qmlsys.mit.edu

Noise Robust Quantum Circuits Search and Training with QuantumNAS and QuantumNAT

Hanrui Wang
MIT HAN Lab

[HPCA'22] QuantumNAS: Noise-adaptive search for robust quantum circuits
[DAC'22] QuantumNAT: Quantum Noise-Aware Training with Noise Injection, Quantization and Normalization
[DAC'22] QOC: Quantum On-Chip Training with Parameter Shift and Gradient Pruning

Torch Quantum

2022/04/29 6.812/6.825 Recitation

MIT HAN LAB

Outline

- Background
- QuantumNAS
- QuantumNAT
- TorchQuantum Library
- Conclusion

Quantum Bit

- Quantum Bit (Qubit)
  - Statevector: contains 2^n complex numbers for n qubit system
  - The square sum of magnitude of 2^n numbers are 1
  - 1 qubit:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \quad a_0, a_1 \in \mathbb{C}$$
$$|a_0|^2 + |a_1|^2 = 1$$

  - 2 qubits:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad a_0, a_1, a_2, a_3 \in \mathbb{C}$$
$$|a_0|^2 + |a_1|^2 + |a_2|^2 + |a_3|^2 = 1$$

Quantum Bit

- Classical bits represented in statevector
  - Classical 0: $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
  - Classical 1: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  - An arbitrary quantum states: $\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = a_0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Quantum Gates

- Qubit gates: operations on one qubit or multiple qubits
- The qubit gates can be represented with matrix format with dimension $2^n \times 2^n$
- All gate matrices are unitary matrices: the conjugate transpose is the same as its inverse
- Single qubit gates:
  - Not (X) gate:

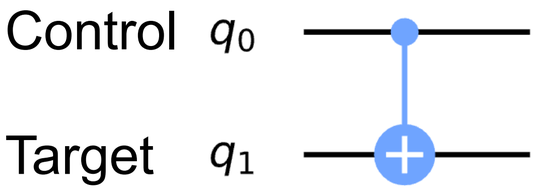
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

  - Parameterized gate: Rotation X (RX) with parameter theta

$$RX(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

Quantum Gates

- 2-qubit gates:
  - Controlled Not (CNOT) gate:



$$CNOT = \begin{array}{c} \\ 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} \text{Output} & & & \\ 00 & 01 & 10 & 11 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Input

  - Controlled Rotation X (CRX) gate

$$CRX(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ 0 & 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

Quantum Gates

- Applying a gate to qubits is performing matrix-vector multiplication between the gate matrix and statevector
  - Apply an X gate to classical state 0, we get 1

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

  - Apply an CNOT gate to state 10, we get 11

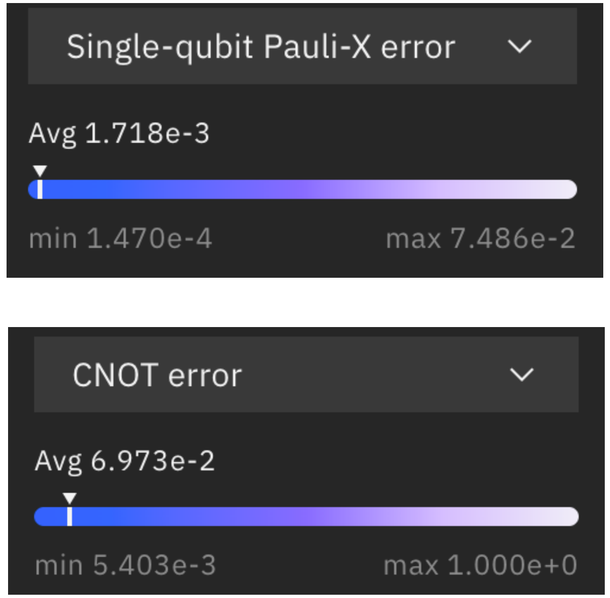
$$CNOT \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Source of Quantum Advantage

- One qubit carries more information then one classic bit
- The statevector length is exponentially to the number of qubits

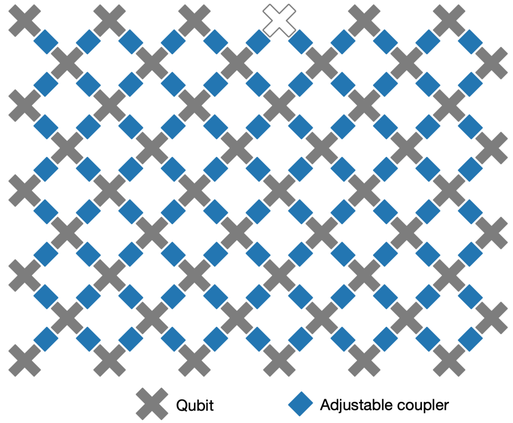
NISQ Era

- Noisy Intermediate-Scale Quantum (NISQ)
  - **Noisy**: qubits are sensitive to environment; quantum gates are unreliable
  - **Limited number** of qubits: tens to hundreds of qubits



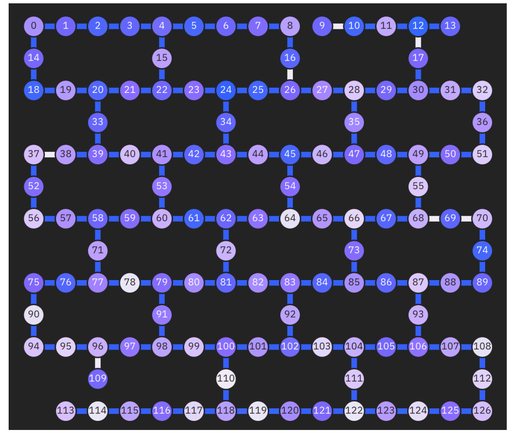
Gate Error Rate

https://quantum-computing.ibm.com/



Google Sycamore

https://www.nature.com/articles/s41586-019-1666-5

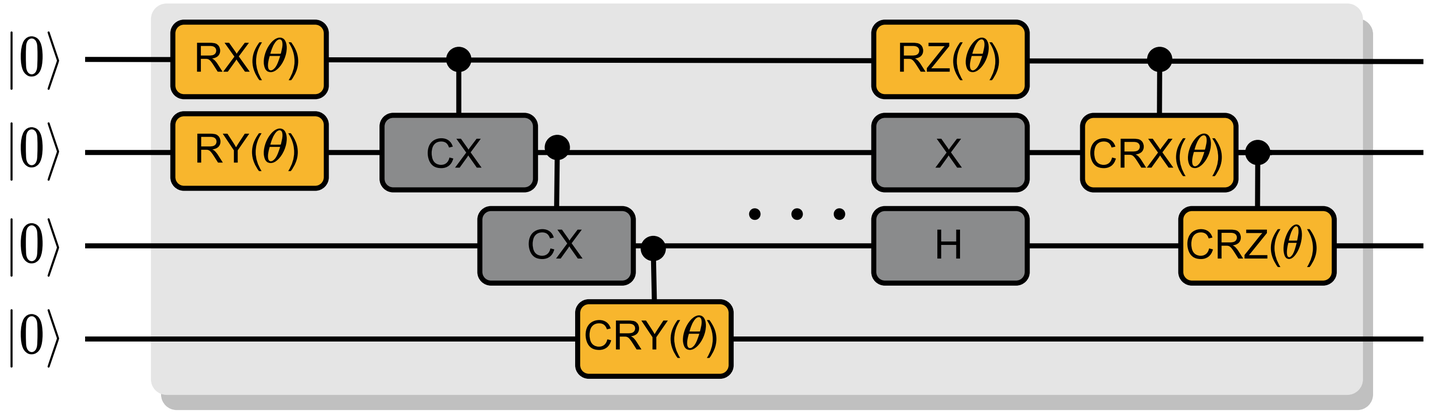


IBM Washington

https://quantum-computing.ibm.com/

Parameterized Quantum Circuits

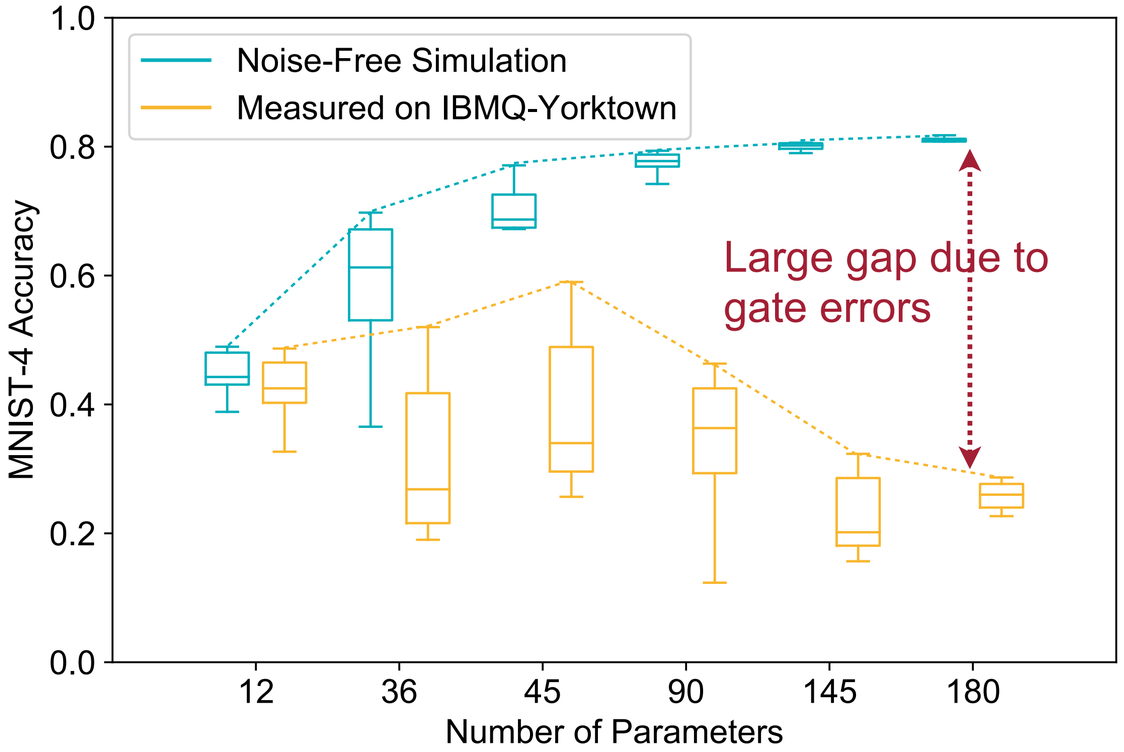
- Parameterized Quantum Circuits (PQC)
- Quantum circuit with fixed gates and parameterized gates



- PQCs are commonly used in **hybrid classical-quantum models** and show promises to achieve quantum advantage
  - Variational Quantum Eigensolver (VQE)
  - Quantum Neural Networks (QNN)
  - Quantum Approximate Optimization Algorithm (QAOA)

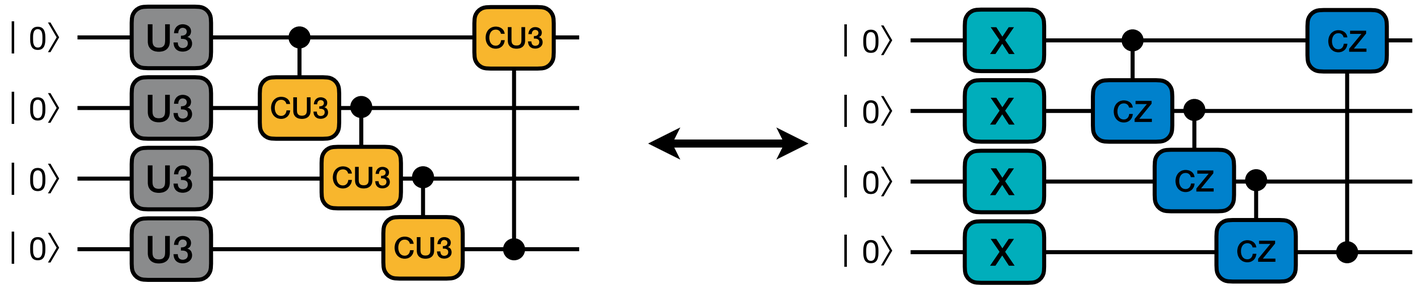
Challenges of PQC — Noise

- Noise **degrades** the Parameterized Quantum Circuit (PQC) reliability
- More parameters increase the noise-free accuracy but degrade the measured accuracy
- Under same #parameters, measured accuracy of different circuit architecture (ansatz) varies a lot
- Therefore, circuit architecture is critical



