國立臺灣大學理學院應用物理研究所

博士論文

Graduate Institute of Applied Physics

College of Science

National Taiwan University

Doctoral Dissertation

量子訓練：從模型壓縮的觀點重新思考混合式量子古典機器學習

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

劉宸銉

Chen-Yu Liu

指導教授: 管希聖 博士

Advisor: Hsi-Sheng Goan Ph.D.

中華民國 114 年 11 月

November, 2025

doi:10.6342/NTU202504677

國立臺灣大學博士學位論文
口試委員會審定書

PhD DISSERTATION ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

量子訓練：從模型壓縮的觀點重新思考混合式量子古典機器學習

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

本論文係劉宸緯（D10245003）在國立臺灣大學應用物理研究所完成之博士學位論文，於民國 114 年 11 月 17 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Graduate Institute of Applied Physics on 17/11/2025 have examined a PhD dissertation entitled above presented by Chen-Yu Liu (D10245003) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

管希聖	張慶瑞	周業訢
（指導教授 Advisor）		
歐家平	賴秀瑛	江振瑞
蔡芸琤		

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Hsi-Sheng Goan, for his invaluable guidance, encouragement, and support throughout my doctoral studies. His profound knowledge, patience, and rigorous attitude toward research have not only shaped this dissertation but also deeply influenced my academic journey.

I am sincerely thankful to my girlfriend, Ya-Chi, for her constant love, patience, and understanding. Her companionship and encouragement have been a source of strength and motivation through the most challenging times of my Ph.D.

I owe my heartfelt appreciation to my parents, whose unwavering support, sacrifices, and faith in me laid the foundation for all of my achievements. Without their continuous encouragement, this work would not have been possible.

I would also like to extend my gratitude to the members of our paper reading group. Their insightful discussions, constructive feedback, and generous sharing of ideas have greatly enriched my research and broadened my perspectives.

Finally, to all those who have supported me in ways both big and small throughout this journey, I offer my sincerest thanks.

doi:10.6342/NTU202504677

摘要

本論文探討量子計算在參數高效學習中的應用，將量子電路重新定位為參數生成器，而非直接用於推論的模型。我們提出Quantum-Train (QT) 架構，利用量子神經網路將需訓練的參數數量從 $O(M)$ 降至 polylog(M)，同時保持推論完全於古典電腦上執行；並進一步提出Quantum Parameter Adaptation (QPA)，將此原理擴展至大型預訓練模型的微調，有效降低微調過程中的參數成本。理論分析涵蓋近似誤差與泛化能力，實驗則驗證了這些方法在影像分類、語言模型微調、洪水預測、強化學習以及颱風路徑預測等任務上的效能。結果顯示，QT 與 QPA 能在大幅壓縮參數的同時維持接近的性能，提供一條以效率與可部署性為核心的量子實用性途徑。透過將量子生成的參數整合進古典機器學習流程，本研究展現了量子與經典協同運作於可擴展且資源高效人工智慧中的潛力。

關鍵字：量子計算、量子機器學習

doi:10.6342/NTU202504677

Abstract

This thesis explores quantum approaches to parameter-efficient learning, reframing quantum circuits as parameter generators for classical models rather than direct inference engines. We introduce Quantum-Train (QT), which employs quantum neural networks to reduce the number of trainable parameters from $O(M)$ to polylog(M) while keeping inference fully classical, and Quantum Parameter Adaptation (QPA), which extends this principle to fine-tuning large pre-trained models with dramatic reductions in adaptation cost. Theoretical analyses examine approximation error and generalization, while empirical studies validate the frameworks across image classification, language model fine-tuning, flood forecasting, reinforcement learning, and typhoon trajectory prediction. Results show that QT and QPA achieve substantial compression with minimal performance loss, offering a realistic pathway toward quantum utility rooted in efficiency and deployability. By integrating quantum-generated parameters into classical machine learning pipelines, this work highlights the potential of quantum–classical synergy for scalable

doi:10.6342/NTU202504677

and resource-efficient AI.

Keywords: Quantum Computing, Quantum Machine Learning

doi:10.6342/NTU202504677

Contents

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

List of Figures

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

List of Tables

doi:10.6342/NTU202504677

doi:10.6342/NTU202504677

Chapter 1 Introduction to Quantum Machine Learning

1.1 What is Machine Learning?

Machine learning (ML) [57] is a subfield of artificial intelligence that focuses on designing algorithms capable of automatically improving their performance from data, rather than relying on explicit rule-based programming. At its core, ML seeks to approximate an underlying function

$$f : \mathcal{X} \to \mathcal{Y},$$

that maps an input domain $\mathcal{X}$ (e.g., images, speech signals, or physical measurements) to an output domain $\mathcal{Y}$ (e.g., class labels, numerical predictions, or control actions). Given a finite training dataset

$$\{(x_i, y_i)\}_{i=1}^{N},$$

the task of ML is to construct a model $\hat{f}$ that generalizes well to unseen inputs.

doi:10.6342/NTU202504677

Machine learning encompasses a wide spectrum of paradigms:

- **Supervised learning** [14], where models are trained with labeled data (e.g., classification and regression).
- **Unsupervised learning** [2], where the task is to extract structure or representations from unlabeled data (e.g., clustering, dimensionality reduction).
- **Reinforcement learning** [34], where agents interact with an environment and learn policies to maximize cumulative rewards.

The success of modern ML has been largely driven by the availability of large datasets, advances in computational hardware, and increasingly sophisticated model architectures. Deep neural networks, in particular, have demonstrated unprecedented capabilities across natural language processing, computer vision, and scientific computing. However, these advances often come at the cost of large-scale parameterization, where state-of-the-art models may contain billions of parameters. This leads to challenges in terms of memory requirements, energy consumption, and deployment on resource-constrained devices.

The pursuit of parameter-efficient learning has therefore become a central theme in contemporary ML research. Techniques such as knowledge distillation, low-rank adaptation, and transfer learning seek to retain model performance while reducing computational and storage overhead. These same challenges motivate the exploration of alternative computational paradigms, including quantum computing, as a means to achieve more efficient representations and learning mechanisms.

doi:10.6342/NTU202504677

1.2 Quantum Computing as an Element of Machine Learning

Quantum computing [60] is an emerging computational paradigm that harnesses the principles of quantum mechanics to process information in fundamentally different ways from classical computers. The basic unit of quantum information is the qubit, which, unlike a classical bit, can exist in a superposition of states. A single qubit is represented by a normalized vector in a two-dimensional Hilbert space,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C},\ |\alpha|^2 + |\beta|^2 = 1.$$

Multiple qubits can be combined to form composite systems, whose state space grows exponentially with the number of qubits. Moreover, qubits can exhibit entanglement, a uniquely quantum correlation that cannot be described by classical probability theory. Computation is performed by applying unitary operations (quantum gates) to qubits, followed by measurement in a chosen basis, most commonly the computational Z-basis. The potential advantages of quantum computing for machine learning stem from several properties:

- **Exponential state space.** An n-qubit system can represent a vector in a 2^n-dimensional Hilbert space, providing a compact encoding of high-dimensional data.
- **Quantum parallelism.** Superposition allows quantum circuits to process multiple computational paths simultaneously, potentially accelerating certain subroutines.
- **Provable hardness.** Sampling from specific families of quantum circuits, such as Instantaneous Quantum Polynomial-time (IQP) circuits [25] or random univer-

doi:10.6342/NTU202504677

sal circuits [8, 58], has been shown to be classically hard under widely believed complexity-theoretic assumptions.

These features suggest that quantum devices may serve as powerful primitives for constructing new machine learning models, particularly when classical approaches face prohibitive resource requirements. In practice, however, near-term quantum hardware is subject to noise, decoherence, and limited qubit counts. As a result, most current research focuses on hybrid quantum–classical approaches, where quantum circuits are used as specialized subroutines integrated into broader classical ML pipelines. This hybridization not only mitigates hardware constraints but also allows quantum computation to be positioned as a complementary resource for data representation, feature transformation, or parameter generation in machine learning tasks.

1.3 Quantum Kernel Method

Kernel methods [28] are a classical machine learning technique that enable nonlinear classification or regression by implicitly mapping input data into a high-dimensional feature space. Given a dataset $\{x_i\}_{i=1}^{N}$, a kernel function $k(x, x')$ computes the inner product of data representations in the feature space,

$$k(x, x') = \langle \phi(x), \phi(x') \rangle,$$

without requiring the explicit evaluation of the feature map $\phi(\cdot)$. Classical support vector machines (SVMs) [27] and Gaussian processes are among the most widely known kernel-based methods.

doi:10.6342/NTU202504677

In the quantum setting, the quantum kernel method [26] constructs kernels by embedding classical data into quantum states. Specifically, a parameterized quantum feature map circuit $U_\phi(x)$ prepares a quantum state

$$|\phi(x)\rangle = U_\phi(x)|0\rangle^{\otimes n},$$

where n is the number of qubits. The kernel between two inputs x and x' is then given by the fidelity between their corresponding quantum states:

$$k(x, x') = |\langle\phi(x)|\phi(x')\rangle|^2.$$

This approach leverages the high-dimensional Hilbert space of quantum systems, potentially enabling the representation of features that are intractable for classical kernels. The quantum kernel method has several attractive properties:

- **Expressive feature maps.** Quantum circuits can generate highly nonlinear transformations of classical data, which may yield richer decision boundaries.
- **Provable separation.** For carefully chosen feature maps, quantum kernels can induce feature spaces that cannot be efficiently simulated by classical computers, providing a potential quantum advantage [31].
- **Compatibility.** Quantum kernels can be directly integrated into established classical kernel-based learners such as SVMs, requiring no modification of the optimization procedure.

Despite these advantages, practical limitations exist. First, estimating the kernel matrix requires repeated measurements of quantum states, leading to potentially large shot

doi:10.6342/NTU202504677

overhead. Second, not all quantum feature maps yield useful kernels: some may result in feature spaces where data points are nearly orthogonal, thereby degrading generalization performance. Finally, recent studies have shown that certain quantum kernels can be approximated efficiently by classical algorithms in practice, which questions their advantage in near-term scenarios [38]. Nevertheless, the quantum kernel method represents a foundational bridge between classical ML theory and quantum computation. It illustrates how quantum states can serve as data representations and motivates further exploration into more flexible and trainable quantum machine learning models, such as parameterized quantum circuits discussed in the next section.

1.4 Parameterized Quantum Circuits as Quantum Neural Networks

Beyond fixed feature maps, a central paradigm in quantum machine learning is the use of parameterized quantum circuits (PQCs), also known as variational quantum circuits. These are quantum circuits whose gates depend on tunable parameters, typically real-valued rotation angles. By adjusting these parameters during training, PQCs can learn task-specific transformations of data, analogous to weight adaptation in classical neural networks.

Parameterized Circuit Structure

A PQC consists of three key components:

1. **Encoding of classical data.** Input data $x \in \mathbb{R}^d$ is mapped to quantum states via an

doi:10.6342/NTU202504677

encoding scheme, such as angle encoding [39],

$$|x\rangle = U_{\text{enc}}(x)|0\rangle^{\otimes n},$$

where $U_{\text{enc}}(x)$ applies parameterized rotations that depend on data values.

2. **Variational layers.** The circuit applies a sequence of parameterized gates with trainable parameters $\boldsymbol{\theta}$. For example, a typical layer may consist of single-qubit rotations followed by entangling gates such as controlled-NOTs:

$$U(\boldsymbol{\theta}) = \prod_{i=1}^{n} R_y^i(\theta_i) \prod_{(i,j)\in\mathcal{E}} \text{CNOT}^{i,j}.$$

3. **Measurement.** At the end of the circuit, observables are measured in the Pauli basis. In particular, the expectation value of a Pauli operator $O_j \in \{X, Y, Z\}^{\otimes n}$ is estimated as

$$z_j(x;\boldsymbol{\theta}) \;=\; \langle 0^{\otimes n}|U_{\text{enc}}(x)^\dagger U(\boldsymbol{\theta})^\dagger O_j U(\boldsymbol{\theta}) U_{\text{enc}}(x)|0^{\otimes n}\rangle.$$

The resulting vector of expectation values $\mathbf{z}(x;\boldsymbol{\theta}) = (z_1, z_2, \ldots, z_K)$ is then passed through a classical softmax layer to define class probabilities:

$$p_{\boldsymbol{\theta}}(y|x) \;=\; \frac{\exp(z_y(x;\boldsymbol{\theta}))}{\sum_{j=1}^{K}\exp(z_j(x;\boldsymbol{\theta}))}.$$

This hybrid readout converts quantum expectation values into a valid probability distribution suitable for classification tasks.

These components define a flexible model class, where the parameterized gates serve as learnable weights that can be trained using optimization methods.

doi:10.6342/NTU202504677

Training PQCs

Training PQCs follows a hybrid quantum–classical workflow. A cost function, often defined in terms of measurement outcomes, is evaluated on a quantum device. Classical optimization algorithms, such as gradient descent or derivative-free methods (e.g., COBYLA, SPSA), are then used to update parameters. Gradient-based optimization can leverage the parameter-shift rule [75], which allows analytical computation of derivatives via shifted quantum circuits.

Challenges in PQCs

While PQCs are conceptually powerful, nevertheless, several challenges arise in practice:

- **Barren plateaus [55].** The optimization landscape of PQCs can suffer from vanishing gradients, especially for deep or unstructured circuits, making training exponentially hard as system size increases.
- **Noise sensitivity.** Current quantum hardware is noisy, and PQCs can be highly sensitive to gate and measurement errors, limiting the achievable circuit depth.
- **Shot complexity.** Estimating expectation values requires repeated circuit evaluations (shots), which increases inference cost and runtime.
- **Expressivity vs. trainability [19].** While highly expressive circuits can approximate a wide range of functions, they may also become untrainable due to overparameterization or barren plateau effects.

doi:10.6342/NTU202504677

PQCs as Quantum Neural Networks

Due to their layered structure and trainable parameters, PQCs are often referred to as quantum neural networks (QNNs). They parallel classical neural networks in that:

- Data is transformed through successive layers of linear and nonlinear operations (here realized by unitary gates and entangling operations).
- Parameters are optimized through iterative training loops guided by a cost function.
- Generalization performance depends on the balance between expressivity and inductive bias.

