

A Scalable Software Framework for Distributed Quantum-HPC Integration: Resource-Oriented Orchestration and Workflow Optimization

Presenter:

Louis Chen @ Imperial College London
Distributed Quantum Computing Group,
Department of Electrical and Electronics Engineering

Co-Authors:

Felix Burt (EEE, ICL) , Kin K. Leung (EEE, ICL)
Collaboration:



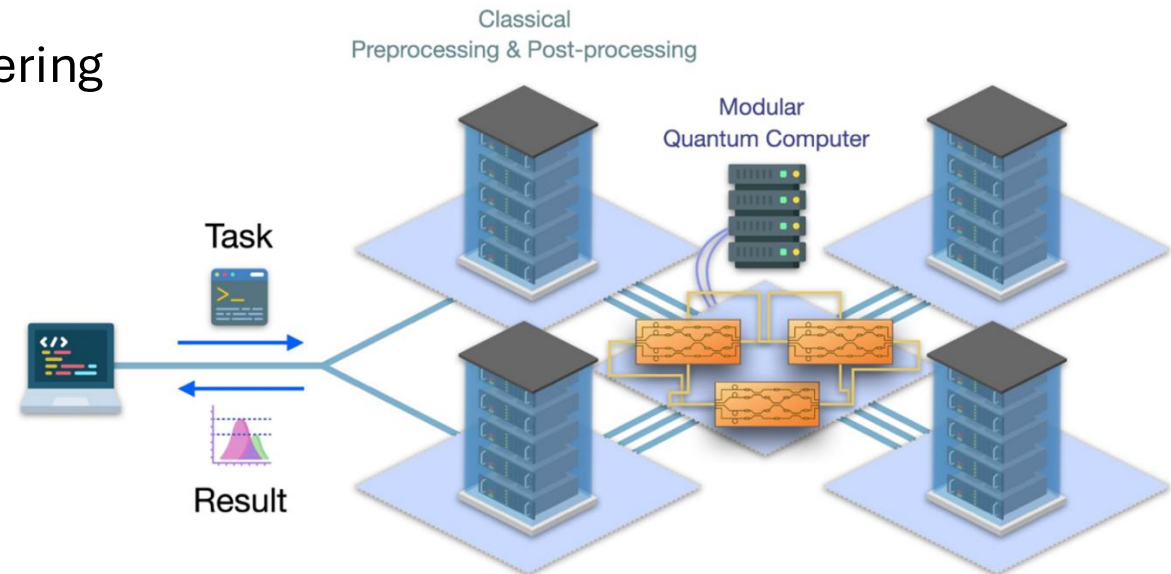
NAR Labs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing



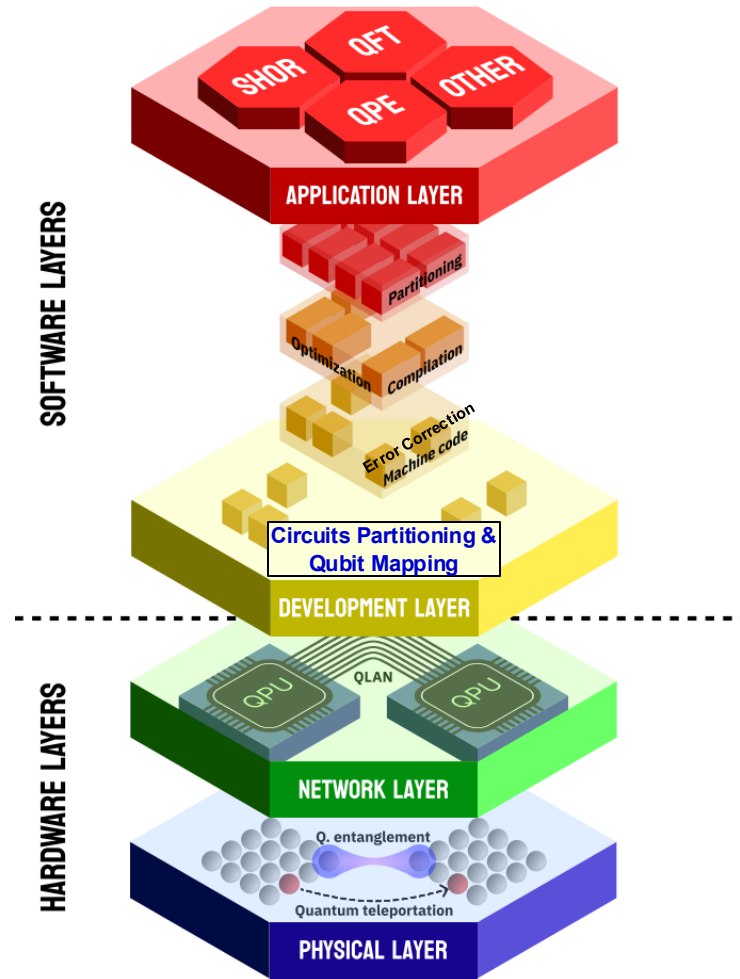
National Quantum
Computing Centre



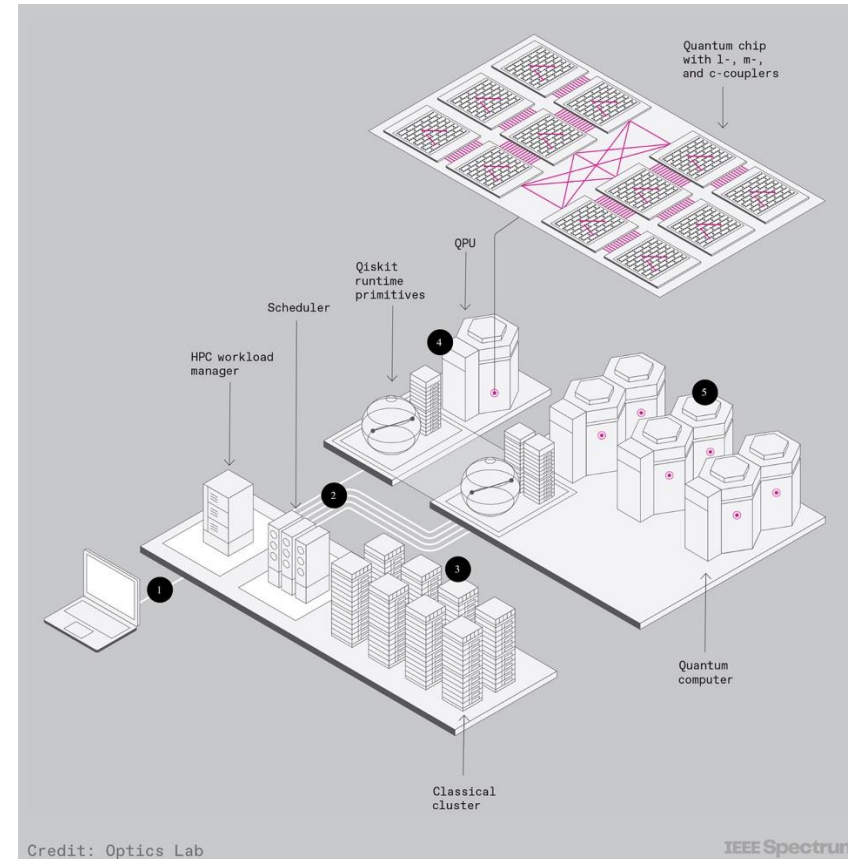
Brookhaven
National Laboratory



Content

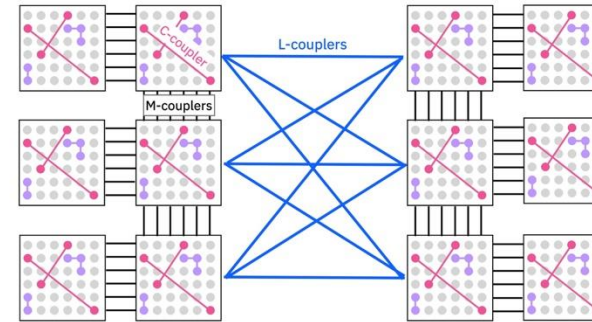
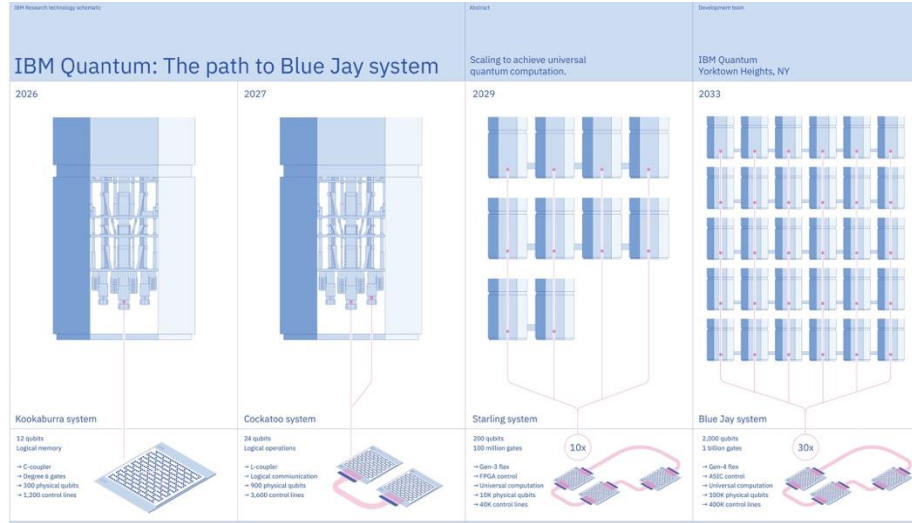


- Introduction for Distributed Quantum Computing and Quantum HPC
- Ideas of Distributed Quantum Computing
 - Circuit Partitioning in Distributed Quantum Computing
 - Resource Efficient Compilation for Distributed Quantum Computing
 - Distributed Photonic Quantum Computing
- Conclusion

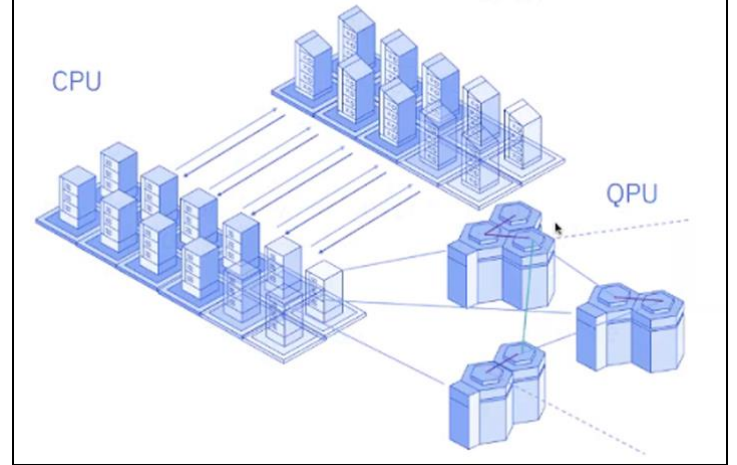


What and Why Distributed Quantum Computing?

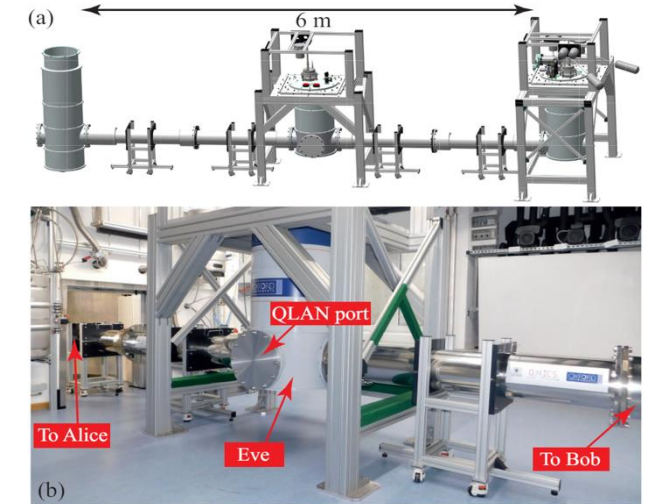
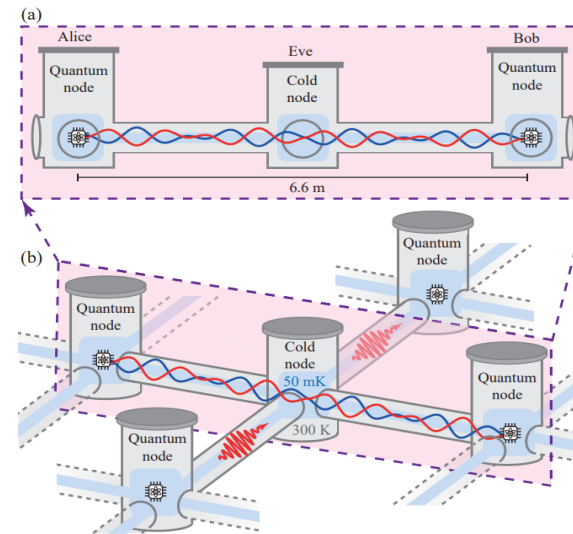
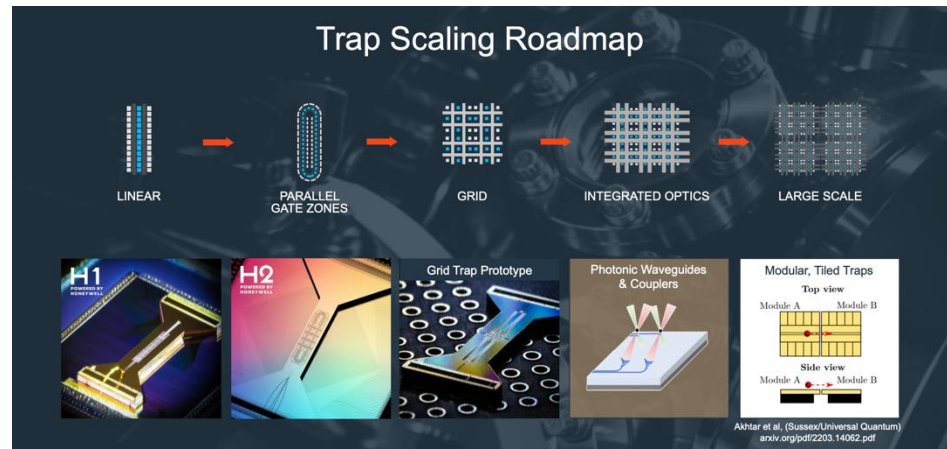
IBM Quantum (Superconducting Circuits – Microwave Controlled)



Dist Quantum-HPC GPU



Quantinuum (Ion Trap – Optical Controlled)

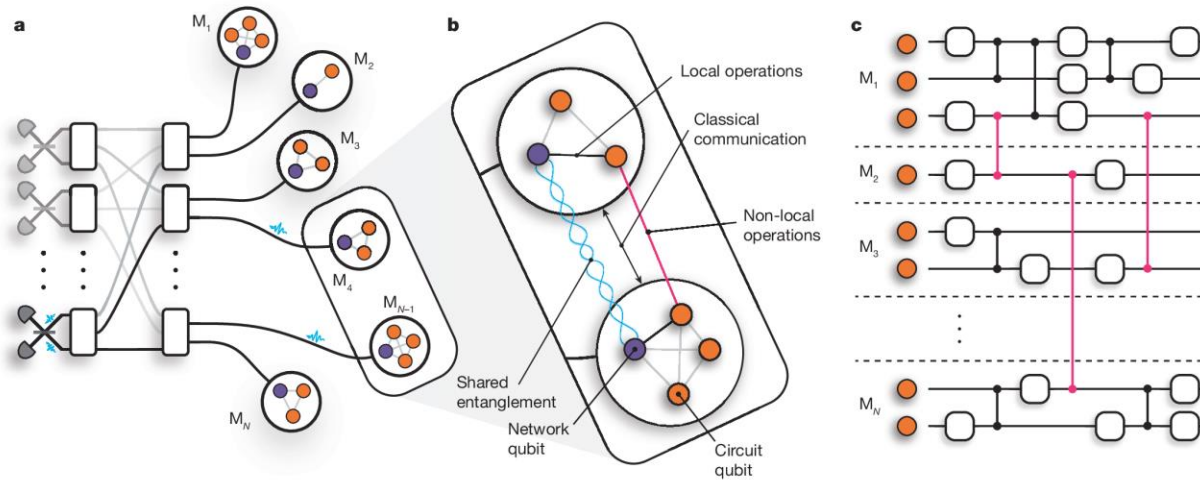


Renger, M., et al. "Cryogenic microwave link for quantum local area networks." *arXiv preprint arXiv:2308.12398* (2023).

Quantum-HPC and Distributed Quantum Computing

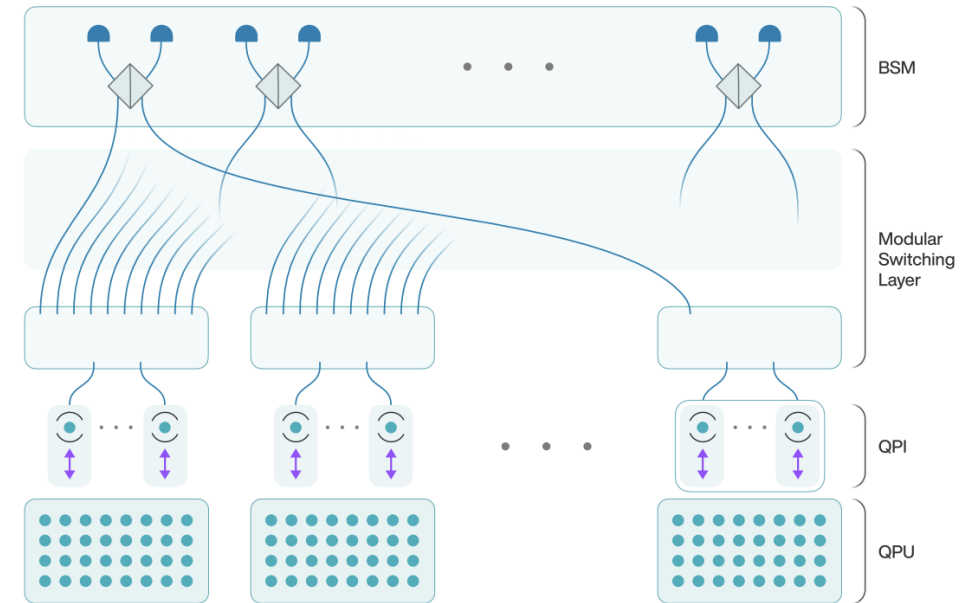
Motivation

Distributed quantum computing offers a scalable pathway to extend QAOA beyond the limitations of individual NISQ devices by parallelising computation across modular quantum processors—enabling practical solutions to large-scale optimisation problems.



Main, D., et al. "Distributed quantum computing across an optical network link." *Nature* (2025): 1-6.

Distributed Quantum Error Correction Code



Sutcliffe, Evan, et al, *arXiv preprint arXiv:2501.14029* (2025).

Circuit Partition in Distributed Quantum Computing

- Quantum advantage requires large numbers of qubits ($\sim 10^6$ for Shor's algorithm [Gidney, arXiv:2505.15917, 2025]).
- Inherent scaling challenges indicate we need modular/distributed quantum computing to reach these numbers [Van Meter, Computer 49, IEEE, 2016].
- Running large programs across multiple quantum processing units (QPUs) requires *partitioning* of logical quantum circuits.
- QPUs are linked via quantum network dedicated to sharing entanglement.
- Slow and noisy inter-QPU links identify shared entanglement (*e-bits*) as the partitioning objective [Caleffi et al., Computer Networks 254, ScienceDirect, 2024].

Challenges for Partitioning Distributed Quantum Circuits

- Circuit partitioning with minimal entanglement has been reduced to various problems. These are typically NP-hard problems [Andrés-Martínez and Heunen, PRA 100, APS, 2019], and often suffer from a lack of generality (limitations on qubit/gate teleportation).

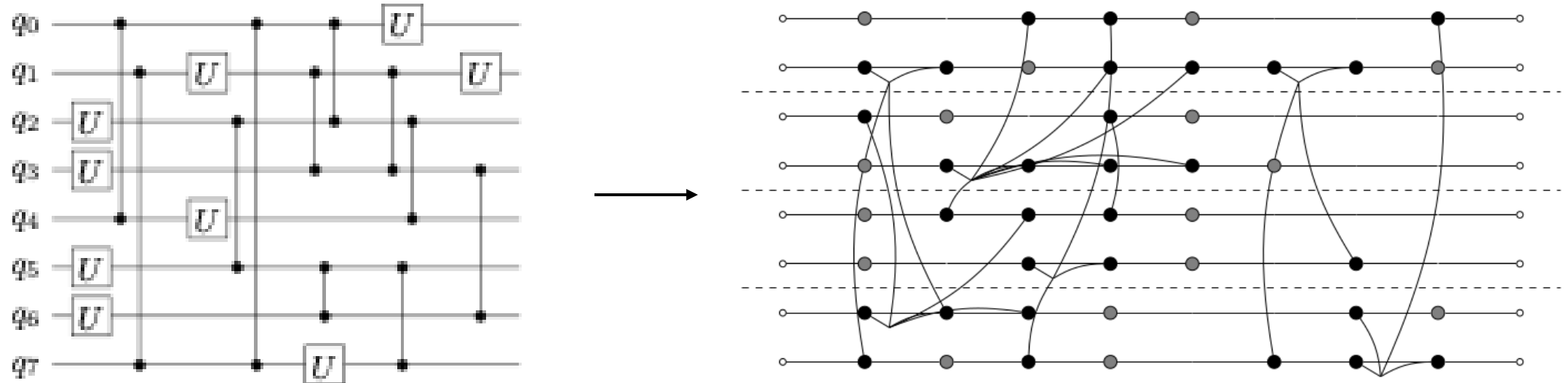
Routing methods try and find the optimal teleportation paths for each qubit in order to cover all two-qubit gates.