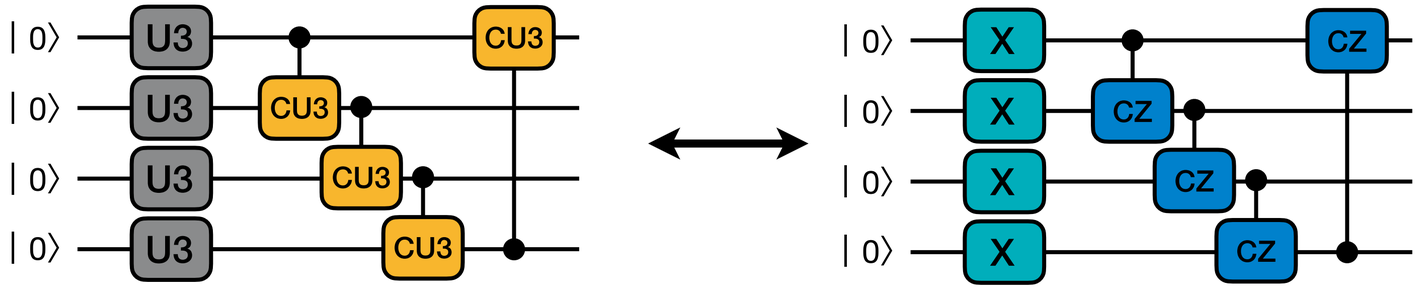
Challenges of PQC — Large Design Space

- Large design space for circuit architecture
  - Type of gates

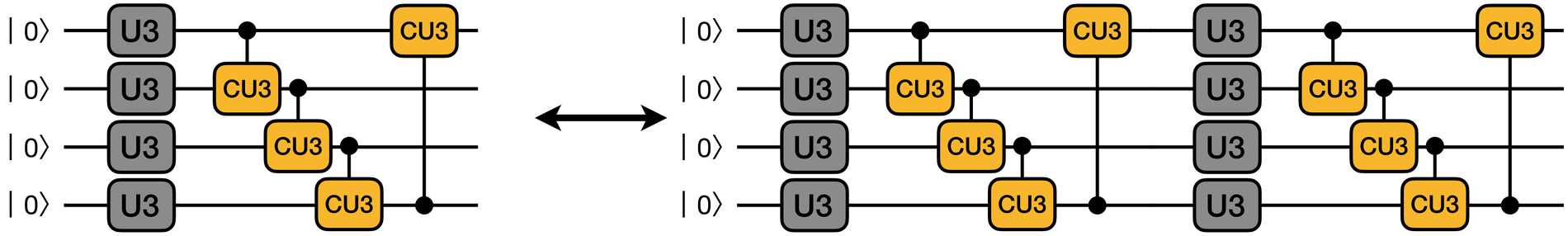


Challenges of PQC — Large Design Space

- Large design space for circuit architecture
  - Type of gates

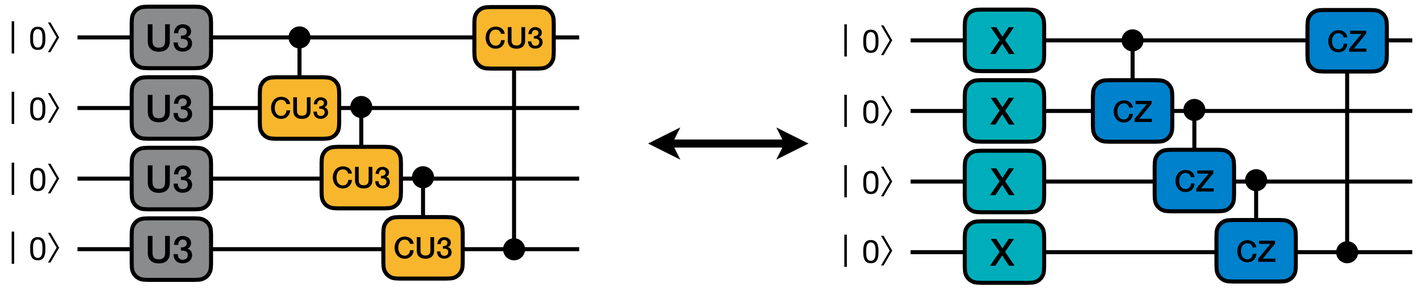


  - Number of gates

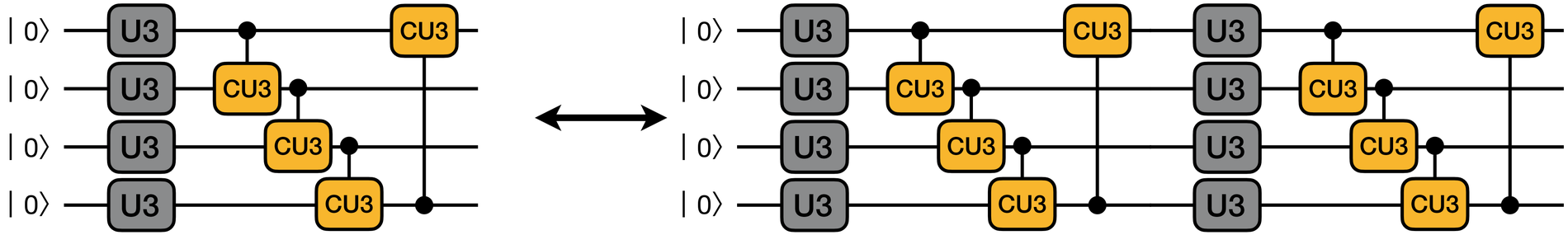


Challenges of PQC — Large Design Space

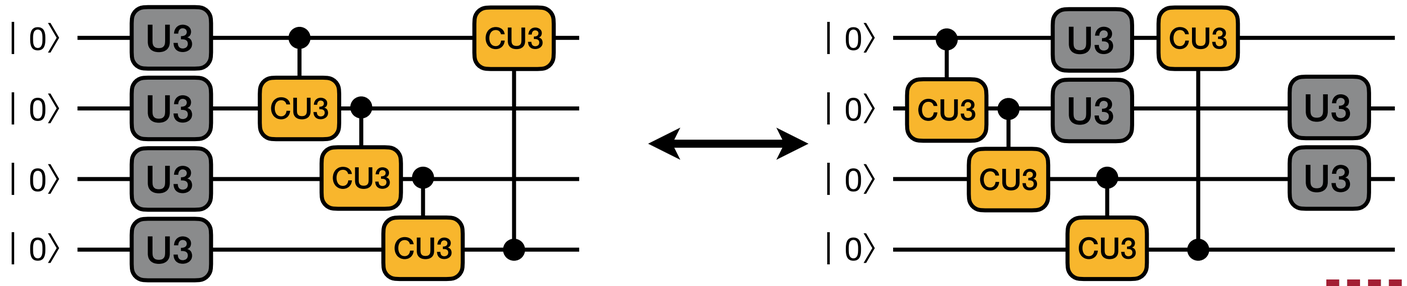
- Large design space for circuit architecture
  - Type of gates



  - Number of gates



  - Position of gates



Goal of QuantumNAS

Automatically & efficiently search for noise-robust quantum circuit

Train one “SuperCircuit”, providing parameters to many “SubCircuits”

Solve the challenge of large design space

(1) Quantum noise feedback in the search loop
(2) Co-search the circuit architecture and qubit mapping

Solve the challenge of large quantum noise

QuantumNAS: Decouple the Training and Search

Naive Search

For q_devices:
  For search episodes: // meta controller
    For circuit training iterations:
      update_parameters(); **Expensive**
    If good_circuit: **break;**

QuantumNAS: Decouple the Training and Search

Naive Search

For q_devices:
  For search episodes: // meta controller
    For circuit training iterations:
      update_parameters(); **Expensive**
    If good_circuit: **break;**

=>

QuantumNAS

For *SuperCircuit* training iterations: **Expensive**
  update_parameters(); **training**

decouple

For q_devices:
  For search episodes:
    sample from ***SuperCircuit***; **Light-Weight**
    If good_circuit: **break;**
    //no training

search

QuantumNAS

- SuperCircuit Construction and Training
- Noise-Adaptive Evolutionary Co-Search of SubCircuit and Qubit Mapping
- Train the Searched SubCircuit
- Iterative Quantum Gate Pruning

QuantumNAS

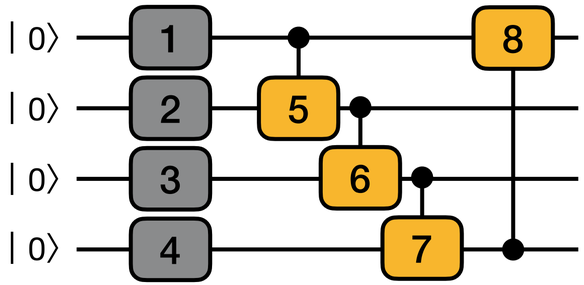
- SuperCircuit Construction and Training
- Noise-Adaptive Evolutionary Co-Search of SubCircuit and Qubit Mapping
- Train the Searched SubCircuit
- Iterative Quantum Gate Pruning

SuperCircuit & SubCircuit

- Firstly construct a design space. For example, a design space of maximum 4 U3 in the first layer and 4 CU3 gates in the second layer
  - Contains 2^8 = 256 candidates

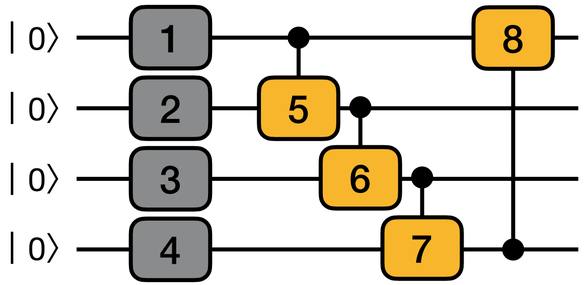
SuperCircuit & SubCircuit

- Firstly construct a design space. For example, a design space of maximum 4 U3 in the first layer and 4 CU3 gates in the second layer
  - Contains 2^8 = 256 candidates
- SuperCircuit: the circuit with the **largest** number of gates in the design space
  - Example: SuperCircuit in U3+CU3 space

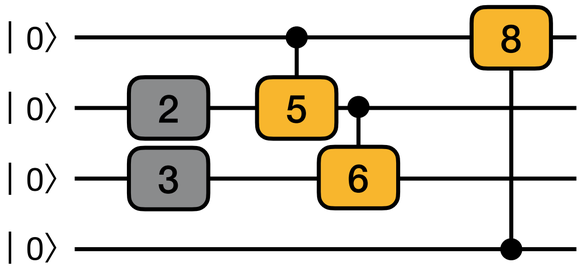


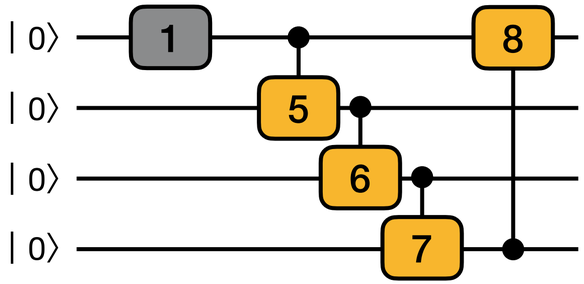
SuperCircuit & SubCircuit

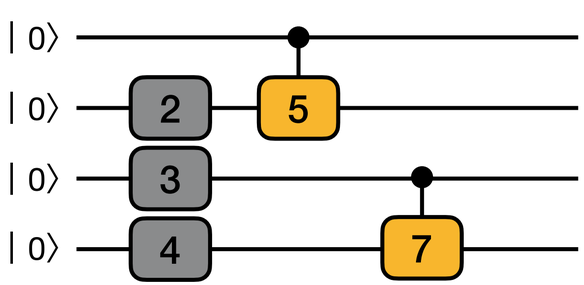
- Firstly construct a design space. For example, a design space of maximum 4 U3 in the first layer and 4 CU3 gates in the second layer
  - Contains 2^8 = 256 candidates
- SuperCircuit: the circuit with the **largest** number of gates in the design space
  - Example: SuperCircuit in U3+CU3 space



- Each candidate circuit in the design space (called SubCircuit) is a **subset** of the SuperCircuit







SuperCircuit Construction

- Why use a SuperCircuit?
  - It enables **efficient** search of circuit architecture candidates with no need of training each of them individually
  - For one SubCircuit candidate, we can directly inherit parameters from SuperCircuit and consider that the SubCircuit can operate **as if it is trained individually from scratch**

SuperCircuit Construction

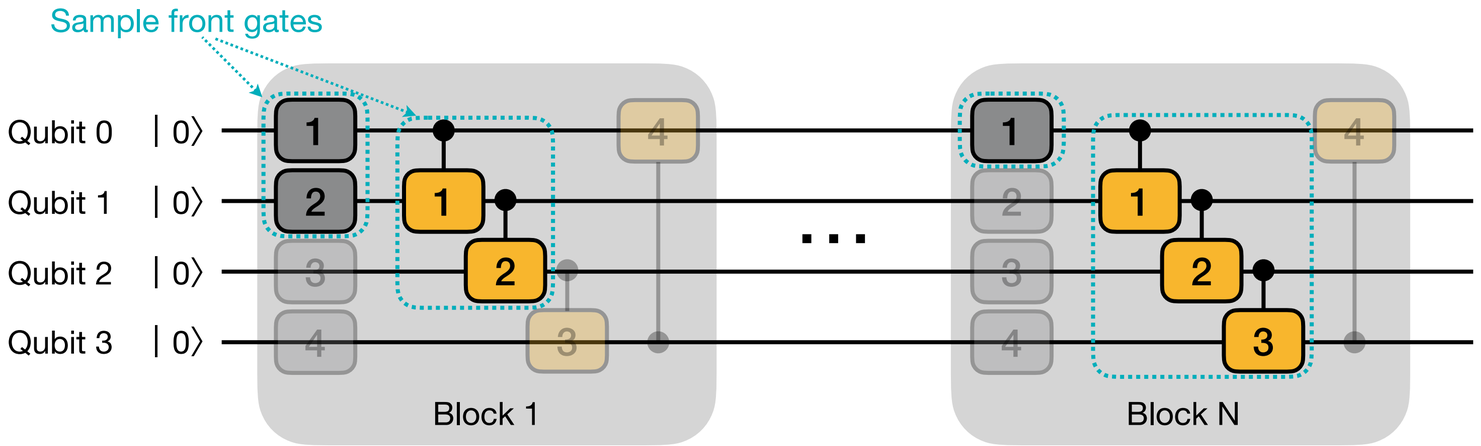
- Why use a SuperCircuit?
  - It enables **efficient** search of circuit architecture candidates with no need of training each of them individually
  - For one SubCircuit candidate, we can directly inherit parameters from SuperCircuit and consider that the SubCircuit can operate **as if it is trained individually from scratch**
- Need to prevent interference of SubCircuits from each other

SuperCircuit Training

- In one SuperCircuit Training step:
  - Sample a gate subset of SuperCircuit (a SubCircuit)
    - Front Sampling and Restricted Sampling
  - Only use the subset to perform the task and updates the parameters in the subset
  - Parameter updates are cumulative across steps

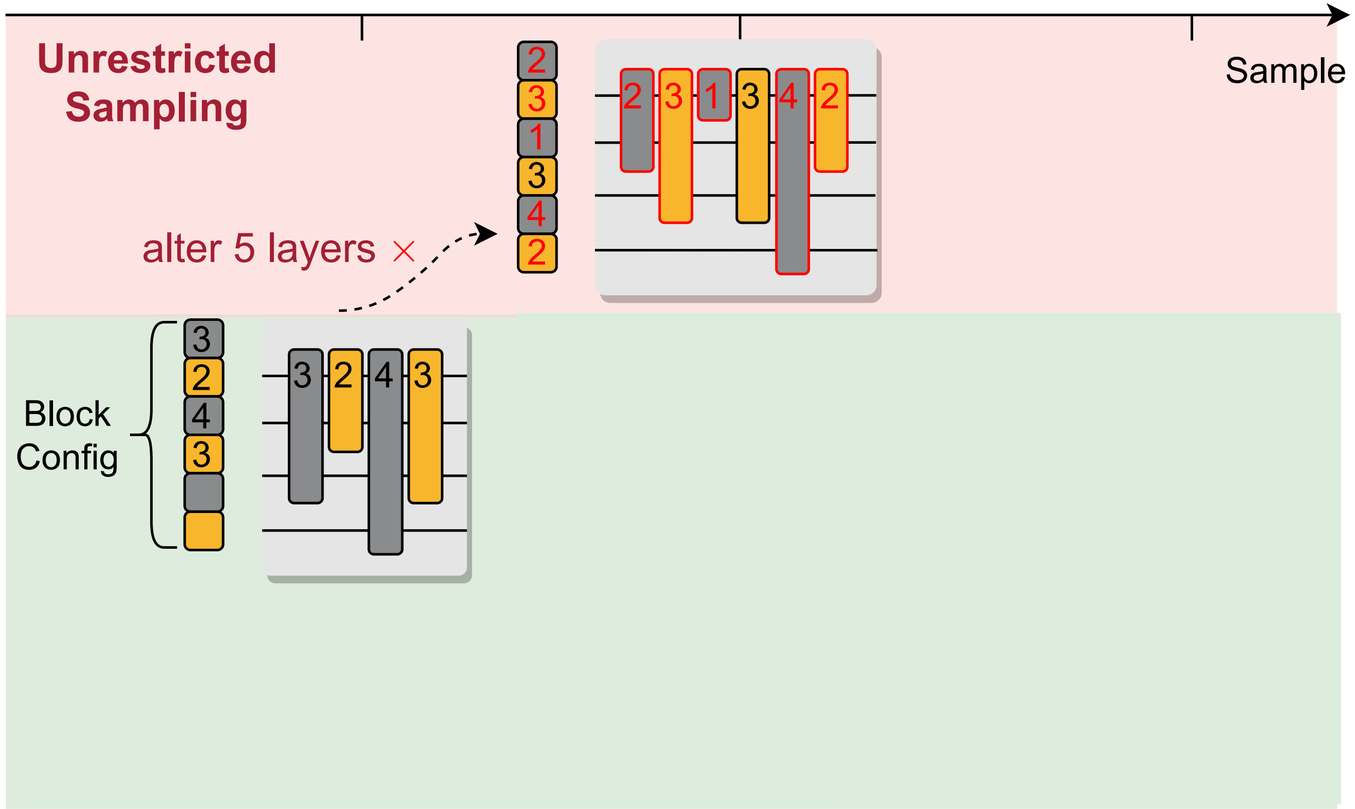
Front Sampling

- During sampling, we first sample total number of blocks, then sample gates within each block
  - Front sampling: Only the **front** several blocks and **front** several gates can be sampled to make SuperCircuit training more stable



