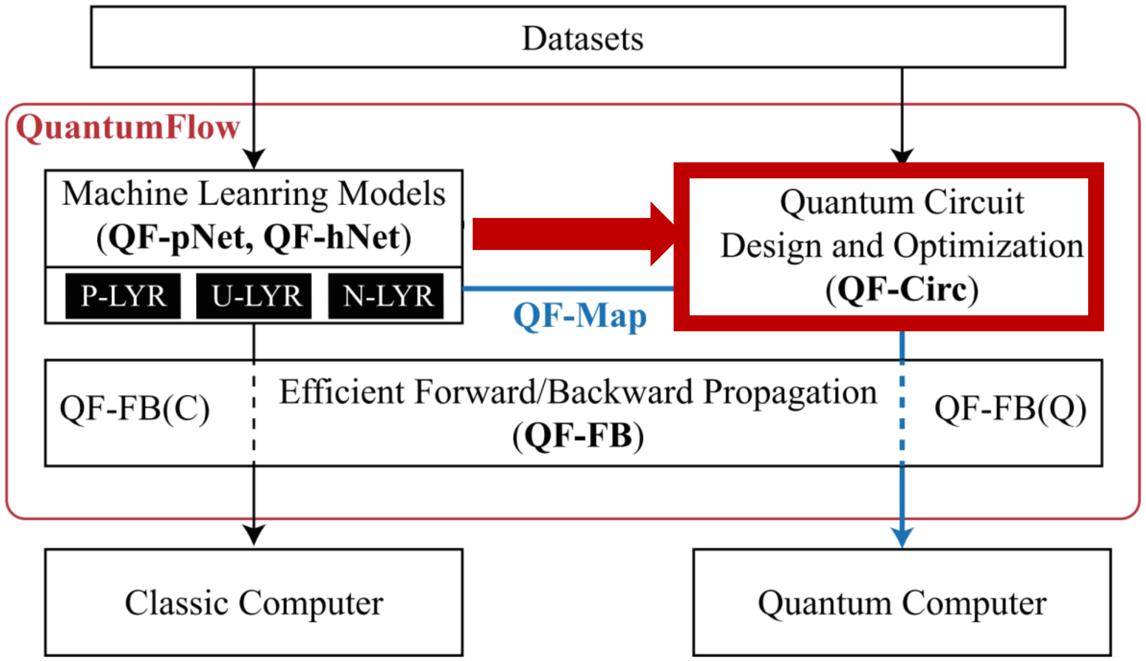
Straightforward mapping, benefiting from the quantum aware design

2. **U-LYR**

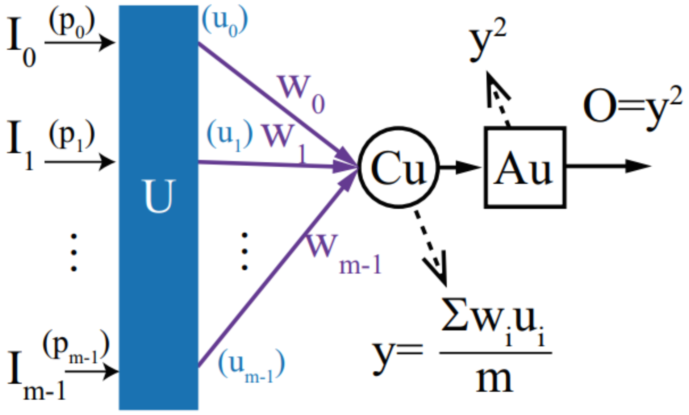
With the help of NN property to achieve quantum advantage

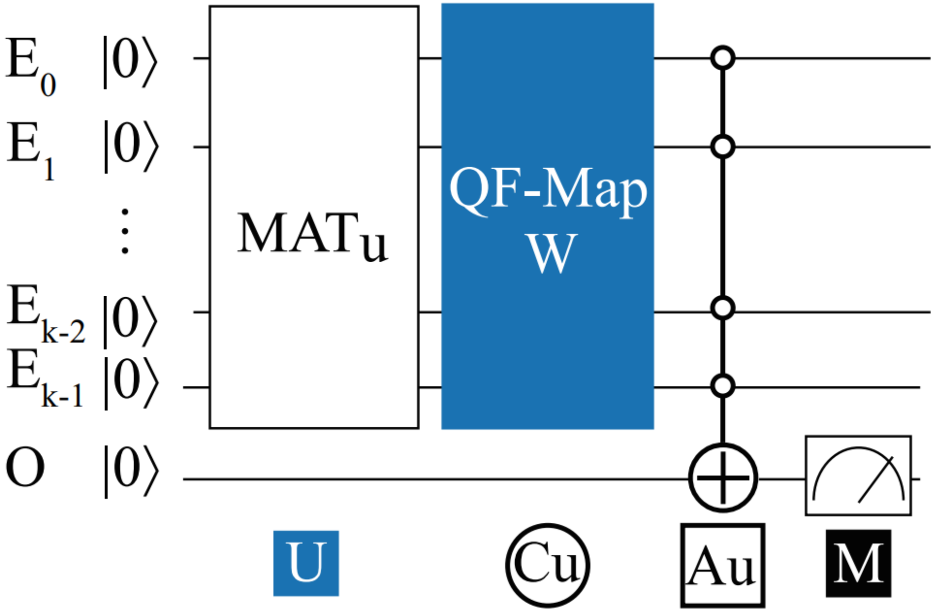
QuantumFlow: Taking NN Property to Design QC