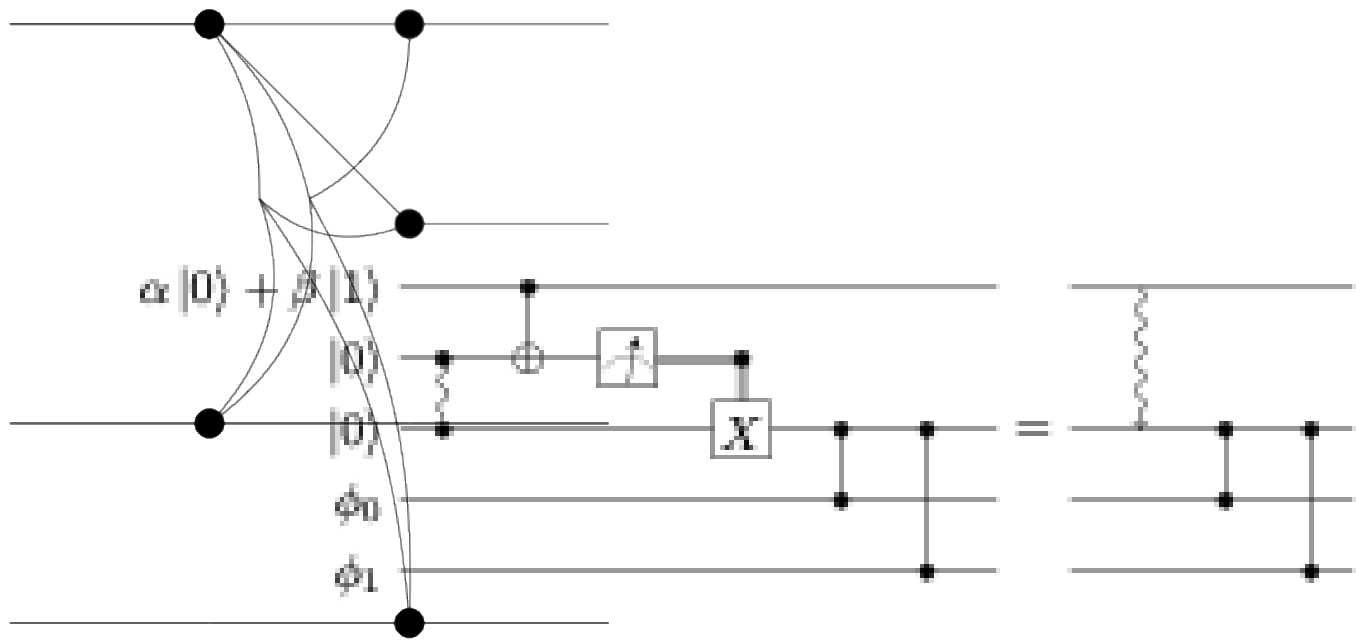
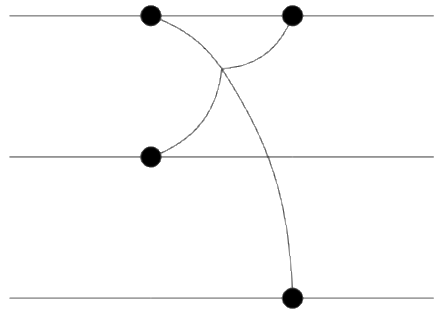
Partitioning methods statically assign qubits to modules in ways that minimise the number of gate teleportations.

- We identified two key challenges to solving this problem effectively:
 1. **Problem formulation:** we need to formulate a problem that is general enough to consider routing and partitioning together.
 2. **Solution techniques:** we need to reduce the problem complexity without losing too much to enable fast, effective solutions.

Temporal Hypergraph Partitioning

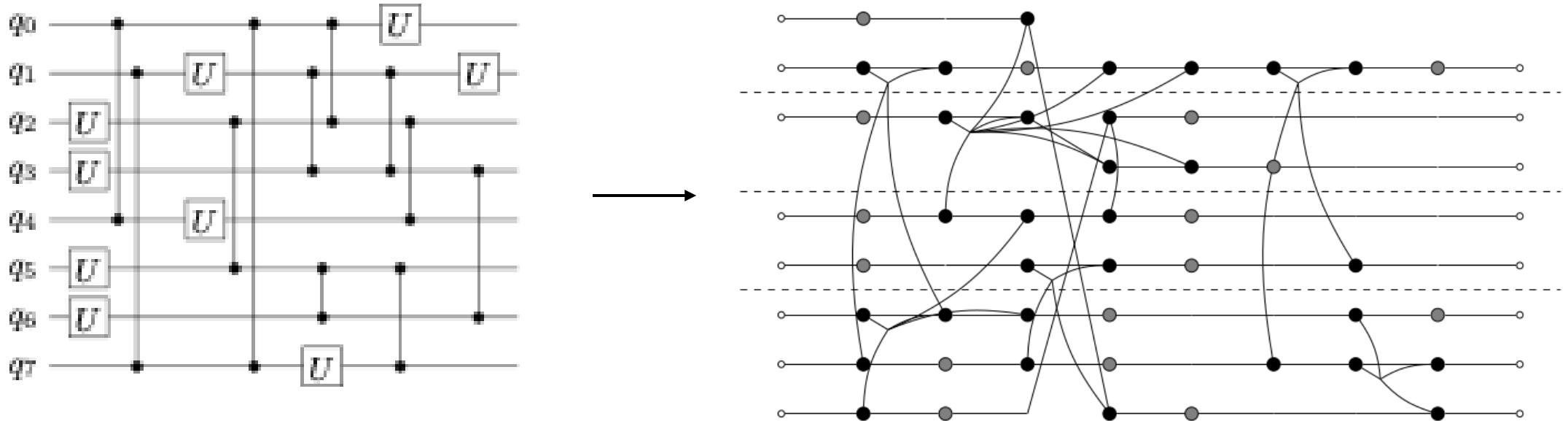
- Blending existing ideas of hypergraph partitioning [Andrés-Martínez and Heunen, PRA 100, APS, 2019], and time-sliced partitioning [Baker et al., ‘Time-Sliced Quantum Circuit Partitioning for Modular Architectures’, Computing Frontiers 17, ACM, 2020], we have developed a representation of quantum circuits as hypergraphs extended in time [Burt et al., QCE, IEEE, 2025][Burt et al., arXiv:2503.19082, 2025].





Temporal Hypergraph Partitioning

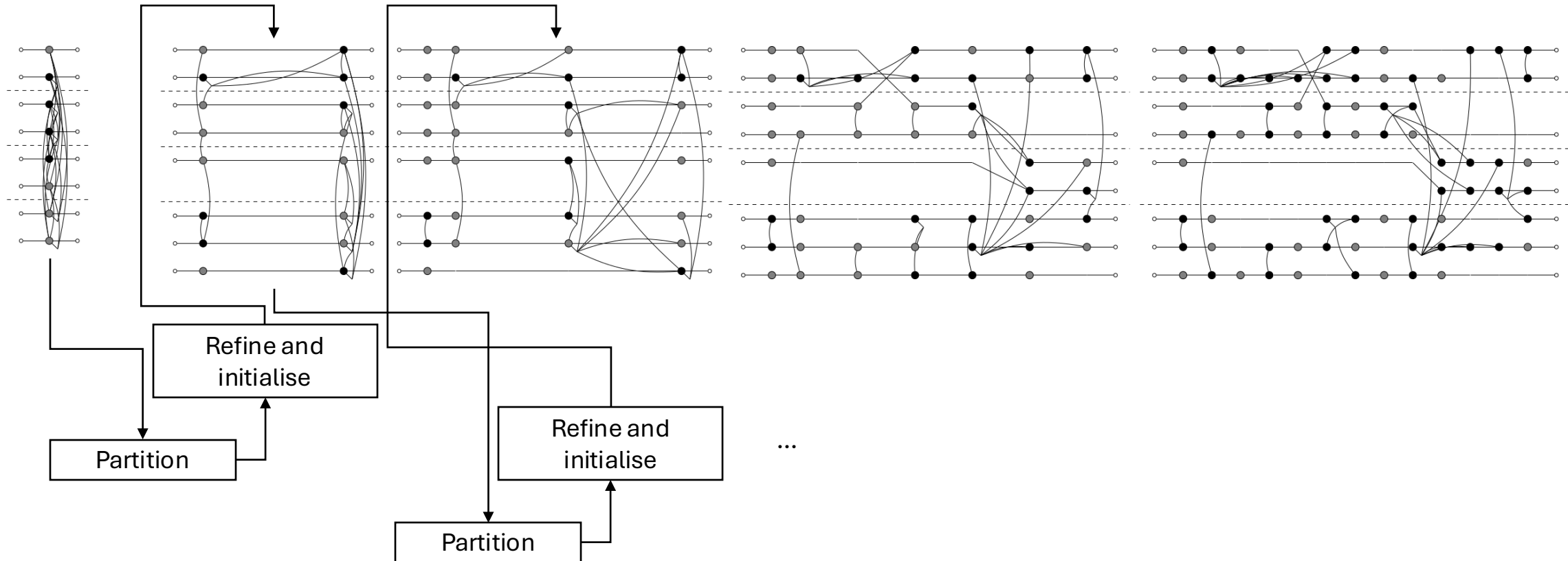
- Blending existing ideas of hypergraph partitioning [Andrés-Martínez and Heunen, PRA 100, APS, 2019], and time-sliced partitioning [Baker et al., ‘Time-Sliced Quantum Circuit Partitioning for Modular Architectures’, Computing Frontiers 17, ACM, 2020], we have developed a representation of quantum circuits as hypergraphs extended in time [Burt et al., QCE, IEEE, 2025][Burt et al., arXiv:2503.19082, 2025].

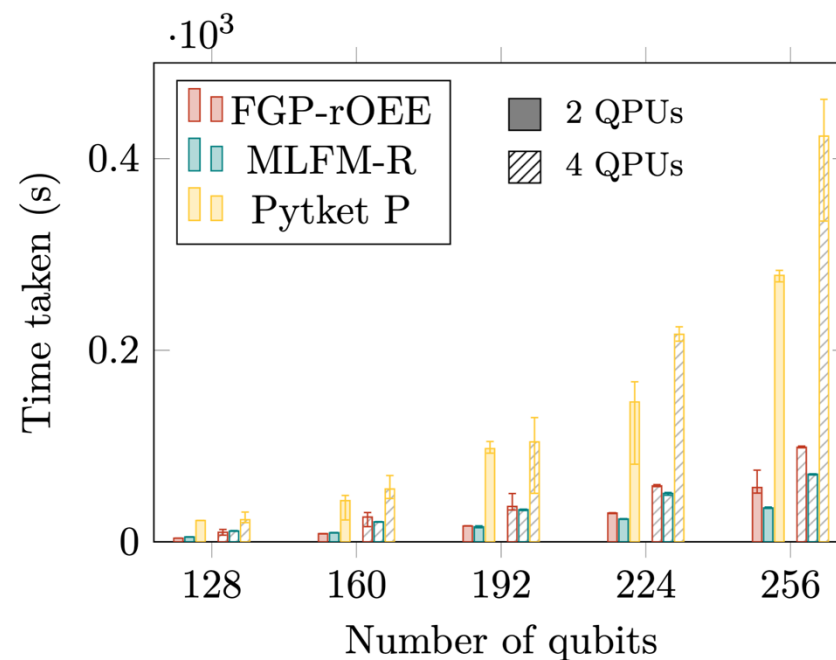
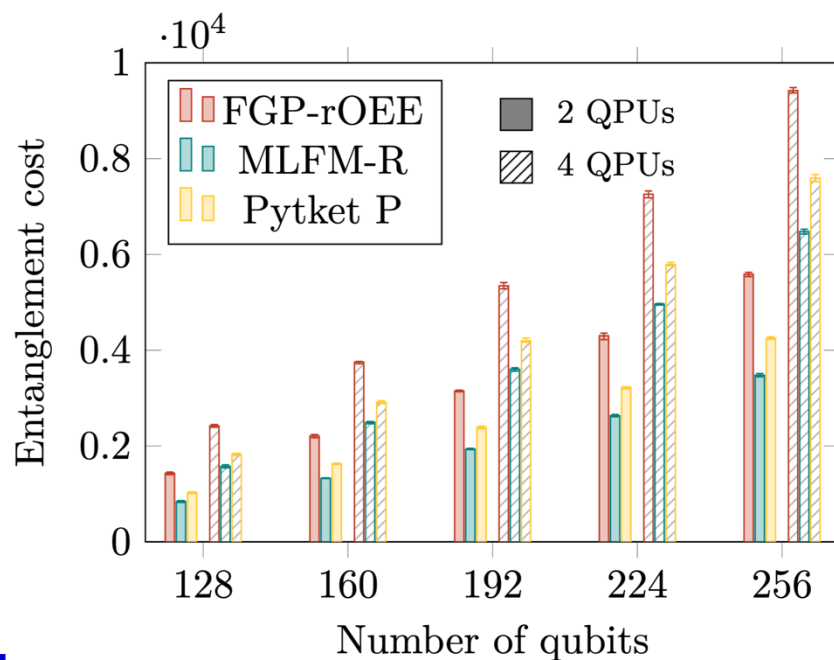
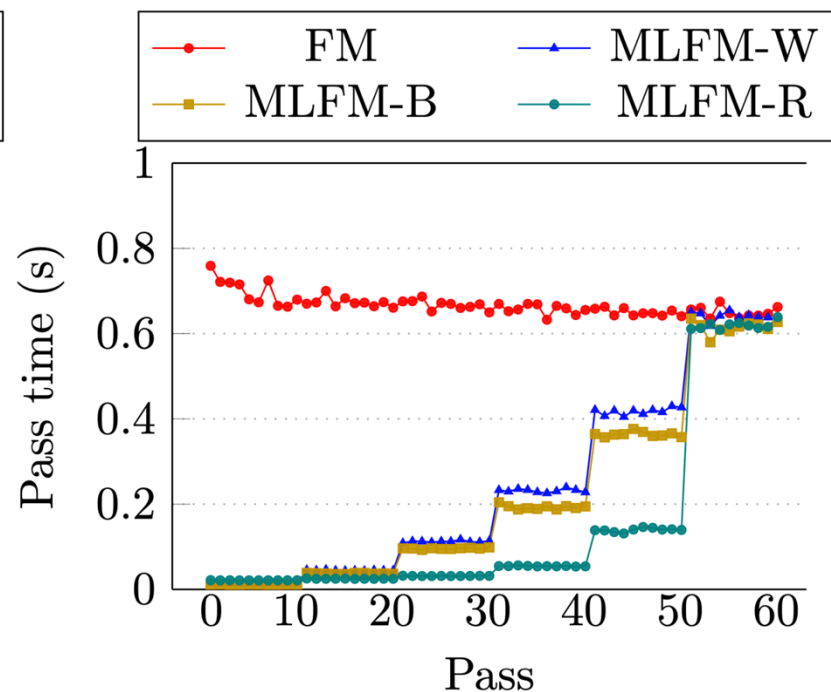
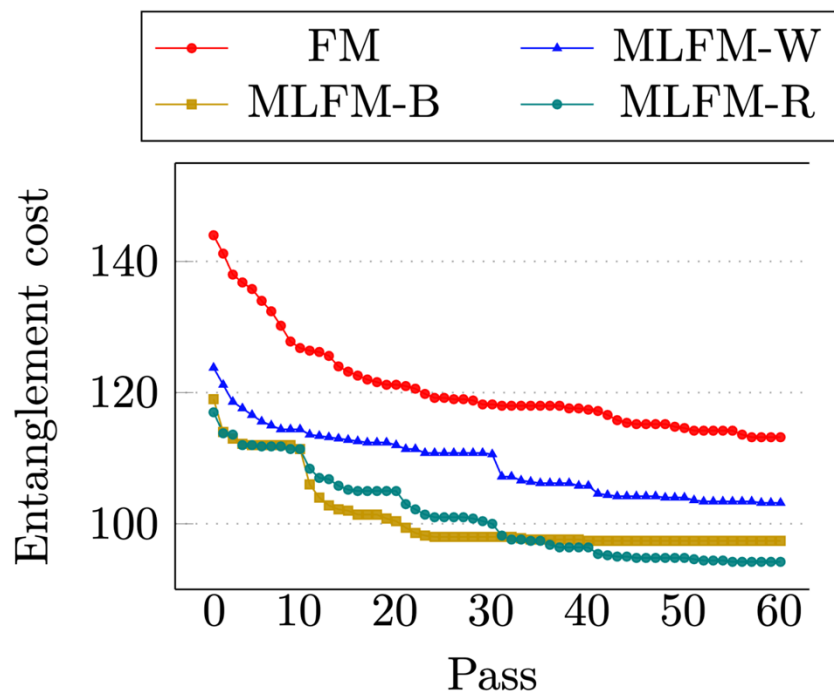


- A partitioned hypergraph of this form tells us exactly how many end-to-end e-bits at each time step of the circuit.

Multilevel Partitioning

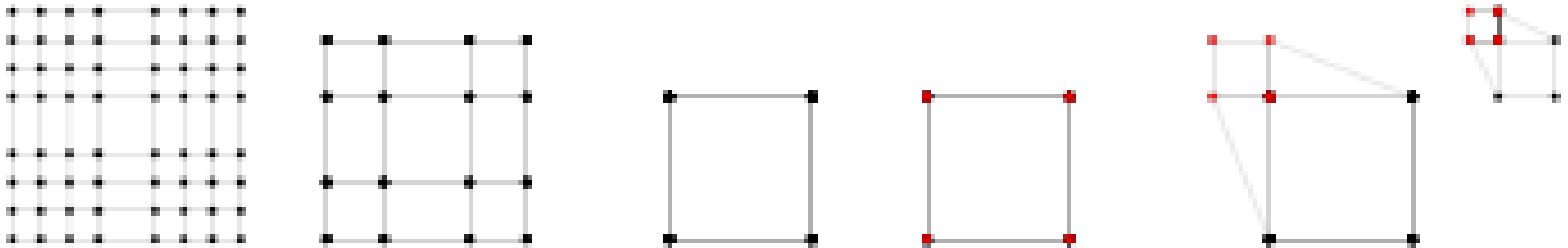
- Problem hypergraphs can be large for deep circuits.
- We use a *multilevel partitioning* framework to enhance the efficiency of heuristic solvers.





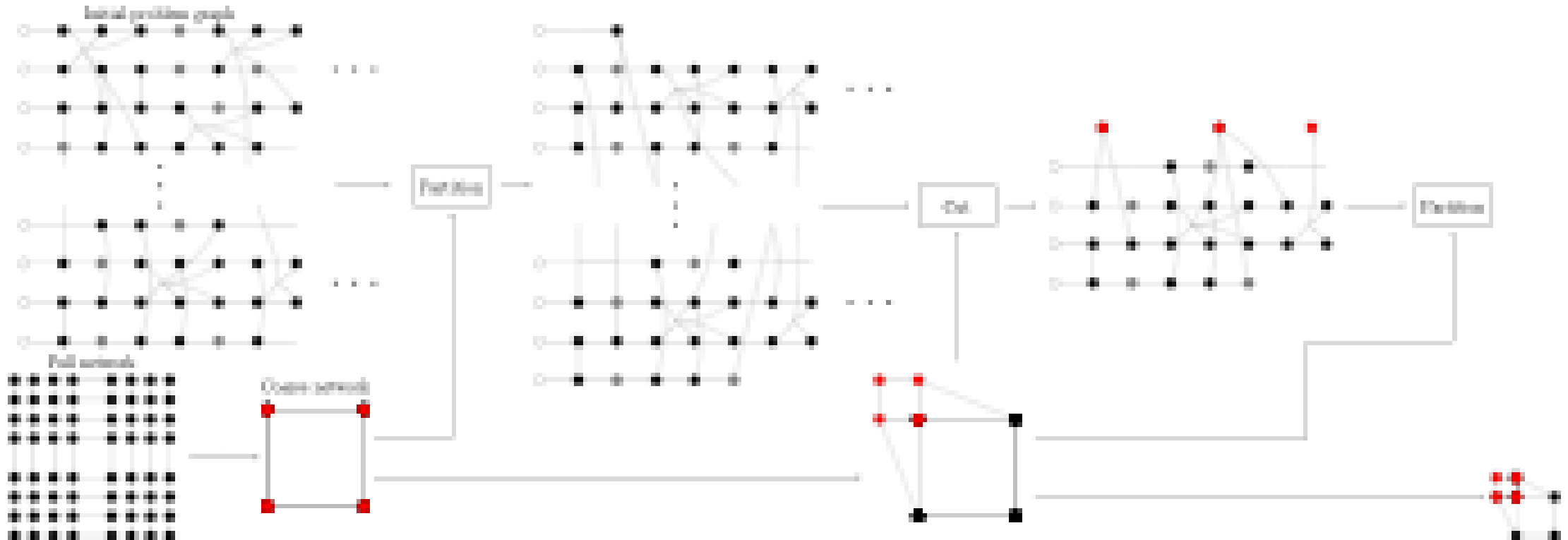
Partition over Larger Networks

- Entanglement distribution paths become significant over larger networks.
- Partitioners must account for auxiliary entanglement costs of generating end-to-end ebits (swapping, purification etc.).
- We can incorporate this into our objective, for some additional complexity.
- To mitigate this, we can employ coarsening on the network level.



Partition over Larger Networks

- Generate series of coarsened representations of network.
- Partition large circuit over coarse network.
- Cut circuits and merge *external* nodes according to their partition.