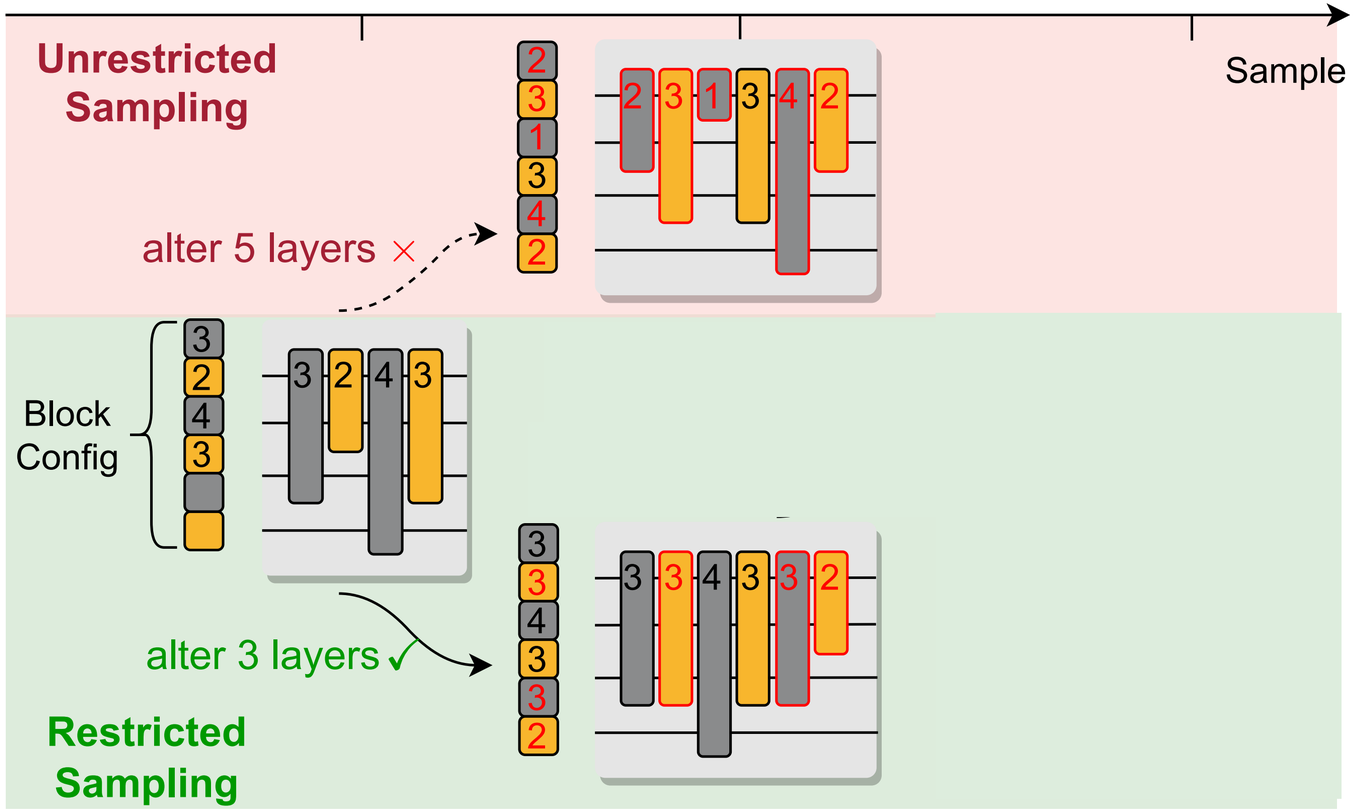
Restricted Sampling

- Restricted Sampling:
  - Restrict the difference between SubCircuits of two consecutive steps
  - For example: restrict to at most 4 different layers



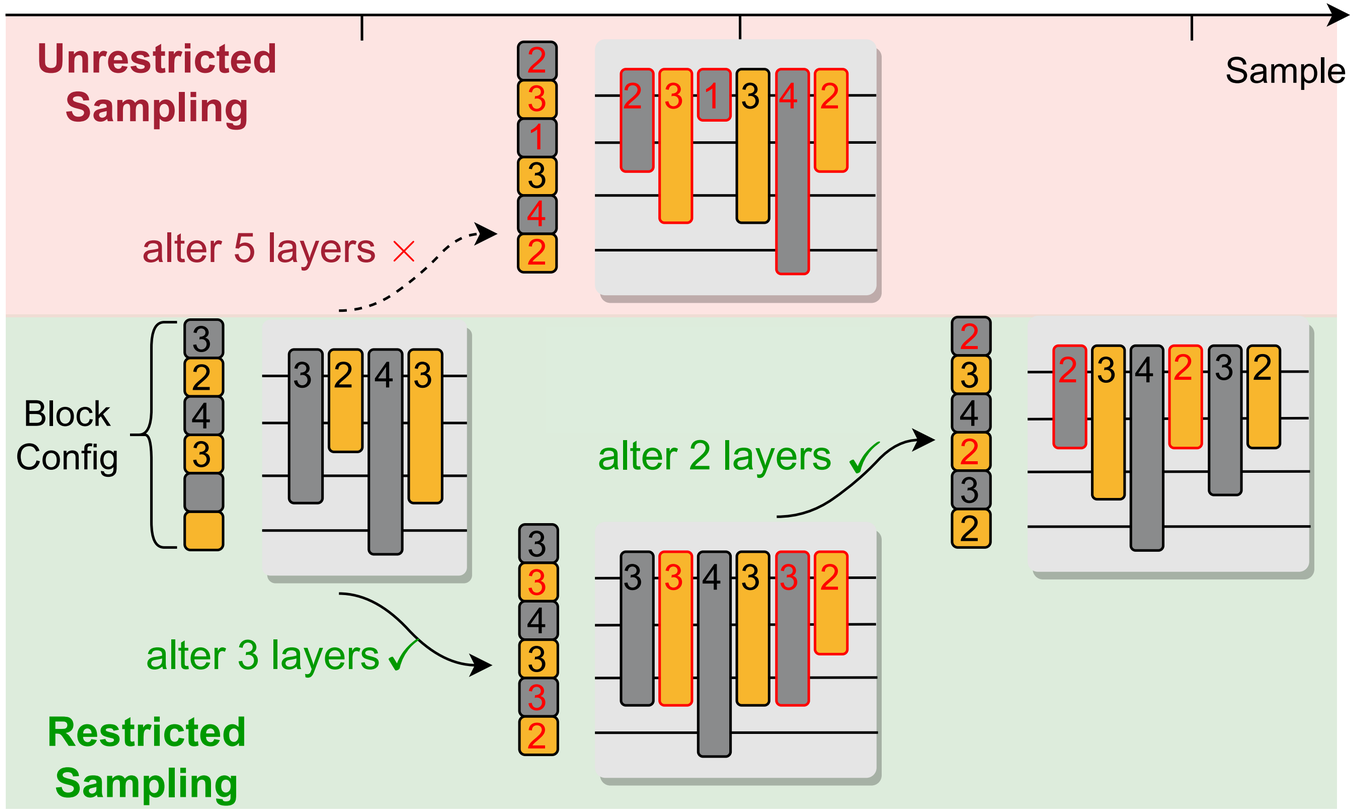
Restricted Sampling

- Restricted Sampling:
  - Restrict the difference between SubCircuits of two consecutive steps
  - For example: restrict to at most 4 different layers



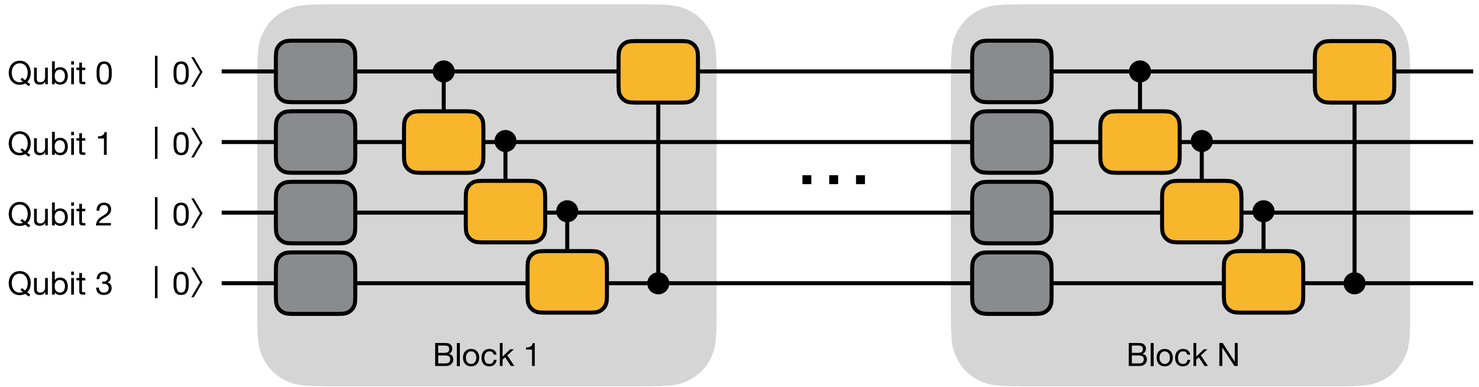
Restricted Sampling

- Restricted Sampling:
  - Restrict the difference between SubCircuits of two consecutive steps
  - For example: restrict to at most 4 different layers