However, unlike classical networks, QNNs inherently exploit properties of quantum mechanics: superposition, entanglement, and interference. These properties enable QNNs to represent and process high-dimensional data in ways that may be challenging to simulate classically.

Positioning Toward Hybrid Models

While PQCs form the foundation of most quantum learning algorithms, their limitations highlight the necessity of hybrid designs. In particular, leveraging classical neural networks for preprocessing or postprocessing, and using quantum circuits for specialized tasks such as feature transformation or parameter generation, offers a more practical and scalable route. This hybrid perspective serves as the basis for the approaches developed in the following chapters.

doi:10.6342/NTU202504677

1.5 Hybrid Quantum–Classical Models

Given the limitations of current quantum devices, fully quantum learning models remain impractical in the near term. Instead, most progress in quantum machine learning (QML) has come from hybrid quantum–classical models [53], which combine the strengths of quantum circuits with the versatility and scalability of classical machine learning. The central idea is to allocate tasks strategically: quantum circuits are used where they may provide advantages in expressivity or representation, while classical components handle preprocessing, optimization, or large-scale data management.

Positioning for This Thesis

The overarching theme of this thesis is to explore quantum–classical hybridization not merely as a way to overcome hardware limitations, but as an opportunity to rethink the design of learning systems. In particular, we focus on frameworks where quantum circuits generate or adapt parameters for classical models, thereby enabling efficient training and inference. This direction leads to the development of Quantum-Train [48] and Quantum Parameter Adaptation [51], which are introduced in Chapters 2 and 3, respectively.

By positioning quantum circuits as tools for parameter-efficient learning, rather than as standalone predictors, we seek to uncover a pathway toward practical quantum advantages in machine learning.

doi:10.6342/NTU202504677

Chapter 2 Quantum-Train

Chapter Overview

This chapter introduces Quantum-Train (QT), a framework that leverages parameterized quantum circuits as generators of classical model parameters. Rather than applying quantum circuits directly for prediction, QT reframes their role as tools for parameter-efficient learning, aligning with the motivation established in Chapter 1. We begin with the core mechanism of parameter generation, proceed to analyze its training process and theoretical guarantees, and conclude with empirical demonstrations across multiple benchmarks. Much of the content in this chapter is based on our published work [48], with minor adaptations for clarity and consistency in the thesis narrative.

2.1 Parameter Generation via Quantum Neural Networks

We now describe the key elements of the Quantum-Train (QT) framework and explain how a quantum state can be employed to generate the weights of a classical neural network (NN). This section establishes the connection between the quantum representation and the classical parameter space, together with the associated parameterization and training strategy.

doi:10.6342/NTU202504677

Consider a classical NN defined by a weight vector $W \in \mathbb{R}^M$. In the QT approach, these M parameters are produced by a quantum circuit acting on $N = \lceil \log_2 M \rceil$ qubits. Since the corresponding Hilbert space has dimension $2^N \geq M$, it provides enough degrees of freedom to encode the full set of parameters.

The circuit prepares a parameterized quantum state

$$|\psi(\theta_{\text{qnn}})\rangle = U(\theta_{\text{qnn}})\,|0\rangle^{\otimes N}, \tag{2.1}$$

with tunable gate parameters θ_{qnn}. From this state, we extract the measurement probabilities of the first M computational basis states, which we collect in a vector $\Psi \in [0,1]^M$:

$$\Psi = \left[\, |\langle\phi_1|\psi\rangle|^2,\ |\langle\phi_2|\psi\rangle|^2,\ \ldots,\ |\langle\phi_M|\psi\rangle|^2 \,\right]^\top, \tag{2.2}$$

where $|\phi_i\rangle$ denotes the binary basis vector associated with the i-th outcome.

For convenience, these basis states can be stacked into a binary matrix $\Phi \in \{0,1\}^{M\times N}$,

$$\Phi = \begin{bmatrix} |\phi_1\rangle \\ |\phi_2\rangle \\ \vdots \\ |\phi_M\rangle \end{bmatrix}, \tag{2.3}$$

so that both Φ and Ψ are available for downstream processing. To couple the quantum outputs with the target NN weights, we construct an augmented input $\Psi^c \in [0,1]^{M\times(N+1)}$ by concatenating each basis vector with its probability,

$$\Psi_i^c = \left[\, |\phi_i\rangle \parallel |\langle\phi_i|\psi\rangle|^2 \,\right], \tag{2.4}$$

doi:10.6342/NTU202504677

where $\|$ denotes concatenation. Each row of Ψ^c contains N binary entries and one probability entry.

Since probabilities lie in $[0, 1]$ while network weights W_i are real-valued, we introduce a learnable mapping $G_{\theta_{\text{mm}}}$, parameterized by θ_{mm}, to bridge the two. We realize $G_{\theta_{\text{mm}}}$ as a multilayer perceptron (MLP) with input size $N + 1$, mapping each augmented vector to a single parameter:

$$G_{\theta_{\text{mm}}} : [0, 1]^{N+1} \rightarrow \mathbb{R}, \quad W_i = G_{\theta_{\text{mm}}}(\Psi_i^c). \tag{2.5}$$

In practice, the mapping is applied in parallel across all rows,

$$G_{\theta_{\text{mm}}} : [0, 1]^{M\times(N+1)} \rightarrow \mathbb{R}^M, \tag{2.6}$$

so that the entire weight vector W is produced in one step.

As a simple illustration, a two-layer mapping takes the form

$$G_{\theta_{\text{mm}}}(\Psi_i^c) = g_2(\sigma(g_1(\Psi_i^c))) \,, \tag{2.7}$$

where g_1 and g_2 are affine transformations, σ is a nonlinear activation (e.g., sigmoid), and the output gives the i-th weight. Because Ψ^c depends on θ_{qnn}, the final weights depend jointly on quantum and classical parameters,

$$W = W(\theta_{\text{qnn}}, \theta_{\text{mm}}). \tag{2.8}$$

This means both the quantum circuit and the mapping model are trained to adapt the resulting NN parameters. Unlike earlier approaches with fixed mappings and weight signs,

doi:10.6342/NTU202504677

the MLP-based mapping allows flexible sign learning and richer inductive bias.

The quantum circuit $U(\theta_{\text{qnn}})$ is constructed from parameterized R_y rotations,

$$R_y(\phi) = \begin{bmatrix} \cos(\frac{\phi}{2}) & -\sin(\frac{\phi}{2}) \\ \sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2}) \end{bmatrix}, \tag{2.9}$$

together with CNOT gates to generate entanglement. The number of tunable quantum parameters grows as $O(\text{polylog}(M))$, with depth determined by the number of blocks L. Choices such as $L = O(N)$ or $O(N^2)$ are common [4, 12, 69], but more generally L can scale polynomially in N. This ansatz therefore offers a compact, scalable mechanism to generate probability distributions over 2^N basis states, which can be mapped efficiently into M neural network weights. An overview of the ansatz structure is given in Fig. 2.1.

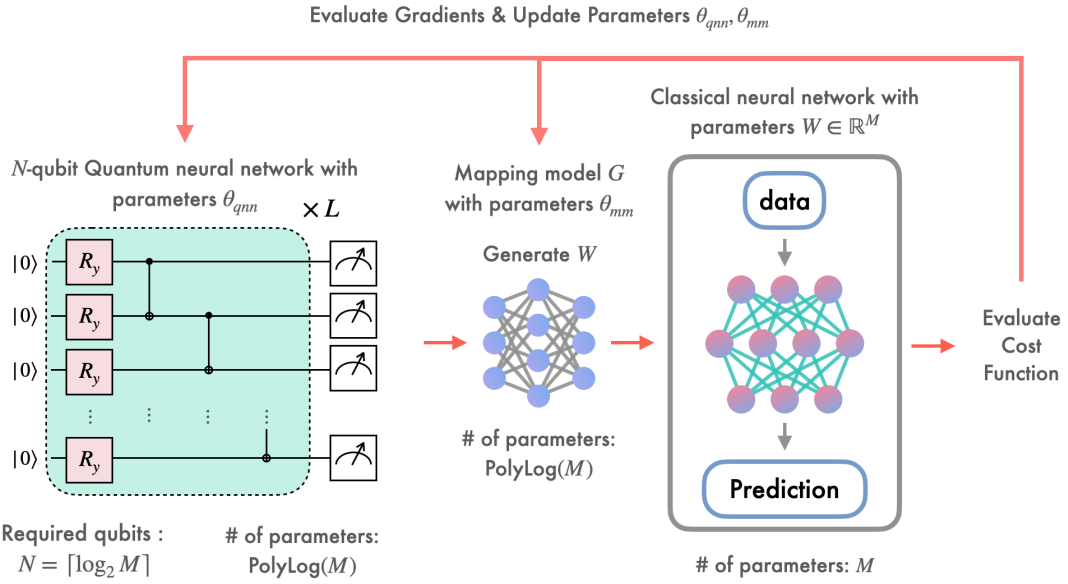


Figure 2.1: Schematic of the Quantum-Train (QT) workflow. An N-qubit variational quantum circuit, composed of parameterized R_y rotations and entangling CNOT gates, is optimized together with a classical mapping model $G_{\theta_{\text{mm}}}$. Their parameters, θ_{qnn} and θ_{mm}, are updated using gradients from a task-specific cost function (e.g., cross-entropy loss). The outputs of the quantum circuit and mapping model define a reduced set of neural network weights W, compressing the parameterization to $O(\text{polylog}(M))$. These weights are then assigned to a classical neural network, which processes data and produces predictions. After training, the entire inference stage is executed purely on classical hardware, demonstrating the practicality of QT as a bridge between quantum parameter generation and conventional machine learning.

doi:10.6342/NTU202504677

2.2 Training Process and Gradient-Based Updates

Having established the dependence $W = W(\theta_{\text{qnn}}, \theta_{\text{mm}})$ in the previous section, we now outline how the Quantum-Train (QT) framework is optimized in practice. During training, the classical neural network defined by the weight vector $W(\theta_{\text{qnn}}, \theta_{\text{mm}})$ is trained to perform the target task, such as classification. Model outputs are compared with labels through a supervised loss function. For classification tasks, we adopt the standard cross-entropy loss:

$$\ell_{\text{CE}} = -\frac{1}{N_{\text{d}}} \sum_{n=1}^{N_{\text{d}}} \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right], \tag{2.10}$$

where y_n represents the ground-truth label, $\hat{y}_n$ the predicted probability, and N_{d} the number of training examples.

Gradient Estimation. Since the weights W are a function of both the quantum parameters θ_{qnn} and the classical mapping parameters θ_{mm}, gradients must be propagated through the hybrid pipeline. By applying the chain rule, the gradient of the loss is expressed as

$$\nabla_{\theta_{\text{qnn}}, \theta_{\text{mm}}} \ell_{\text{CE}} = \left(\frac{\partial W}{\partial(\theta_{\text{qnn}}, \theta_{\text{mm}})} \right)^T \cdot \nabla_W \ell_{\text{CE}}, \tag{2.11}$$

where the Jacobian $\frac{\partial W}{\partial(\theta_{\text{qnn}}, \theta_{\text{mm}})}$ captures the sensitivity of the generated NN weights to variations in both sets of underlying parameters. This formulation highlights how changes in the quantum circuit and mapping model jointly influence the performance of the classical NN. We explore several training settings in this thesis. For ideal experiments, exact state-vector simulation is used, where gradients can be computed directly through automatic differentiation. To more closely mimic hardware conditions, we also consider finite-shot simulations and noise-aware settings. In these cases, gradients with respect to θ_{qnn} are ob-

doi:10.6342/NTU202504677

tained using the parameter-shift rule and its generalizations [67, 75], which remain valid even under sampling noise.

Parameter Updates. At iteration t, both θ_{qnn} and θ_{mm} are updated via a gradient-based optimizer:

$$\theta_{\text{qnn}}^{(t+1)},\ \theta_{\text{mm}}^{(t+1)} \ =\ \theta_{\text{qnn}}^{(t)},\ \theta_{\text{mm}}^{(t)} + \eta\, \nabla_{\theta_{\text{qnn}},\theta_{\text{mm}}}\, \ell_{\text{CE}}, \tag{2.12}$$

with learning rate η. Because the quantum and classical parts interact during training, careful tuning of η and gradient normalization can have a strong effect on stability and convergence.

Efficiency and Deployment. As illustrated in Fig. 2.1, the QT framework achieves parameter efficiency by requiring only $O(\text{polylog}(M))$ parameters in the QNN, while the generated classical NN holds M parameters. Importantly, after training, the entire model can be deployed without quantum resources, since inference runs purely on the classical NN. This property ensures the framework' s practicality, especially in contexts where access to quantum hardware is constrained.