UNIVERSITY OF NOTRE DAME



2. U-LYR



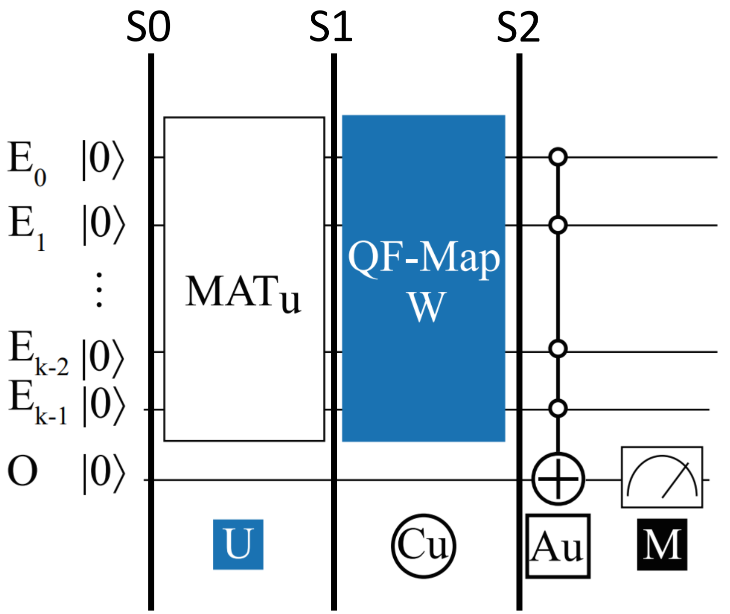


$[0, 0.9, 0, 0, 0, 0.1, 0, 0, 1.0, 0.5, 0.5, 0, 0, 0, 0]^T$

$\downarrow$ 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:

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

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

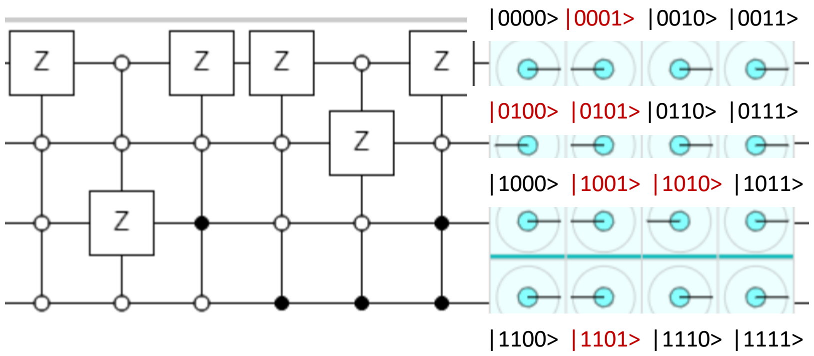
S1 -> S2:

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

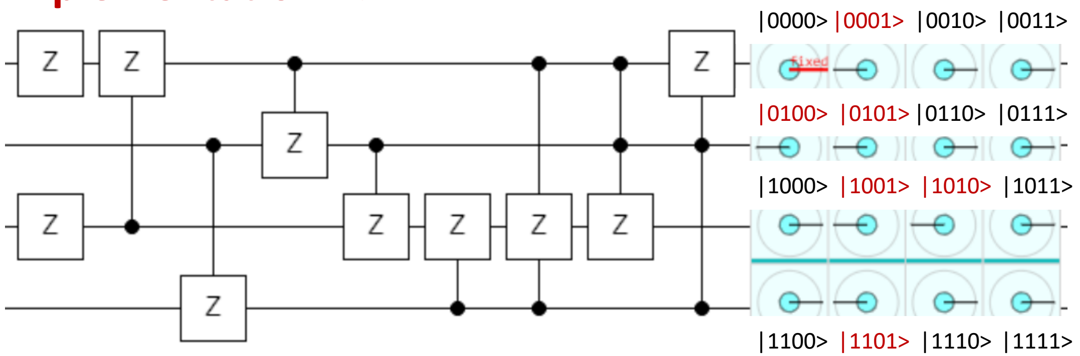
|0000> |0001> |0010> |0011> |0100> |0101> |0110> |0111> |1000> |1001> |1010> |1011> |1100> |1101> |1110> |1111>