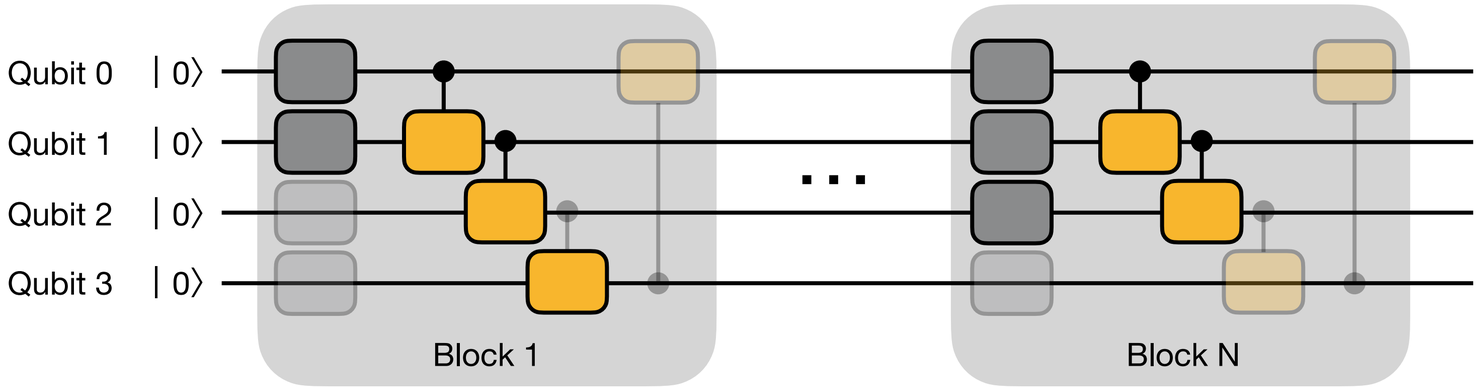
Train SuperCircuit for Multiple Steps

- In one SuperCircuit Training step: Sample and Train



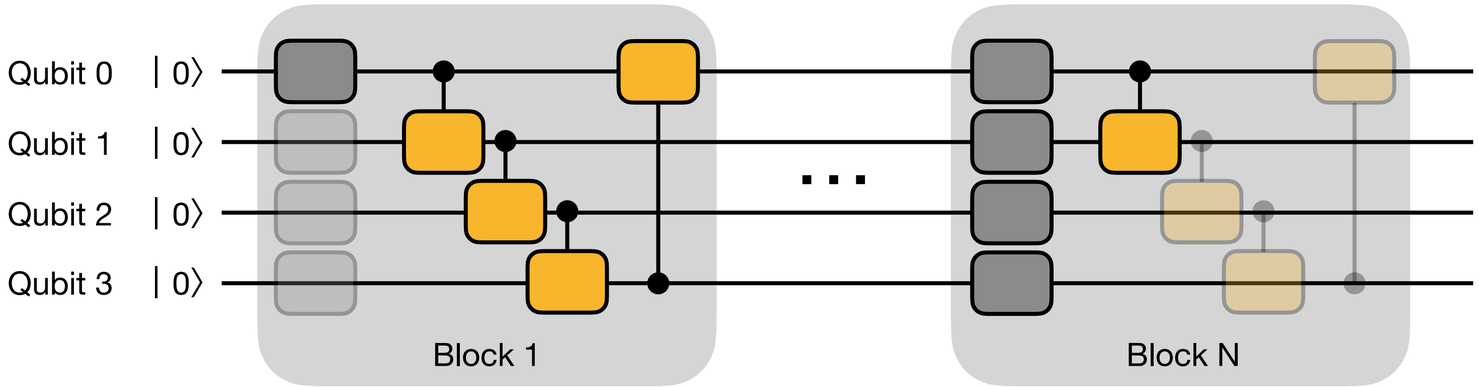
Train SuperCircuit for Multiple Steps

- In one SuperCircuit Training step: Sample and Train

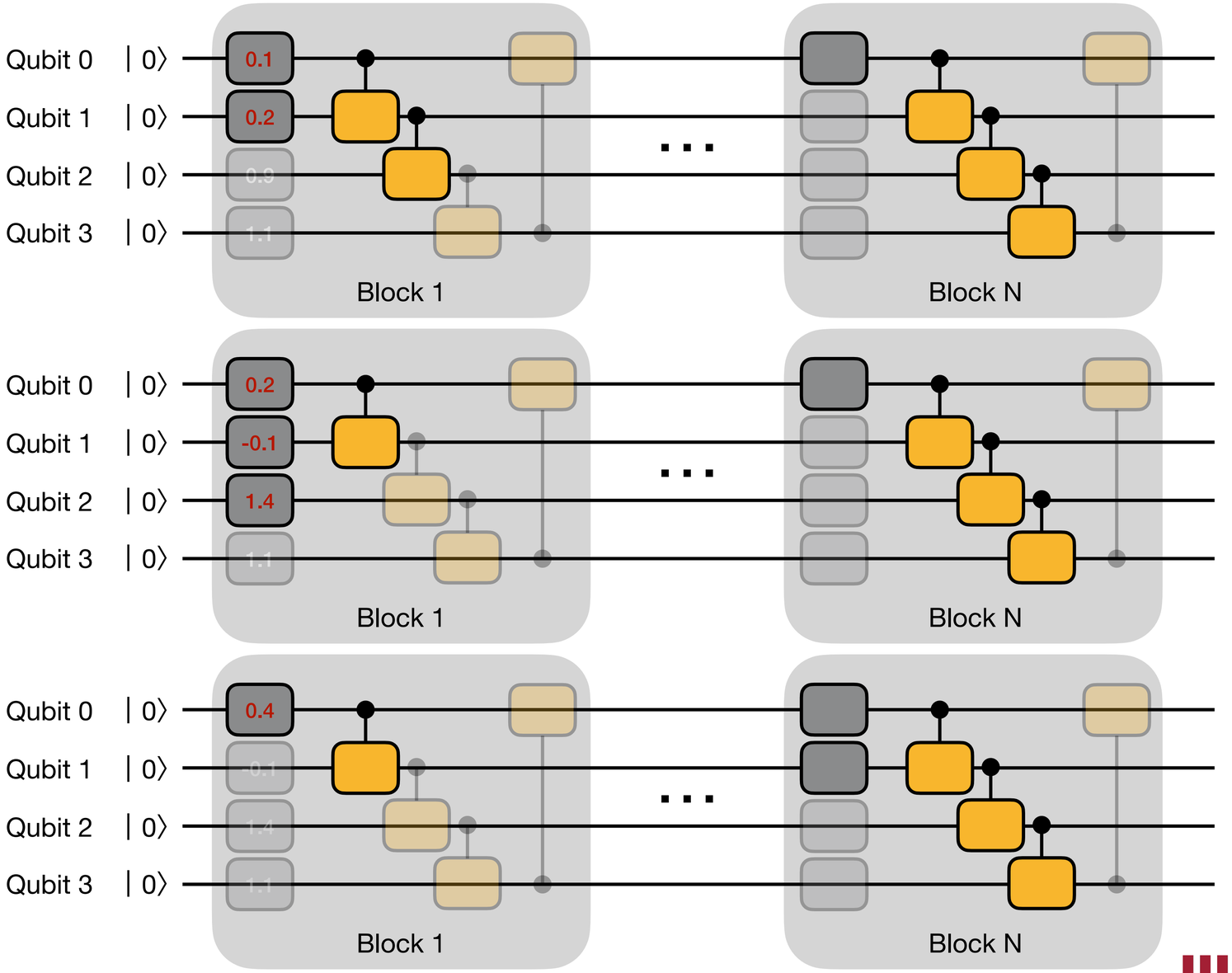


Train SuperCircuit for Multiple Steps

- In one SuperCircuit Training step: Sample and Train

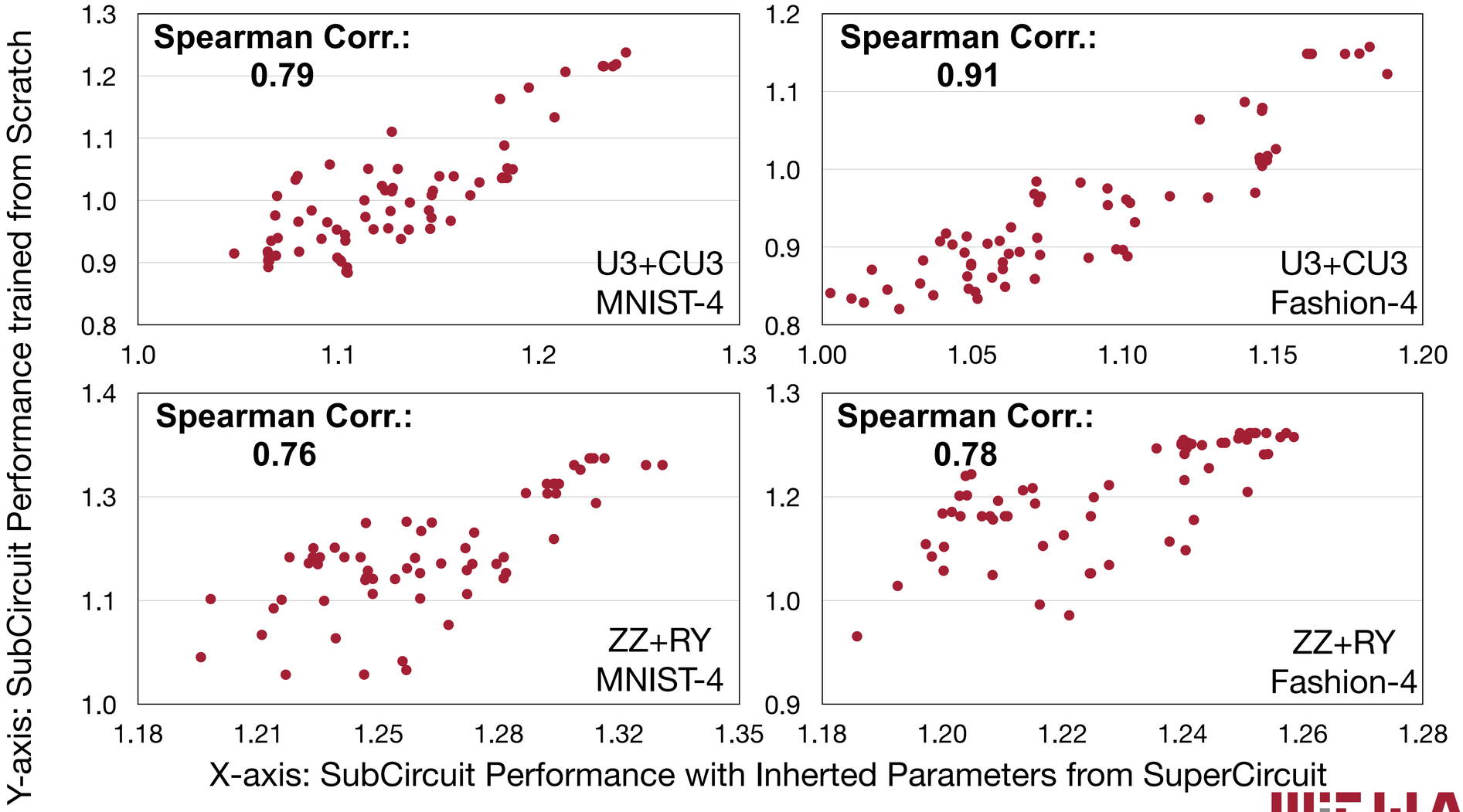


Train SuperCircuit for Multiple Steps



How Reliable is the SuperCircuit?

- Inherited parameters from SuperCircuit can provide accurate relative performance

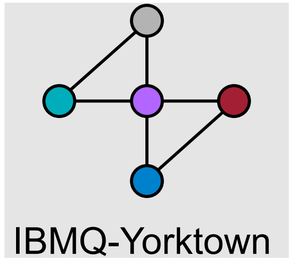


QuantumNAS

- SuperCircuit Construction and Training
- Noise-Adaptive Evolutionary Co-Search of SubCircuit and Qubit Mapping
- Train the Searched SubCircuit
- Iterative Quantum Gate Pruning

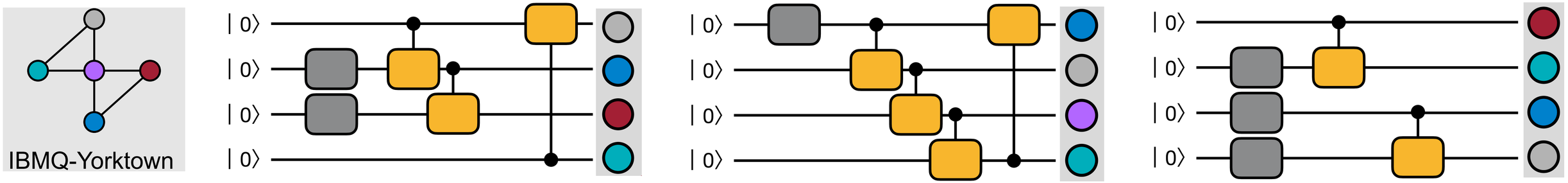
Noise-Adaptive Evolutionary Co-Search

- Search the best SubCircuit and its qubit mapping on target device



Noise-Adaptive Evolutionary Co-Search

- Search the best SubCircuit and its qubit mapping on target device



Evolutionary Search Engine
Mutation
Crossover

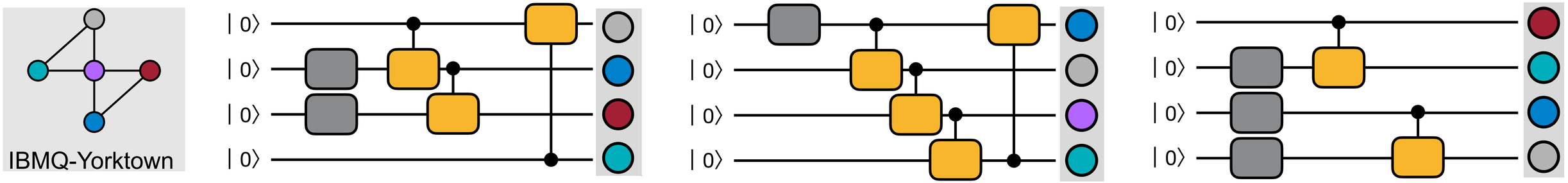
SubCircuit and Qubit Mapping pairs

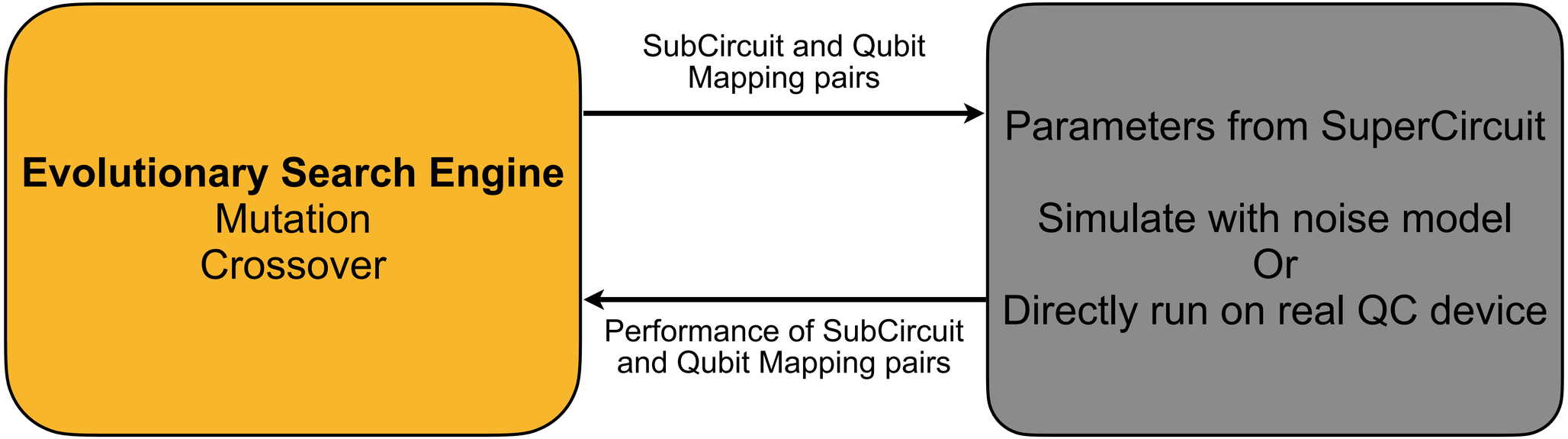
Parameters from SuperCircuit

Simulate with noise model
Or
Directly run on real QC device

Noise-Adaptive Evolutionary Co-Search

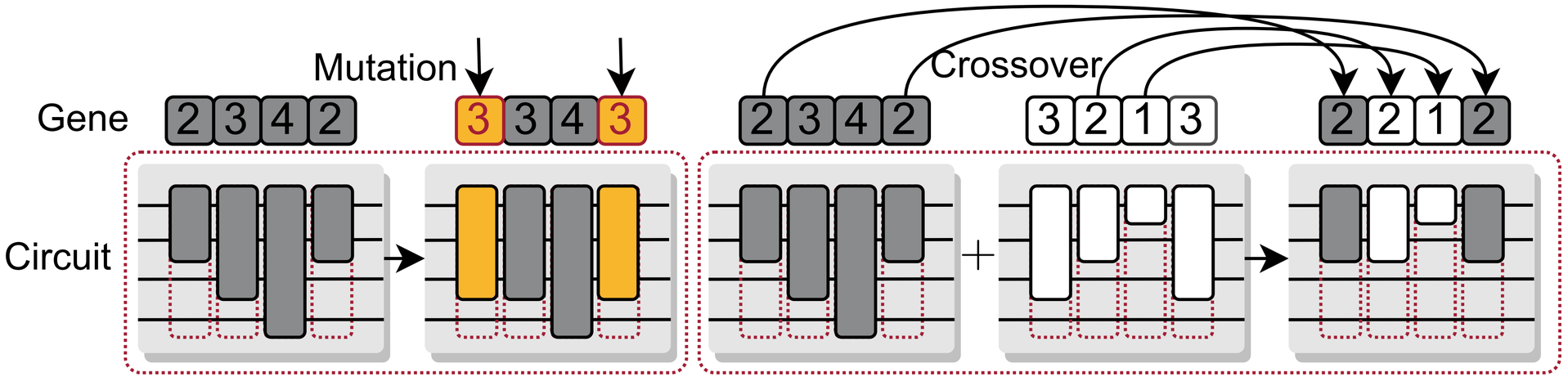
- Search the best SubCircuit and its qubit mapping on target device





Mutation and Crossover

- Mutation and crossover create new SubCircuit candidates



QuantumNAS

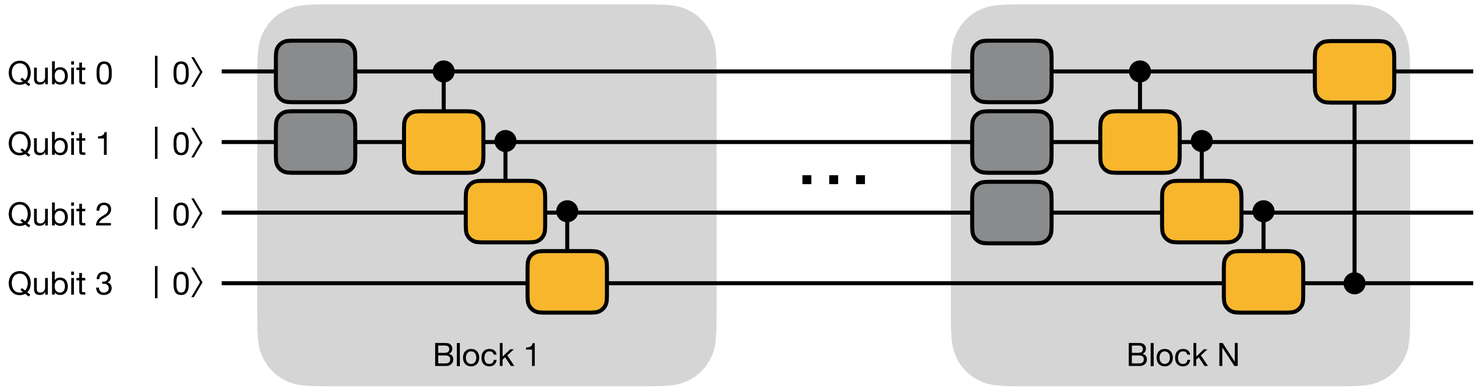
- SuperCircuit Construction and Training
- Noise-Adaptive Evolutionary Co-Search of SubCircuit and Qubit Mapping
- Train the Searched SubCircuit
- Iterative Quantum Gate Pruning

QuantumNAS

- SuperCircuit Construction and Training
- Noise-Adaptive Evolutionary Co-Search of SubCircuit and Qubit Mapping
- Train the Searched SubCircuit
- Iterative Quantum Gate Pruning