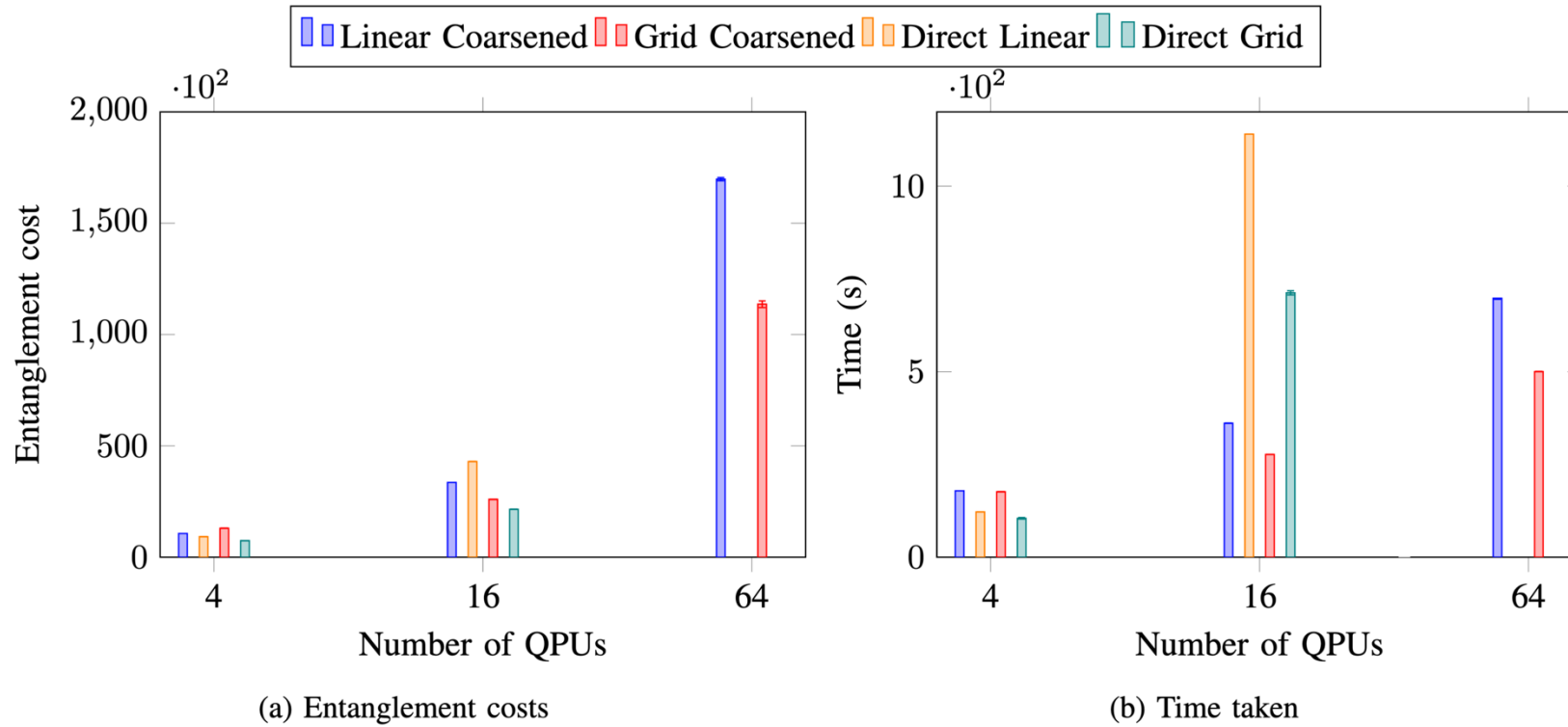
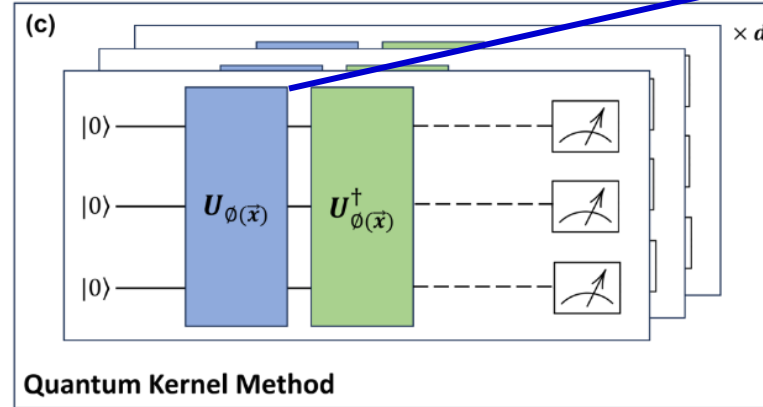
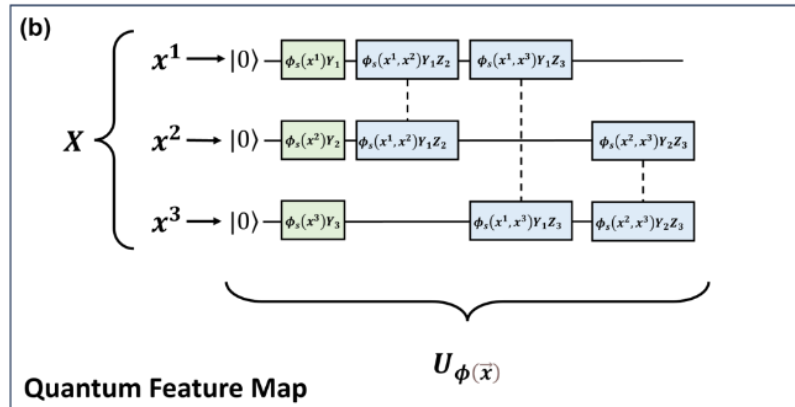
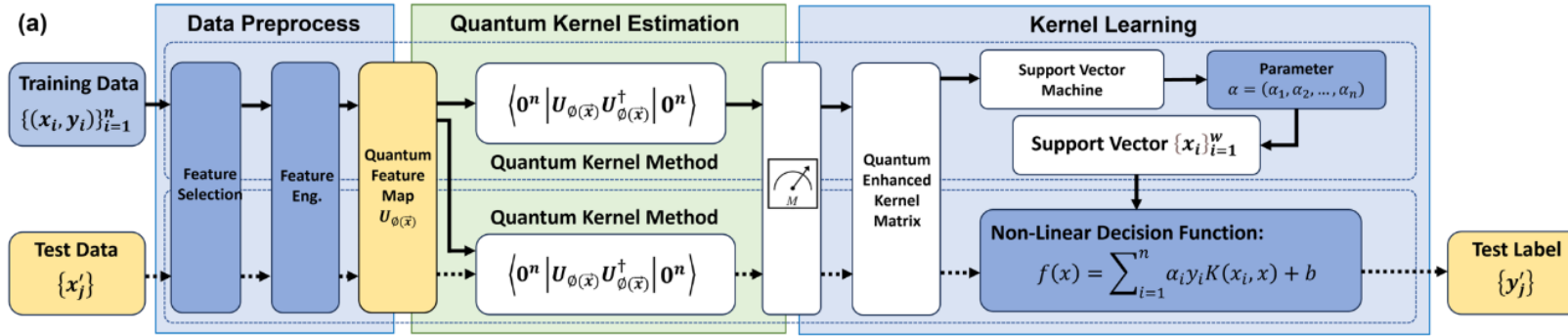


Fig. 12: Comparison of partitioning over coarsened networks and direct partitioning for grid and linear networks. We use network coarsening factors 2, 4 and 8, for 4, 15 and 64 QPUs, respectively. A 256 qubit CP -fraction circuit with 0.5 two-qubit gate proportion is used.

Resource Efficient Compilation for Distributed Quantum Computing

Gate-Based NISQ Quantum Algorithms

- Gate-based Quantum Machine Learning Algorithm



Kernel Function:

A commonly used quantum kernel in QSVM is the Gaussian kernel, adapted for quantum computing:

$$K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{2\sigma^2}\right) \quad (7)$$

where x_i and x_j are input data vectors and σ denotes the kernel's width parameter.

Unitary Gate:

$$U_{\Phi}(x) = \exp\left(i \sum_{S \subseteq [n]} \phi_S(x) \prod_{k \in S} Z_k\right)$$

In this expression, S represents either the k -th element or a set of k elements drawn from n , which generally indicates the connectivity among various qubits or data points. The indices k range from 1 to n , and Z_k represents the application of R_Z operations

Chen, Kuan-Cheng, et al. "Quantum-Enhanced Support Vector Machine for Large-Scale Multi-class Stellar Classification." International Conference on Intelligent Computing. Singapore: Springer Nature Singapore, 2024.

Validation of Large-Scale Quantum Computing

Example: Quantum Machine Learning Algorithm:

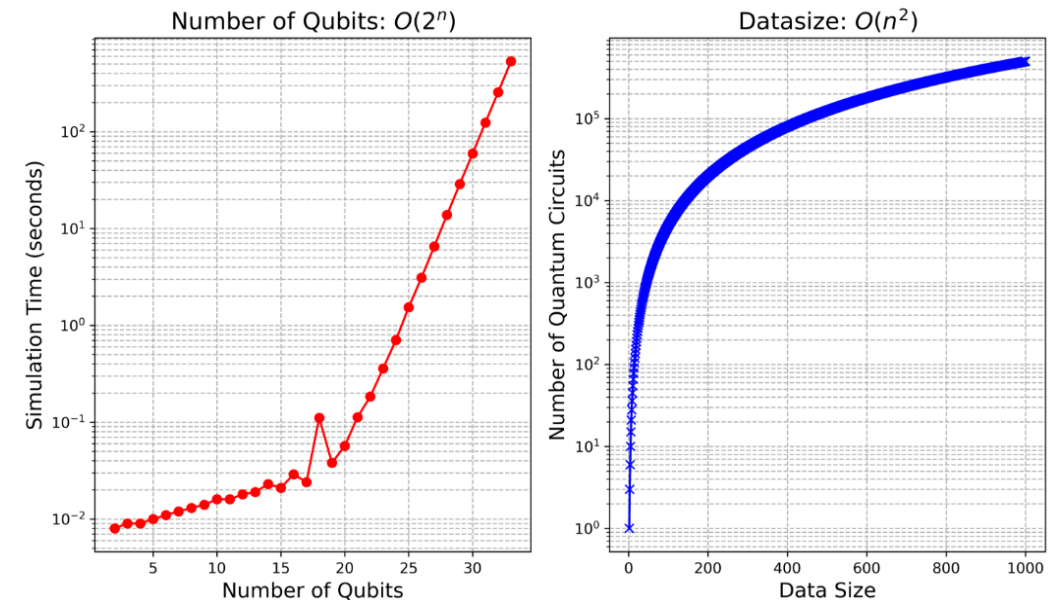
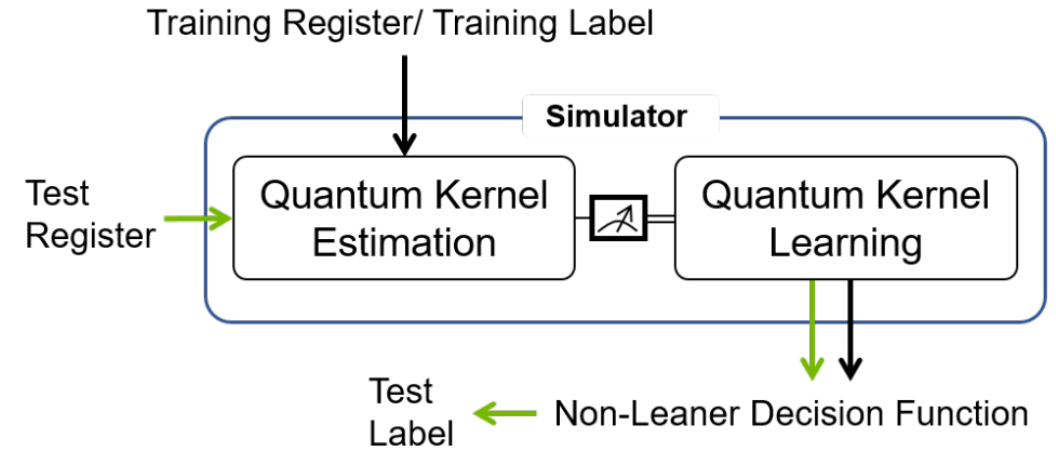
Quantum-enhanced Support Vector Machines (SVMs) leverage quantum computing to achieve exponential speedups in training and classification tasks for SVMs, addressing the computational challenges faced in large-scale big-data applications.

Cost of Quantum Circuit Simulation with Naïve State Vector on CPU:

- Exponential growth with an increase in qubits
 - HPC with 3 PB memory can only simulate 47 qubits arbitrary circuit.
 - Simulate one 40-qubit circuit for QSVM takes roughly one day.
- Quadratic growth with an increase in data size
 - Insufficient training data will affect the model's performance.

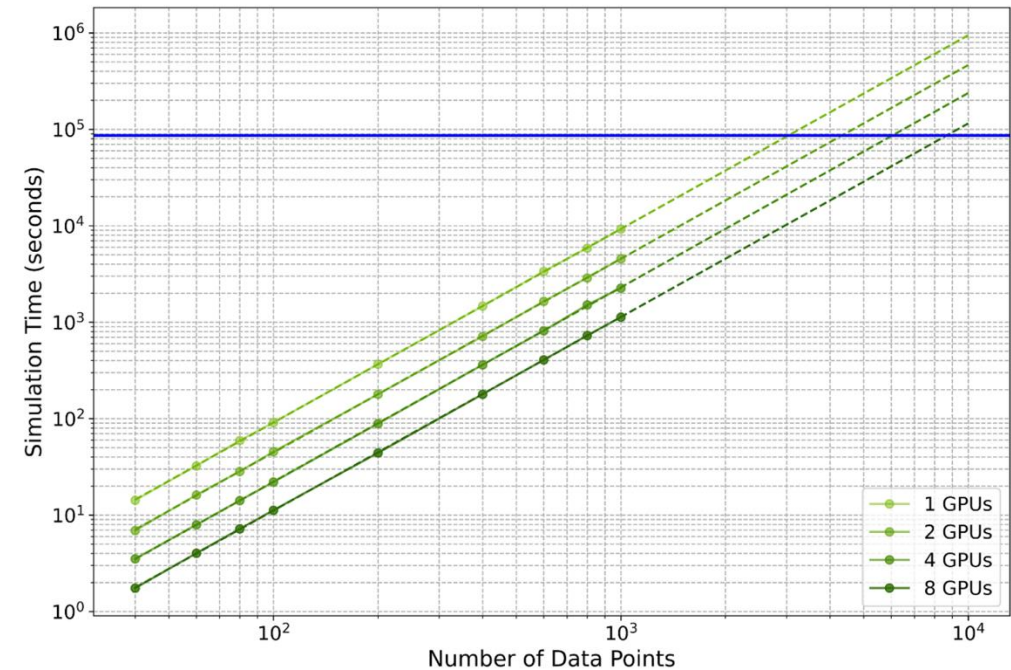
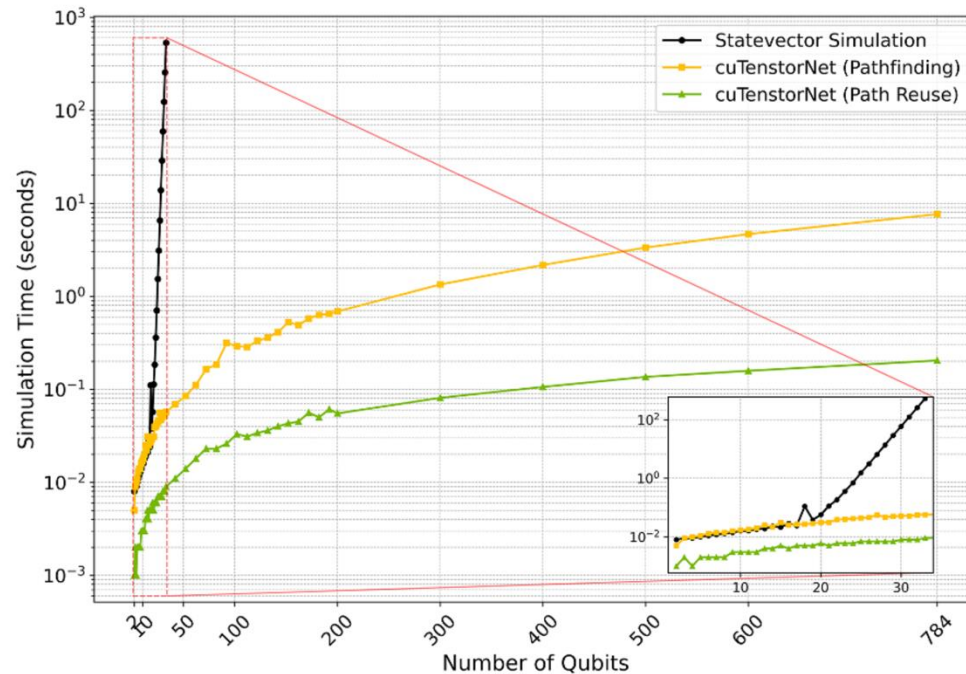
Solution for Large-Scale Qubit Systems with Large Datasets

- cuQuantum SDK with cuTensorNet approach can reduce the computational complex for simulating large-scale qubit system (toward a thousand of qubits).
- Multi-GPU with Message Passing Interface (MPI) in DGX clusters allows for efficient scaling of dataset in QSVMs.



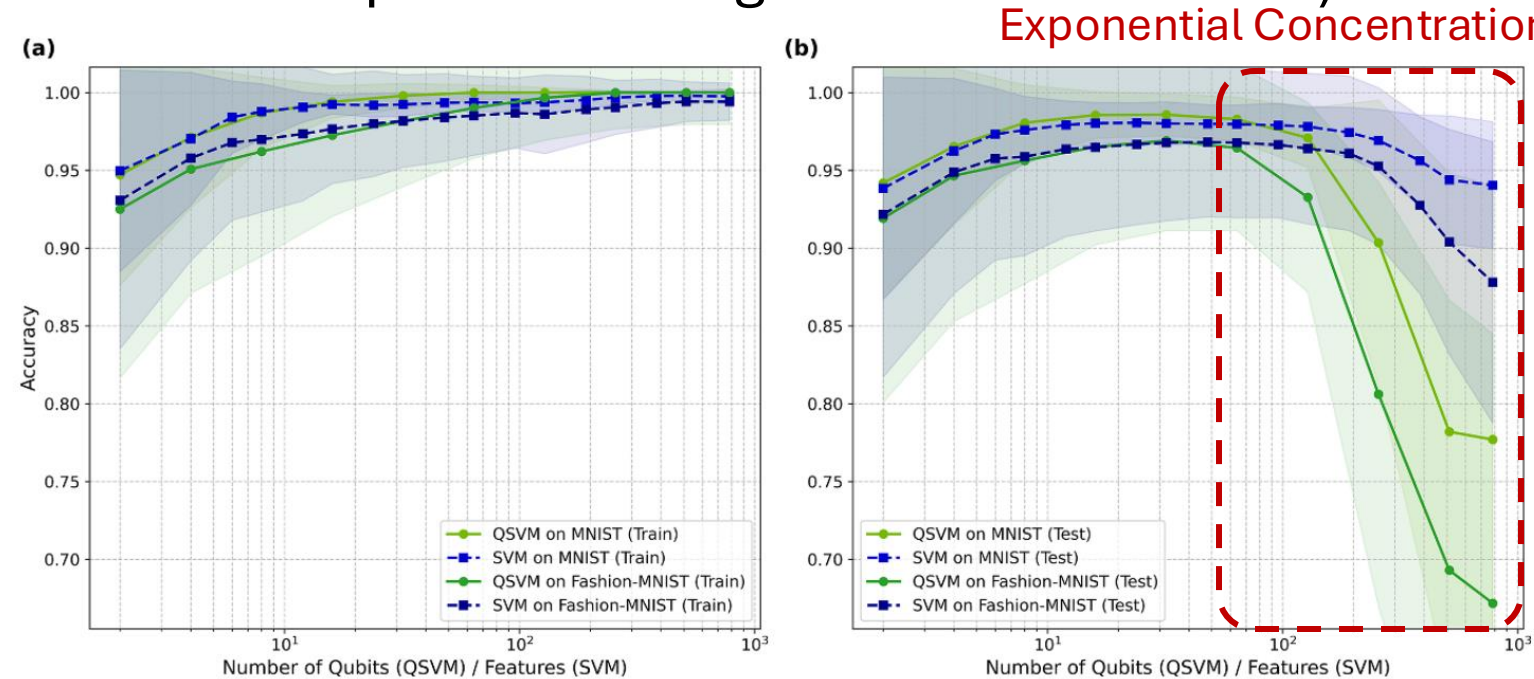
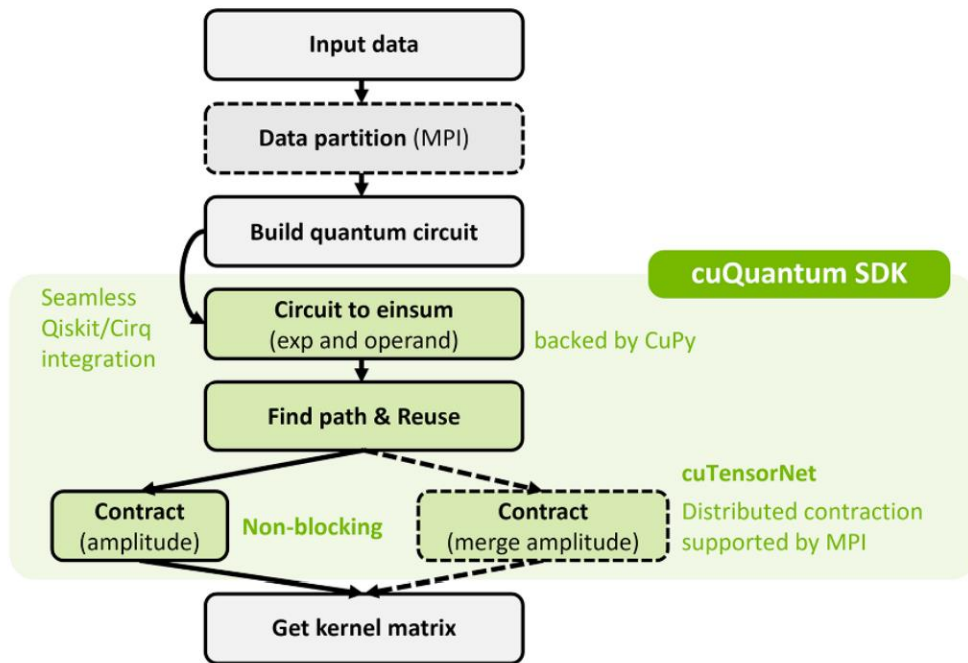
Validation of Large-Scale Quantum Computing

- **Reduction in Computational Complexity:** The **tensor network** simulation workflow, enhanced by Nvidia GPU, drastically reduces the computational complexity from exponential in the number of qubits $O(2^n)$ to polynomial $O(n^2)$, enabling more efficient processing of large datasets and complex quantum circuit simulations.
- **Multi-GPU Execution and Efficiency:** Utilizing multi-GPU configurations, the cuTensorNet framework achieves significant computational speedups by employing strategies such as path reuse and asynchronous tensor contractions, which minimize delays and optimize GPU resource utilization, resulting in a substantial increase in throughput over traditional CPU-based methods.
- **Scalability with Multi-GPU Systems:** The integration of cuTensorNet with MPI on multi-GPU systems demonstrates strong linear scalability, effectively decreasing the computation time for QSVM simulations across large datasets. This approach allows for real-time processing enhancements and fosters the practical application of quantum simulations in real-world scenarios.