2.3 Theoretical Perspective on Approximation Error

To analyze the theoretical behavior of the Quantum-Train (QT) framework, we draw upon classical tools from statistical learning theory [57, 63], focusing in particular on approximation error. Our goal is to characterize how well a QT-generated neural network can approximate the optimal hypothesis defined in the classical setting. Let $\mathbb{F}_{\text{NN}}$ denote the hypothesis space of classical neural networks. Suppose we are given N_d training samples

doi:10.6342/NTU202504677

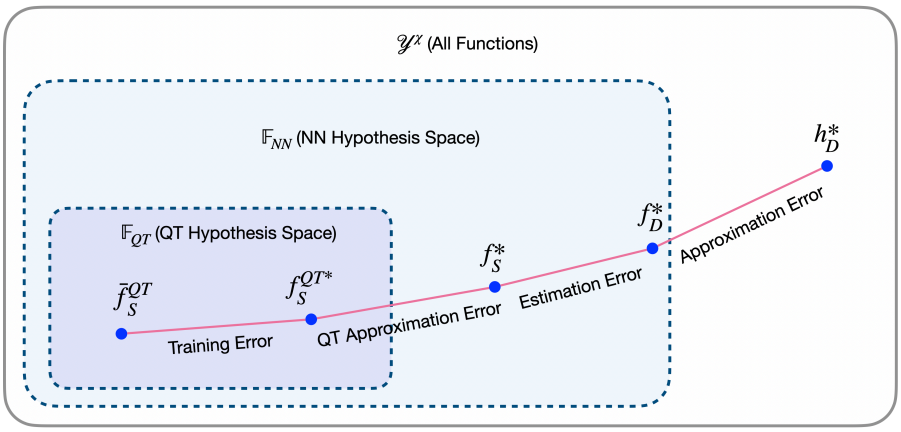


Figure 2.2: **Illustration of the error decomposition framework in Quantum-Train (QT).** The target function h_D^* denotes the true mapping from the input space χ to the label space $\mathcal{Y}$ under the full data distribution $\mathcal{D}$. The function f_D^* corresponds to the best classical neural network hypothesis achievable in expectation with respect to $\mathcal{D}$. The empirical optimum f_S^* is defined as the best classical model trained on a finite sample set $S \subset \mathcal{D}$, and serves as a benchmark for idealized classical training on S. Within the QT setting, $f_S^{\mathrm{QT}*}$ denotes the optimal network produced by the quantum-to-classical parameter generation process, reflecting both the expressivity of the parameterized quantum circuit $U(\theta_{\mathrm{qnn}})$ and the capacity of the mapping model $G_{\theta_{\mathrm{mm}}}$. Finally, $\bar{f}_S^{\mathrm{QT}}$ represents the practically instantiated QT model under finite-shot measurements, and is therefore affected by sampling noise and stochastic variability.

drawn i.i.d. from a data distribution $\mathcal{D}$. For a loss function ℓ and target function $h_{\mathcal{D}}^*$, the optimal network with respect to $\mathcal{D}$ is denoted $f_{\mathcal{D}}^* \in \mathbb{F}_{\mathrm{NN}}$. Its expected loss is

$$\mathcal{L}_{\mathcal{D}}(f_{\mathcal{D}}^*) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\ell\big(h_{\mathcal{D}}^*(\mathbf{x}), f_{\mathcal{D}}^*(\mathbf{x})\big)\right], \tag{2.13}$$

and the corresponding empirical loss over dataset $S \subset \mathcal{D}$ is

$$\mathcal{L}_S(f_{\mathcal{D}}^*) = \frac{1}{N_d}\sum_{n=1}^{N_d} \ell\big(h_{\mathcal{D}}^*(\mathbf{x}_n), f_{\mathcal{D}}^*(\mathbf{x}_n)\big). \tag{2.14}$$

We distinguish between $f_{\mathcal{D}}^*$ (the ideal hypothesis with respect to the true distribution), f_S^* (the best classical hypothesis achievable on dataset S), and $\bar{f}_S$ (the hypothesis returned by training). Without QT, the standard error decomposition expresses the expected loss of $\bar{f}_S$

doi:10.6342/NTU202504677

as

$$\mathcal{L}_{\mathcal{D}}(\bar{f}_S) = \underbrace{\mathcal{L}_{\mathcal{D}}(f^*_{\mathcal{D}})}_{\text{Approximation Error}} + \underbrace{\mathcal{L}_{\mathcal{D}}(f^*_S) - \mathcal{L}_{\mathcal{D}}(f^*_{\mathcal{D}})}_{\text{Estimation Error}}$$
$$+ \underbrace{\mathcal{L}_{\mathcal{D}}(\bar{f}_S) - \mathcal{L}_{\mathcal{D}}(f^*_S)}_{\text{Training Error}}$$
$$\leq \mathcal{L}_{\mathcal{D}}(f^*_{\mathcal{D}}) + 2\hat{\mathcal{R}}_S(\mathbb{F}_{\text{NN}}) + v, \tag{2.15}$$

where $\hat{\mathcal{R}}_S(\mathbb{F}_{\text{NN}})$ denotes the empirical Rademacher complexity of the hypothesis class and v accounts for optimization bias.

To incorporate QT, let $\mathbb{F}_{\text{QT}} \subset \mathbb{F}_{\text{NN}}$ denote the restricted hypothesis space defined by quantum parameter generation. As before, $\bar{f}_S^{\text{QT}}$ is the model returned by training with QT, while $f_S^{\text{QT}*}$ is the optimal hypothesis within $\mathbb{F}_{\text{QT}}$. We define the QT approximation error as

$$\mathcal{L}_{\mathcal{D}}(f_S^{\text{QT}*}) - \mathcal{L}_{\mathcal{D}}(f^*_S),$$

which measures the gap between the best QT-generated hypothesis and the best classical hypothesis achievable on S. Analogously, the QT training error is

$$\mathcal{L}_{\mathcal{D}}(\bar{f}_S^{\text{QT}}) - \mathcal{L}_{\mathcal{D}}(f_S^{\text{QT}*}),$$

reflecting the distance between the trained QT model and its own optimum. The error

doi:10.6342/NTU202504677

decomposition in the QT case therefore becomes:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(\bar{f}_S^{\mathrm{QT}}) = & \underbrace{\mathcal{L}_{\mathcal{D}}(f_{\mathcal{D}}^*)}_{\text{Approximation Error}} + \underbrace{\mathcal{L}_{\mathcal{D}}(f_S^*) - \mathcal{L}_{\mathcal{D}}(f_{\mathcal{D}}^*)}_{\text{Estimation Error}} \\ & + \underbrace{\mathcal{L}_{\mathcal{D}}(f_S^{\mathrm{QT}*}) - \mathcal{L}_{\mathcal{D}}(f_S^*)}_{\text{QT Approximation Error}} \\ & + \underbrace{\mathcal{L}_{\mathcal{D}}(\bar{f}_S^{\mathrm{QT}}) - \mathcal{L}_{\mathcal{D}}(f_S^{\mathrm{QT}*})}_{\text{QT Training Error}} \\ & \leq \mathcal{L}_{\mathcal{D}}(f_{\mathcal{D}}^*) + 2\hat{\mathcal{R}}_S(\mathbb{F}_{\mathrm{NN}}) + \left[\mathcal{L}_{\mathcal{D}}(f_S^{\mathrm{QT}*}) - \mathcal{L}_{\mathcal{D}}(f_S^*)\right] + v. \end{aligned} \tag{2.16}$$

Theorem 1 (QT Approximation Error Bound)**.** *Let f_S^* denote the optimal empirical hypothesis in the classical setting, and let f_S^{QT*} denote the optimal QT-generated hypothesis produced from a parameterized quantum circuit $U(\theta_{qnn})$ of depth D, a mapping model $G_{\theta_{mm}}$ with effective width h_g, and m measurement shots. Under the assumptions of universal approximation for $G_{\theta_{mm}}$, bounded activation derivatives, and universality of the QNN gate set such that the unitary approximation error satisfies $\epsilon \leq O(1/\exp(D^{1/c_{sk}}))$, the QT approximation error satisfies*

$$\mathcal{L}_{\mathcal{D}}(f_S^{QT*}) - \mathcal{L}_{\mathcal{D}}(f_S^*) \leq O\left(\frac{1}{\sqrt{h_g}}\right) + O\left(\frac{1}{\exp(D^{1/c_{sk}})}\right) + O\left(\frac{1}{\sqrt{m}}\right). \tag{2.17}$$

This bound highlights how each element of QT contributes to performance. The $O(1/\sqrt{h_g})$ term arises from the capacity of the mapping model; widening $G_{\theta_{\mathrm{mm}}}$ reduces this error, consistent with universal approximation results. The $O(1/\exp(D^{1/c_{sk}}))$ term reflects the expressivity of the quantum circuit, with c_{sk} stemming from the Solovay – Kitaev theorem [15]. Smaller values of c_{sk} indicate greater efficiency in approximating arbitrary unitaries, and suggest that circuit design can play an important role in reducing error without requiring prohibitively deep circuits. The $O(1/\sqrt{m})$ term quantifies statis-

doi:10.6342/NTU202504677

tical noise due to finite measurement shots, an unavoidable factor in NISQ hardware.

In our experiments, we employ R_y–CNOT ansatz circuits, which are not universal. Hence the exponential scaling term predicted in Theorem 1 does not strictly apply, though the qualitative trend of improved approximation with depth is observed. The theorem therefore serves as a general guideline, while the empirical results illustrate the practical trade-offs among circuit depth, mapping capacity, and measurement budget. A detailed proof of the bound is provided in Appendix 5.4.

Overall, the decomposition underscores the interplay between quantum expressivity, classical mapping power, and finite-shot effects. Careful joint design of the QNN and mapping model is therefore essential for minimizing approximation error within QT.

2.4 Results on Model Complexity and Efficiency

In this section, we report the outcomes of numerical simulations designed to evaluate the Quantum-Train (QT) framework. The results are organized around several central aspects: (i) model complexity and efficiency, (ii) generalization error, (iii) the role of the mapping model, (iv) comparison with existing classical compression techniques, (v) robustness to noise and finite-shot effects, and (vi) overall practicality of the QT approach.

All simulations were performed using the TorchQuantum package [74]. We first benchmarked QT on the MNIST dataset [16], where the baseline is a convolutional neural network (CNN) containing $M = 6690$ trainable parameters. The input data consists of 28×28 grayscale images, and the corresponding number of qubits required is $\lceil \log_2(6690) \rceil = 13$. The training set comprises $60{,}000$ samples, with an additional $10{,}000$ reserved for testing. Parameters of both the quantum circuit ($\vec{\phi}$) and the mapping model

doi:10.6342/NTU202504677

($\vec{\gamma}$) were initialized randomly. Training proceeded for 50 epochs with the Adam optimizer [35], using a batch size of 128 and learning rate 10^{-4}.

The architecture of the classical CNN and the QT variants (with different numbers of QNN blocks) is summarized in Table 2.1, which also records test accuracy, parameter size, and compression ratio (defined as the ratio between the parameter count of the baseline CNN and the QT model). Training losses for models with different QNN block counts are shown in Fig. 2.3(a). For FashionMNIST [77], we adopt the same setup, with results reported in Table 2.2 and Fig. 2.3(b).

We then extended the study to CIFAR-10 [37], which is substantially more challenging due to its 32×32 color images. The baseline CNN contains $285{,}226$ parameters, requiring $\lceil \log_2(285226) \rceil = 19$ qubits in the QT setting. Training was conducted for 1000 epochs with a batch size of 1000 and learning rate 10^{-4}. Again, we varied the number of QNN blocks to evaluate scaling effects. Detailed results are provided in Table 2.3, with training curves presented in Fig. 2.3(c).

To further analyze the differences between QT-based and classical training, we compared the resulting parameter distributions, as shown in Fig. 2.4. For MNIST and FashionMNIST we display results with 16 QNN blocks, while for CIFAR-10 we show the case with 95 QNN blocks. Despite using significantly fewer trainable parameters (as demonstrated in Tables 2.1–2.3), QT achieves test accuracies comparable to the classical baselines. Notably, the parameter distributions differ: QT-trained networks tend to have larger weight magnitudes and sparser density near zero, in contrast to the smoother distributions observed in classical networks. This reflects the unique optimization characteristics of QT, where M-dimensional weight vectors are generated from only $O(\text{polylog}(M))$ pa-

doi:10.6342/NTU202504677

rameters. The divergence in parameter statistics highlights QT' s distinctive approach to parameter efficiency and suggests alternative generalization properties compared to conventional compression schemes.

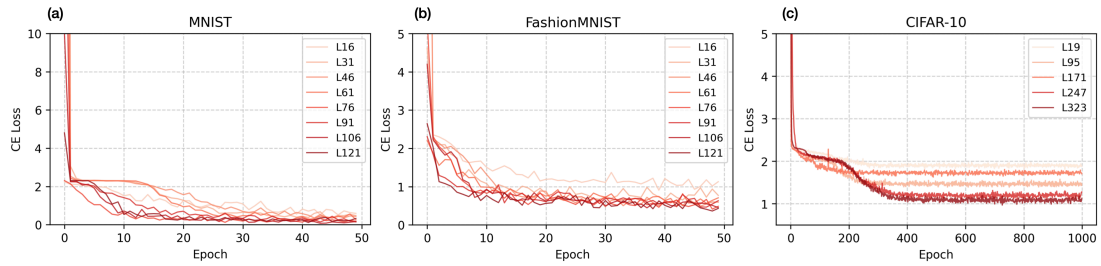


Figure 2.3: Cross-entropy (CE) loss during training of QT models on three datasets: (a) MNIST, (b) FashionMNIST, and (c) CIFAR-10. Each curve corresponds to a QNN with a different number of blocks (denoted by "L" followed by the block count). For MNIST and FashionMNIST, results are shown over 50 epochs, while CIFAR-10 is trained for 1000 epochs due to its higher complexity. Increasing the number of QNN blocks generally produces smoother and more stable convergence, demonstrating the effect of circuit depth on the learning dynamics.

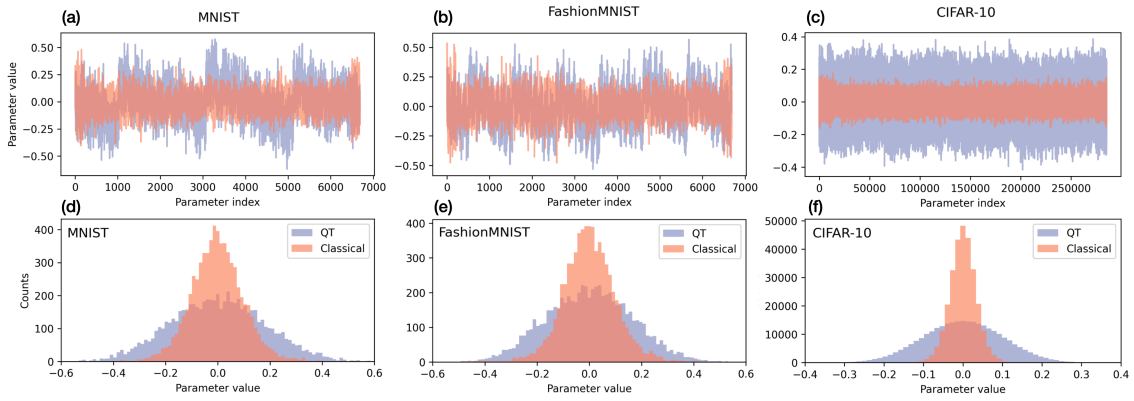


Figure 2.4: Comparison of parameter values between QT-trained networks and classically trained neural networks. The top row (a–c) shows parameter values plotted against their indices for MNIST, FashionMNIST, and CIFAR-10, with QT results obtained using 16 QNN blocks for MNIST and FashionMNIST, and 95 blocks for CIFAR-10. The bottom row (d–f) presents the corresponding parameter distributions. Blue curves correspond to QT models, while orange curves denote classical baselines. QT models exhibit distinctive weight distributions, typically characterized by larger magnitudes and sparser regions near zero, reflecting the effect of generating parameters from $O(\text{polylog}(M))$ quantum parameters.