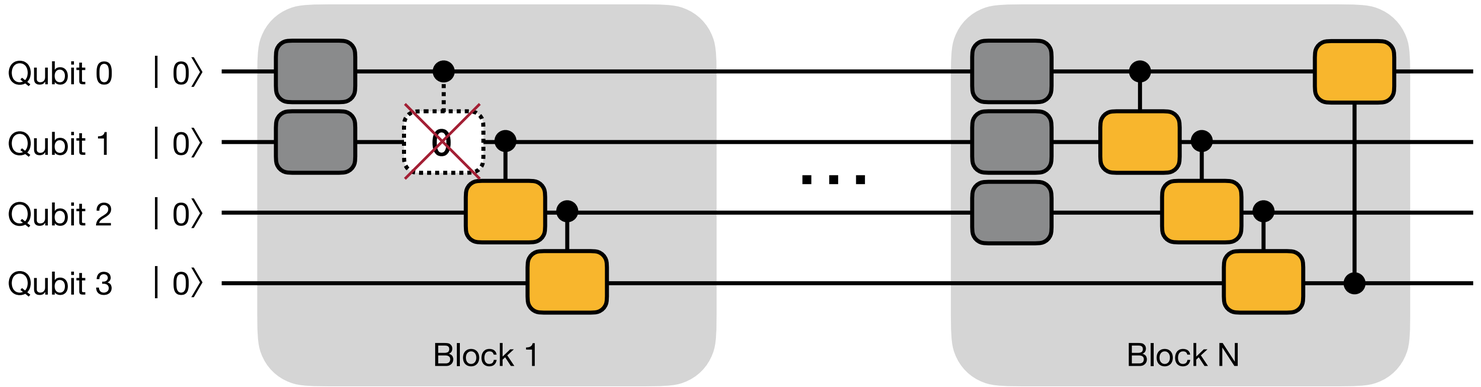
Iterative Pruning

- Some gates have parameters close to 0
  - Rotation gate with angle close to 0 has small impact on the results
- Iteratively prune small-magnitude gates and fine-tune the remaining parameters



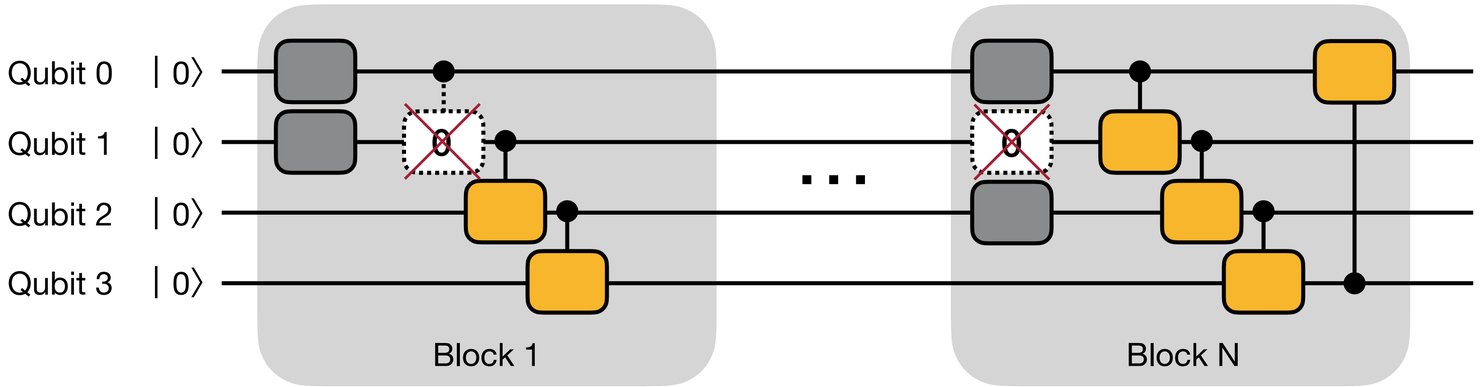
Iterative Pruning

- Some gates have parameters close to 0
  - Rotation gate with angle close to 0 has small impact on the results
- Iteratively prune small-magnitude gates and fine-tune the remaining parameters



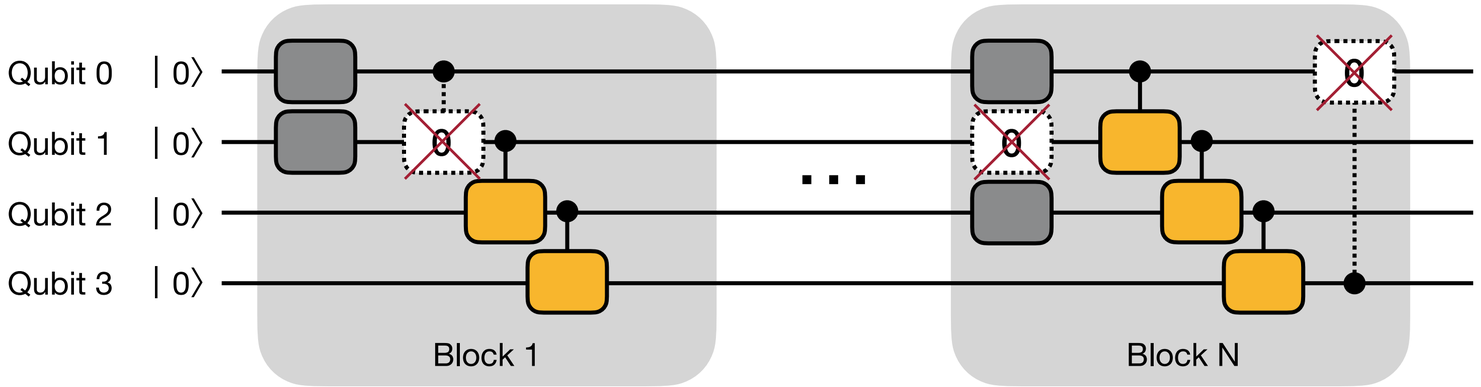
Iterative Pruning

- Some gates have parameters close to 0
  - Rotation gate with angle close to 0 has small impact on the results
- Iteratively prune small-magnitude gates and fine-tune the remaining parameters



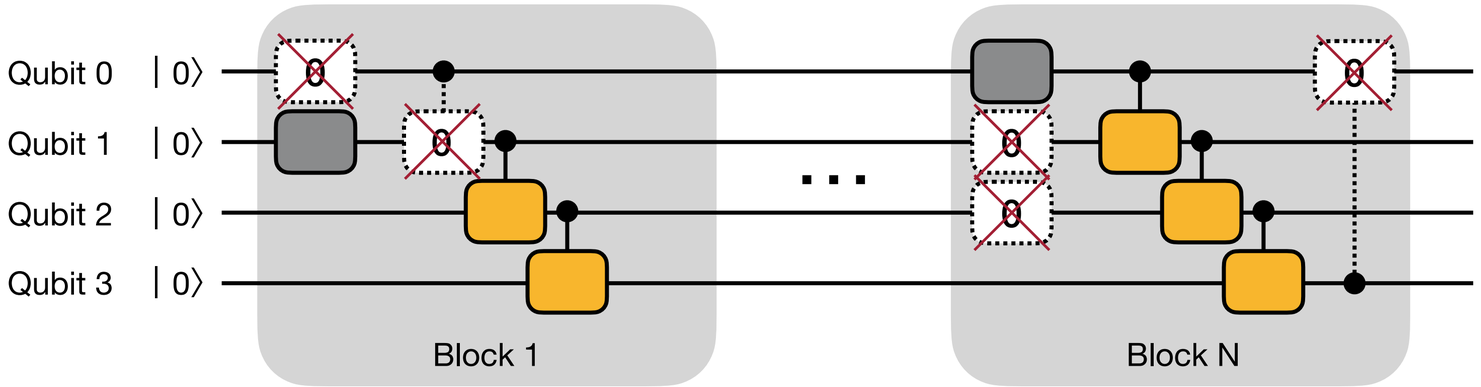
Iterative Pruning

- Some gates have parameters close to 0
  - Rotation gate with angle close to 0 has small impact on the results
- Iteratively prune small-magnitude gates and fine-tune the remaining parameters



Iterative Pruning

- Some gates have parameters close to 0
  - Rotation gate with angle close to 0 has small impact on the results
- Iteratively prune small-magnitude gates and fine-tune the remaining parameters

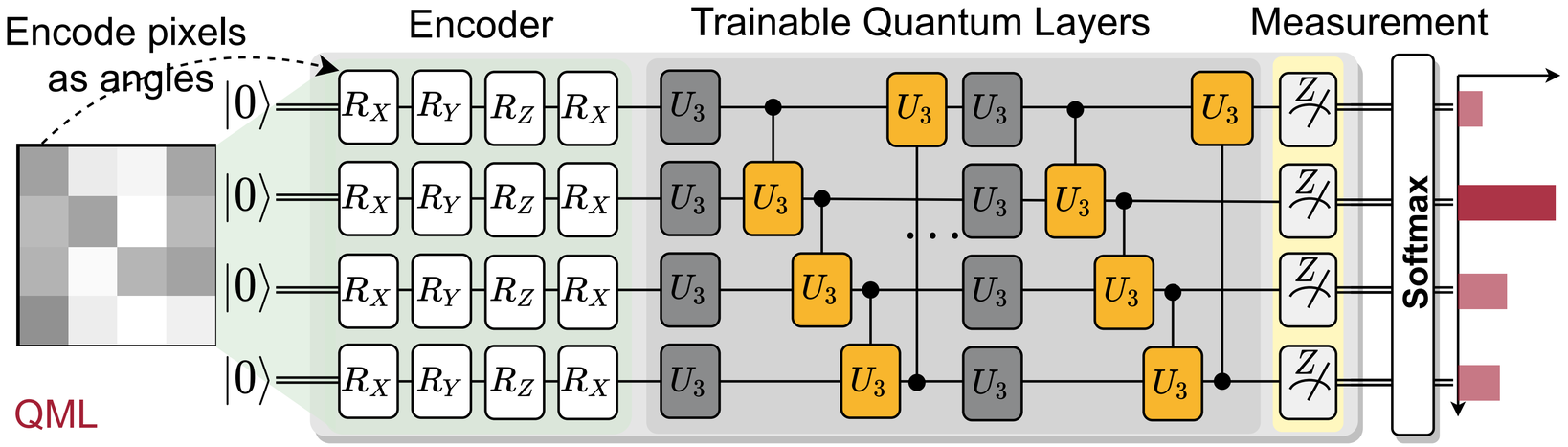


Evaluation Setups: Benchmarks and Devices

- Benchmarks
  - QML classification tasks: MNIST 10-class, 4-class, 2-class, Fashion 4-class, 2-class, Vowel 4-class
  - VQE task molecules: H2, H2O, LiH, CH4, BeH2
- Quantum Devices
  - IBMQ
  - #Qubits: 5 to 65
  - Quantum Volume: 8 to 128