Validation of Large-Scale Quantum Computing

- Validating the Quantum Kernel Method Using Tensor-Network Simulation with NVIDIA Multi-GPU Processing
- Gate-based quantum machine learning (QML) algorithms can suffer from barren plateaus or exponential concentration—leading to inefficient training—as the number of qubits increases. (the case here follows Superconducting Circuit Architecture)

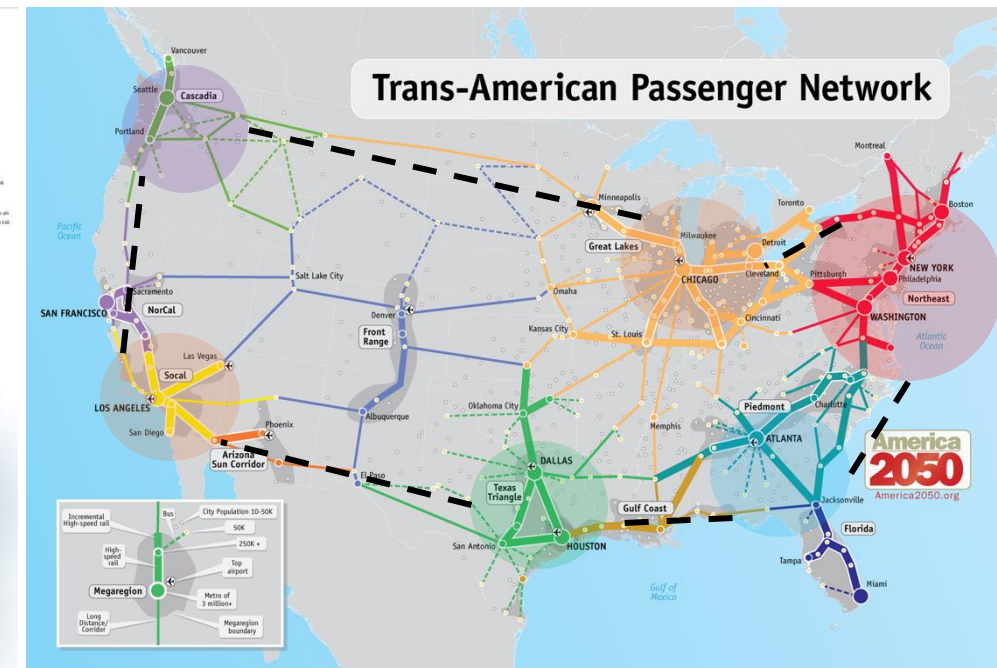
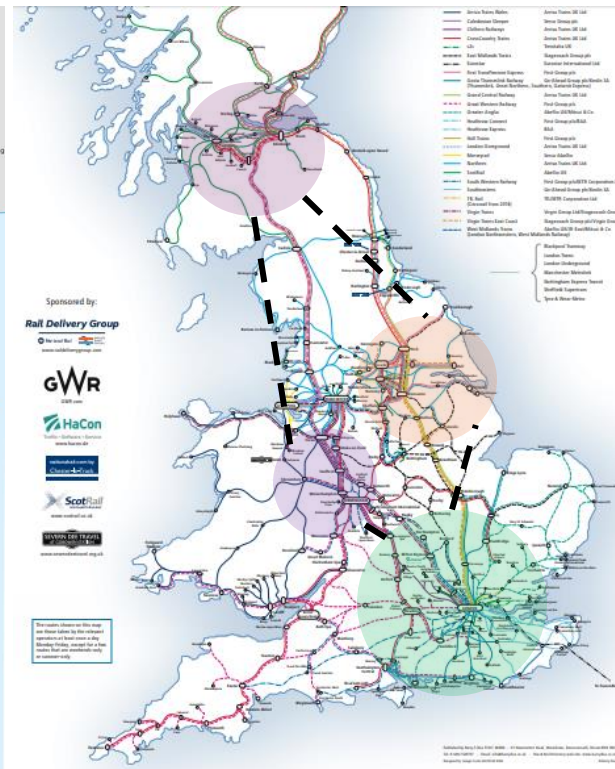
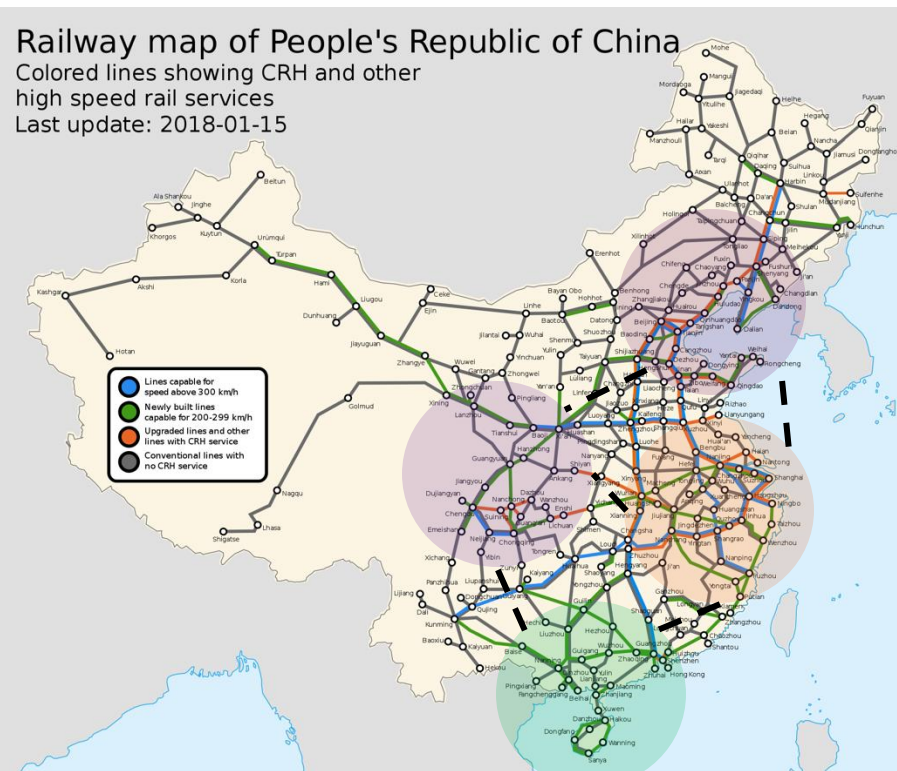


Chen, Kuan-Cheng, et al. "Validating Large-Scale Quantum Machine Learning: Efficient Simulation of Quantum Support Vector Machines Using Tensor Networks." *Machine Learning: Science and Technology* (2024).

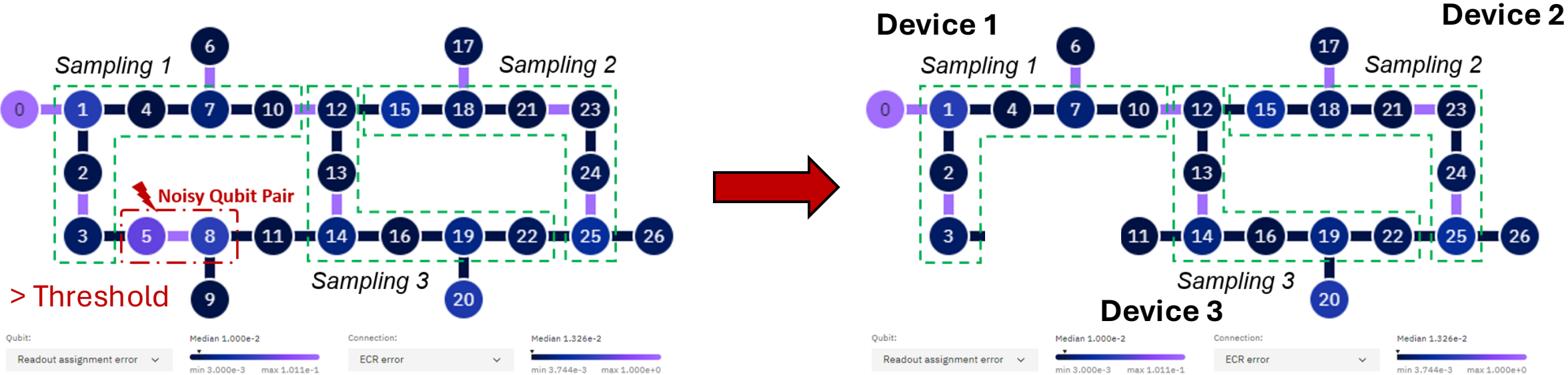
Motivation of Applying DQC for QAOA

The real-world problem:

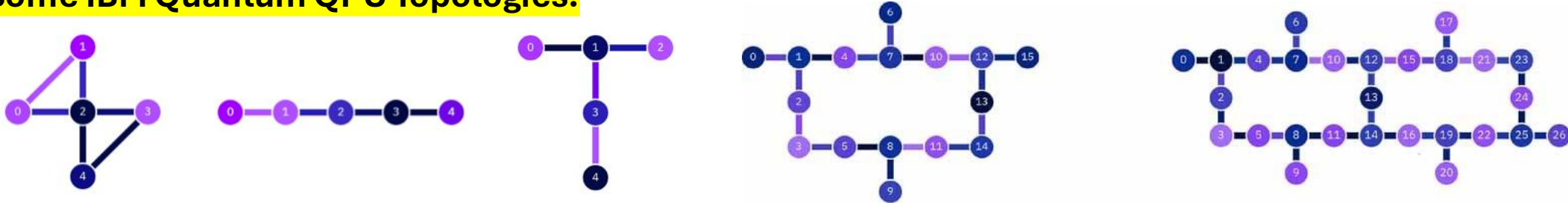
In real-world optimization problems, data is often not homogeneously distributed. For example, in railway optimization, individual areas or regions may have dense railway networks, whereas inter-regional connections often rely on a few main routes. This formulation aligns well with the problem formulation of distributed quantum computing, where tasks can be divided based on varying data densities.



Noise-Aware Distributed QAOA Compilation Strategy



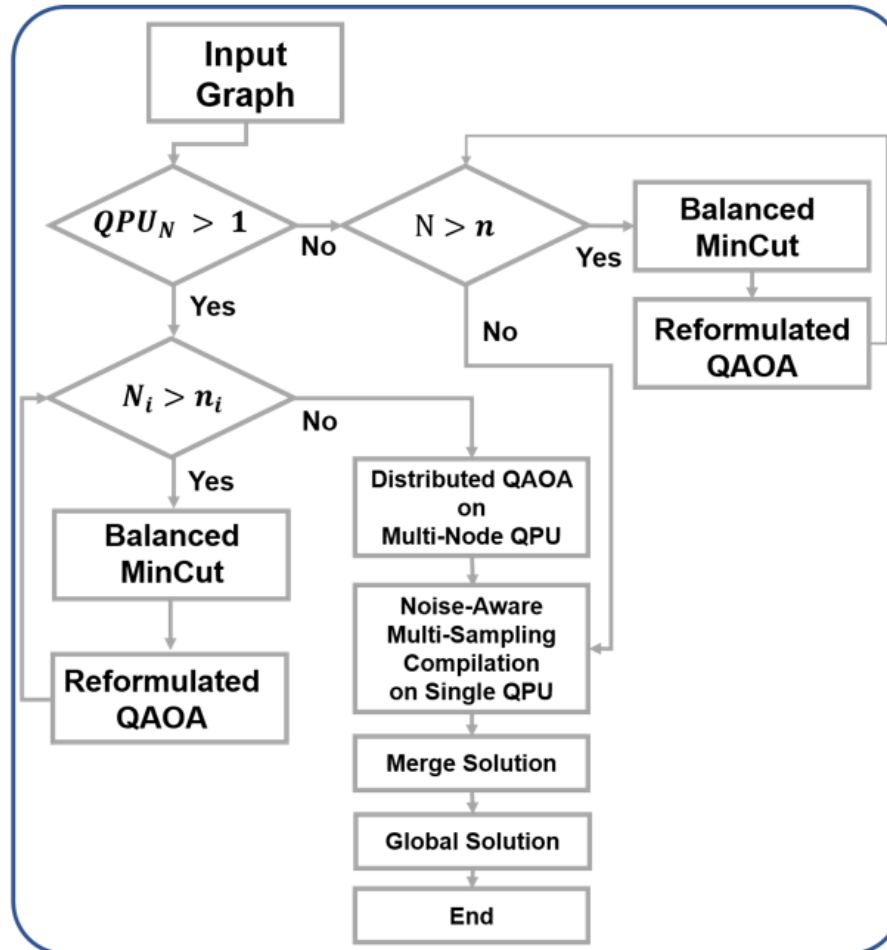
Some IBM Quantum QPU Topologies:



Chen et al., IEEE QCE 25'

QAOA Algorithms on Noisy Hardware

This Noise-Aware Distributed QAOA algorithm optimally partitions large graphs based on QPU capacity and node size, applying balanced MinCut (based on problem-informed formulation, for example city distribution) and reformulated QAOA strategies to ensure efficient multi-node execution with noise-aware compilation.



Algorithm 1 Noise-Aware Distributed QAOA

Require: Graph $G = (V, E)$, Threshold η , Node Capacity n_i ,
QPU Capacity QPU_N

Ensure: Global Solution S

1: **Procedure** Distributed-QAOA(G, η, n_i, QPU_N):

2: **if** $QPU_N > 1$ **then**

3: **if** $|V| > n_i$ **then**

4: Perform Balanced MinCut on G

5: **else**

6: Perform Reformulated QAOA on G

7: **end if**

8: **else**

9: **if** $|V| > \eta$ **then**

10: Perform Balanced MinCut on G

11: **else**

12: Perform Reformulated QAOA on G

13: **end if**

14: **end if**

15: **if** $|V| \leq \eta$ **then**

16: Execute Distributed QAOA on Multi-Node QPU

17: Apply Noise-Aware Multi-Sampling Compilation on Single QPU

18: Merge Partial Solutions from QAOA

19: Obtain Global Solution S

20: **end if**

21: **return** Global Solution S

Noise-Aware Distributed QAOA Compilation Strategy

Research Framework in this work:

A. Step 1: Threshold Filtering

The initial step in our noise-aware compilation strategy involves defining an error rate threshold η for both single-readout error rates and two-qubit gate error rates. Each qubit q_i with a single-readout error rate e_i and each two-qubit gate g_{ij} between qubits q_i and q_j with an error rate e_{ij} are evaluated against this threshold:

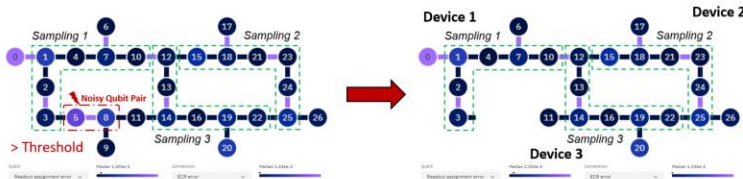
- q_i is valid if $e_i < \eta$
- g_{ij} is valid if $e_{ij} < \eta$

Using these criteria, we construct a subgraph G' from the original QPU graph G by retaining only the nodes and edges that meet the error rate criteria:

$$G' = (Q', E')$$

$$Q' = \{q_i \in Q \mid e_i < \eta\}$$

$$E' = \{g_{ij} \in E \mid e_{ij} < \eta\}$$

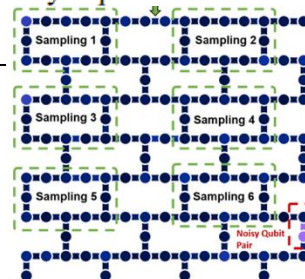


B. Step 2: Noise-Aware Symmetrical Sampling

For a QAOA problem requiring n qubits, we identify k symmetrical sampling areas. Each sampling area is a subgraph S_k containing n qubits. The number of symmetrical sampling areas k is determined by the number of valid qubits and gates in the filtered subgraph G' . To ensure optimal performance, we prioritize sampling areas with the highest fidelity. The fidelity of a block is defined as the product of the readout errors of the selected qubits and the two-qubit gate errors among them:

$$\text{Fidelity} = \prod_{i=1}^n e_i \times \prod_{j=1}^m e_{ij}$$

We evaluate the fidelity for each potential sampling area and select k subgraphs S_1, S_2, \dots, S_k with the highest fidelity, ensuring each subgraph S_i contains n qubits. Each subgraph S_i maintains symmetry and is topologically equivalent to the other subgraphs.



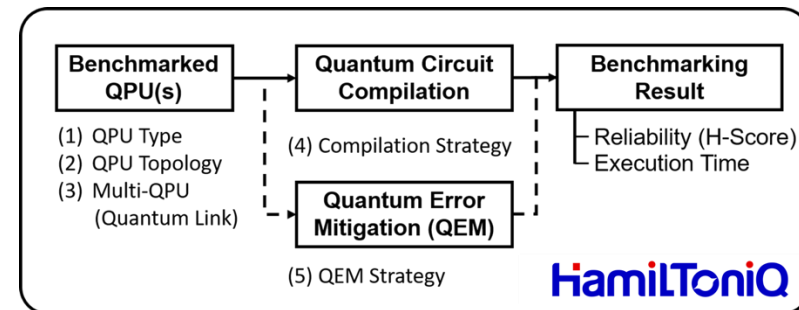
C. Step 3: Compilation

For each symmetrical sampling area S_i :

- 1) Map the QAOA circuit to the n qubits in S_i .
- 2) Execute the QAOA circuit on each S_i independently.
- 3) Collect results from all sampling areas to improve the robustness and reliability of the computations.



Evaluation

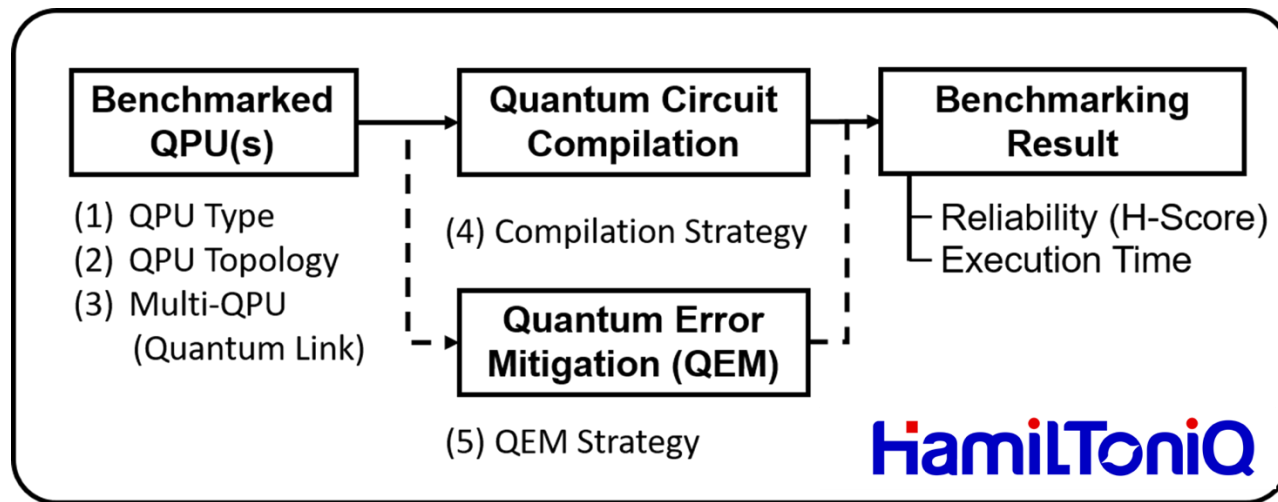


Chen et al.,
IEEE QCE 25'

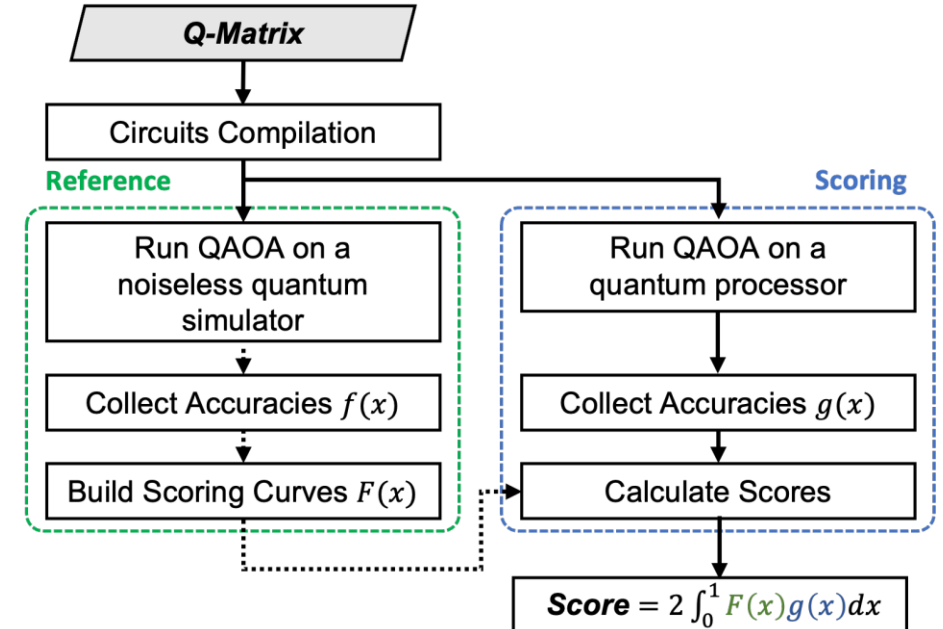
HamilToniQ: Application-Oriented QPU Benchmarking Toolkit

- Include all factors (QPU, compilation, QEM, ...).
- H-Scores are comparable.
- Automatic, easy to use.