Before analyzing the QT framework in detail, we first introduce the Quantum-Classical Machine Learning (QCML) architecture [5, 54], which we use as a baseline for comparison (see Fig. 2.5). QCML integrates classical neural networks and quantum neural networks

doi:10.6342/NTU202504677

Model	Topology	Test acc. (%)	Para. size	C. R.
Classical CNN	(Conv_layers, max_pooling)*2, 192-20, 20-10	98.45	6690	1
QT-16	QNN: (R_y block)*16 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	81.08	457	14.64
QT-31	QNN: (R_y block)*31 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	84.69	652	10.26
QT-46	QNN: (R_y block)*46 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	87.78	847	7.89
QT-61	QNN: (R_y block)*61 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	88.43	1042	6.42
QT-76	QNN: (R_y block)*76 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	90.05	1237	5.40
QT-91	QNN: (R_y block)*91 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	92.41	1432	4.67
QT-106	QNN: (R_y block)*106 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	93.81	1627	4.11
QT-121	QNN: (R_y block)*121 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	93.05	1822	3.67
QT-136	QNN: (R_y block)*136 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	94.91	2017	3.31
QT-151	QNN: (R_y block)*151 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	94.33	2212	3.02

Table 2.1: Comparative performance of Classical CNN and QT models with varying numbers of QNN blocks for MNIST dataset.

Model	Topology	Test acc. (%)	Para. size	C. R.
Classical CNN	(Conv_layers, max_pooling)*2, 192-20, 20-10	88.14	6690	1
QT-16	QNN: (R_y block)*16 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	65.69	457	14.64
QT-31	QNN: (R_y block)*31 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	73.48	652	10.26
QT-46	QNN: (R_y block)*46 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	76.05	847	7.89
QT-61	QNN: (R_y block)*61 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	74.92	1042	6.42
QT-76	QNN: (R_y block)*76 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	77.31	1237	5.40
QT-91	QNN: (R_y block)*91 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	77.51	1432	4.67
QT-106	QNN: (R_y block)*106 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	80.20	1627	4.11
QT-121	QNN: (R_y block)*121 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	80.74	1822	3.67
QT-136	QNN: (R_y block)*136 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	83.55	2017	3.31
QT-151	QNN: (R_y block)*151 + Mapping Model: (14-4, 4-20, 20-4, 4-1)	83.19	2212	3.02

Table 2.2: Comparative performance of Classical CNN and QT models with varying numbers of QNN blocks for FashionMNIST dataset.

Model	Topology	Test acc. (%)	Para. size	C. R.
Classical CNN	(Conv_layers, max_pooling)*2, 2048-128, 128-64, 64-10	62.50	285226	1
QT-19	QNN: (R_y block)*19 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	28.73	17482	16.31
QT-95	QNN: (R_y block)*95 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	44.76	18926	15.07
QT-171	QNN: (R_y block)*171 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	47.42	20370	14.00
QT-247	QNN: (R_y block)*247 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	56.45	21814	13.07
QT-323	QNN: (R_y block)*323 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	60.69	23258	12.26
QT-361	QNN: (R_y block)*361 + Mapping Model: (20-40, 40-200, 200-40, 40-1)	61.08	23980	11.89

Table 2.3: Comparative performance of Classical CNN and QT models with varying numbers of QNN blocks for CIFAR-10 dataset.

doi:10.6342/NTU202504677

in a sequential pipeline, aiming to leverage the complementary strengths of both. In this setup, a classical NN performs initial feature extraction from the input data. The extracted features are then encoded into quantum states through parameterized gates, which serve as inputs to a multi-layer quantum circuit. Expectation values from quantum measurements are subsequently fed into a downstream classical NN that produces the final predictions.

This hybrid design can improve expressivity and efficiency relative to purely classical or purely quantum models. We focus on QCML as a baseline rather than architectures that rely on full angle encoding of raw data, since direct encoding of high-dimensional inputs (such as images) would require prohibitively many qubits and deep circuits. QCML, by contrast, benefits from classical preprocessing while still incorporating a quantum processing stage, making it a more practical and meaningful comparator for QT.

In our experiments, the QCML models are instantiated as follows. For MNIST, the classical front-end (Part 1 in Fig. 2.5) is a CNN with 4693 parameters. The QNN in the middle consists of 13 qubits arranged in 13 blocks, totaling 169 parameters. The downstream classical component (Part 2) contributes another 140 parameters, leading to a total of 5002 parameters for the QCML model. The FashionMNIST configuration mirrors this setup, as the two datasets share the same dimensionality. For CIFAR-10, the front-end CNN contains 145,977 parameters, followed by the same 13-qubit, 13-block QNN (169 parameters), and a 140-parameter classical output head, giving a total of 146,286 parameters. In all cases, the majority of parameters reside in the classical components, with the QNN contributing only marginally. As a result, QCML achieves performance similar to that of the purely classical baselines, a point we return to in the following section.

doi:10.6342/NTU202504677

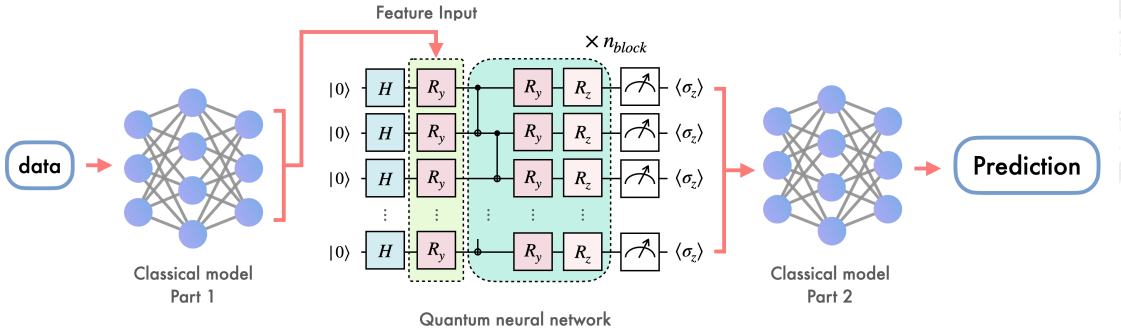


Figure 2.5: Architecture of the Hybrid Quantum–Classical Machine Learning (QCML) model used as a baseline. Input data are first processed by a classical neural network (Part 1), producing features that are encoded into a quantum state through Hadamard gates and parameterized rotations (R_y, R_z). These features are then processed by a multi-block QNN, and the qubit expectation values $\langle\sigma_z\rangle$ are measured. The outputs are passed to a second classical network (Part 2) that produces the final predictions. QCML combines classical feature extraction with quantum processing to provide a meaningful comparison point for QT.

2.5 On Generalization Error

The results in Fig. 2.6(a – c) reveal a consistent pattern: classical models tend to exhibit a larger gap between training and testing accuracy compared to QT, indicating higher generalization error [11]. Generalization error, which quantifies how well a model performs on unseen data, is formally defined as

$$\text{gen}(\alpha) = R(\alpha) - R_S(\alpha), \tag{2.18}$$

where α denotes the learnable parameters. Here,

$$R(\alpha) = \mathbb{E}_{(x,y)\sim P}[\ell(\alpha; x, y)] \tag{2.19}$$

is the expected loss over the data distribution P, and

$$R_S(\alpha) = \frac{1}{N_\text{d}} \sum_{n=1}^{N_\text{d}} \ell(\alpha; x_n, y_n) \tag{2.20}$$

doi:10.6342/NTU202504677

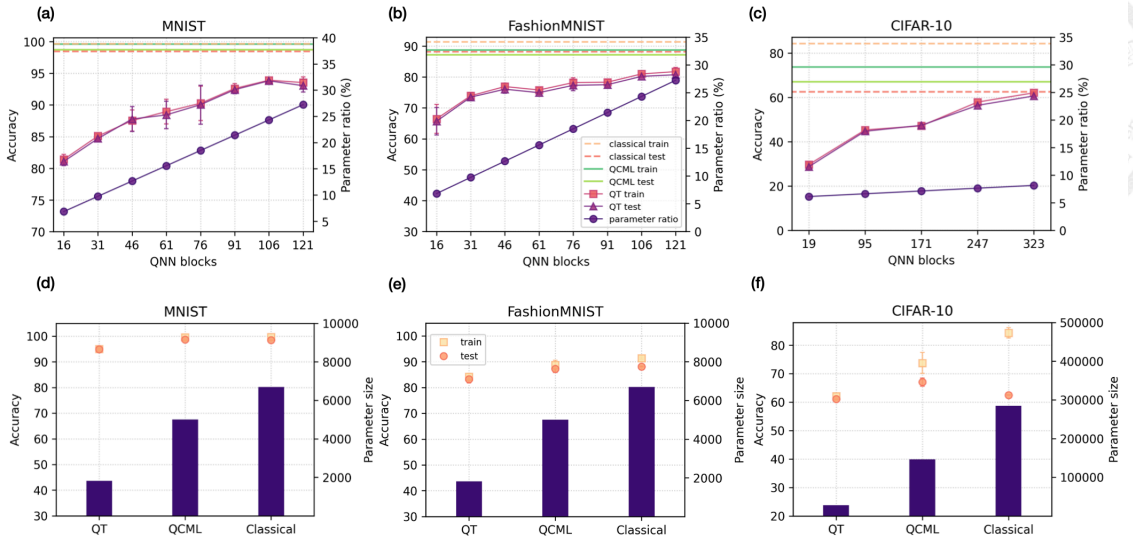


Figure 2.6: Comparison of accuracy and parameter efficiency for QT, QCML, and classical models on MNIST, FashionMNIST, and CIFAR-10. Top row (a–c): training accuracy of QT models with different numbers of QNN blocks, showing that deeper circuits generally improve accuracy while maintaining compact parameterization. Bottom row (d–f): accuracy versus parameter size across QT, QCML, and classical models. QT achieves high accuracy with dramatically fewer parameters (136 blocks for MNIST/FashionMNIST and 361 blocks for CIFAR-10), illustrating its potential for efficient model compression.

is the empirical loss computed on the training set of size N_d.

As shown in Fig. 2.7, QT models consistently achieve lower generalization errors than their classical counterparts, with QCML models showing similar behavior to QT. This suggests that quantum-enhanced models possess stronger generalization capabilities, despite operating with significantly fewer trainable parameters. In contrast, classical models display clearer signs of overfitting, consistent with their higher parameter counts relative to sample size.

From a theoretical perspective, one might expect that increasing the number of parameters N_t would worsen generalization due to overfitting, while enlarging the training set n improves it by better approximating the true loss. This balance between model complexity and dataset size is central to understanding generalization in both classical and quantum machine learning. Prior results [11] suggest that generalization error scales pro-

doi:10.6342/NTU202504677

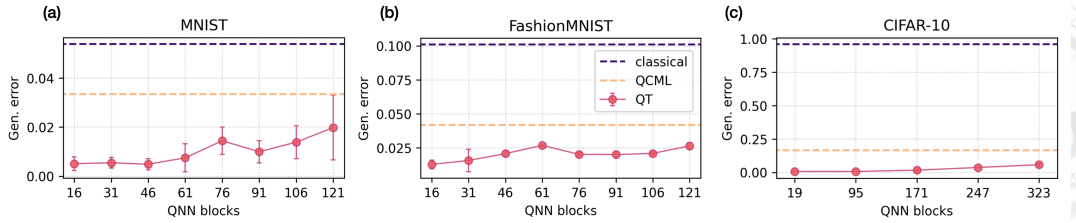


Figure 2.7: Generalization error for QT, QCML, and classical models on (a) MNIST, (b) FashionMNIST, and (c) CIFAR-10. For MNIST, QT error decreases with more QNN blocks, indicating improved learning with circuit depth. On FashionMNIST, QT error grows with block count but remains below QCML and classical levels. For CIFAR-10, generalization error stays low and relatively stable across different QNN depths.

portionally to $\text{gen}(\alpha) \sim O(\sqrt{N_t/n})$, emphasizing the dependence on both parameter count and training data volume.

Our experiments highlight several dataset-specific behaviors. For MNIST, the generalization error of QT models increases gradually with more QNN blocks, reflecting the added model complexity. FashionMNIST follows a similar trend, although QT still maintains lower error levels than both classical and QCML models. For CIFAR-10, QT generalization error remains low and relatively stable, with only a modest rise at 247 blocks, underscoring QT' s robustness in handling more complex datasets.

Finally, we consider the notion of over-parameterization. In classical learning theory [20], over-parameterization occurs when $N_t \gg n$. For QNN-based models, however, the threshold is lower and typically identified as $N_t > n$. In our MNIST and FashionMNIST settings, none of the models (classical, QCML, or QT) are over-parameterized, since $n = 6 \times 10^4$ while parameter counts are on the order of 10^3. For CIFAR-10, the classical baseline with $285{,}226$ parameters exceeds the $50{,}000$ training samples, entering the over-parameterized regime. By contrast, QT models, which use around $23{,}258$ parameters, remain below this threshold, explaining their improved generalization behavior. These observations collectively suggest that QT not only provides parameter efficiency but also

doi:10.6342/NTU202504677

helps mitigate overfitting, especially in data-limited scenarios.