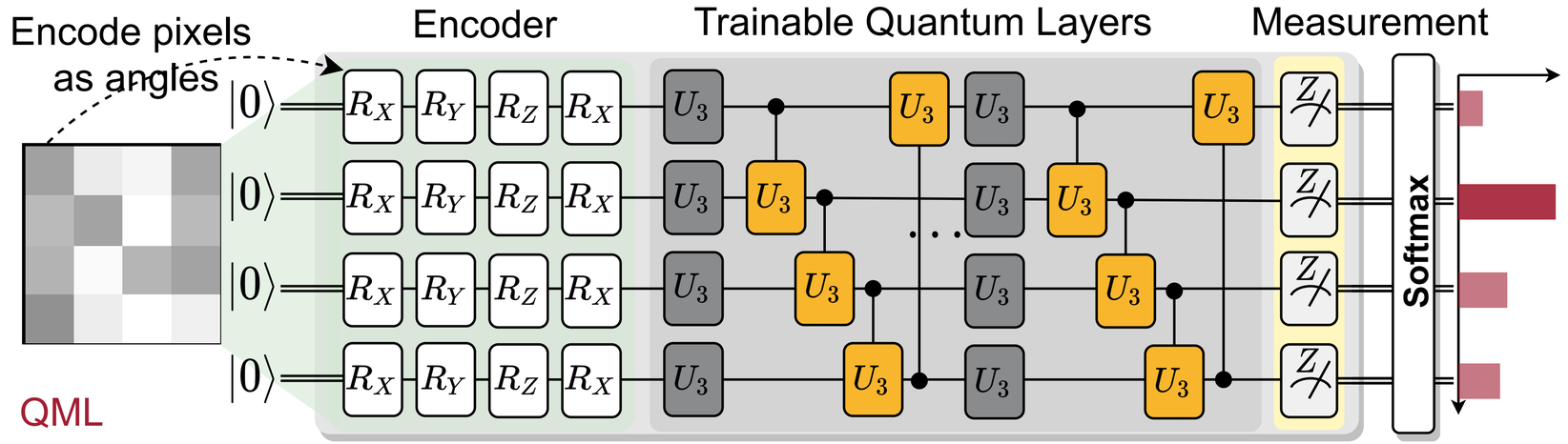
Benchmarks: QNN and VQE

- Quantum Neural Networks: classification

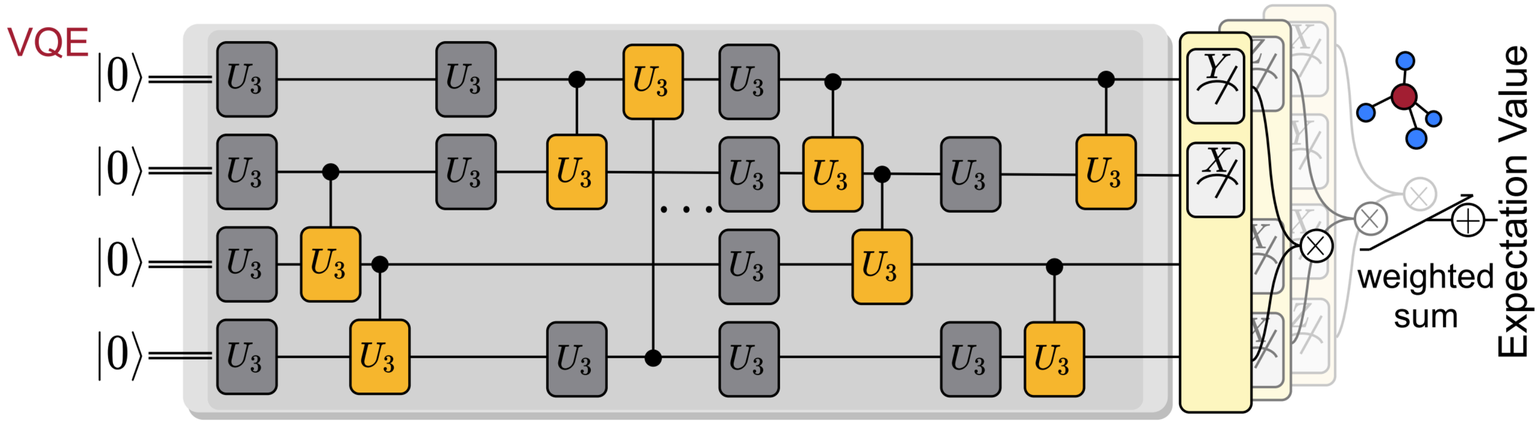


Benchmarks: QNN and VQE

- Quantum Neural Networks: classification

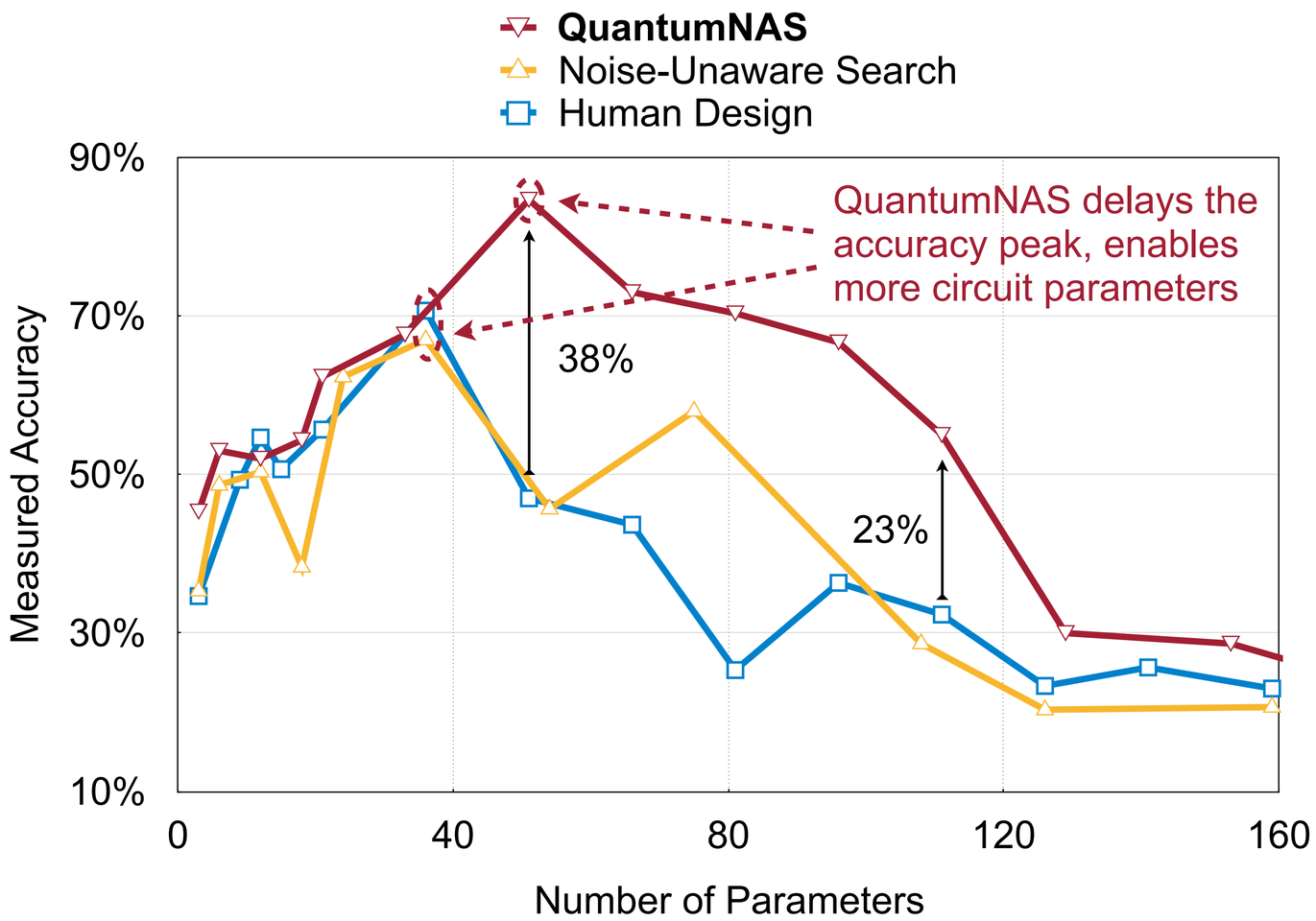


- Variational Quantum Eigensolver: finds the ground state energy of molecule Hamiltonian



QML Results

- 4-classification: MNIST-4 U3+CU3 on IBMQ-Yorktown



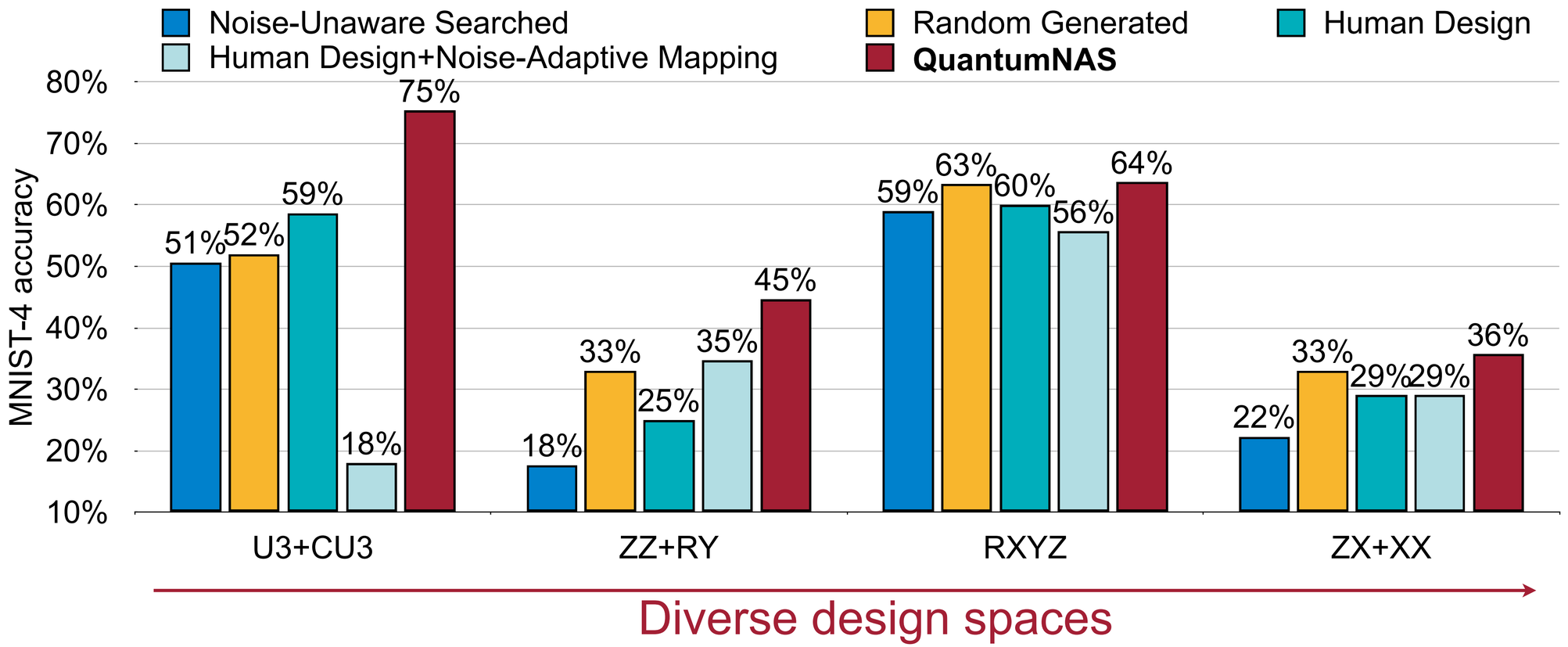
Analysis of Searched Circuit

- QuantumNAS searched circuit has fewer #gates and #params but higher accuracy

MNIST-2	Depth	#Gates	#Params	**QuantumNAS**
Human Design	64	135	36	**88%**
QuantumNAS	70	116	22	**92%**

Consistent Improvements on Diverse Design Spaces

- ‘U3+CU3’, ’ZZ+RY’, ‘RXYZ’, ‘ZX+XX’ spaces with different gates



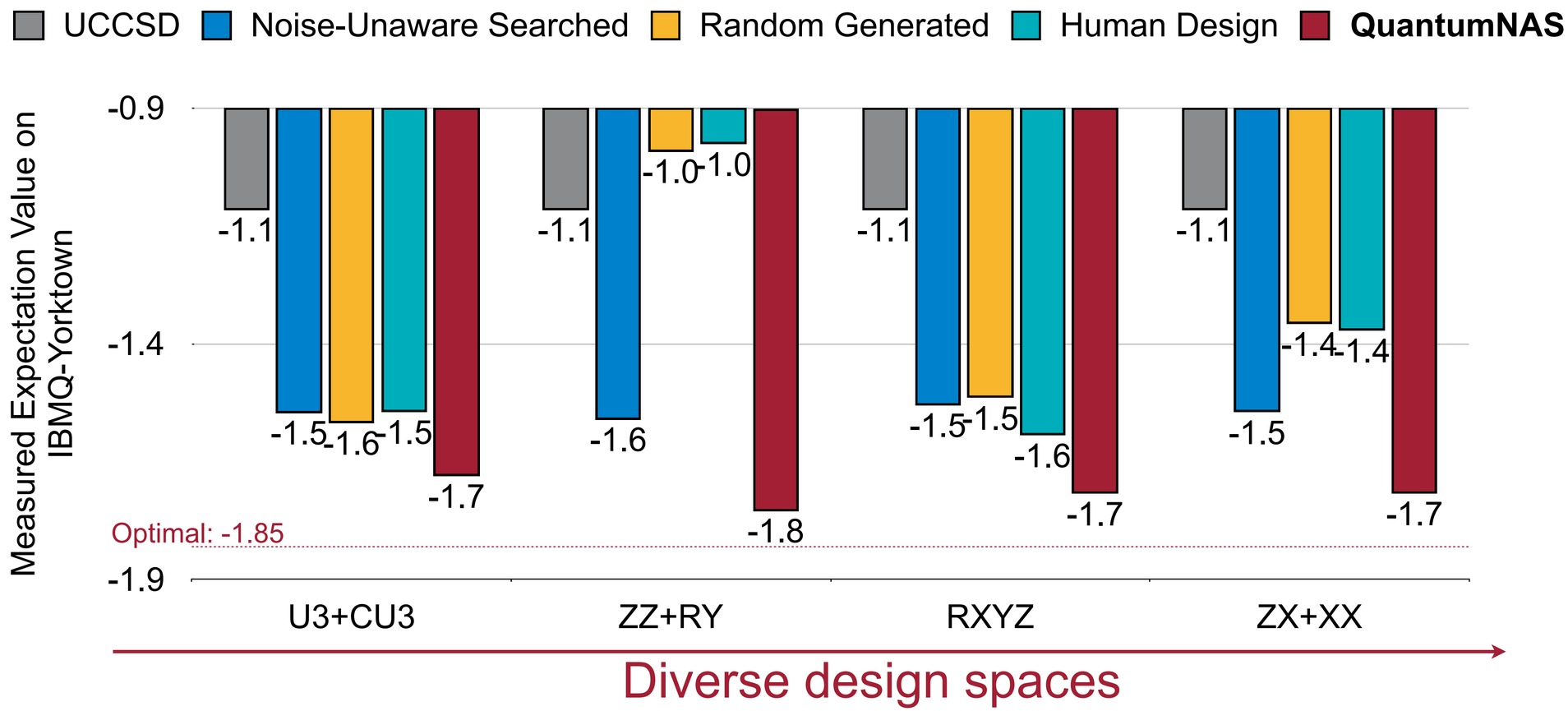
Torch Quantum

qmlsys.mit.edu

MIT HAN LAB

Consistent Improvements on Diverse Design Spaces

- H2 in different design spaces on IBMQ-Yorktown



Consistent Improvements on Diverse Devices

- QuantumNAS is effective for different real quantum devices
- On different 5-Qubit devices
- MNIST-4, Fashion-4, Vowel-4, MNIST-2, Fashion-2 averaged accuracy

Diverse devices

Method	Noise-Unaware Searched	Random	Human	**QuantumNAS**
Belem (5Q, 16QV)	47%	50%	67%	**76%**
Quito (5Q, 16QV)	73%	68%	74%	**79%**
Athens (5Q, 32QV)	50%	63%	68%	**77%**
Santiago (5Q, 32QV)	74%	73%	75%	**80%**

Scalable to Large #Qubits

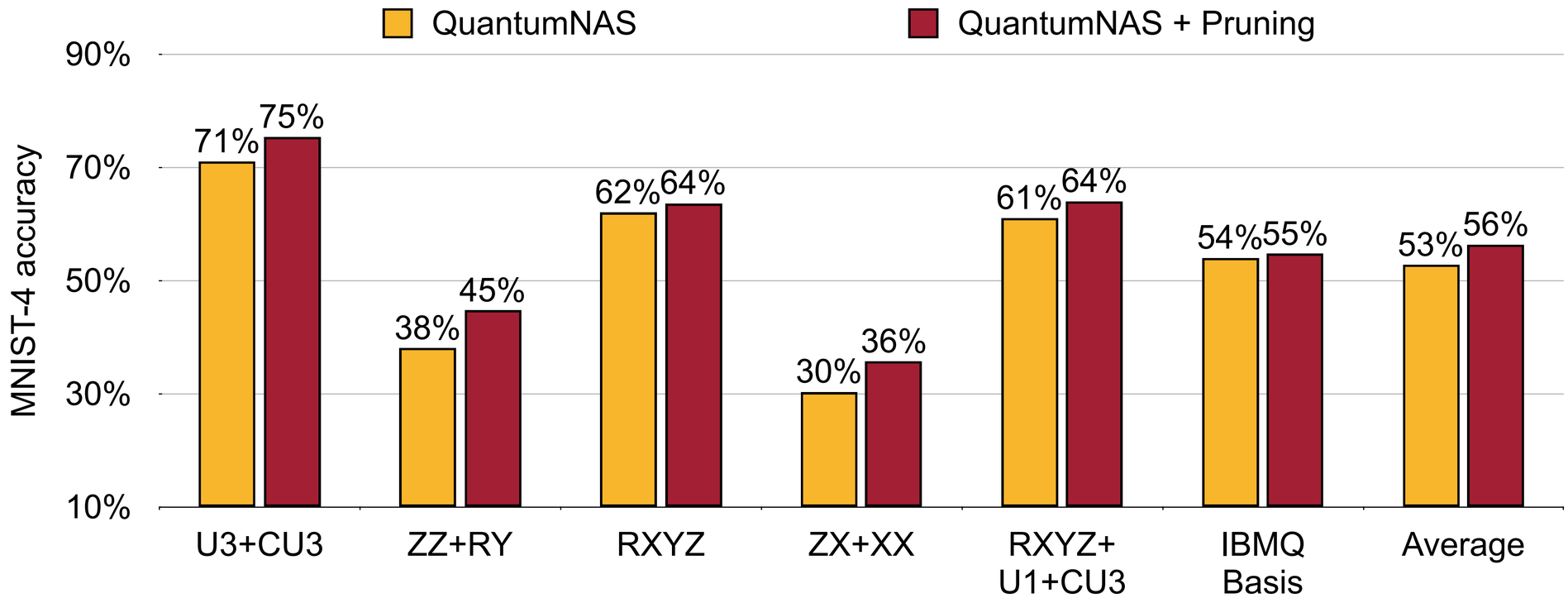
- On large devices
- MNIST-10 accuracy

More Qubits

Method	Noise-Unaware Searched	Random	Human	**QuantumNAS**
Melbourne (15Q, 8QV, use 15Q)	11%	10%	15%	**32%**
Guadalupe (16Q, 32QV, use 16Q)	14%	12%	10%	**15%**
Montreal (27Q, 128QV, use 21Q)	13%	7%	14%	**16%**
Manhattan (65Q, 32QV, use 21Q)	11%	11%	15%	**18%**