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

Implementation 1 (example in Quirk):

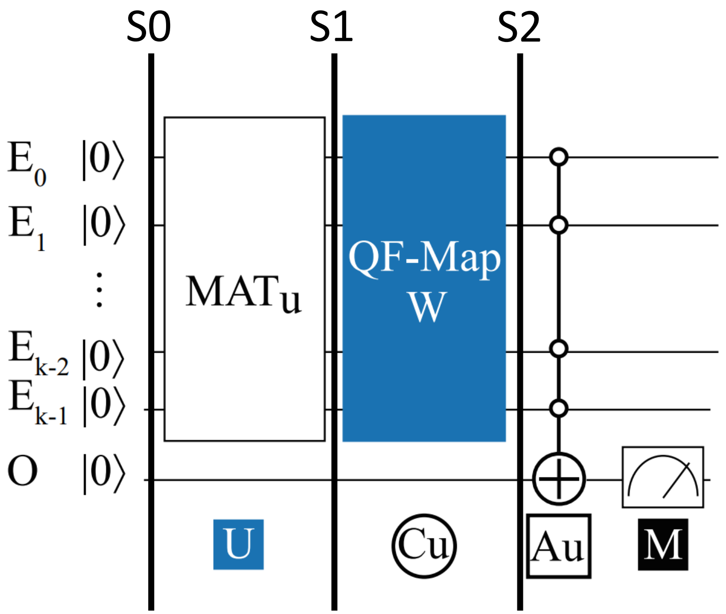


Implementation 2:



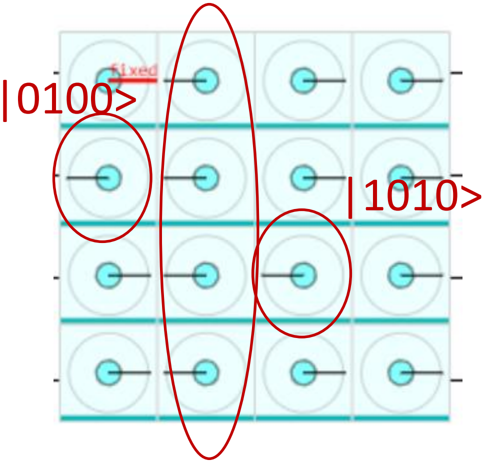
[ref] Tacchino, F., et al., 2019. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1), pp.1-8.

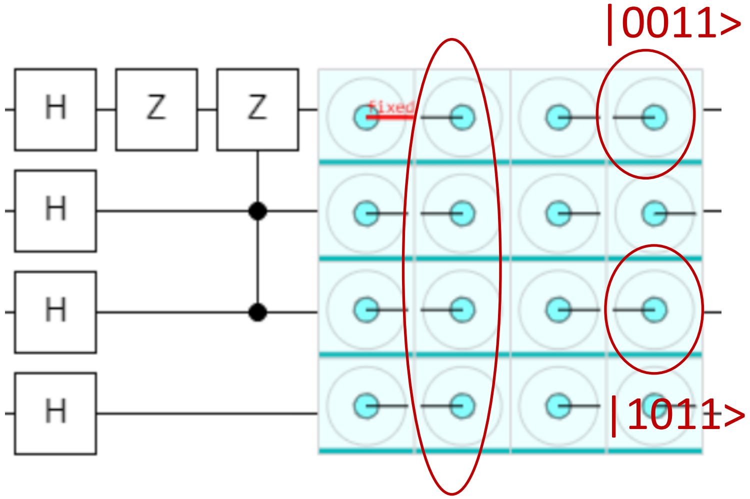
QuantumFlow: Taking NN Property to Design QC



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.