2.6 Beyond MLP Mapping Models

Up to this point, multilayer perceptrons (MLPs) have been employed as the default choice for the classical mapping model $G_{\theta_{\text{mm}}}$ in the QT framework. However, this is not a fundamental restriction. In principle, any sufficiently expressive and efficient architecture can serve as the mapping mechanism to transform quantum measurement probabilities into the target neural network weights. In this section, we examine extensions beyond the standard MLP design. We begin with architectural variations of MLPs themselves, and then introduce a tensor-network (TN) based alternative, focusing on matrix product state (MPS) mappings. Such TN-based mappings offer a compact yet powerful representation, particularly suitable for high-dimensional cases. Both the theoretical motivation and empirical evaluations are presented.

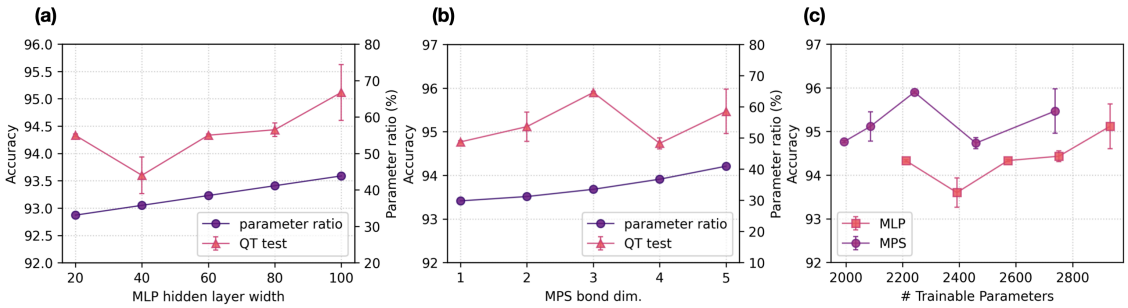


Figure 2.8: Performance of different mapping architectures within QT. (a) Impact of varying hidden layer width in MLP-based mappings. (b) Effect of bond dimension in MPS-based mappings. (c) Comparison of test accuracy against total parameter count for MLP and MPS mappings.

doi:10.6342/NTU202504677

2.6.1 Impact of MLP Architecture

To assess how MLP structure influences QT performance, we varied the width of the hidden layer in the mapping model $G_{\theta_{\text{mm}}}$ and evaluated classification accuracy on MNIST. The corresponding setting aligns with the "20" parameter in the mapping portion of Table 2.1. Results are shown in Fig. 2.8(a). As anticipated from Theorem 1, enlarging the hidden layer enhances the model's approximation ability. Test accuracy increases steadily with layer width, confirming that a more expressive mapping network enables QT to better approximate the target neural network weights. This provides empirical evidence consistent with the theoretical $O(1/\sqrt{h_g})$ scaling term in the QT approximation error bound.

2.6.2 Tensor Network Mapping Model

Although MLPs provide a natural and flexible baseline for the mapping model, the QT framework does not rely exclusively on this architecture. To explore the broader design space, we implemented a tensor-network (TN) based alternative, focusing on a matrix product state (MPS) structure. While developing a rigorous approximation bound for TN-based mappings is left for future work, our experiments demonstrate the feasibility and advantages of this structured approach within QT.

Tensor networks have proven to be powerful tools in both quantum many-body physics and machine learning, owing to their ability to efficiently capture correlations in high-dimensional data. Their compact representations make them particularly suitable for translating quantum measurement statistics into neural network parameters. Following the supervised MPS formulation in [70], we parameterize the mapping function as a contraction

doi:10.6342/NTU202504677

between an MPS weight tensor T and an input-dependent feature map $\Xi(\Psi_i^c)$. For each parameter $W_i \in \mathbb{R}$, the mapping is defined as:

$$T_{s_1,s_2,\ldots,s_{N+1}} = \sum_{\boldsymbol{\alpha}} A_{s_1}^{\alpha_1} A_{s_2}^{\alpha_1\alpha_2} \cdots A_{s_{N+1}}^{\alpha_N}, \tag{2.21}$$

where $\Psi_i^c \in \mathbb{R}^{N+1}$ (Eq. 2.4) denotes the concatenated basis state and its measurement probability. The associated feature map is expressed as

$$\Xi^{s_1,\ldots,s_{N+1}}(\mathbf{x}) = \xi^{s_1}(x_1) \otimes \xi^{s_2}(x_2) \otimes \cdots \otimes \xi^{s_{N+1}}(x_{N+1}), \tag{2.22}$$

with each local embedding

$$\xi^{s_j}(x_j) = \begin{bmatrix} x_j \\ 1 - x_j \end{bmatrix}. \tag{2.23}$$

The final mapping output is then

$$G_{\theta_{\text{TN}}}(\Psi_i^c) = T \cdot \Xi(\Psi_i^c) = W_i, \tag{2.24}$$

where the MPS core tensors $A_{s_j}^{\alpha_j}$ are trainable parameters and the bond dimension r controls expressivity. This design plays a role analogous to the hidden layer width in the MLP-based mapping (Eq. 2.4), but with a structured representation that can potentially scale more efficiently.

Empirical results are summarized in Fig. 2.8. Panel (b) shows that increasing the bond dimension consistently improves accuracy, reflecting the greater expressivity of larger MPS mappings. Panel (c) compares MPS- and MLP-based mappings under matched parameter budgets. Strikingly, the MPS approach achieves comparable or superior accuracy with fewer parameters, highlighting the promise of structured mappings as parameter-

doi:10.6342/NTU202504677

efficient alternatives within QT.

2.7 Comparison with Classical Compression Methods

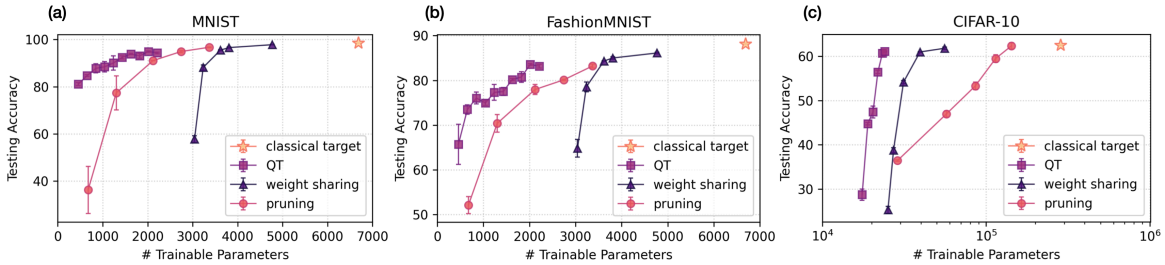


Figure 2.9: Testing accuracy versus number of trainable parameters for different compression strategies across MNIST, FashionMNIST, and CIFAR-10. The original full classical models are marked by orange stars. QT results are shown in purple (squares), weight sharing in blue (triangles), and pruning in red (circles). For CIFAR-10, the horizontal axis is plotted on a logarithmic scale to highlight the large baseline parameter count and the substantial reduction achieved by QT. Across datasets, the curves illustrate the trade-off between parameter count and performance, with QT consistently maintaining strong accuracy even at significantly reduced parameter budgets.

While QT demonstrates a clear ability to reduce the number of trainable parameters, it is important to benchmark its performance against established compression techniques from classical machine learning. A wide variety of approaches exist in this domain [59], including network pruning [6], weight sharing [61], low-rank matrix and tensor decomposition [36], and knowledge distillation [23, 42]. More recently, parameter-efficient fine-tuning methods such as Low-Rank Adaptation (LoRA) [30] have been widely adopted in large-scale settings.

For the purposes of this comparison, we focus on methods that, like QT, directly reduce the number of parameters to be trained, and do so without requiring pre-trained teacher models. Based on these criteria, we select pruning, weight sharing, and LoRA as representative baselines. In the following subsections, we first compare QT against pruning and weight sharing, then examine its relationship with LoRA, both as a point of

doi:10.6342/NTU202504677

comparison and as a potential complementary strategy.

2.7.1 Weight Sharing

Weight sharing is among the simplest approaches for reducing model size, relying on the reuse of weights across layers or within the structure of a single layer. The main challenge lies in determining which weights can be shared, and to what extent, without incurring unacceptable loss of performance for a given task [59]. Existing methods often cluster similar weights and replace them with a shared centroid, or incorporate regularization terms in the training objective to encourage weights to converge toward common values. However, such techniques frequently assume access to a pre-trained model in order to identify weight similarities, which can limit their applicability in scenarios where training must begin from scratch.

In this work, we employ a more direct variant of weight sharing that avoids dependence on pre-trained models. Specifically, we represent the weight matrix $W \in \mathbb{R}^{m \times n}$ using a smaller set of shared row vectors $\mathbf{v}_k \in \mathbb{R}^n$ with $k < m$. Each row W_i is then assigned to one of these shared vectors in sequence. By adjusting the number of distinct shared vectors, we can directly control the number of trainable parameters and thus tune the balance between model compactness and predictive performance.

This method offers a simple yet effective means of regulating model size during training. As shown in Fig. 2.9, test accuracy under weight sharing (blue triangles) is reported for MNIST, FashionMNIST, and CIFAR-10. Each point corresponds to the average of three independent runs, confirming the robustness of the approach. The results illustrate how reducing the number of distinct weight vectors gradually lowers accuracy, thereby

doi:10.6342/NTU202504677

making explicit the trade-off between compression and performance.

2.7.2 Pruning

The second baseline considered against QT is pruning, one of the most established techniques for reducing neural network complexity. In its conventional form, pruning is applied after pre-training: weights deemed less important are removed to obtain a smaller, more efficient network. This strategy typically reduces storage requirements, improves runtime efficiency, and maintains performance after retraining the pruned model. For fairness, however, our comparison avoids reliance on pre-trained models. Instead, we implement random pruning, in which a fixed proportion of weights is eliminated at initialization according to a predefined pruning ratio. The selection of pruned weights is random, and to mitigate stochastic variability, each setting is repeated three times and the results averaged. As shown in Fig. 2.9, pruning performance is reported as red circles for MNIST, FashionMNIST, and CIFAR-10. The results highlight the expected trade-off: as the pruning ratio increases, model accuracy progressively declines, though moderate pruning retains competitive performance.

2.7.3 Comparison of QT and Other Methods

Fig. 2.9 summarizes the results of QT, weight sharing, and pruning across the three benchmark datasets. The accuracy trends reveal the inherent trade-off between parameter count and predictive performance for all compression methods. Notably, QT consistently achieves higher or comparable accuracy relative to weight sharing and pruning, despite operating with substantially fewer parameters. This consistent advantage across MNIST,

doi:10.6342/NTU202504677

FashionMNIST, and CIFAR-10 underscores QT's ability to retain strong predictive power while dramatically reducing trainable parameters, demonstrating its potential as a practical alternative to classical compression techniques.

2.7.4 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) [30] represents a modern parameter-efficient technique that has proven highly effective in large-scale deep learning, particularly for fine-tuning foundation models. Unlike pruning or weight sharing, LoRA is based on a low-rank decomposition of parameter updates, enabling efficient adaptation while freezing the majority of pre-trained weights. Specifically, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, the update is parameterized as:

$$W_0 + \Delta W = W_0 + BA, \tag{2.25}$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$. Here, only the low-rank factors A and B are optimized during fine-tuning, significantly reducing the number of trainable parameters while maintaining or even improving performance. Conceptually, LoRA shares a similar philosophy with QT: both aim to compress parameter representations while preserving expressivity.

In our experiments, LoRA is applied to all layers of the same CNN baseline used in earlier sections, with rank values $r \in \{1, 2, 4\}$. The results, shown in Fig. 2.10, indicate that LoRA achieves higher accuracy than QT under these settings, and also surpasses the performance of weight sharing and pruning in Fig. 2.9.

Importantly, QT and LoRA need not be viewed as competing approaches. Since QT

doi:10.6342/NTU202504677

functions as a parameter generation framework rather than being tied to specific parameter structures, it can also be adapted to generate the LoRA factors A and B. This opens the possibility of combining the benefits of both methods, yielding compact low-rank updates generated through quantum-assisted parameterization.

2.7.5 Combination of QT and LoRA

A key advantage of QT is its modular design, which allows integration with classical compression techniques such as LoRA. In this hybrid approach, QT serves as the parameter generation mechanism for the low-rank adaptation matrices introduced by LoRA. Instead of directly optimizing the LoRA matrices A and B, the QT framework generates their entries via quantum-classical mappings, effectively combining quantum parameter generation with low-rank decomposition.

Formally, QT produces the LoRA matrices in a manner analogous to Eq. **??**. For each element of the concatenated LoRA matrices, the mapping is expressed as:

$$G_{\theta_{\text{TN}}}(\Psi_i^c) = [A \mid B]_i, \tag{2.26}$$

where $[A \mid B]_i$ denotes the i-th element of the flattened matrices A and B, and Ψ_i^c is the corresponding quantum-classical input as defined in the QT framework. This formulation enables QT to generate the complete set of LoRA adaptation parameters using quantum-enhanced mappings.

In our experiments, we consider LoRA with rank $r = 4$, leading to matrices $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$, with a total of 1948 trainable parameters. The QT-LoRA hybrid employs an MPS-based mapping model with bond dimension 2, and QNNs of varying depths

doi:10.6342/NTU202504677

with block sizes $\{32, 48, 64, 80, 96, 112, 128, 192\}$. The corresponding qubit requirement is $\lceil \log_2 1948 \rceil = 11$.

The results, shown in Fig. 2.10, compare QT-LoRA against standalone QT and standalone LoRA under different parameter budgets. Notably, QT-LoRA consistently outperforms LoRA in the low-parameter regime, where quantum-generated parameters help preserve expressivity despite tighter constraints. These findings suggest that QT-LoRA represents a promising hybrid compression strategy, combining the structural efficiency of low-rank adaptation with the generative power of quantum parameterization.

While these results highlight the flexibility of QT in augmenting state-of-the-art classical methods, we emphasize that this section serves primarily as an illustration of QT's versatility. The central focus of this work remains on the theoretical foundations and core performance of QT itself.