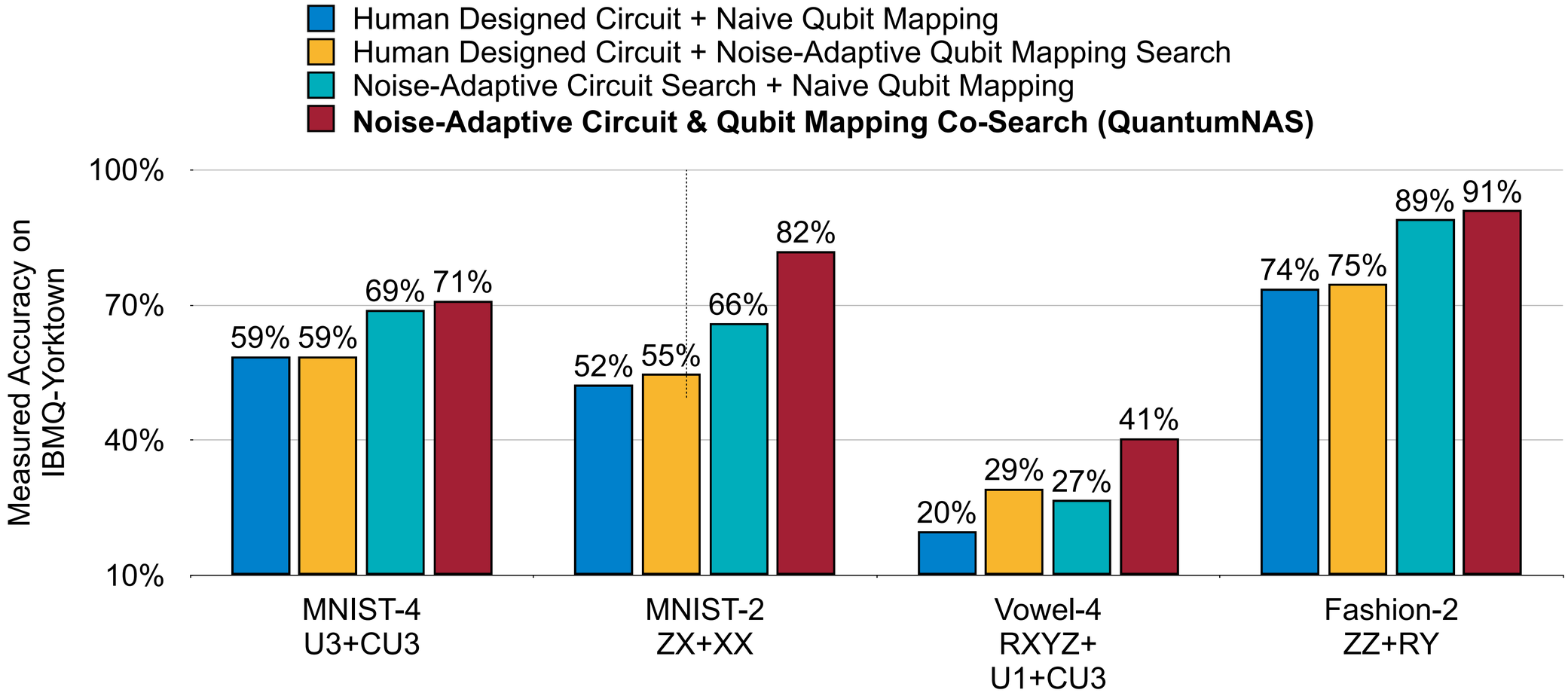
Effective of Quantum Gate Pruning

- For MNIST-4, Quantum gate pruning improves accuracy by 3% on average



Effect of Circuit & Qubit Mapping Co-Search

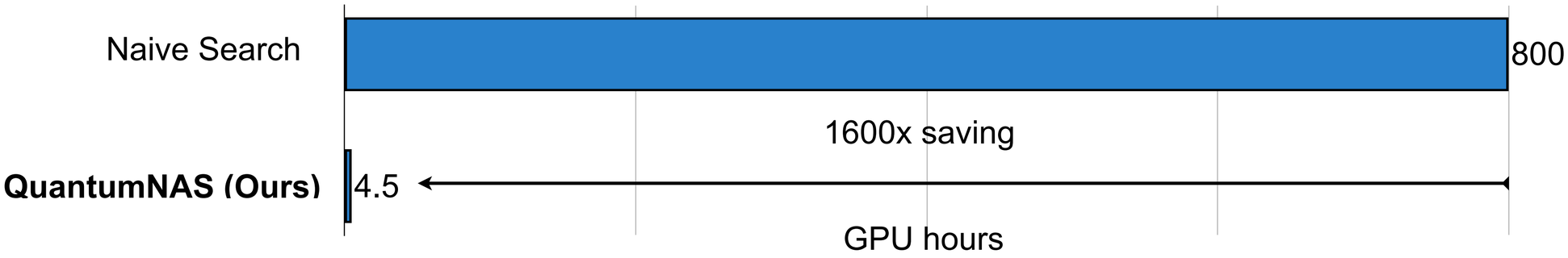
- Co-search is better than only search architecture/mapping



Time Cost

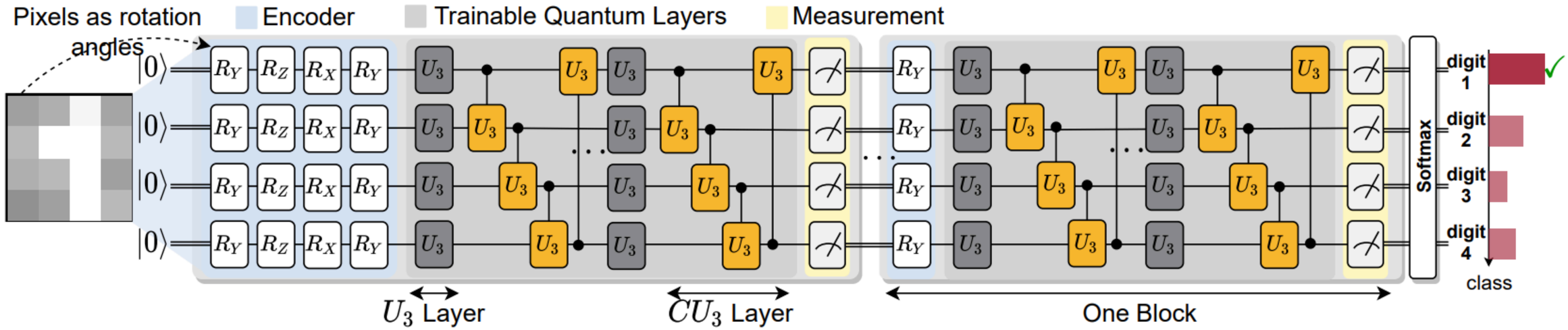
- On 1 Nvidia Titan RTX 2080 ti GPU

#qubits \ Step	SuperCircuit Training	Noise-Adaptive Co-search	SubCircuit Training	Deployment on Real QC
4 Qubits	0.5h	3h	0.5h	0.5h
15 Qubits	5h	5h	5h	1h
21 Qubits	20h	10h	15h	1h



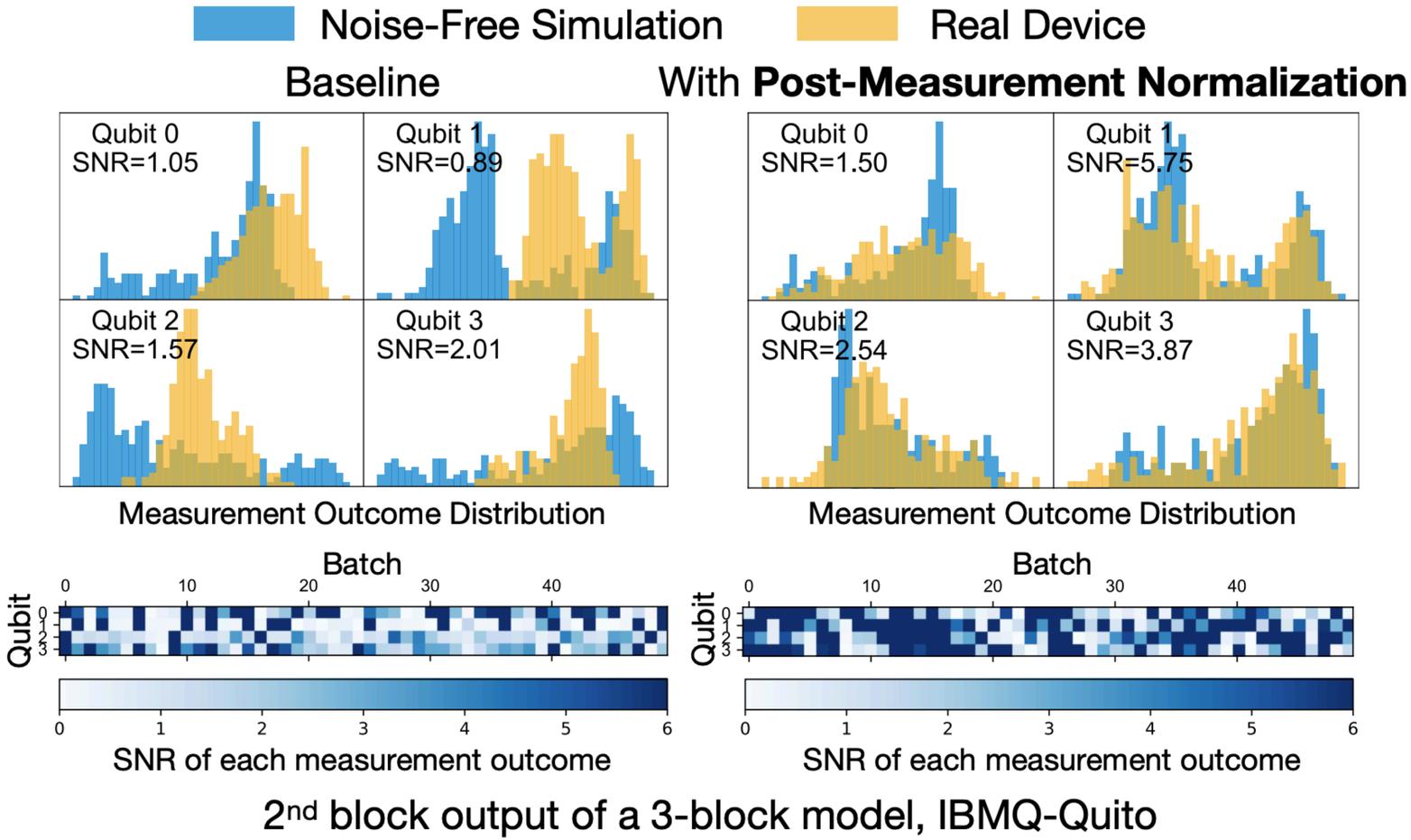
QuantumNAT

- QuantumNAS: find the **circuit architecture** robust to noise
- QuantumNAT: further make the **parameter** robust to noise



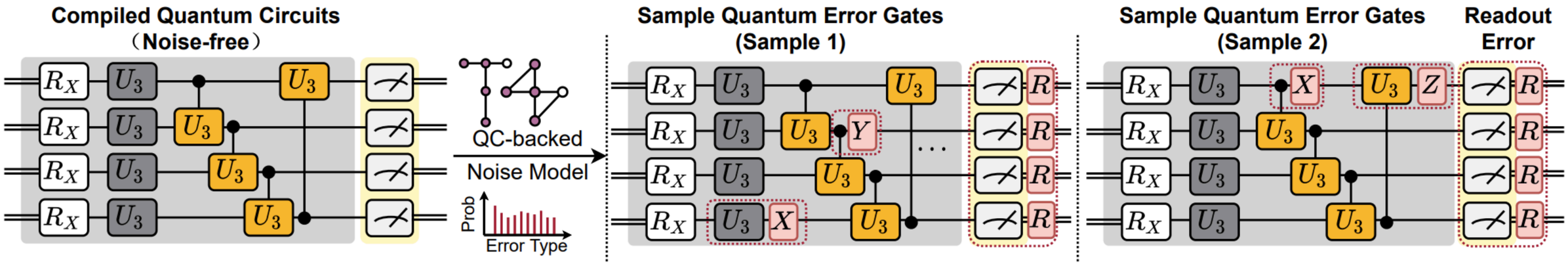
Post-Measurement Normalization

- Normalize the measurement outcomes on the batch dimension



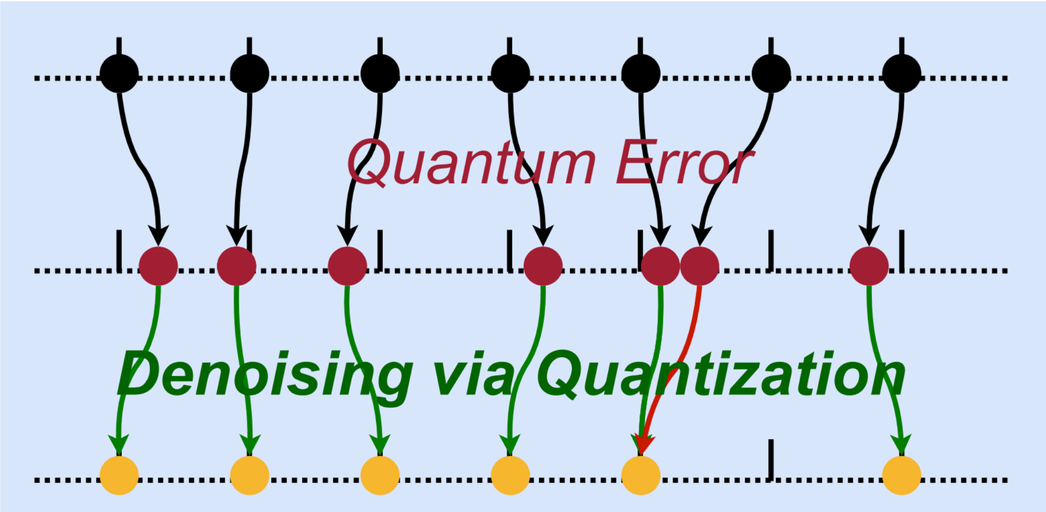
Noise Injection

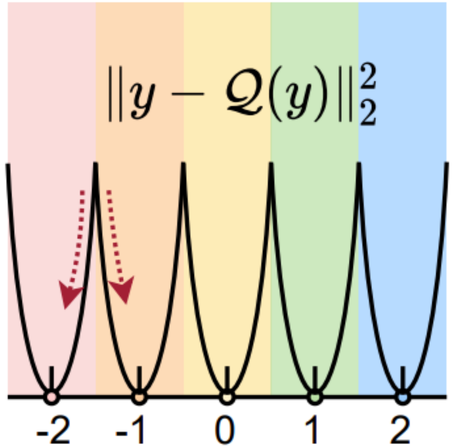
- Insert noise gate during training, according to the noise model
- For each step, sample new positions for noise gates