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

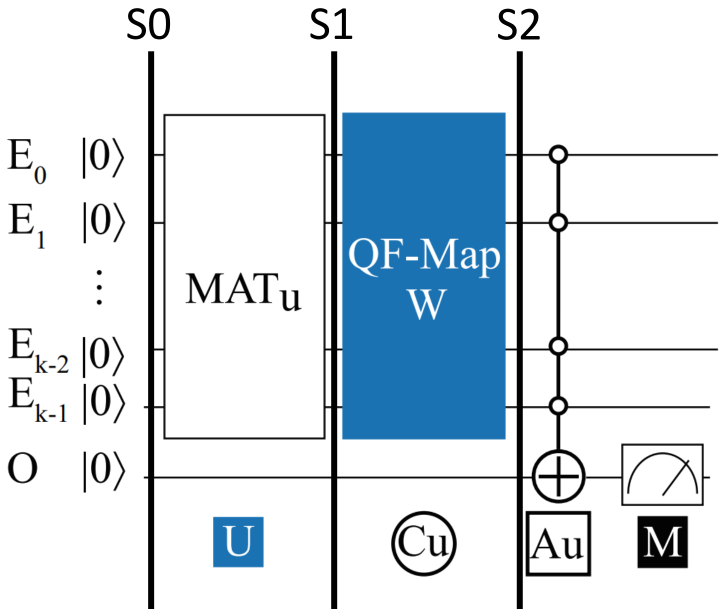
ori + - - +

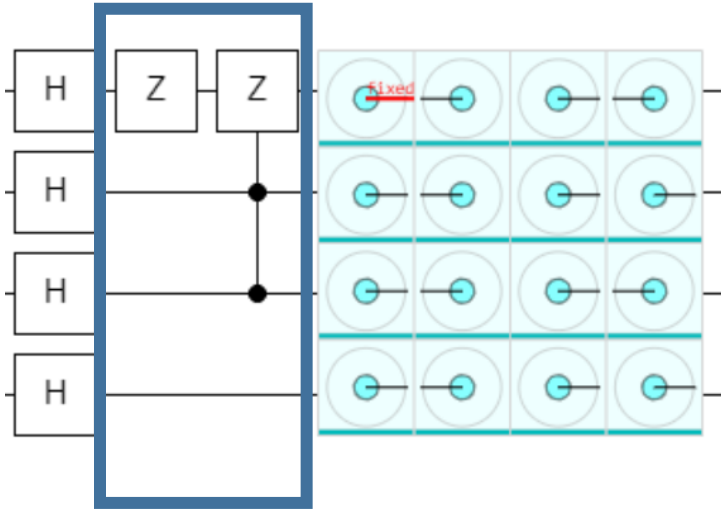
fin - + + -

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

QuantumFlow: Taking NN Property to Design QC

UNIVERSITY OF NOTRE DAME





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

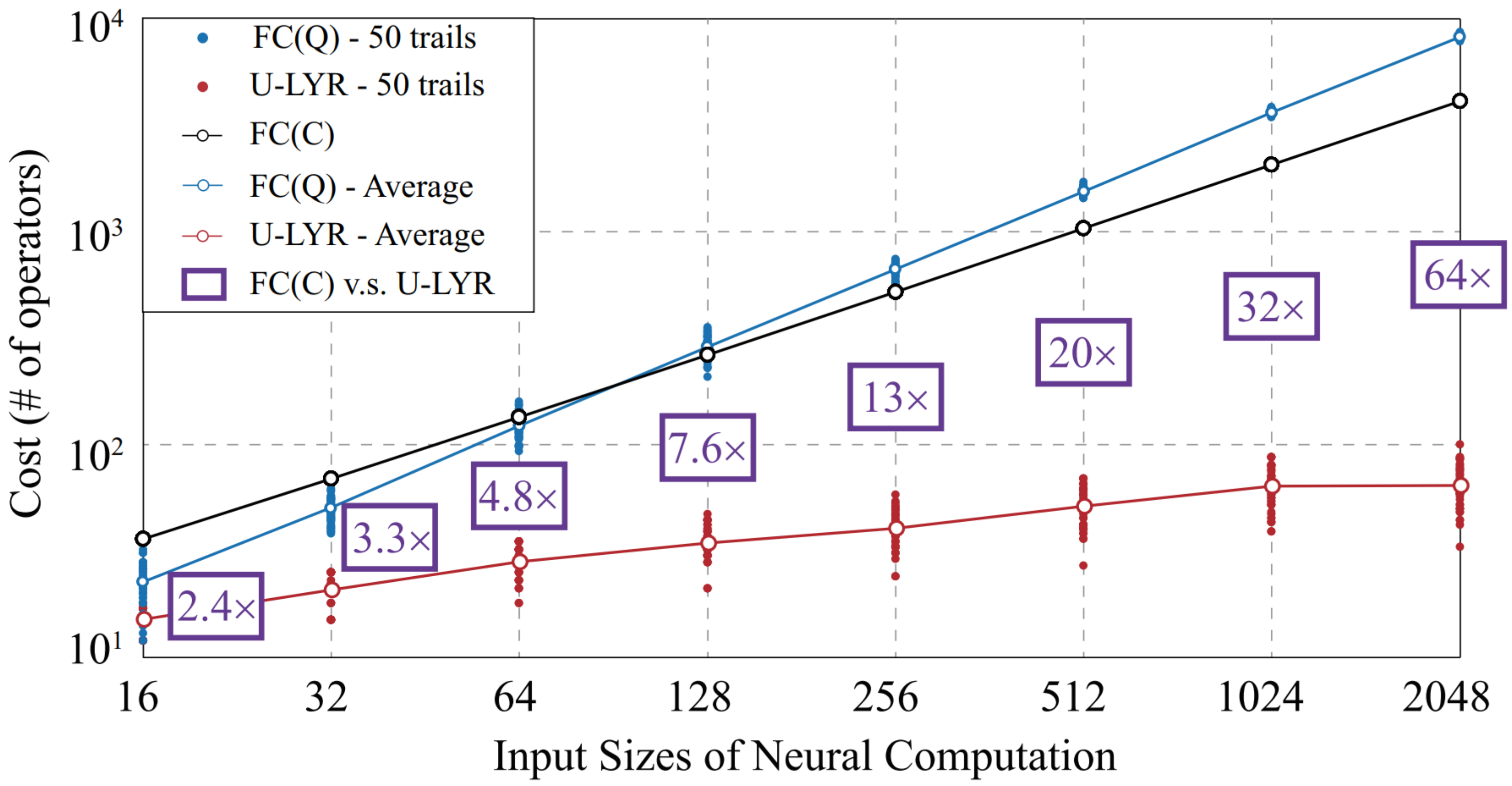
Gates	Cost
Z	1
CZ	1
C^2Z	3
C^3Z	5
C^4Z	6
…	…
C^kZ	2k-1