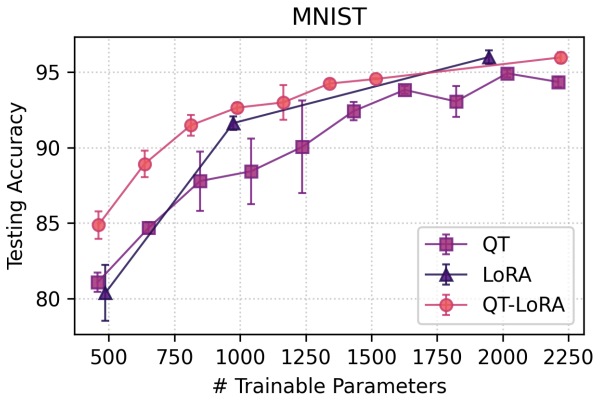


Figure 2.10: Comparison of QT, LoRA, and QT-LoRA on MNIST. QT-LoRA integrates quantum parameter generation with low-rank adaptation by generating the LoRA matrices via an MPS-based quantum mapping. The hybrid approach outperforms standalone LoRA in the low-parameter regime, demonstrating the benefits of combining quantum-enhanced mappings with classical low-rank compression.

doi:10.6342/NTU202504677

2.8 Effect of Noise and Finite Measurement Shots

Having demonstrated the flexibility of QT, including its integration with classical compression techniques, we now focus on its robustness under realistic quantum conditions. In particular, we investigate the impact of noise and finite measurement shots, two key challenges in near-term quantum hardware, with special attention to how they affect learning dynamics and gradient estimation.

2.8.1 Model Accuracy Across Configurations

To evaluate robustness, we benchmark QT under finite-shot settings and simulated hardware noise. For the finite-shot case, gradients are computed using the parameter-shift rule [68]. Tables 2.4 and 2.5 report results on the MNIST-10 classification task, comparing MLP-based and MPS-based mapping models. In these experiments, the QNN circuit depth is fixed to 151 blocks. For the mapping models, the MLP configuration follows earlier sections, while the MPS model is implemented with bond dimension 4. Noise is introduced using backend profiles from IBM Quantum simulators, specifically `ibm_fez` and `ibm_torino`, which emulate realistic device-level noise [1]. Although these simulations do not correspond to direct execution on hardware, they provide a useful proxy for understanding QT behavior in noisy environments.

Results show a clear contrast between the two mapping models. QT with MLP mapping suffers substantial performance degradation under both hardware noise and shot noise, with test accuracy falling sharply compared to noiseless simulation. By contrast,

[1] IBM Quantum provides noise models derived from the calibration data of physical backends: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.providers.aer.noise.NoiseModel.

doi:10.6342/NTU202504677

QT with an MPS-based mapping maintains significantly higher performance under the same conditions. The structured representation provided by MPS appears to mitigate the instability caused by noisy gradients and finite statistics. These findings suggest that incorporating structured mappings such as tensor networks can improve the resilience of QT, enabling stable training even in non-ideal quantum environments. This provides a practical pathway toward noise-aware design of QT architectures, bridging the gap between idealized simulation and near-term hardware execution.

Table 2.4: Comparison of QT models with MLP mapping under different noise settings and measurement shots.

MLP Mapping (MNIST-10)		
Test acc. (%)	**# Shot count**	**Noise model**
94.33	∞	—
28.04	8192	—
27.47	8192	ibm_fez
18.51	8192	ibm_torino
28.11	8.2×10^4	—
26.85	8.2×10^4	ibm_fez
18.18	8.2×10^4	ibm_torino
22.73	3.2×10^5	—
23.99	3.2×10^5	ibm_fez
26.95	3.2×10^5	ibm_torino

Table 2.5: Comparison of QT models with MPS mapping under different noise settings and measurement shots.

MPS Mapping (MNIST-10)		
Test acc. (%)	**# Shot count**	**Noise model**
94.40	∞	—
83.52	8192	—
85.89	8192	ibm_fez
88.23	8192	ibm_torino
87.33	8.2×10^4	—
83.85	8.2×10^4	ibm_fez
82.39	8.2×10^4	ibm_torino
86.17	3.2×10^5	—
83.75	3.2×10^5	ibm_fez
83.89	3.2×10^5	ibm_torino

doi:10.6342/NTU202504677

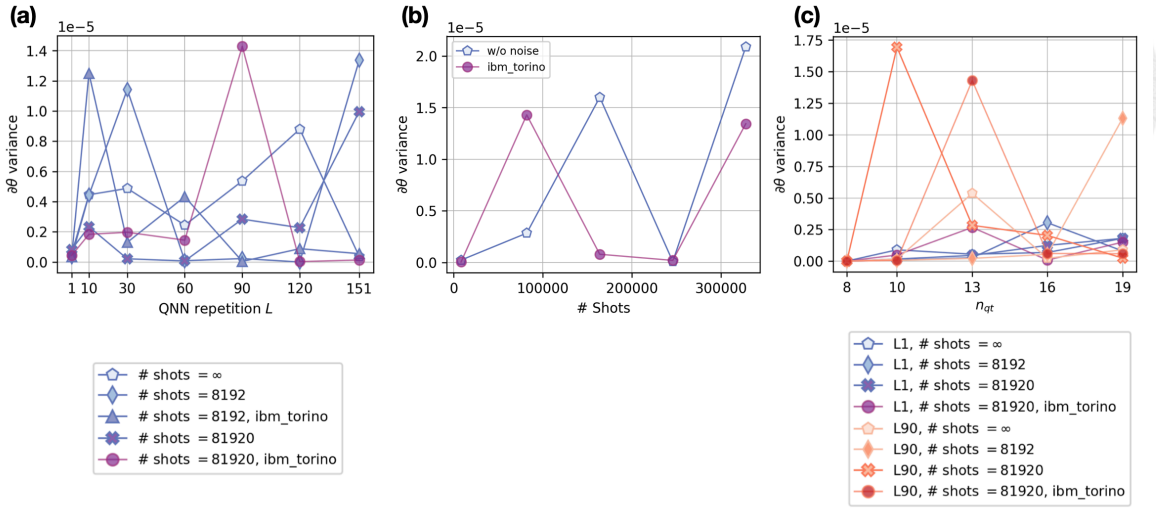


Figure 2.11: Gradient variance analysis for QNN parameters within the QT framework under different configurations. (a) Variance as a function of the number of QNN repetitions L. (b) Variance as a function of measurement shot count. (c) Variance across different numbers of qubits n_{qt}. Across all settings, the gradients remain stable, with no evidence of vanishing behavior or barren plateaus.

2.8.2 Variance of QNN Gradients

Beyond accuracy, it is important to examine how noise and finite-shot sampling affect gradient behavior in QT. A central design principle of QT is the use of expressive QNNs —often constructed from universal gate sets—to approximate the distribution of classical NN weights. While this expressivity is beneficial, it raises concerns about the dimensionality of the underlying dynamical Lie algebra (DLA), which in other variational quantum algorithms (VQAs) has been linked to barren plateaus [65]. Such plateaus manifest as exponentially vanishing gradients, severely hindering optimization. In this subsection, we empirically investigate the variance of QNN gradients under different conditions and discuss why QT appears more robust compared to expectation-value–based approaches.

Fig. 2.11 reports gradient variance across three key factors: (a) QNN depth L, (b) measurement shot count, and (c) qubit number N. In Fig. 2.11(a), we vary the QNN depth from $L \in \{1, 10, 30, 60, 90, 120, 151\}$, tracking gradient variance during training under

doi:10.6342/NTU202504677

both noiseless simulation and noisy simulation with IBM's `ibm_torino` noise model. For finite-shot experiments, we consider shot counts of $\{\infty, 8192, 81920\}$. In Fig. 2.11(b), we fix the depth to $L = 90$ and analyze how gradient variance scales with increasing shot counts, ranging from 8192 to 8192×40, again under both noiseless and noisy conditions. In Fig. 2.11(c), we explore the effect of qubit count by varying $n_{\text{qt}} \in \{8, 10, 13, 16, 19\}$. Both shallow circuits ($L = 1$) and deeper circuits ($L = 90$) are considered, with the qubit count controlled by adjusting the size of the corresponding classical neural network. Each configuration is tested across multiple shot counts and noise conditions.

Across all experiments, we observe no systematic suppression of gradients and no evidence of barren plateaus. The gradients remain stable and informative even under finite-shot sampling and realistic noise, in contrast to the exponential decay typical of deep VQAs. This robustness suggests that QT avoids the pathological gradient landscapes that often plague variational training. To explain this empirically observed stability, we note that QT optimizes a probability-based objective derived directly from measurement distributions, rather than expectation values of Pauli observables. This structural difference appears to mitigate the concentration-of-measure effects that drive barren plateau phenomena in standard VQAs, allowing QT to maintain usable gradients even in deep and noisy settings.

Variance collapse in expectation-value gradients. Let $\mathcal{O} = \sum_i \lambda_i |\phi_i\rangle\langle\phi_i|$ be an observable diagonal in the computational basis (e.g., Pauli-Z operators or energy observables). Its expectation value with respect to a parameterized state $|\psi(\theta)\rangle$ is

$$\langle \mathcal{O} \rangle_\theta = \langle \psi(\theta) | \mathcal{O} | \psi(\theta) \rangle = \sum_{i=1}^{2^n} \lambda_i \, p_i(\theta), \tag{2.27}$$

doi:10.6342/NTU202504677

where $p_i(\theta)$ denotes the probability of outcome i. Differentiating with respect to a circuit parameter yields

$$\frac{\partial \langle \mathcal{O} \rangle_\theta}{\partial \theta} = \sum_{i=1}^{2^n} \lambda_i \frac{\partial p_i(\theta)}{\partial \theta}. \tag{2.28}$$

In deep or highly expressive circuits, many of the terms $\lambda_i \cdot \frac{\partial p_i}{\partial \theta}$ are individually small and often carry opposite signs. Their aggregation introduces destructive cancellations, leading to a pronounced reduction in gradient variance:

$$\mathrm{Var}\left(\frac{\partial \langle \mathcal{O} \rangle}{\partial \theta}\right) \ll \mathrm{Var}\left(\frac{\partial p_i}{\partial \theta}\right). \tag{2.29}$$

This "variance collapse" effect is exacerbated when (1) the eigenvalue spectrum $\{\lambda_i\}$ includes alternating signs or irregular structure, and (2) the Hilbert space dimension grows, as is typical in deep variational quantum circuits.

Probability-based gradients in QT. In contrast, QT bypasses expectation values entirely. The QNN outputs the measurement probabilities directly:

$$C_i(\theta) = p_i(\theta) = |\langle \phi_i | \psi(\theta) \rangle|^2, \tag{2.30}$$

with gradient

$$\frac{\partial C_i(\theta)}{\partial \theta} = \frac{\partial p_i(\theta)}{\partial \theta}. \tag{2.31}$$

Because no summation over basis states is performed, there are no cancellation effects. Consequently,

$$\mathrm{Var}\left(\frac{\partial C_i}{\partial \theta}\right) \gg \mathrm{Var}\left(\frac{\partial \langle \mathcal{O} \rangle}{\partial \theta}\right). \tag{2.32}$$

Thus, while QT still propagates gradients through classical mappings via the chain rule, the quantum component $\frac{\partial p}{\partial \theta}$ retains higher variance and remains robust. This structural

doi:10.6342/NTU202504677

property explains why QT avoids the barren plateau behavior that plagues expectation-based VQAs.

Empirically, this robustness is confirmed in Fig. 2.11, where gradient variance remains stable across increasing depth, qubit count, and under noisy or finite-shot conditions. By working directly with measurement probabilities, QT preserves gradient signal strength, enabling scalable optimization even in practical NISQ environments.

2.9 Practicality of Quantum-Train

Over the past decades, the size and complexity of machine learning models have grown at an extraordinary pace, with parameter counts rising exponentially. This trend is expected to continue, ultimately reaching a point where training such massive models becomes infeasible with existing hardware or prohibitively costly to scale. Even today, doubling GPU resources does not reliably yield proportional speedups. By contrast, the QT framework leverages only a polylogarithmic number of parameters in generating classical models, potentially shifting the paradigm away from linear hardware scaling and toward quantum-assisted training efficiency.

A key challenge in many quantum machine learning (QML) approaches lies in data encoding. Methods such as angle encoding scale poorly: larger input data requires deeper circuits and more qubits, often making them impractical. Preprocessing and compressing data before encoding can reduce circuit depth, but risks discarding valuable information. QT circumvents this bottleneck by maintaining classical inputs and outputs, using quantum states not to encode data but to generate classical neural network (NN) parameters. In this way, QT retains the advantages of Hilbert space expressivity while avoiding the

doi:10.6342/NTU202504677

overhead of direct quantum data encoding.

The QCML baseline aims to combine the strengths of classical preprocessing and postprocessing with quantum blocks in the middle. While QCML demonstrates parameter efficiency and strong performance (see Fig. 2.6 and Fig. 2.7), both QCML and standard QML architectures require access to quantum devices for inference. This requirement presents a major scalability barrier, particularly as recent LLM studies show that inference often dominates deployment cost [66]. By contrast, QT produces a fully classical model after training, deployable without any quantum resources. This property makes QT especially attractive in practice, bridging the gap between quantum-enhanced training and classical inference.

To assess QT under realistic conditions, we extend beyond idealized state vector simulations and examine robustness under hardware noise models and finite-shot sampling. Experiments with IBM's `ibm_fez` and `ibm_torino` noise profiles show that QT maintains strong performance, especially when paired with structured mappings such as tensor networks (MPS). These results underscore QT's viability in near-term settings where noise and limited shots are unavoidable.

The scalability of QT also aligns well with advances in fault-tolerant quantum computing. Recent demonstrations include IBM's fault-tolerant quantum memory architecture that preserves 12 logical qubits for nearly one million syndrome cycles using 288 physical qubits at 0.1% error rate [9], and neutral-atom platforms achieving 48 logical qubits with hypercube connectivity [7]. Since QT requires only $N = \lceil \log_2 M \rceil$ qubits for a model with M parameters, a billion-parameter model would demand merely 30 qubits. Unlike data-encoding approaches, which often require qubits proportional to input size, QT's

doi:10.6342/NTU202504677

qubit requirement scales polylogarithmically with model size, making it compatible with anticipated near-term logical qubit capacities.