Workflow of HamilToniQ



How to calculate H-Score:

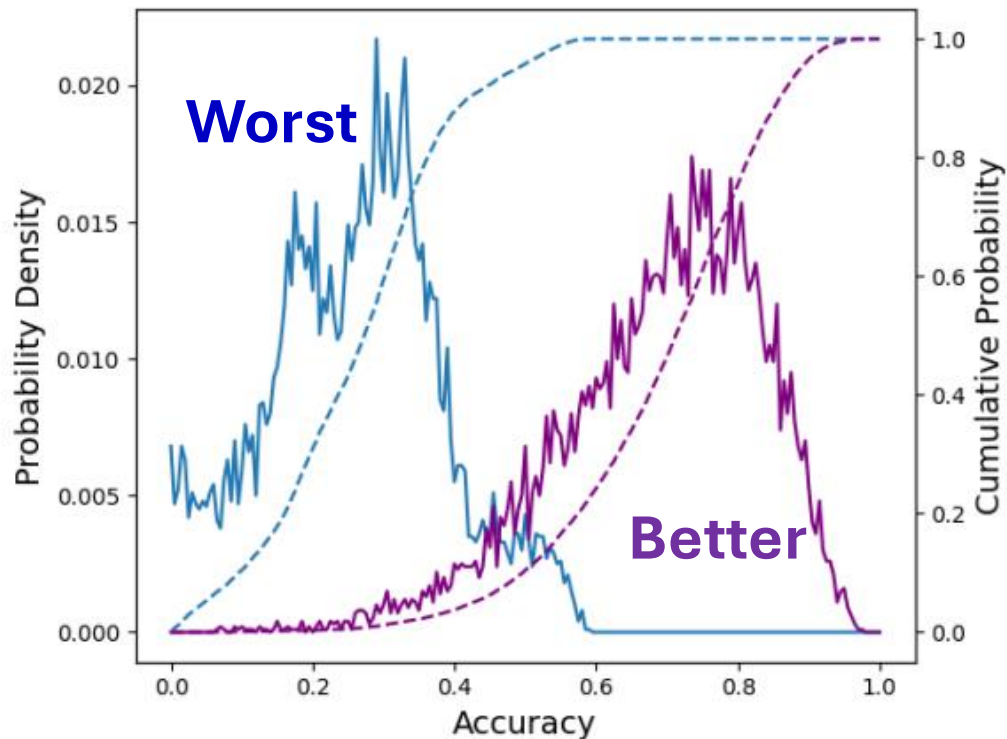


Xu, Xiaotian, Kuan-Cheng Chen, and Robert Wille. "HamilToniQ: An Open-Source Benchmark Toolkit for Quantum Computers." *IEEE QCE'24*

HamilToniQ: Application-Oriented QPU Benchmarking Toolkit

Math Behind the H-Score:

$$C = 2 \int_0^1 F(x) f(x) dx + 2 \int_0^1 F(x) \psi(x) dx$$



ibm_cairo	0.96	0.81	0.69	0.51	0.37	0.30	0.25	0.21	0.17
ibm_lagos	0.94	0.85	0.72	0.57	0.44	0.35	0.29	0.23	0.19
ibm_perth	0.94	0.78	0.65	0.46	0.34	0.27	0.23	0.20	0.16
ibm_nairobi	0.94	0.74	0.63	0.44	0.35	0.26	0.22	0.19	0.16
ibm_guadalupe	0.94	0.70	0.58	0.42	0.33	0.27	0.23	0.20	0.17
ibm_auckland	0.92	0.73	0.58	0.43	0.32	0.26	0.23	0.19	0.16
ibm_athens	0.88	0.63	0.46	0.36	0.30	0.25	0.23	0.20	0.18
ibm_burlington	0.86	0.59	0.44	0.32	0.27	0.23	0.21	0.18	0.16
	1	2	3	4	5	6	7	8	9
	n_layers								

Benchmarking results on several IBM QPUs with 5 qubits.

Xu, Xiaotian, Kuan-Cheng Chen, and Robert Wille. "HamilToniQ: An Open-Source Benchmark Toolkit for Quantum Computers." *IEEE QCE'24 Submitted*

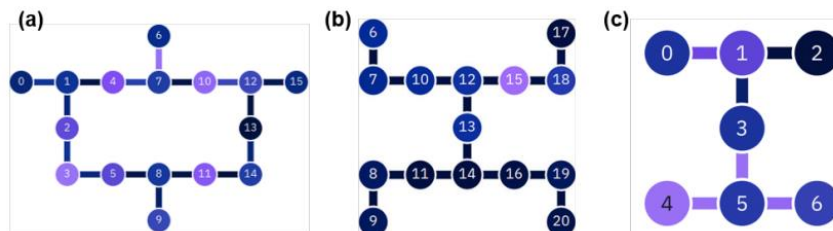
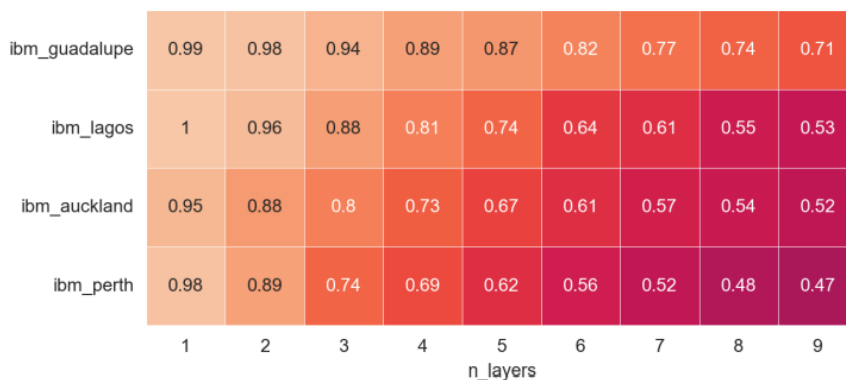
HamilToniQ: Application-Oriented QPU Benchmarking Toolkit

What HamilToniQ Toolkit can benchmark?

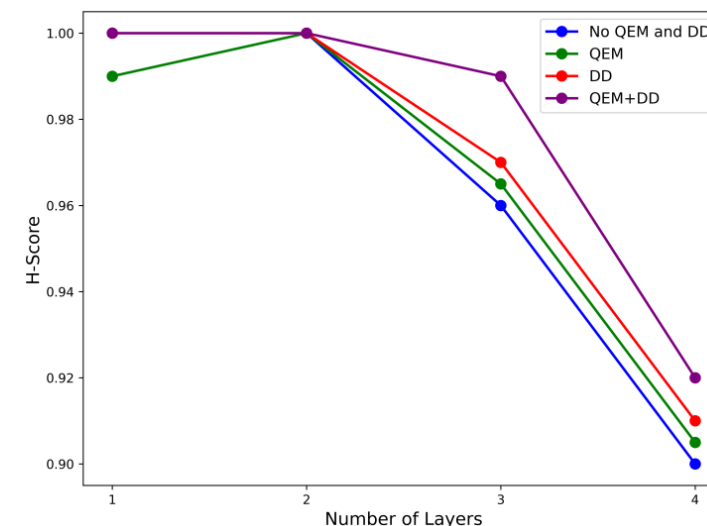
(1) QPUs on quantum cloud server



(2) QPU Topology



(3) Quantum Error Mitigation

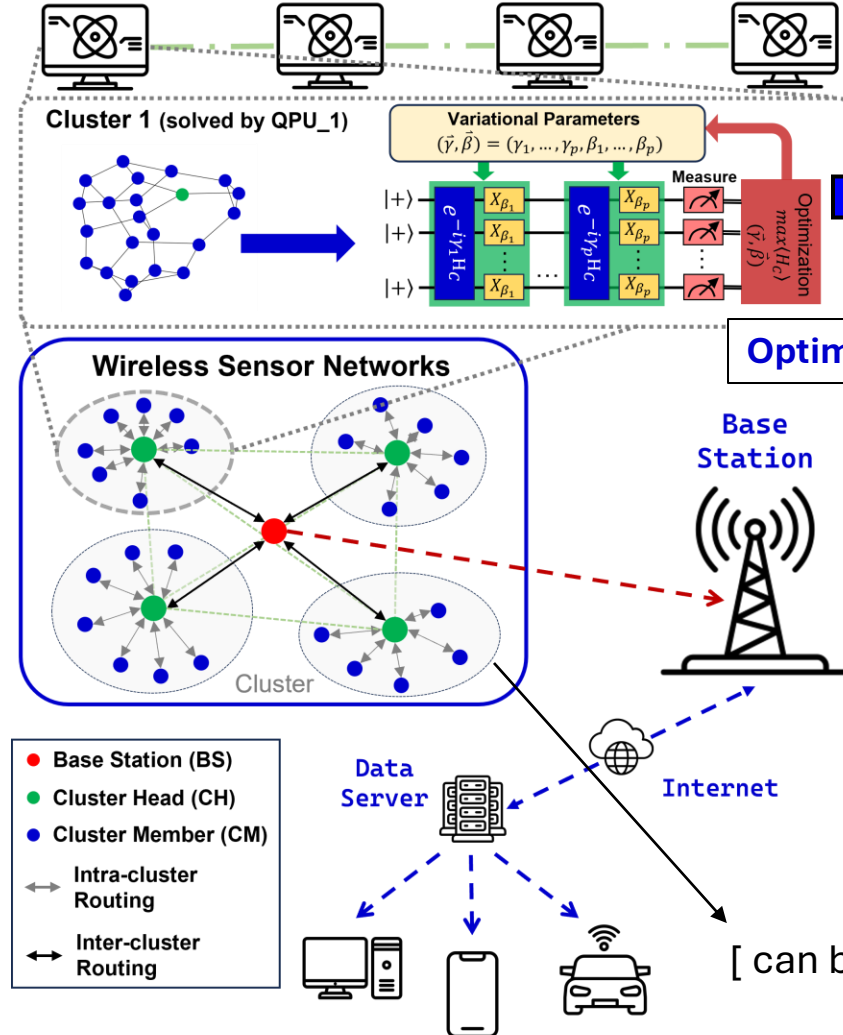


Chen et al.,
IEEE QCE 25'

Xu, Xiaotian, Kuan-Cheng Chen, and Robert Wille. "HamilToniQ: An Open-Source Benchmark Toolkit for Quantum Computers." *IEEE QCE'24 Submitted*

Task Partition Perspective –

Application to solve large-scale wireless communication network problems



To ensure flow conservation, the following constraint is imposed for each node $i \in V_s$:

$$\sum_{j:(i,j) \in E_s} x_{ij} - \sum_{j:(j,i) \in E_s} x_{ji} = b_i, \quad (4)$$

where b_i is the net flow at node i :

$$b_i = \begin{cases} 1 & \text{if } r_i = \text{Sensor,} \\ 0 & \text{if } r_i = \text{CH,} \\ -\sum_{i \in S} b_i & \text{if } r_i = \text{BS.} \end{cases}$$

Energy constraints are enforced to ensure that the energy consumed by a node does not exceed its initial energy E_i :

$$\sum_{j:(i,j) \in E_s} c_{ij} x_{ij} \leq E_i. \quad (5)$$

These constraints are incorporated into the QUBO objective using penalty terms, ensuring that feasible solutions satisfy both flow conservation and energy limitations.

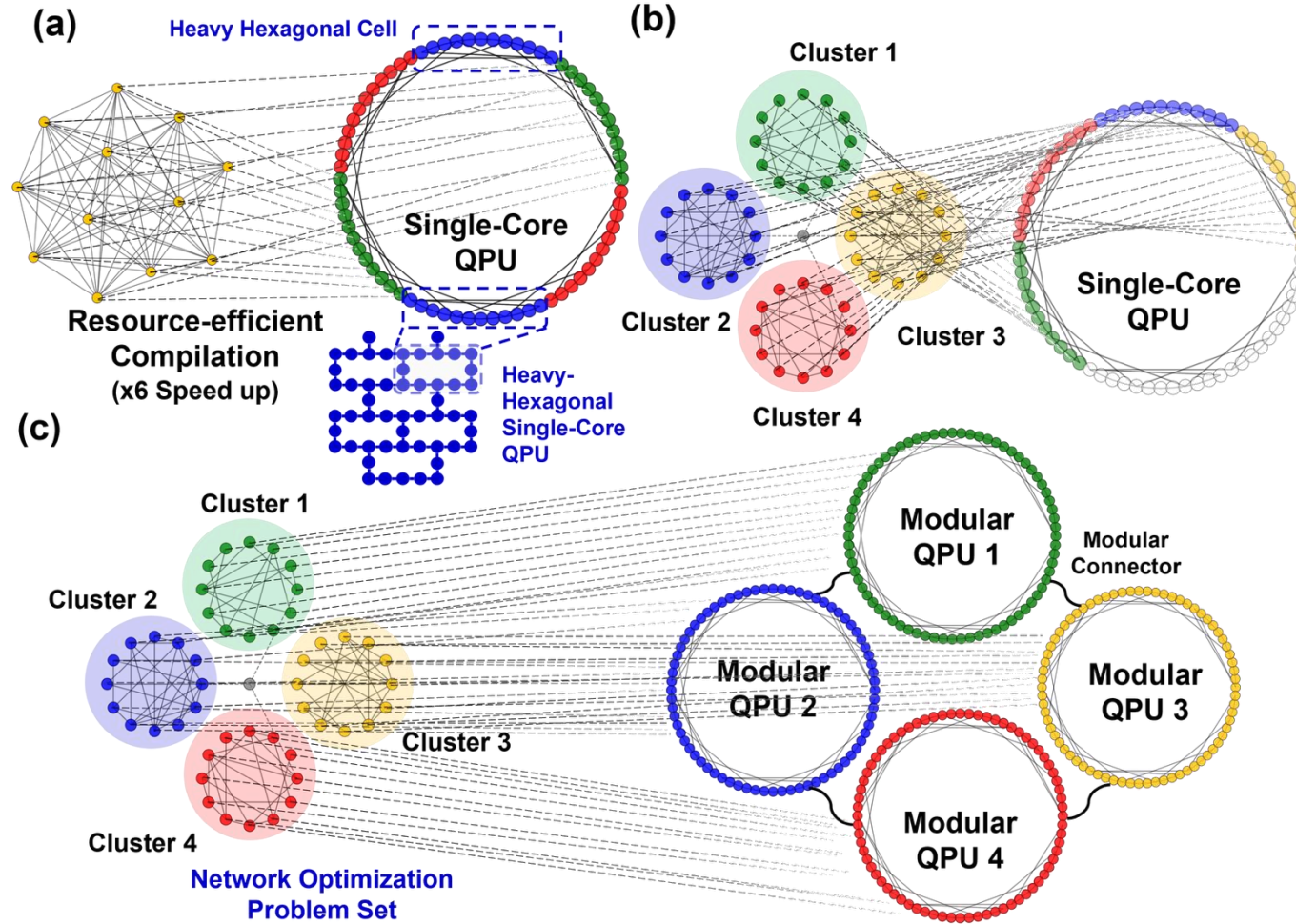
Quadratic unconstrained binary optimization (QUBO) Formulation:

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j) \in E_s} c_{ij} x_{ij} \\ & + \lambda_{\text{flow}} \sum_{i \in V_s} \left(\sum_{j:(i,j) \in E_s} x_{ij} - \sum_{j:(j,i) \in E_s} x_{ji} - b_i \right)^2 \\ & + \lambda_{\text{energy}} \sum_{i \in V_s} \left(\sum_{j:(i,j) \in E_s} c_{ij} x_{ij} - E_i \right)^2. \end{aligned}$$

[can be classically partitioning the sub-QUBO formulation by breadth-first search (BFS)]

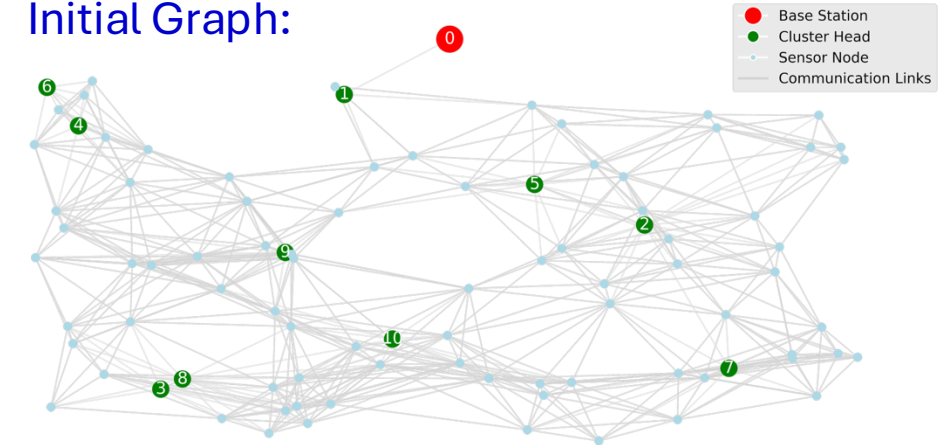
Solving Large-Scale Wireless Communication Network Problems

Compilation Strategy for Dist-QAOA:

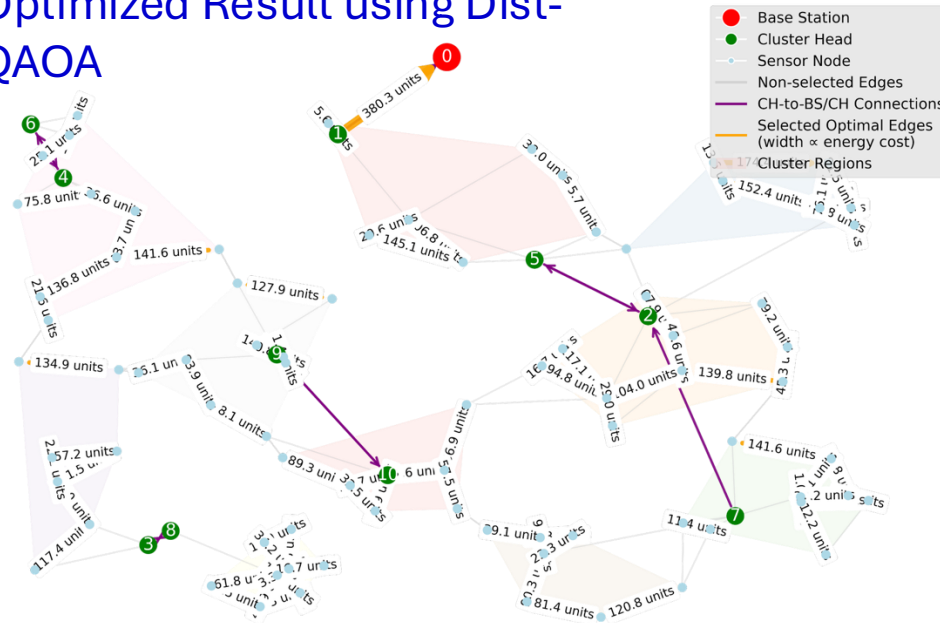


Example for Distributed QAOA problem benchmarking

Initial Graph:



Optimized Result using Dist-QAOA



Greedy Search: -70%

Dist-QAOA: -83%

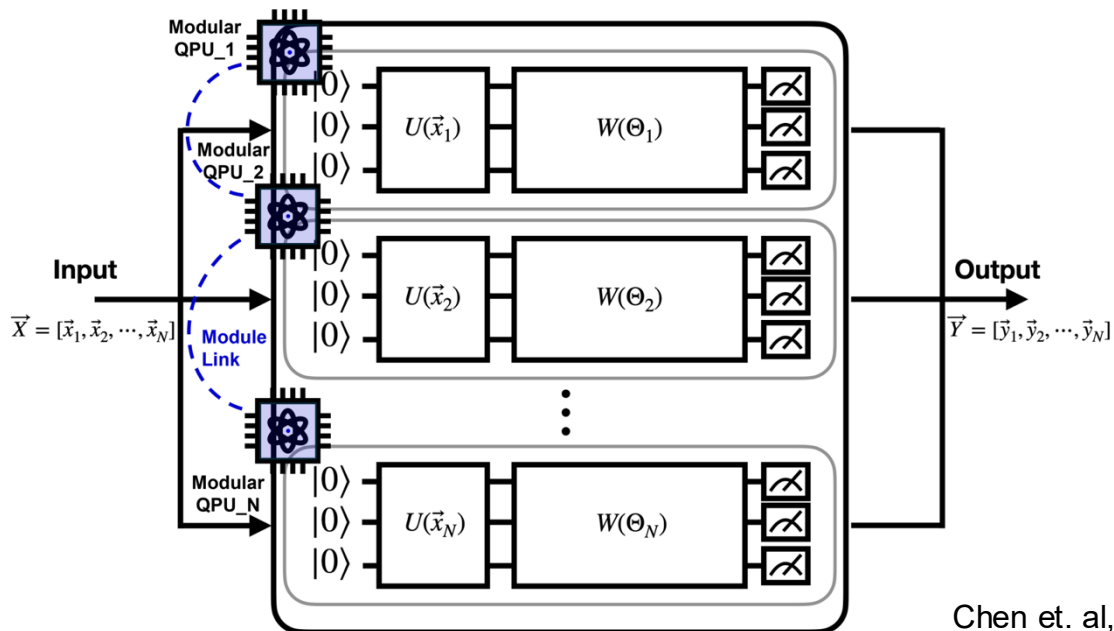
Distributed Quantum Long Short-Term Memory

Contribution

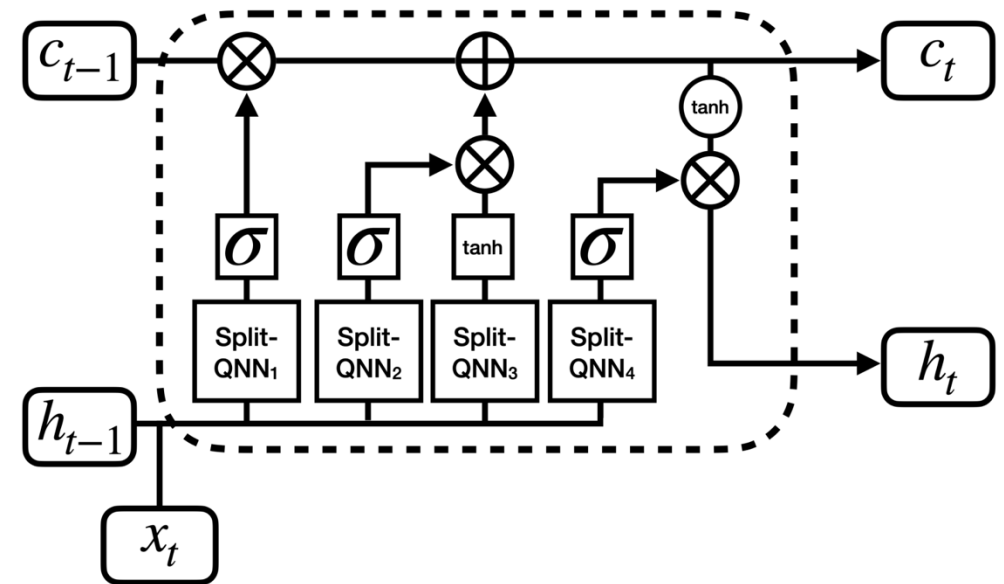
We introduce a Distributed Quantum Long Short-Term Memory (QLSTM) architecture that partitions variational quantum circuits across modular QPUs. This enables scalable, parallel execution of quantum-enhanced LSTM gates while preserving temporal modeling capacity.

Framework

The proposed system decomposes input vectors into subcomponents, each processed by a separate quantum module (QPU) executing a variational quantum circuit. These modular outputs are integrated into a QLSTM cell composed of quantum neural gate blocks (Split-QNNs), supporting distributed learning of sequential dynamics.



Chen et. al, IWCMC 2025



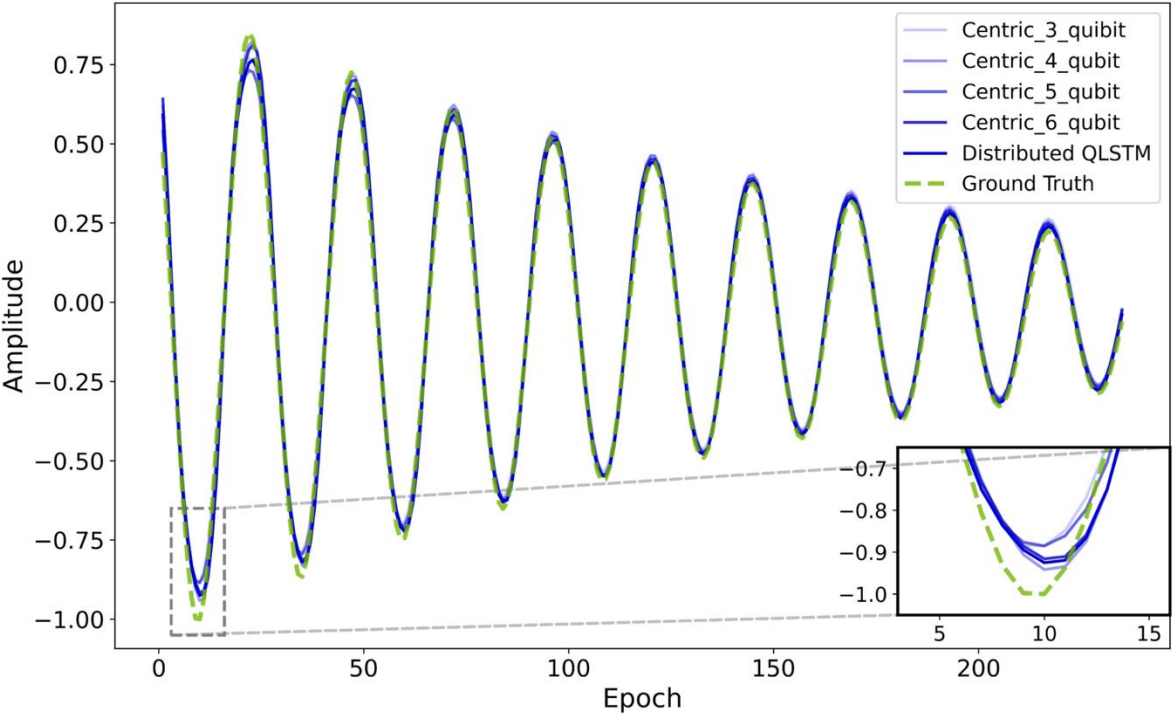
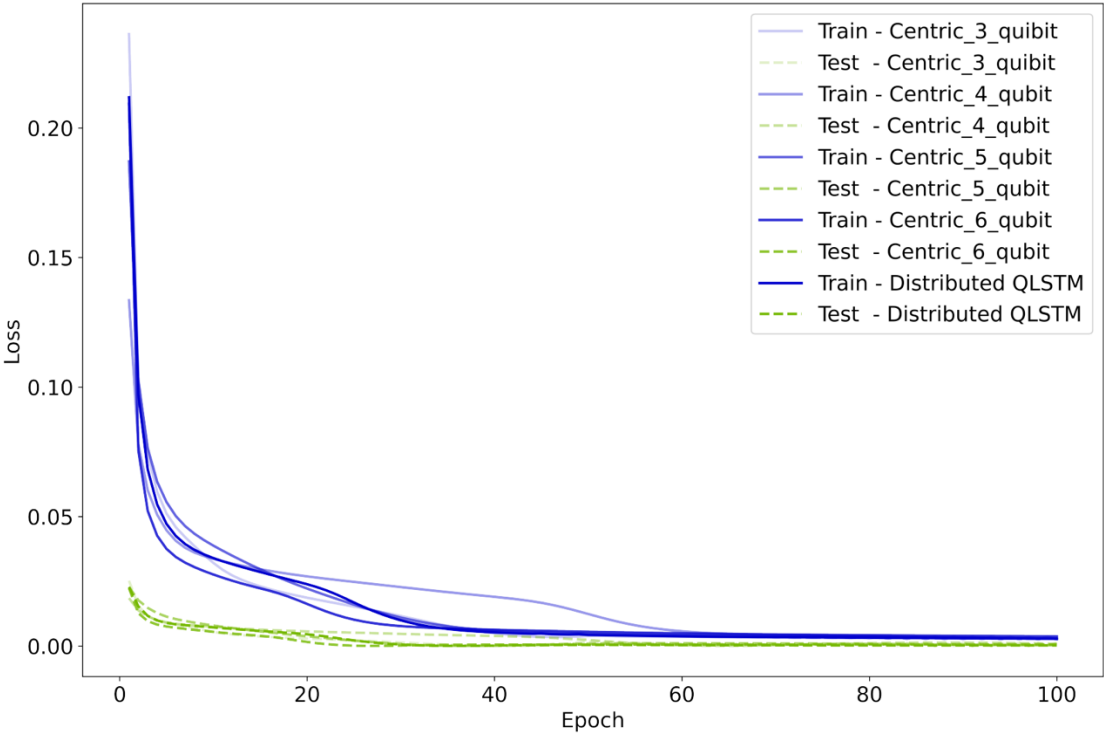
Result Analysis

Damped Harmonic Oscillator

We focus on a damped pendulum with parameters chosen to yield nontrivial oscillatory decay:

$$\frac{d^2\theta}{dt^2} + \frac{b}{m} \frac{d\theta}{dt} + \frac{g}{L} \sin(\theta) = 0.$$

Model	R-square	Testing Epoch Convergence
Centric QLSTM (3 qubit)	0.9898	81
Centric QLSTM (4 qubit)	0.9913	68
Centric QLSTM (5 qubit)	0.9938	38
Centric QLSTM (6 qubit)	0.9903	27
Distributed QLSTM	0.9936	31



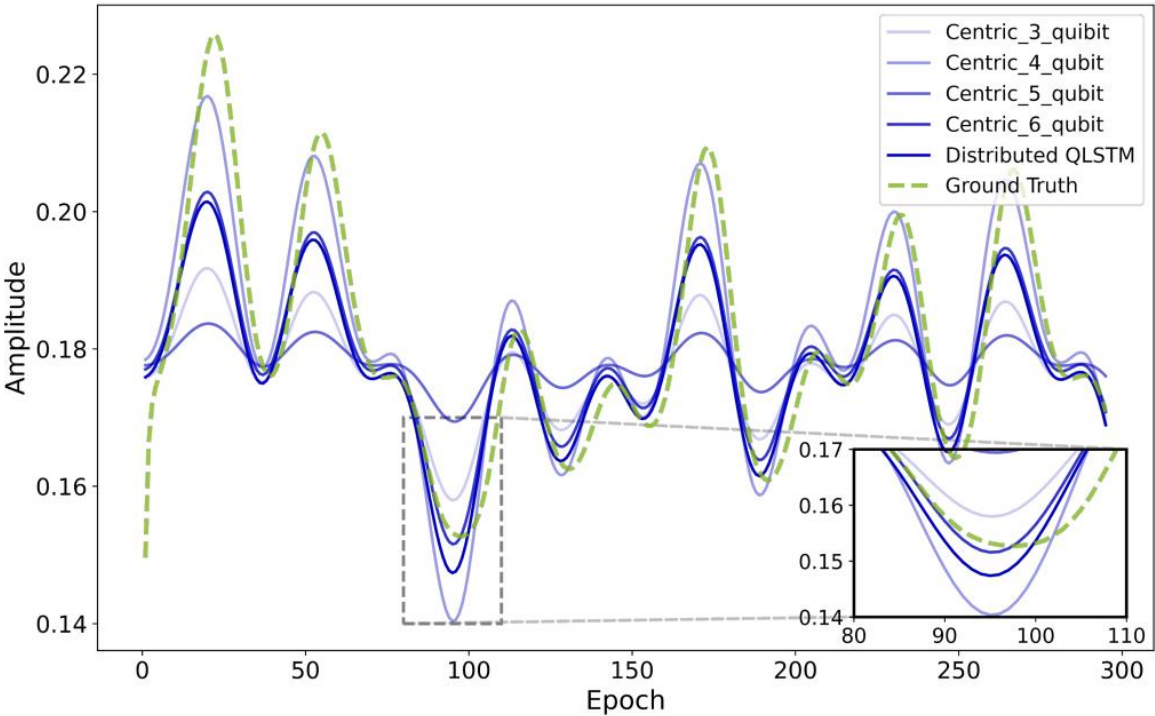
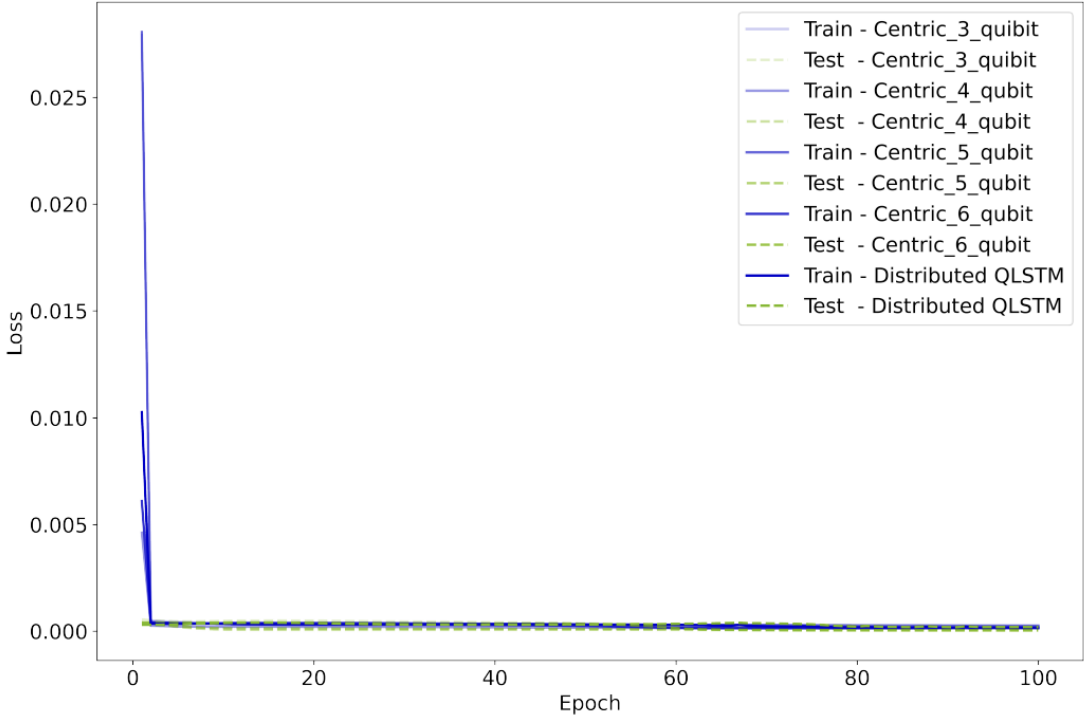
Result Analysis

Nonlinear Autoregressive Moving Average Sequences (NARMA)

A representative example is the NARMA-2 relation:

$$y_{t+1} = 0.4 y_t + 0.4 y_t y_{t-1} + 0.6 u_t^3 + 0.1$$

Model	R-square	Testing Epoch Convergence
Centric QLSTM (3 qubit)	0.5477	2
Centric QLSTM (4 qubit)	0.8611	2
Centric QLSTM (5 qubit)	0.2651	2
Centric QLSTM (6 qubit)	0.7799	2
Distributed QLSTM	0.7523	2













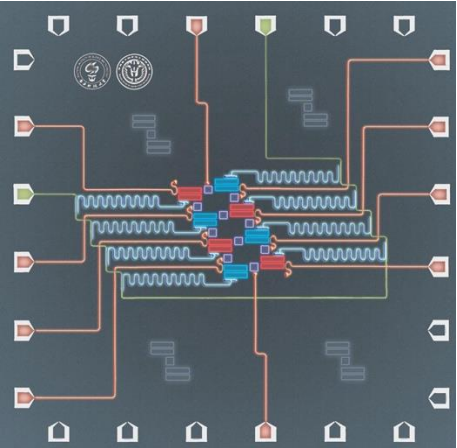
Distributed Photonic Quantum Computing

Hybrid Quantum Machine Learning

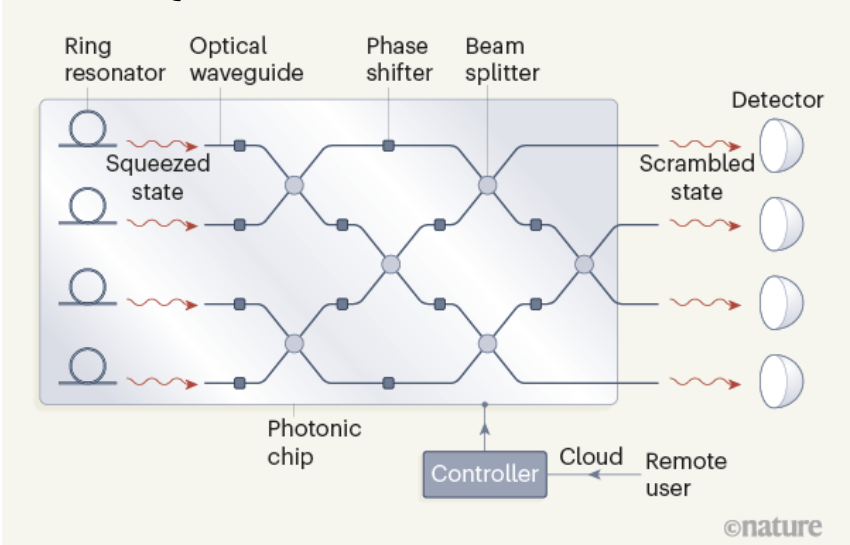
- Different between Continuous Variable (CV) and Qubit

	CV	Qubit
Basic element	Qumodes	Qubits
Relevant operators	Quadratures \hat{x}, \hat{p} Mode operators \hat{a}, \hat{a}^\dagger	Pauli operators $\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z$
Common states	Coherent states $ \alpha\rangle$ Squeezed states $ z\rangle$ Number states $ n\rangle$	Pauli eigenstates $ 0/1\rangle, \pm\rangle, \pm i\rangle$
Common gates	Rotation, Displacement, Squeezing, Beamsplitter, Cubic Phase	Phase shift, Hadamard, CNOT, T-Gate
Common measurements	Homodyne $ x_\phi\rangle\langle x_\phi $, Heterodyne $\frac{1}{\pi} \alpha\rangle\langle\alpha $, Photon-counting $ n\rangle\langle n $	Pauli eigenstates $ 0/1\rangle\langle 0/1 , \pm\rangle\langle\pm , \pm i\rangle\langle\pm i $

Operator	Gate(s)	Matrix
Pauli-X (X)	 \oplus	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$



Photonic QC:

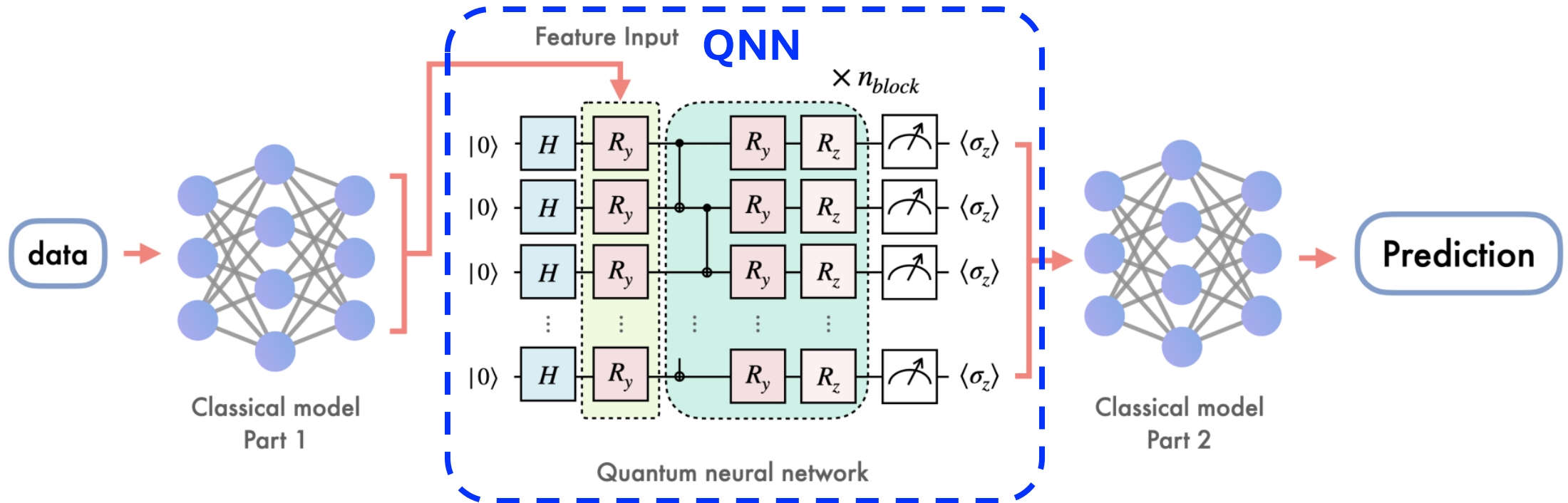


Hybrid Quantum Machine Learning

Feature	Continuous-Variable (CV)	Discrete-Variable (DV) Qubits
Encoding Type	Analog quadratures (e.g., light field amplitudes)	Binary superpositions of
Hilbert Space	Infinite-dimensional	Finite-dimensional
Scalability	High via multiplexing (e.g., time/frequency modes)	Challenging due to crosstalk and decoherence
HW Requirements	Operates at room temperature; uses standard optical components	Requires cryogenic temperatures and specialized hardware
Noise Resilience	Inherent robustness; Gaussian states offer some protection	Susceptible to decoherence; necessitates complex error correction
Error Correction	Emerging schemes (e.g., GKP codes); still under development	Established protocols (e.g., surface codes, qLDPC); resource-intensive
Algorithm Maturity	Fewer algorithms; active area of research	Extensive algorithm library; well-established frameworks
Measurement	Homodyne/heterodyne detection (Gaussian); PNRD (non-Gaussian); continuous outcomes	Projective measurements; discrete outcomes
Use Cases	Quantum sensing, communication, certain machine learning applications	Broad applications including cryptography, simulation, and general-purpose computing