Post-Measurement Quantization

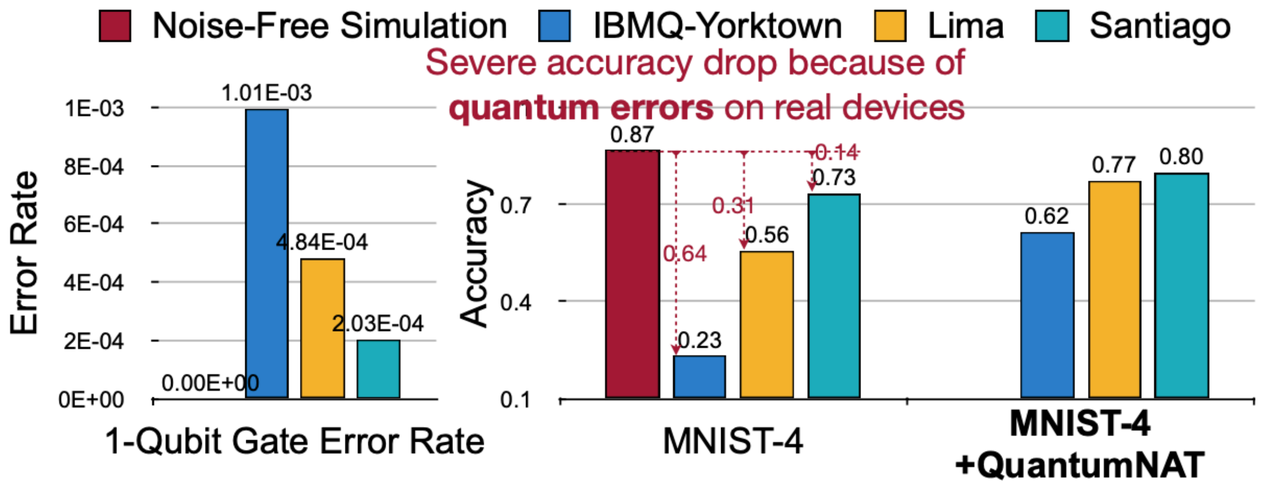
- Quantization provides denoising effects
- Quadratic penalty loss to encourage measurement outcomes close to quantization centroids

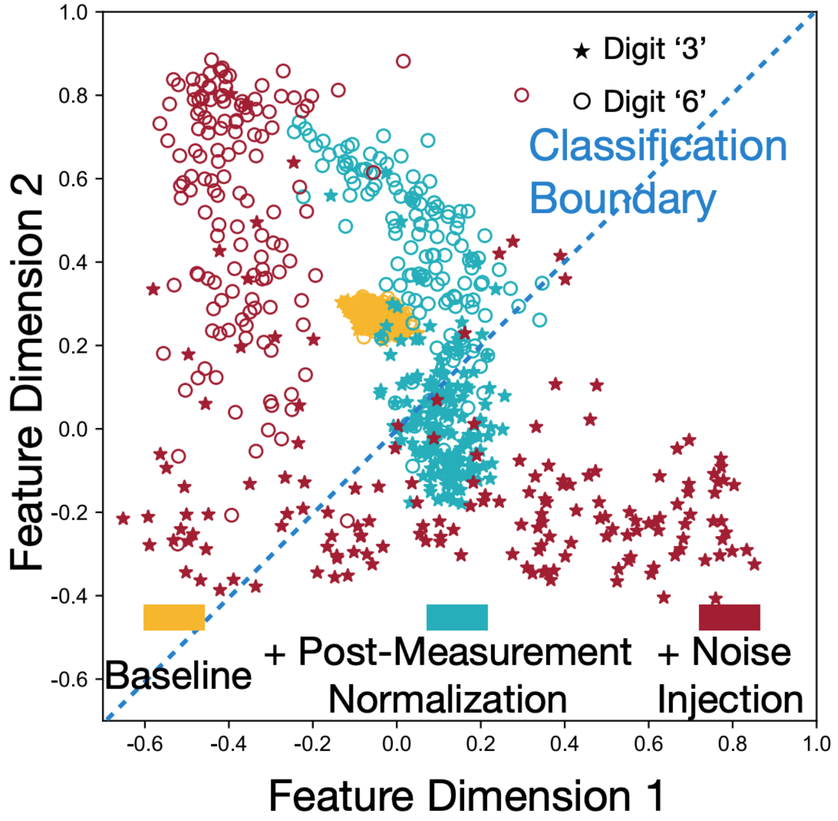




Evaluation

- On classification tasks with QNN

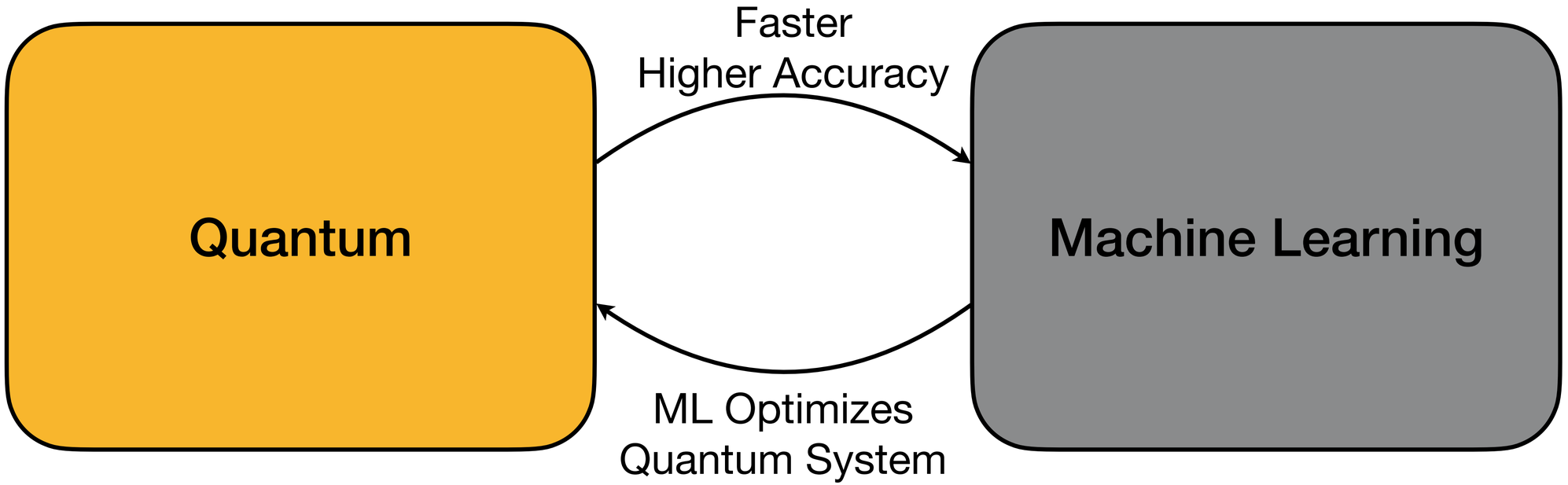




Torch Quantum

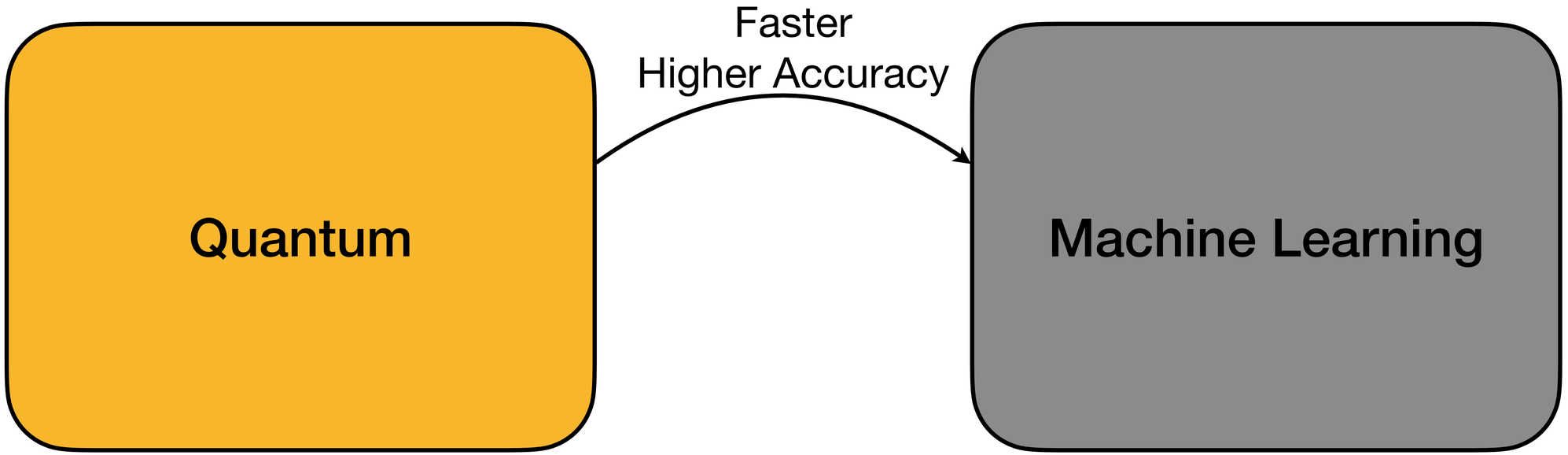
Open-source: TorchQuantum

- TorchQuantum — An open-source library for interdisciplinary research of quantum computing and machine learning
- https://github.com/mit-han-lab/torchquantum



Open-source: TorchQuantum

- TorchQuantum — An open-source library for interdisciplinary research of quantum computing and machine learning
- https://github.com/mit-han-lab/torchquantum

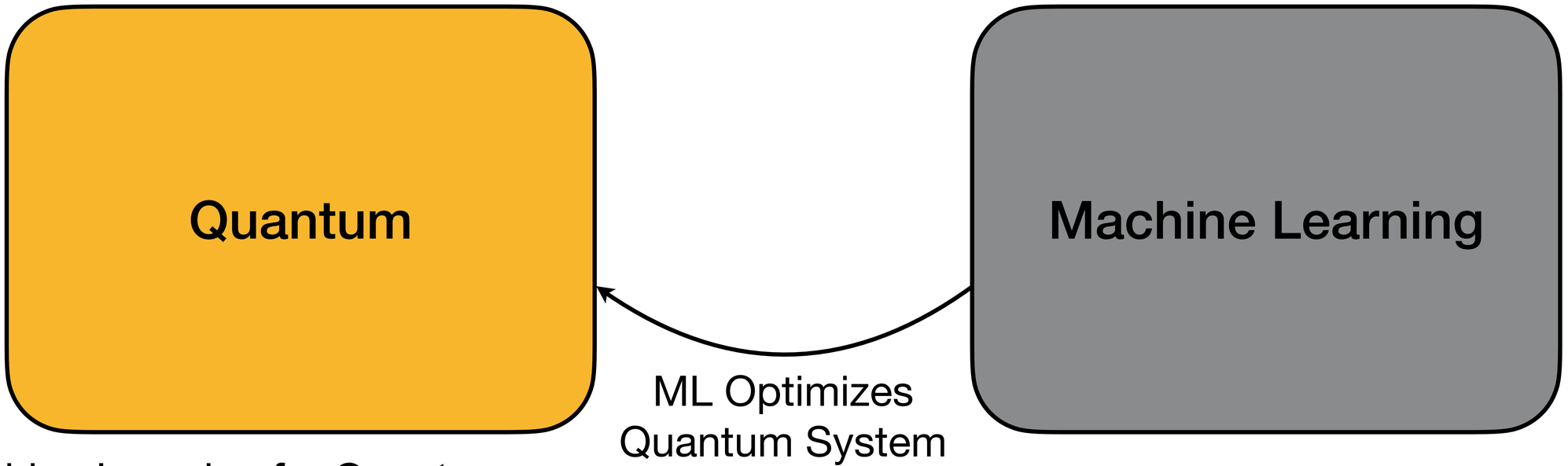


- Quantum for Machine learning
  - Quantum neural networks
  - Quantum kernel methods

Torch Quantum

Open-source: TorchQuantum

- TorchQuantum — An open-source library for interdisciplinary research of quantum computing and machine learning
- https://github.com/mit-han-lab/torchquantum



- Machine Learning for Quantum
  - ML for quantum compilation (qubit mapping, unitary synthesis)
  - …

Torch Quantum

TorchQuantum

- Features
  - Easy construction of **parameterized quantum circuits** such as Quantum Neural Networks in PyTorch
  - Support **batch mode inference and training** on GPU/CPU, supports highly-parallelized training
  - Support **easy deployment** on real quantum devices such as IBMQ
  - Provide tutorials, videos and example projects of QML and using ML to optimize quantum computer system problems

PyTorch Implementations

- Statevector

```
_state = torch.zeros(2 ** self.n_wires, dtype=C_DTYPE)
_state[0] = 1 + 0j
```

- Quantum Gates

```
'cnot': torch.tensor([[1, 0, 0, 0],
                      [0, 1, 0, 0],
                      [0, 0, 0, 1],
                      [0, 0, 1, 0]], dtype=C_DTYPE),
```

PyTorch Implementations

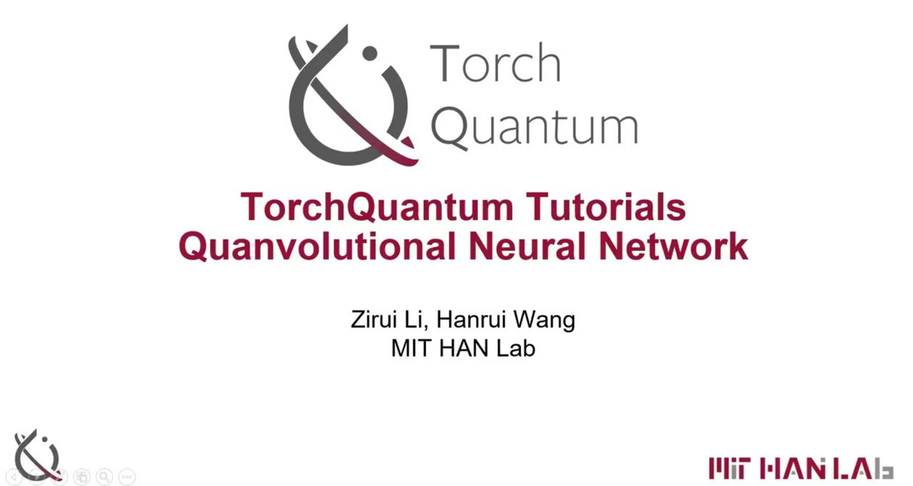
- Quantum Gates

```
def crx_matrix(params):
    theta = params.type(C_DTYPE)
    co = torch.cos(theta / 2)
    jsi = 1j * torch.sin(-theta / 2)

    matrix = torch.tensor([[1, 0, 0, 0],
                           [0, 1, 0, 0],
                           [0, 0, 0, 0],
                           [0, 0, 0, 0]], dtype=C_DTYPE, device=params.device
                          ).unsqueeze(0).repeat(co.shape[0], 1, 1)
    matrix[:, 2, 2] = co[:, 0]
    matrix[:, 2, 3] = jsi[:, 0]
    matrix[:, 3, 2] = jsi[:, 0]
    matrix[:, 3, 3] = co[:, 0]

    return matrix.squeeze(0)
```

- Matrix-vector multiplication: torch.einsum and torch.bmm

Examples and tutorials

- Tutorial Colab and videos



Thank you for listening!

- **QuantumNAS** exploits **SuperCircuit-**based co-search for most **noise-robust** circuit architecture and qubit mapping
- Iterative **quantum gate pruning** to further remove redundant gates
- **QuantumNAT** exploits noise injection, post-measurement Normalization, quantization to improve robustness of parameters
- Improves MNIST 2-class accuracy from 88% to **95%**,10-class from 15% to **34%**
- Save search cost by over **1,000 times**
- Open-sourced **TorchQuantum** library for Quantum + ML research

https://github.com/mit-han-lab/torchquantum

qmlsys.mit.edu

qmlsys.mit.edu