We also investigate barren plateau phenomena, a common obstacle in deep variational circuits. Empirical results across varying depth, qubit count, and finite-shot settings show no evidence of systematic gradient suppression. QT' s probability-based gradients remain stable, unlike expectation-value–based objectives prone to collapse. This inherent robustness reduces the need for elaborate mitigation strategies, though continued advances in ansatz design and initialization [62, 80] will further strengthen QT' s resilience.

Finite-shot effects are another important consideration. The QT approximation error bound in Theorem 1 includes an $O(1/\sqrt{m})$ term, reflecting the uncertainty from limited measurement shots. While additional shots can improve accuracy, our empirical results reveal no strict monotonic relationship—performance may also depend on architecture and noise settings. This highlights the need for efficient shot allocation strategies. Insights from quantum tomography show that achieving fidelity $F(|\psi\rangle, |\phi\rangle) \geq 1 - \epsilon$ can require exponentially many samples in N [24]. In our experiments, scaling shots linearly with 2^N provides a reasonable balance between cost and performance. Further, generative approaches to quantum state tomography [1] could be integrated into QT to reduce sampling costs by simulating quantum outcomes efficiently.

In summary, QT addresses three major limitations faced by both QML and classical compression: (1) the difficulty of quantum data encoding, (2) the inefficiency of classical compression techniques, and (3) the high inference costs of quantum-enhanced models. By generating classical NN weights via quantum measurements and mapping them through classical models (e.g., MLPs or MPS), QT enables scalable, parameter-efficient

doi:10.6342/NTU202504677

training while producing models that run entirely on classical hardware. Theoretical analysis establishes error bounds that clarify trade-offs between quantum circuit depth, mapping expressivity, and shot count, while empirical studies show competitive performance compared to weight sharing and pruning. Moreover, QT integrates naturally with modern methods such as LoRA, and even surpasses LoRA in the low-parameter regime when used in the QT-LoRA configuration.

The ability to train with quantum resources yet deploy classically positions QT as a practical and forward-looking hybrid paradigm. While our current focus is on classification tasks, the framework extends naturally to reinforcement learning, generative modeling, and large-scale architectures. Continued development of hardware-specific designs, structured mappings, and hybrid training strategies may further unlock QT' s potential, contributing to scalable quantum-classical machine learning in the era of increasingly complex models.

doi:10.6342/NTU202504677

Chapter 3 Quantum Parameter Adaptation

Chapter Overview

In this chapter we extend the principles of Quantum-Train (Chapter 2) to the fine-tuning of large pre-trained models. We introduce Quantum Parameter Adaptation (QPA), a framework that leverages parameterized quantum circuits (PQCs) as generators for the trainable parameters of parameter-efficient fine-tuning (PEFT) methods. QPA offers a compression perspective, enabling significant reductions in the number of trainable parameters for large language models (LLMs) while maintaining competitive or improved performance. The results presented in this chapter are based on our published work at ICLR 2025.

doi:10.6342/NTU202504677

3.1 Beyond Quantum-Train

Earlier studies of Quantum-Train (QT) [41, 50] have primarily targeted small- to medium-scale models with fewer than one million parameters, the largest reaching approximately 0.28M. While these results establish proof-of-concept, they do not yet capture the scale of contemporary machine learning practice. To convincingly demonstrate the practicality of quantum-assisted parameter generation, it is necessary to scale QT beyond benchmark tasks such as handwritten digit classification or simplified reinforcement learning environments, toward model sizes and domains that reflect modern usage.

Scaling is now central to both quantum and classical machine learning. The rise of large language models (LLMs)—including GPT-2 [64], GPT-3 [10], and the more recent Gemma-2 [71]—illustrates this trend. These models, with billions of parameters, underpin a wide range of natural language applications but pose significant computational challenges. Full fine-tuning of LLMs is often prohibitively expensive, motivating the development of parameter-efficient fine-tuning (PEFT) methods such as Low-Rank Adaptation (LoRA) [30], Weight-Decomposed LoRA (DoRA) [52], Prefix-Tuning (PT) [40, 79], and adapters [29, 43]. These approaches substantially reduce the number of trainable parameters while maintaining strong downstream performance, yet the balance between efficiency and effectiveness remains an open challenge.

Building on the insight that parameterized quantum circuits (PQCs) can function as generative models for parameters, we propose to extend this principle to the PEFT setting. Specifically, we introduce **Quantum Parameter Adaptation (QPA)**, a framework that combines quantum parameter generation with classical mapping models (e.g., MLPs) to produce PEFT parameters. Rather than training LoRA matrices, DoRA decompositions,

doi:10.6342/NTU202504677

or adapter modules directly, QPA uses quantum–classical mappings to generate these parameters from a compact set of quantum circuit parameters, further reducing trainable complexity.

Figure 3.1 summarizes the QPA framework, situating it alongside standard QML, quantum parameter generation, and PEFT approaches. QPA integrates these perspectives by generating the additional PEFT parameters required for LLM adaptation via QNNs, while preserving classical inference after training.

Empirical results on Gemma-2 [71] and GPT-2 [64] demonstrate that QPA can reduce the parameter footprint of LoRA, DoRA, PT, and Feed-Forward Adapters (FFAs) by nearly an order of magnitude (e.g., $10^6 \rightarrow 10^5$), while achieving comparable or even superior perplexity on text generation tasks. To the best of our knowledge, QPA represents **the first application of quantum computing to fine-tuning large-scale classical LLMs**, bridging parameter-efficient quantum learning with practical NLP applications. Crucially, inference remains entirely classical, ensuring deployability without runtime quantum resources.

Theoretical analysis of QPA's convergence, trainability, and learnability properties is an important direction for future work. These investigations will further clarify the role of quantum parameter generation in modern parameter-efficient adaptation, and its potential to extend beyond language models into other large-scale domains.

3.2 Parameter-Efficient Fine-Tuning (PEFT) Methods

Fine-tuning large language models (LLMs) is computationally demanding due to their vast parameter counts. Parameter-Efficient Fine-Tuning (PEFT) methods address

doi:10.6342/NTU202504677

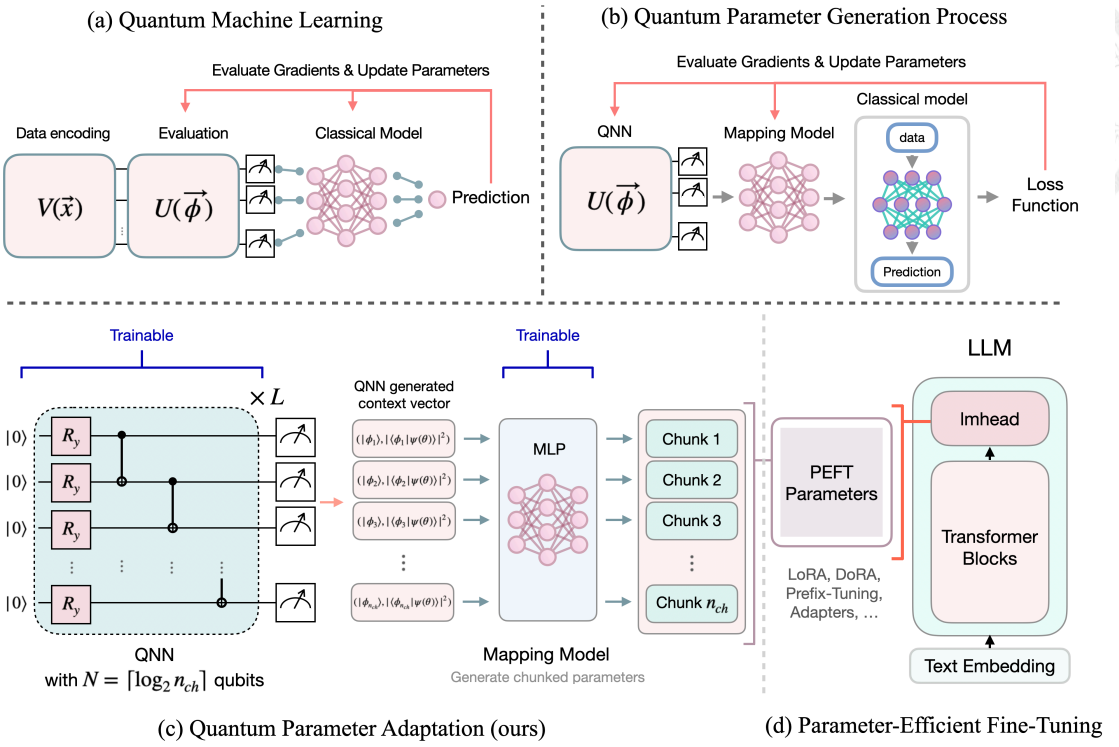


Figure 3.1: Overview of (a) Quantum Machine Learning [53, 56], (b) Quantum Parameter Generation [47], (c) our proposed Quantum Parameter Adaptation (QPA), and (d) Parameter-Efficient Fine-Tuning methods [29].

this challenge by reducing the number of trainable parameters while preserving, or in some cases even improving, task performance. Among the most widely adopted PEFT techniques are Low-Rank Adaptation (LoRA) [30] and Weight-Decomposed LoRA (DoRA) [52]. These methods are motivated by the observation that fine-tuning primarily induces low-rank changes in the model' s weight space. Instead of updating the full weight matrix, they introduce small trainable matrices that represent low-rank decompositions of the update. This significantly reduces the trainable parameter budget while still capturing the essential adaptations required for new tasks. Another family of approaches leverages adapters, lightweight feed-forward layers inserted between existing layers of a pre-trained model [29, 43]. During adaptation, only the parameters of these inserted modules are updated, while the original model weights remain frozen. This modular strategy minimizes training overhead and enables efficient reuse across tasks. Prefix-Tuning (PT) [40, 79]

doi:10.6342/NTU202504677

adopts a different philosophy: rather than modifying internal weight structures, it introduces tunable prefix vectors that are prepended to the input embeddings or hidden states at each layer. During fine-tuning, only these prefix vectors are learned, while the core model remains untouched. Collectively, these PEFT methods reduce computational and memory costs by constraining fine-tuning to smaller, task-specific components. This efficiency has been key to enabling large-scale LLM adaptation in practice and provides the foundation for integrating quantum parameter generation within fine-tuning pipelines.

3.3 Quantum Parameter Generation for Efficient Adaptation

In methods such as LoRA and DoRA, a central assumption is that weight updates can be expressed as low-rank matrices. While effective, these approaches are inherently restricted to discrete rank choices $r \in \mathbb{Z}^+$. Consequently, the parameter space between ranks r and $r+1$ remains unexplored. This rigidity limits the ability to perform more fine-grained adjustments, potentially constraining adaptation performance. Similar constraints appear in other PEFT techniques—for example, in Prefix-Tuning (PT), the number of trainable parameters is tightly coupled to the input sequence length, restricting flexibility. To overcome these limitations, we draw inspiration from quantum circuit–based compression techniques, where parameterized quantum circuits (PQCs) and classical mapping models are jointly employed to represent large neural networks using far fewer parameters. Extending this idea, we hypothesize that PQC-based quantum neural networks (QNNs), when combined with mapping models, can serve as efficient parameter generators for PEFT methods. The key insight is that only a small number of QNN pa-

doi:10.6342/NTU202504677

rameters are needed to control measurement probabilities, yet these probabilities span a high-dimensional Hilbert space. This property suggests that quantum parameter generation enables a compact yet expressive representation of fine-tuning updates. In particular, leveraging the Hilbert space structure allows QPA to interpolate smoothly between parameter regimes that would otherwise be discrete (e.g., between low-rank settings in LoRA), thereby providing a more detailed and flexible adaptation mechanism. Such quantum-enhanced parameterization offers the potential to improve both efficiency and effectiveness in PEFT, moving beyond the rigidity of classical designs.

3.3.1 Quantum Circuit–Based Model Parameter Generation

We now describe the parameter generation process that differentiates QPA from conventional QML approaches. Consider a target neural network (NN) with parameters $\boldsymbol{a} = (a_1, a_2, \ldots, a_m)$, where m is the number of parameters. To generate these parameters, we construct a parameterized quantum circuit (PQC) with $N = \lceil \log_2 m \rceil$ qubits and L layers, defined by the ansatz:

$$|\psi(\boldsymbol{\theta})\rangle = \left(\prod_{i=1}^{N-1} \text{CNOT}^{i,i+1} \prod_{j=1}^{N} R_Y^j(\theta_j^{(L)}) \right)^L |0\rangle^{\otimes N}, \tag{3.1}$$

where R_Y^j denotes a tunable single-qubit rotation on qubit j, indexed by layer L, and CNOT is the controlled-NOT gate. This PQC spans a Hilbert space of size $2^N \geq m$, producing 2^N measurement probabilities $p_i(\boldsymbol{\theta}) = |\langle \phi_i | \psi(\boldsymbol{\theta}) \rangle|^2 \in [0, 1]$ for basis states $|\phi_i\rangle$.

The parameter count of $\boldsymbol{\theta}$ depends on both N and L. Typically, L scales as $O(N)$ or $O(N^2)$ depending on the circuit family [4, 12, 69], but can more generally be polynomial

doi:10.6342/NTU202504677

in N. Thus, with $O(\text{polylog}(m))$ PQC parameters, we obtain at least m measurement outcomes, which can then be mapped to the target NN parameters.

Mapping Model. The measurement probabilities are mapped to the target parameters $\boldsymbol{a} \in \mathbb{R}^m$ using a classical mapping model G with parameters $\boldsymbol{b}$. The input to G is the binary encoding of the basis state (length N) concatenated with the corresponding probability, yielding:

$$G_{\boldsymbol{b}}(|\phi_i\rangle, p_i(\boldsymbol{\theta})) = a_i, \quad \forall i \in \{1, \ldots, m\}. \tag{3.2}$$

Only the first m basis states are used, ensuring full coverage of all target parameters. Since the input dimension is $N+1$, the parameter size of $\boldsymbol{b}$ also scales as $O(\text{polylog}(m))$. Together, the PQC and $G_{\boldsymbol{b}}$ serve as a parameter generator for the NN, with $\boldsymbol{\theta}$ and $\boldsymbol{b}$ tuned to minimize a task-specific loss $\mathcal{L}$.