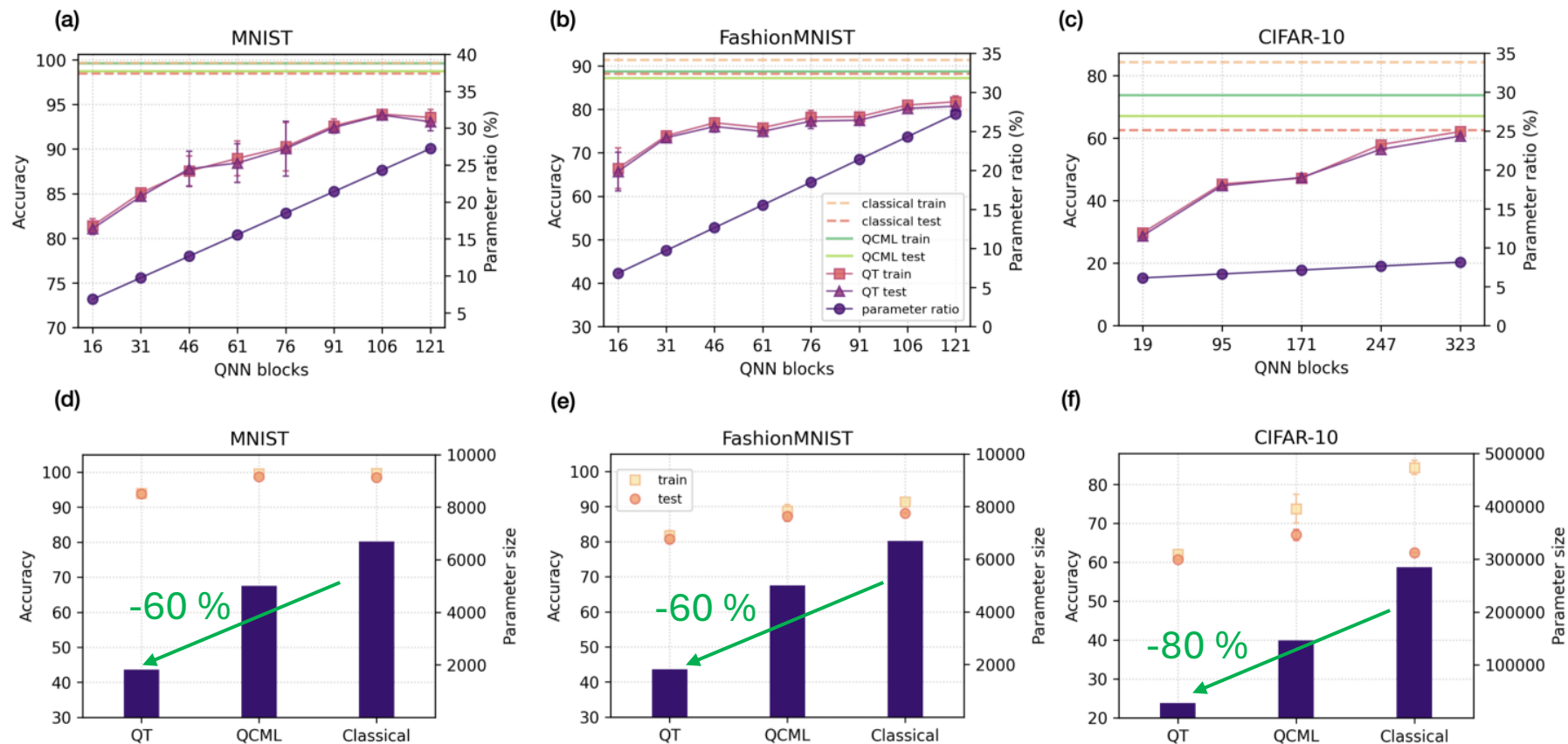
Hybrid Quantum Machine Learning

- Rethinking Hybrid Quantum-Classical Machine Learning in the Model Compression Perspective



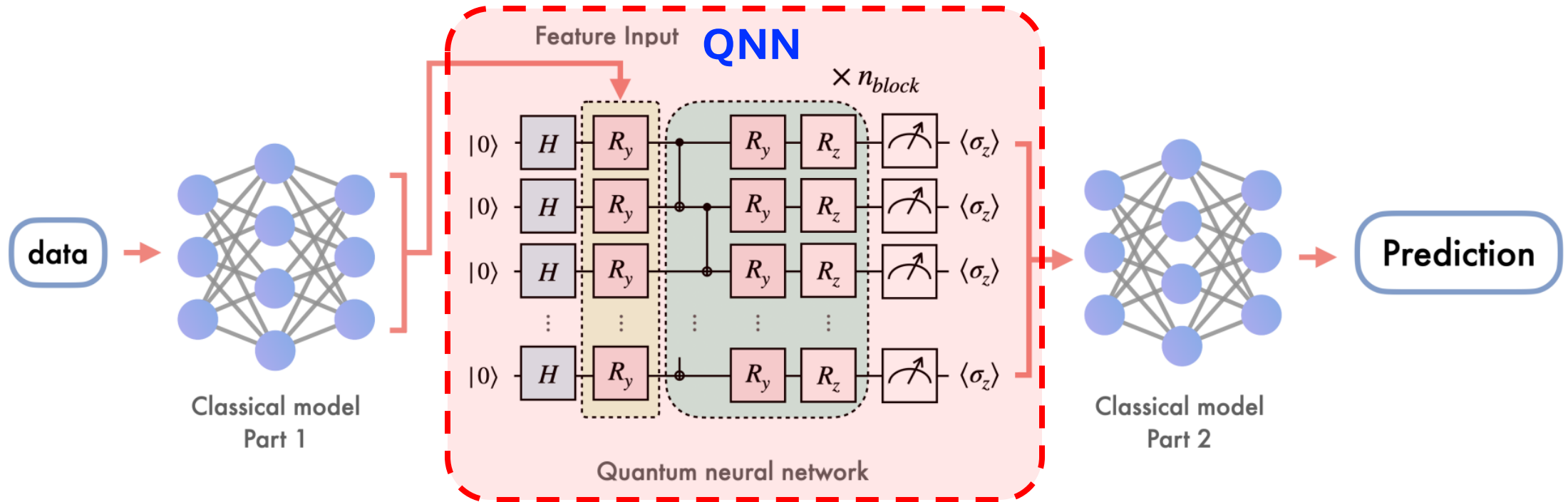
Hybrid Quantum Machine Learning

- Rethinking Hybrid Quantum-Classical Machine Learning in the Model Compression Perspective



Hybrid Quantum Machine Learning

- Rethinking Hybrid Quantum-Classical Machine Learning in the Model Compression Perspective

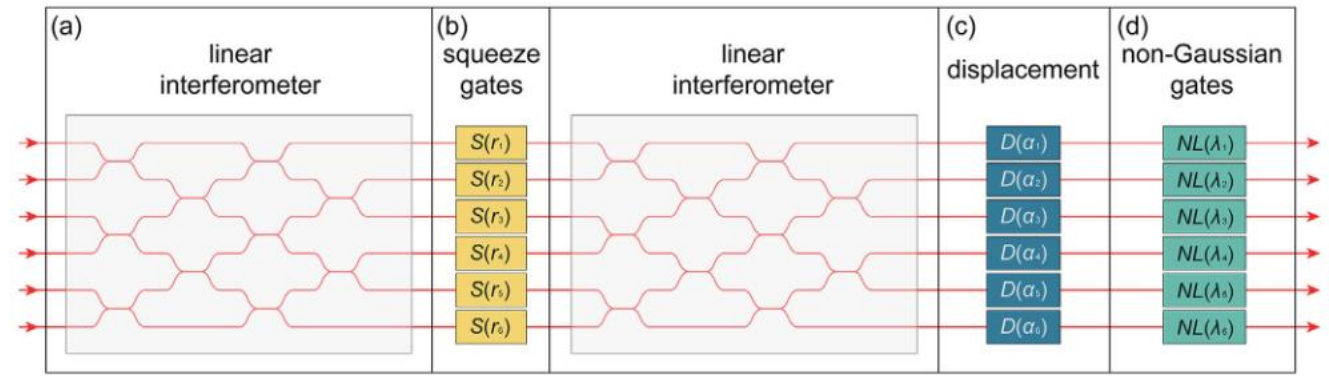
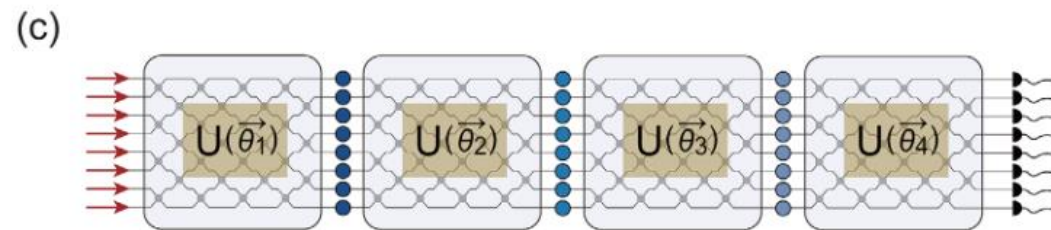
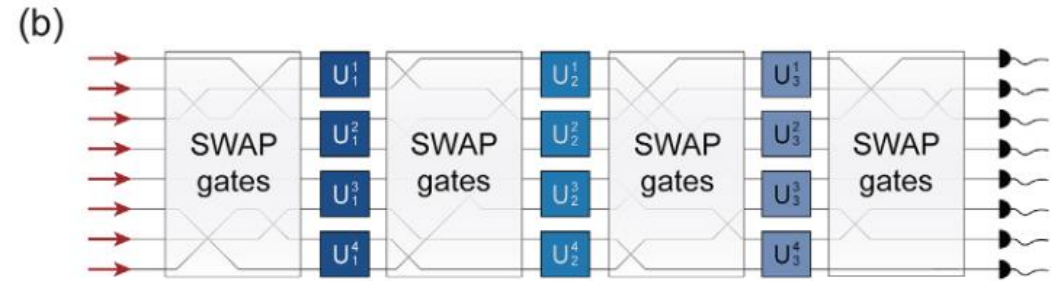
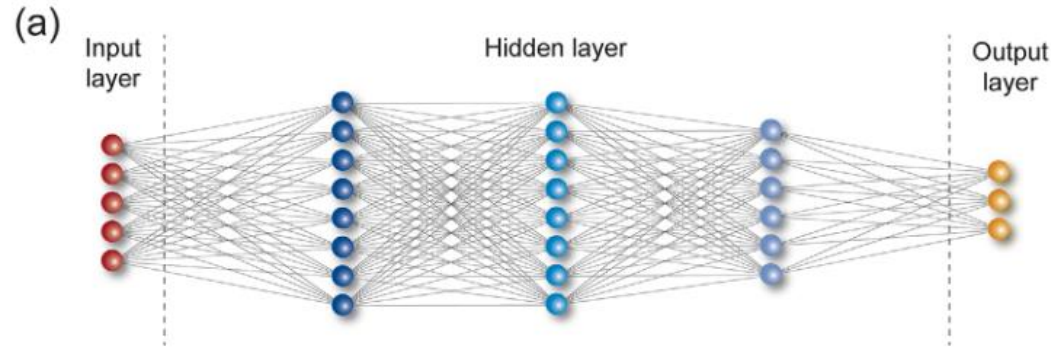


Problems:

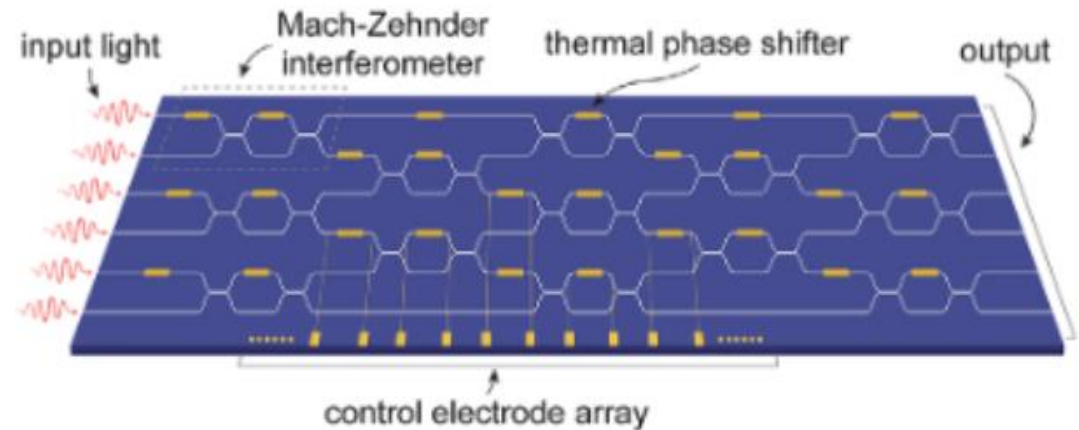
(1) Partition Cost; (2) Large Noise in Communication Channel

Hybrid Quantum Machine Learning with Photonic QC

Photonic QNNs integrate quantum mechanics and optical systems to enable scalable, noise-resilient machine learning for quantum and classical data



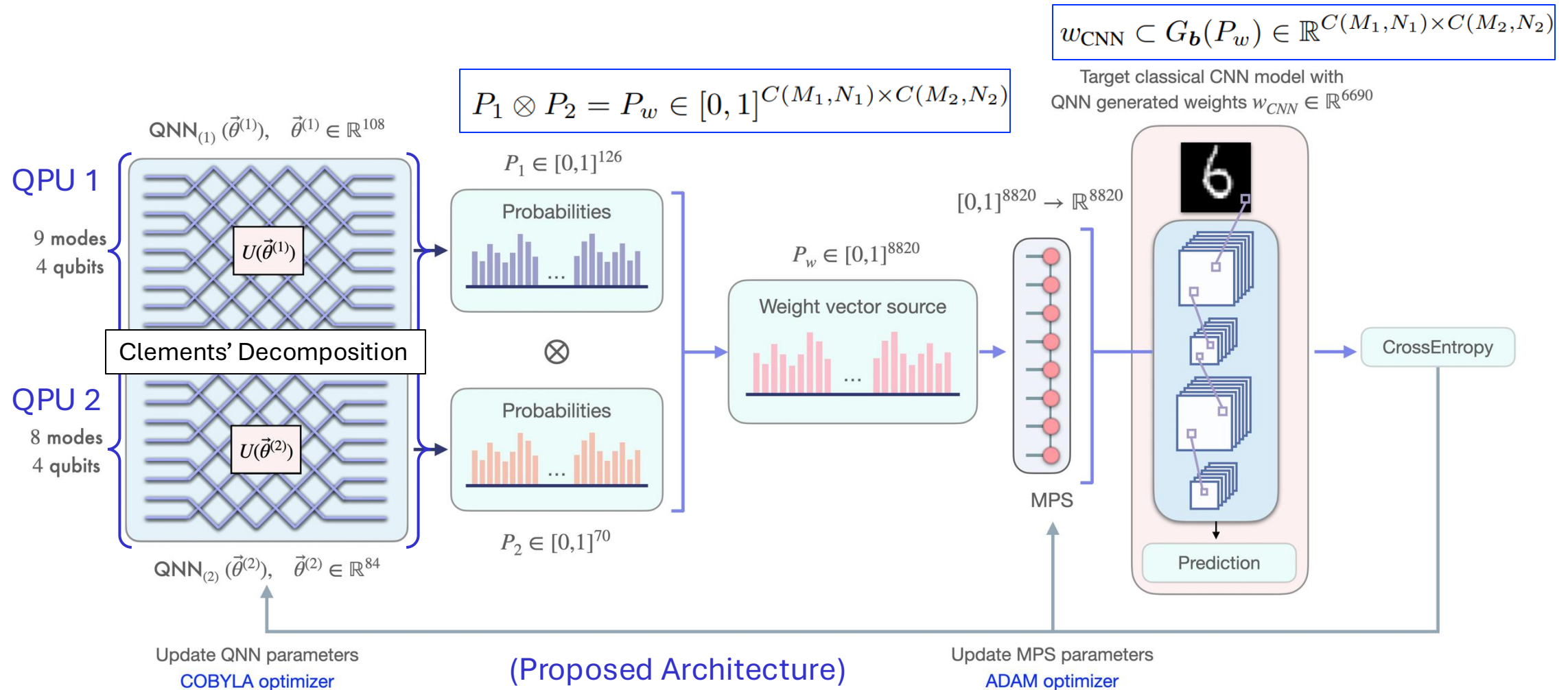
Photonic Chip:



Yu, Shang, et al. "Shedding light on the future: Exploring quantum neural networks through optics." *Advanced Quantum Technologies* (2024): 2400074.

Preliminary: Photonic Quantum Neural Networks

- Distributed Photonic QNNs



Distributed-Aware and No Need Microwave-Optical Transduction

Preliminary: Photonic Quantum Neural Networks

Gradient Estimation of Photonic Quantum Circuit Compressed Parameters. The target NN parameters w_{CNN} are generated through the use of QNNs coupled with a mapping model. The quantum-dependent parameters, denoted as $(\vec{\theta}^{(i)}, \mathbf{b})$, influence the target NN parameters through the quantum state preparation and measurement steps. The gradient of the loss function, which captures the effect of the quantum parameters, is expressed as:

$$\nabla_{\vec{\theta}^{(i)}, \mathbf{b}} \mathcal{L} = \left(\frac{\partial w_{\text{CNN}}}{\partial (\vec{\theta}^{(i)}, \mathbf{b})} \right)^T \cdot \nabla_{w_{\text{CNN}}} \mathcal{L}. \quad (4)$$

Here, $\frac{\partial w_{\text{CNN}}}{\partial (\vec{\theta}^{(i)}, \mathbf{b})}$ represents the Jacobian matrix, which quantifies the sensitivity of the classical parameters w_{CNN} to variations in the quantum parameters $(\vec{\theta}^{(i)}, \mathbf{b})$.

Parameter Update of Photonic Quantum Circuit Compressed Parameters. The learning rate η is a critical factor, particularly given the complex dynamics introduced by the quantum-classical interface. The update rule for the quantum parameters is defined as:

$$\vec{\theta}_{t+1}^{(i)}, \mathbf{b}_{t+1} = \vec{\theta}_t^{(i)}, \mathbf{b}_t + \eta \nabla_{\vec{\theta}^{(i)}, \mathbf{b}} \mathcal{L}. \quad (5)$$

This update ensures that the quantum parameters are optimized to improve the performance of the target NN. The equation provides a high-level representation of the gradient update in an exact quantum state simulation. However, in practical applications using real quantum hardware or specific backend providers, the gradient calculation must incorporate the parameter shift rule and its variants (Mitarai et al., 2018; Schuld et al., 2019).