Worst case: all gates

$O(k^2)$

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%.

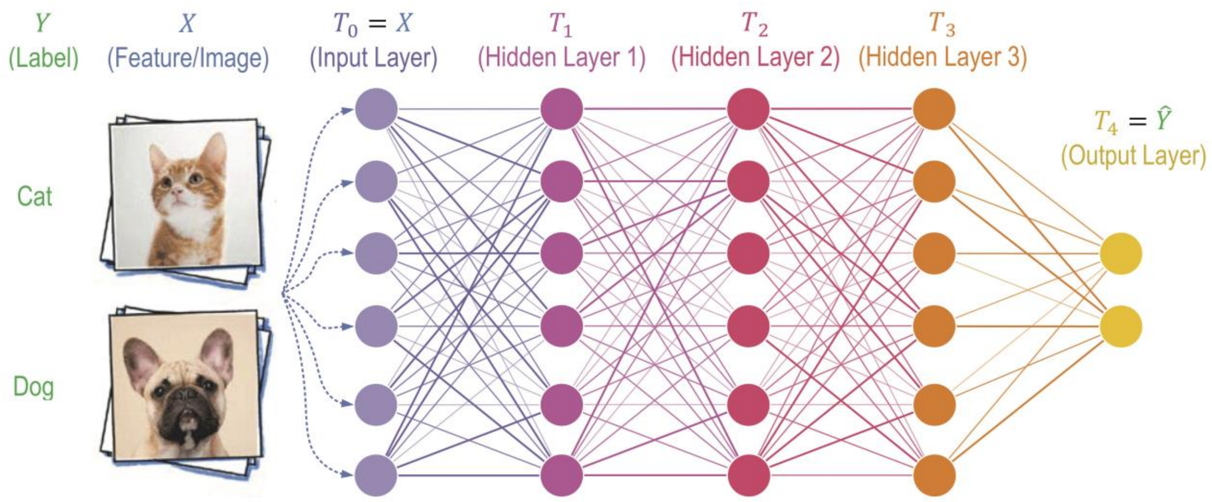
[ref of FFNN] Tacchino, F., et al., 2019. Quantum implementation of an artificial feed-forward neural network. *arXiv preprint arXiv:1912.12486*.

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*.

Key Takeaways



CPU/GPU Accelerator

FPGA/ASIC Accelerator

Quantum Accelerator

- What is the Quantum Friendly Neural Network?

  QF-Nets

- How to automatically map NN to QC?

  Co-Design

- Can we achieve quantum advantage by implementing NN on QC?

  Yes, we can!

Thank You!

wjiang2@nd.edu