Algorithm 1 Photonic Quantum-Train Forward Pass

Require: Input tensor $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ (height \times width \times channels), PQC parameters $\theta \in \mathbb{R}^d$, Photonic modes $\{M_k\}_{k=1}^K$, Measurement budget N_{samp} , CNN template $\mathcal{S} = \{(n_j, \mathbf{s}_j)\}$ (param counts \times shapes)

Ensure: Class logits $\mathbf{y} \in \mathbb{R}^K$

PHOTONIC PARAMETER GENERATION

```
1:  $\{\theta^{(k)}\}_{k=1}^K \leftarrow \text{split}(\theta)$   $\triangleright$  Clements-decomposed MZI meshes
2: for  $k \in \{1, \dots, K\}$  do  $\triangleright$  Parallel QNN execution
3:    $\mathbf{p}^{(k)} \leftarrow \text{norm}(\text{QNN}_k(\theta^{(k)}, N_{\text{samp}}))$   $\triangleright$ 
    $C(M_k, N_k)$ -dim probabilities per Eq. (1)
```

```
4: end for
5:  $\mathbf{P}_w \leftarrow \text{vec}(\bigotimes_{k=1}^K \mathbf{p}^{(k)})_{1:m}$   $\triangleright$  Truncated to  $m$  params via Eq. (1)
```

QUANTUM-CLASSICAL MAPPING

```
6:  $\mathbf{v} \leftarrow \text{MPS}(\mathbf{P}_w; \chi)$   $\triangleright G_b$  with  $\chi$  from Table I
7:  $\mathbf{v} \leftarrow \mathbf{v} - \mu_{\mathbf{v}}$   $\triangleright$  Centering
```

PARAMETER ALLOCATION

```
8:  $\mathcal{W} \leftarrow \{\}; c \leftarrow 0$ 
9: for  $(n_j, \mathbf{s}_j) \in \mathcal{S}$  do
10:    $\mathcal{W}[j] \leftarrow \text{reshape}(\mathbf{v}[c:c+n_j], \mathbf{s}_j)$   $\triangleright$  Param slicing with  $c \leq m$ 
11:    $c \leftarrow c + n_j$ 
12: end for
```

CLASSICAL INFERENCE

```
13:  $\mathbf{y} \leftarrow \text{Sequential} ($ 
14:   Conv2D( $\mathcal{W}[1]$ ), MaxPool,
15:   Conv2D( $\mathcal{W}[2]$ ), AvgPool,
16:   Flatten, Linear( $\mathcal{W}[3]$ ),
17:   ReLU, Linear( $\mathcal{W}[4]$ ))( $\mathbf{x}$ )
18: return  $\mathbf{y}$ 
```

Configuration of
mapping model

IMPERIAL

Hyperparameter	Meaning	Value
Input size	Input of the mapping model ($ \phi_i\rangle, \langle\phi_i \psi(\vec{\theta}^{(i)})\rangle ^2$) (Liu et al., 2024c)	$\lceil \log_2 m \rceil + 1$
Bond dimension	Main structure parameter of the MPS mapping model	$1 \sim 10$

Result

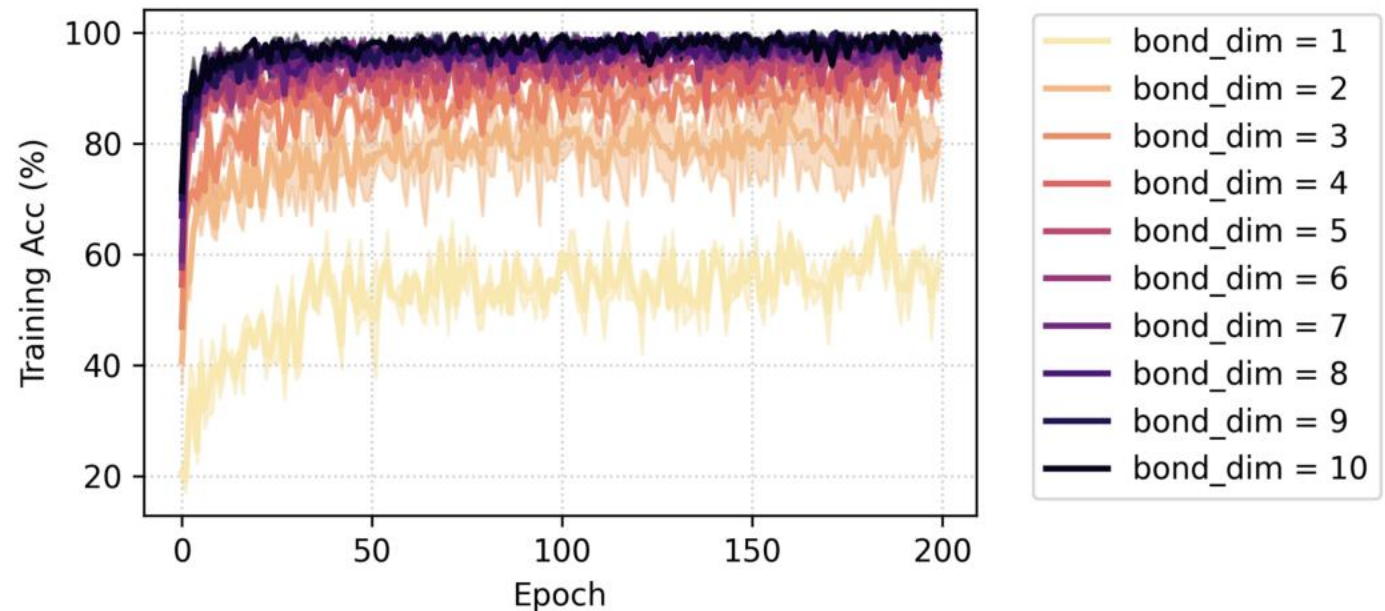
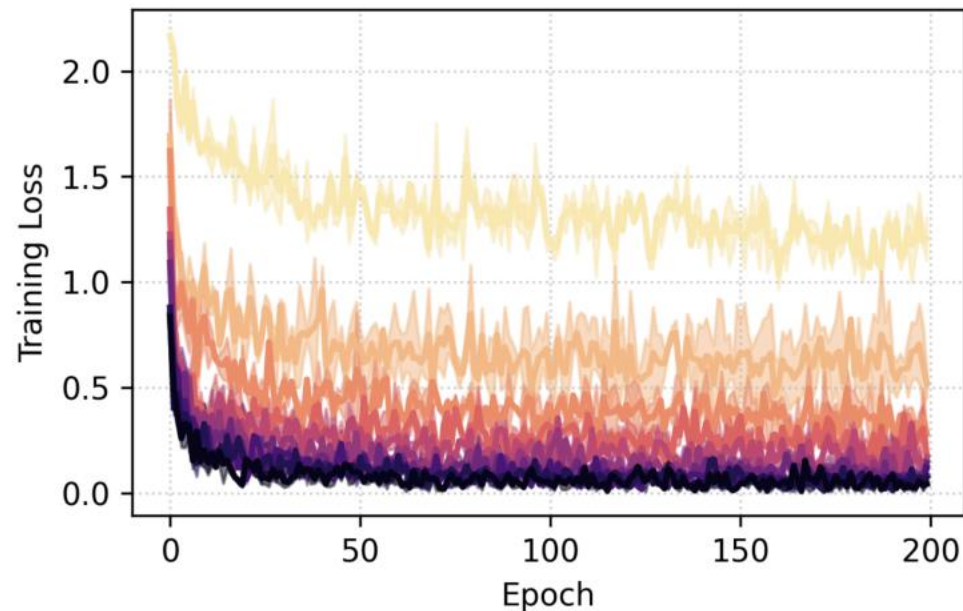
Benchmark: Classical Model (Baseline)

# of training parameters	Training accuracy (%)	Testing accuracy (%)	Generalization error
6690	99.983 ± 0.02	96.890 ± 0.31	0.1690 ± 0.005

Photonic QT with different bond dimensions

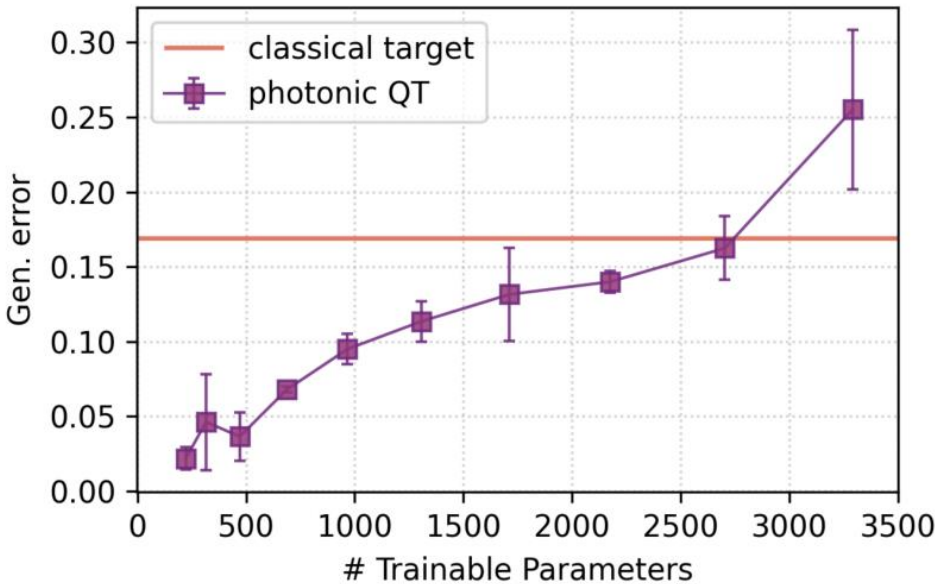
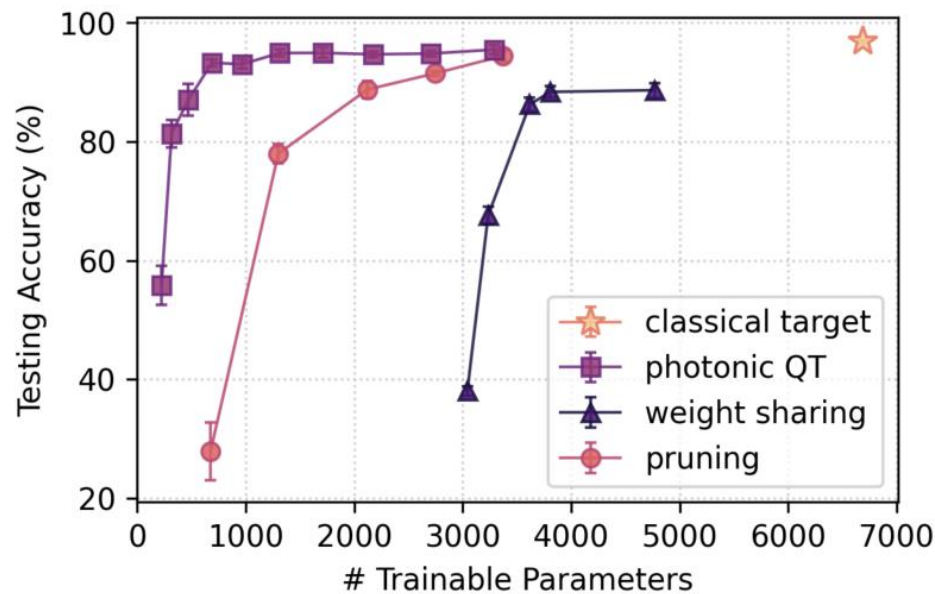
Bond dimension	# of training parameters	Training accuracy (%)	Testing accuracy (%)	Generalization error
1	223	58.256 ± 2.34	55.775 ± 3.27	0.0219 ± 0.007
2	316	83.340 ± 2.77	81.375 ± 2.28	0.0462 ± 0.032
3	471	88.693 ± 1.67	87.057 ± 2.66	0.0364 ± 0.016
4	688	93.916 ± 0.45	93.292 ± 0.62	0.0679 ± 0.002
5	967	95.450 ± 0.39	93.042 ± 0.77	0.0950 ± 0.010
6	1308	96.953 ± 0.02	94.917 ± 0.60	0.1135 ± 0.013
7	1711	97.773 ± 0.22	94.957 ± 0.82	0.1315 ± 0.031
8	2176	97.866 ± 0.78	94.707 ± 0.47	0.1399 ± 0.007
9	2703	98.373 ± 0.12	94.835 ± 0.48	0.1624 ± 0.021
10	3292	98.990 ± 0.34	95.502 ± 0.84	0.2552 ± 0.053

10 times compression



Result

Method	# of training parameters	Testing accuracy (%)
Original	6690	96.890 \pm 0.31
Weight sharing	4770	88.666 \pm 1.207
Pruning	3370	94.443 \pm 0.923
Photonic QT (bond dimension = 10)	3292	95.502 \pm 0.84
Photonic QT (bond dimension = 4)	688	93.292 \pm 0.62
Gate-based QT	1365	92.172 \pm 0.35

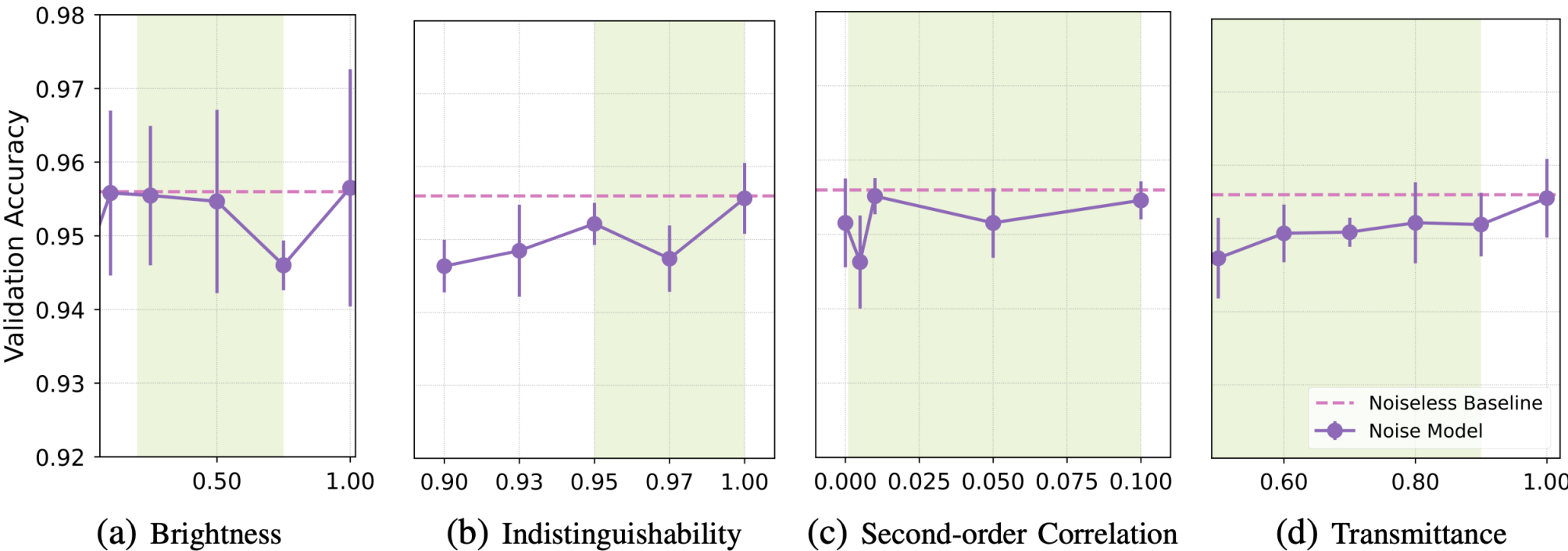


Noise Resilience

TABLE V

KEY HARDWARE PARAMETERS FOR STATE-OF-THE-ART SINGLE-PHOTON SOURCES AND INTEGRATED PHOTONIC CIRCUITS.

Symbol	Figure of merit	Physical meaning	Typical value
β	Brightness	Single-photon emission probability per clock cycle	0.2–0.75 [59]
I	Indistinguishability	Wave-packet overlap (HOM visibility)	> 0.95 [60]
$g^{(2)}(0)$	Second-order correlation	Multiphoton emission probability	10^{-1} – 10^{-3} [61]
T	Transmittance	Probability that a photon survives propagation, coupling and detection	0.4–0.9 [62], [63]

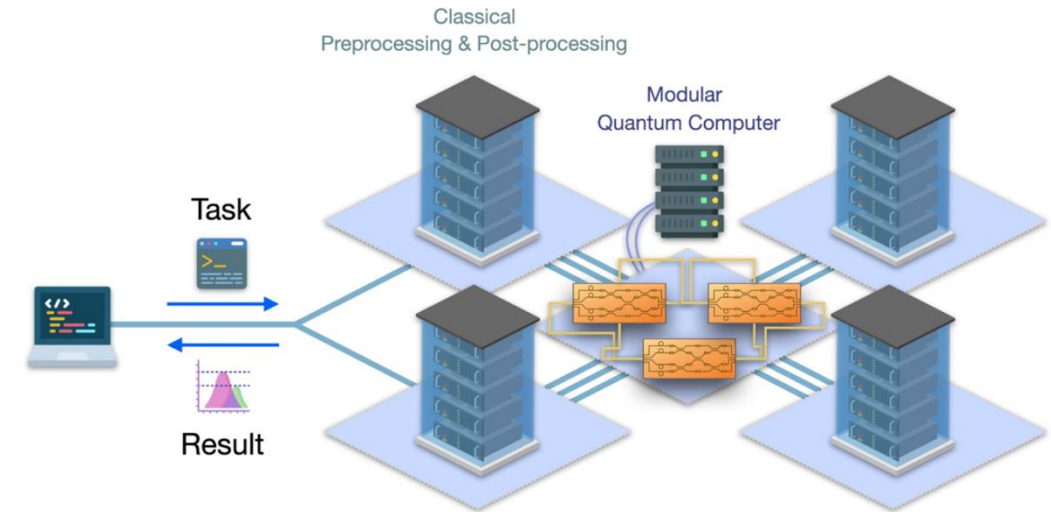
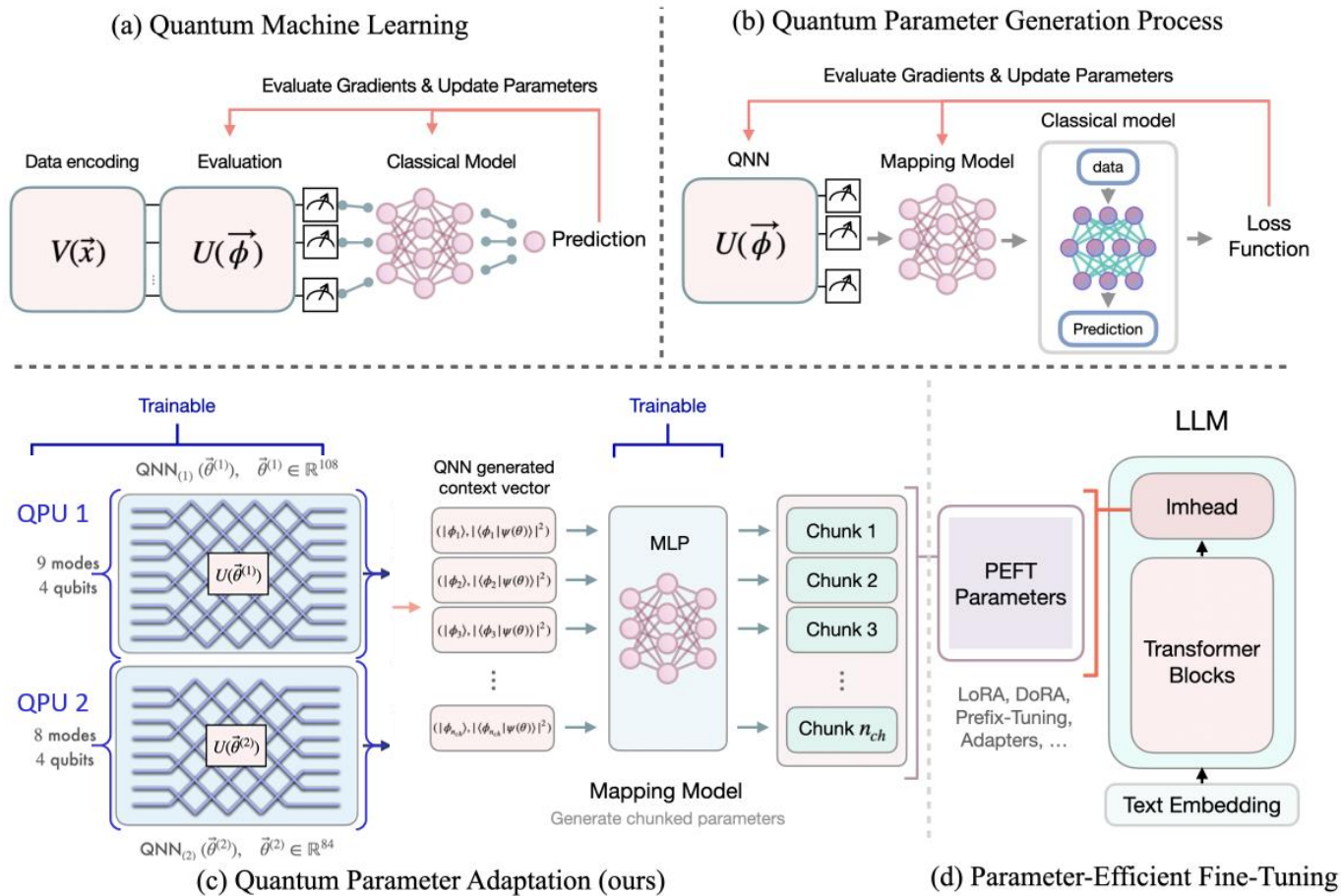


Conclusion

- **Distributed Photonic QNNs:** Hybrid training approach using photonic QNNs to generate weights for classical models, enabling efficient compression.
- **Classical Inference:** Offloads the most compute-intensive training phase to quantum hardware; inference runs classically on CPUs/GPUs for Quantum-HPC applications.
- **High Efficiency:** Achieves up to 90% parameter reduction with competitive accuracy, outperforming classical compression techniques.
- **Noise Resilience:** Noise analysis shows our photonic QNN framework maintains robust validation accuracy across practical photonic imperfections, demonstrating strong noise resilience.

Future Work – Distributed Quantum HPC for Artificial Intelligence

- Working on more complex and larger-parameter models — for example, fine-tuning LLMs.
- Using quantum computers for training and classical computers (CPU or GPU) for inference.



Liu et al, ICLR 2025

Future Work – Quantum for Humanity



FinTech

Credit Scoring, Fraud Detection etc.



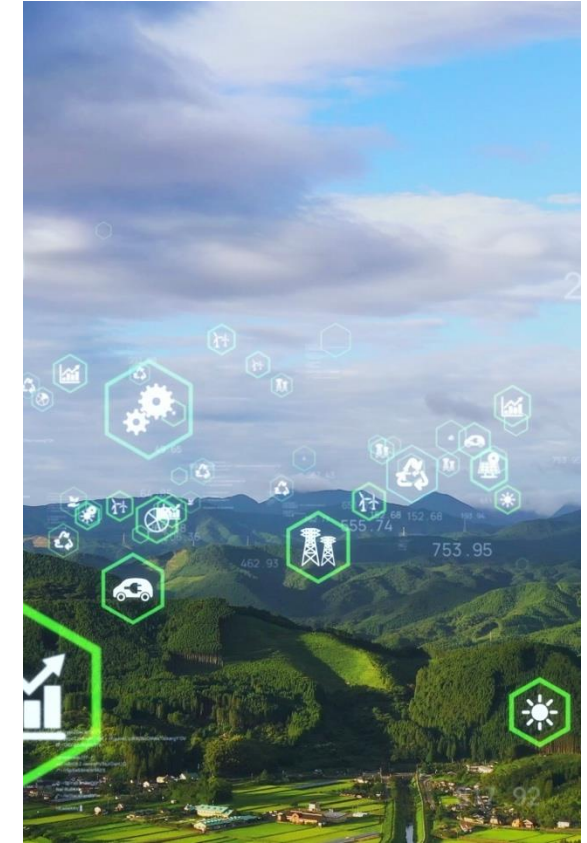
Manufacturing

Defect Detection, Predictive Maintenance etc.



Medical Diagnosis

Disease Diagnosis, Personalized Medicine etc.



Net Zero

Renewable Energy Forecasting, Pollution Control etc.

Than you for your time!
Any Question?