Gradient Estimation. Because the NN parameters $\boldsymbol{a}$ are implicitly defined through the PQC–mapping pipeline, the gradient of the loss with respect to the quantum parameters is:

$$\nabla_{\boldsymbol{\theta},\boldsymbol{b}} \mathcal{L} = \left(\frac{\partial \boldsymbol{a}}{\partial(\boldsymbol{\theta}, \boldsymbol{b})} \right)^T \cdot \nabla_{\boldsymbol{a}} \mathcal{L}, \tag{3.3}$$

where $\frac{\partial \boldsymbol{a}}{\partial(\boldsymbol{\theta},\boldsymbol{b})}$ is the Jacobian describing the sensitivity of the classical parameters $\boldsymbol{a}$ to the quantum parameters $(\boldsymbol{\theta}, \boldsymbol{b})$. In practice, gradients with respect to $\boldsymbol{\theta}$ are estimated using the parameter-shift rule and its variants [56, 68].

Parameter Update. The quantum and classical parameters are updated via:

$$\boldsymbol{\theta}_{t+1}, \boldsymbol{b}_{t+1} = \boldsymbol{\theta}_t, \boldsymbol{b}_t + \eta \nabla_{\boldsymbol{\theta},\boldsymbol{b}} \mathcal{L}, \tag{3.4}$$

doi:10.6342/NTU202504677

where η is the learning rate. This update ensures joint optimization across the quantum and classical components of the generator.

Applications. This parameter generation approach has been applied to multiple domains: CNNs for image classification [46, 47, 49], LSTMs for time-series forecasting (e.g., flood prediction) [41], and policy-gradient reinforcement learning in CartPole and MiniGrid [50]. Across these tasks, PQC-based parameter generation achieved substantial reductions in trainable parameter counts while maintaining competitive performance.

Table 3.1: Configuration of the mapping model $\tilde{G}_{\boldsymbol{b}}$, with N representing the number of qubits for each task.

Hyperparameter	**Meaning**	**Value**
Input size	Input of the mapping model $(\|\phi_i\rangle, \|\langle\phi_i\|\psi(\boldsymbol{\theta})\rangle\|^2)$	$N+1$
Hidden dimension	Main structure of the MLP mapping model	[32, 64, 128, 128, 64, 32, n_{mlp}]

3.4 Batched Parameter Generation in Quantum Parameter Adaptation

For a target NN model with m parameters, the required number of qubits in quantum parameter generation is

$$N = \lceil \log_2 m \rceil.$$

For example, scaling to $m = 10^9$ (one billion) parameters would require $N = 30$ qubits. While a 30-qubit simulation is within reach of current quantum hardware and classical simulators, the practical cost is prohibitive: simulating 2^{30} amplitudes requires roughly 16 GB of GPU memory and several seconds per circuit evaluation [22], rendering it unsuitable for machine learning workloads that demand repeated, large-scale training iterations.

doi:10.6342/NTU202504677

Chunked parameter generation. To address this challenge, we introduce a batched generation strategy in which the m parameters of the target NN are divided into n_{ch} chunks, each of size n_{mlp}. Thus,

$$n_{ch} = \left\lceil \frac{m}{n_{mlp}} \right\rceil.$$

In this formulation, each quantum basis state is mapped to a batch of n_{mlp} parameters instead of a single one.

The mapping model, now denoted $\tilde{G}_{\boldsymbol{b}}$, takes as input the computational basis state $|\phi_i\rangle$ and its corresponding probability $p_i(\boldsymbol{\theta}) = |\langle\phi_i|\psi(\boldsymbol{\theta})\rangle|^2$, and outputs an entire chunk of parameters:

$$\boldsymbol{a} = (\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_{n_{ch}}), \tag{3.5}$$

$$\tilde{G}_{\boldsymbol{b}}(|\phi_i\rangle, p_i(\boldsymbol{\theta})) = \tilde{a}_i, \quad \forall i \in \{1, \ldots, n_{ch}\}, \tag{3.6}$$

$$\tilde{a}_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,n_{mlp}}). \tag{3.7}$$

This expansion is implemented by using a decoder-style MLP in $\tilde{G}_{\boldsymbol{b}}$, where the output dimension per basis state is increased from 1 to n_{mlp} (see Table 3.1).

Qubit reduction. With chunking, the number of qubits required becomes:

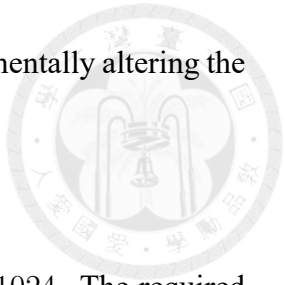
$$N = \left\lceil \log_2 n_{ch} \right\rceil = \left\lceil \log_2 \left\lceil \tfrac{m}{n_{mlp}} \right\rceil \right\rceil, \tag{3.8}$$

effectively reducing the qubit count by approximately $\lceil \log_2 n_{mlp} \rceil$ compared to the original scheme. The standard method is recovered as the special case $n_{mlp} = 1$.

While the enlarged output layer of $\tilde{G}_{\boldsymbol{b}}$ introduces additional classical parameters, the memory required to store the quantum state decreases by a factor of $1/n_{mlp}$. This

doi:10.6342/NTU202504677

reduction substantially improves simulation feasibility without fundamentally altering the quantum-to-classical parameter generation process.



Example. Consider $m = 10^9$ parameters with chunk size $n_{mlp} = 1024$. The required qubit count drops from $N = 30$ to

$$N = \left\lceil \log_2 \left\lceil \frac{10^9}{1024} \right\rceil \right\rceil = 20.$$

This 33% reduction in qubits translates to a memory saving of $1/1024$, as the state dimension shrinks from 2^{30} to 2^{20}. Such savings not only ease hardware requirements but also accelerate classical simulation, making batched generation a practical enabler for scaling QPA to billion-parameter models.

3.4.1 From Model Tuning to General Parameter-Tuning Tasks

In QPA, the scope of quantum parameter generation extends beyond producing the full parameter set of a target NN. Instead, it focuses on generating the parameters of a parameter-efficient fine-tuning (PEFT) method, thereby enabling significantly larger models and tasks to be addressed. When combined with batched parameter generation, this approach further amplifies scalability. Following the earlier notation, $\boldsymbol{a}$ now denotes the trainable parameters of a chosen PEFT method.

LoRA case. Consider Low-Rank Adaptation (LoRA), where a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ is updated as

$$W_0 + \Delta W = W_0 + BA,$$

doi:10.6342/NTU202504677

with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and rank $r \ll \min(d,k)$. In this setting, QPA generates the elements of A and B according to Eq. 3.6 and Eq. 3.7, so that $\boldsymbol{a}$ corresponds precisely to the LoRA parameters. The required qubit count becomes

$$N = \left\lceil \log_2 \left\lceil \frac{r(d+k)}{n_{mlp}} \right\rceil \right\rceil \quad \text{(LoRA case)}. \tag{3.9}$$

DoRA case. For Weight-Decomposed LoRA (DoRA), the decomposition introduces an additional magnitude vector of length k, yielding a total parameter count of $r(d+k)+k$. The qubit requirement is therefore

$$N = \left\lceil \log_2 \left\lceil \frac{r(d+k)+k}{n_{mlp}} \right\rceil \right\rceil \quad \text{(DoRA case)}. \tag{3.10}$$

Generalization. These formulations illustrate how QPA adapts naturally to a wide range of PEFT methods, by interpreting $\boldsymbol{a}$ as the corresponding adaptation parameters. The schematic in Fig. 2.1 depicts this process in the context of a single LLM layer, which is the focus of the experiments described in Sec. 3.5. The parameter optimization follows the gradient evaluation and update rules in Eq. 3.3 and Eq. 3.4, while for tasks where gradient computation is infeasible, gradient-free optimizers such as Nelder‑Mead or COBYLA may be applied.

3.5 Performance of Quantum Parameter Adaptation

The goal of this section is to empirically evaluate the effectiveness of QPA as introduced in Sec. 3.3. Specifically, we aim to test whether QPA can significantly reduce the number of trainable parameters while preserving or surpassing the performance of es-

doi:10.6342/NTU202504677

tablished PEFT methods. The underlying hypothesis is that the high-dimensional Hilbert space offers an efficient representation mechanism for adaptation.

All experiments are conducted using quantum circuit simulation implemented in PyTorch and TorchQuantum [73]. At this stage, quantum noise is not included, and the quantum state amplitudes (i.e., probabilities) are obtained exactly. A discussion of finite-shot sampling and noise effects is deferred to Sec. 3.9.

We benchmark QPA against several widely used PEFT methods—LoRA, DoRA, Prefix-Tuning (PT), and Feed-Forward Adapters (FFA)—by evaluating perplexity in text generation tasks. Experiments are performed on WikiText-2 using Gemma-2 (2B) and GPT-2 (80M) as base models.

Gemma-2 is selected as a recent state-of-the-art LLM[1], offering competitive performance compared to models such as Phi-2-2B [32], LLaMA2-7B [72], and Mistral-7B [33]. Full hyperparameter details are provided in Sec. 3.7.

To isolate the contribution of QPA, we simplify the PEFT setup by freezing all layers of Gemma-2 and GPT-2 and fine-tuning only the final linear projection layer ("lmhead"), which contains 38.59M parameters in GPT-2 and 0.52B parameters in Gemma-2. Unless otherwise specified, the QNN depth (repetition) is fixed at $L = 8$.

Low-rank adaptation methods with QPA. We first apply QPA to LoRA and DoRA, using QPA to generate the corresponding low-rank matrices (and the additional magnitude vector in DoRA). For this experiment, the low-rank dimension is fixed at $r = 4$. To explore parameter scaling, we vary the chunk size n_{mlp}, which controls the number of

[1] We use the Gemma-2 version released on August 8th, 2024: https://huggingface.co/google/gemma-2-2b.

doi:10.6342/NTU202504677

generated parameters: - GPT-2 with LoRA: $n_{mlp} \in \{256, 512, 1024, 2048, 4096, 8192\}$ - GPT-2 with DoRA: $n_{mlp} \in \{512, 1024, 2048, 4096, 8192\}$ - Gemma-2 with LoRA: $n_{mlp} \in \{1024, 4096, 8192, 16258, 32768, 65536\}$ - Gemma-2 with DoRA: $n_{mlp} \in \{512, 1024, 2048, 4096, 8192\}$

For baseline comparison, we evaluate LoRA and DoRA directly by varying their rank $r \in \{1, 2, 4, 8, 16, 32\}$.

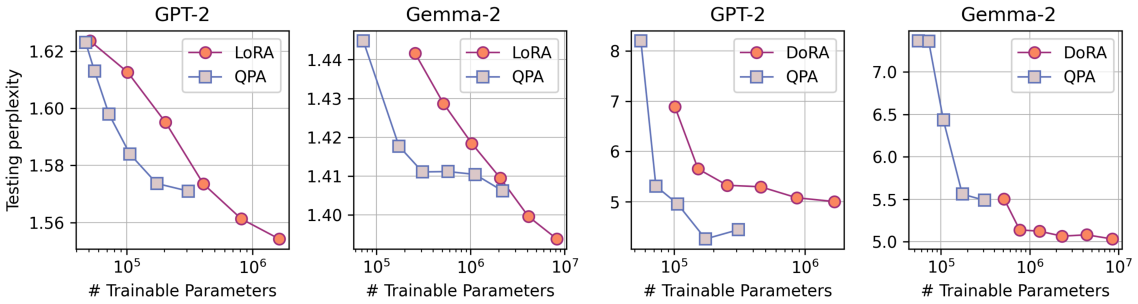


Figure 3.2: Testing perplexity of GPT-2 and Gemma-2 models compared to the number of trainable parameters for LoRA, DoRA, and QPA on the WikiText-2 dataset.

Table 3.2: Training parameters and testing perplexity for GPT-2 and Gemma-2 using PEFT and QPA methods, with emphasis on configurations achieving the most significant parameter reductions. Complete results are provided in Fig. 3.2 and Fig. 3.3.

PEFT Method	**GPT-2**		**Gemma-2**	
	# Params (%)	**PPL**	**# Params (%)**	**PPL**
LoRA	0.52	1.595	0.19	1.418
QPA LoRA (Ours)	**0.27**	**1.583**	**0.03**	**1.417**
DoRA	4.36	5.003	0.09	5.504
QPA DoRA (Ours)	**0.27**	**4.955**	**0.05**	**5.487**
PT	1.01	2.225	0.20	1.530
QPA PT (Ours)	**0.18**	**2.327**	**0.01**	**1.540**
FFA	0.76	1.763	0.40	1.439
QPA FFA (Ours)	**0.18**	**1.689**	**0.01**	**1.507**

In the results, shown in Fig. 3.2, QPA consistently outperforms LoRA and DoRA across a wide range of parameter configurations.

For **GPT-2**, QPA achieves lower test perplexity than LoRA, particularly in the low-parameter regime, highlighting its efficiency in resource-constrained settings. Even at larger parameter counts, QPA maintains performance on par with or better than LoRA.

doi:10.6342/NTU202504677

The best trade-off is observed with 106,264 trainable parameters and a perplexity of 1.583 for QPA, compared to 204,100 parameters and a perplexity of 1.595 for LoRA. This corresponds to a 52.06% reduction in parameter count alongside a 0.75% improvement in performance.

For **Gemma-2**, QPA demonstrates clear gains over LoRA at smaller parameter budgets, while the performance gap narrows at higher counts. The best result is obtained with 173,888 trainable parameters and a perplexity of 1.417 for QPA, compared to 1,032,192 parameters and a perplexity of 1.418 for LoRA—equivalent to a reduction to 16.84% of the parameters and a 0.07% improvement in performance.

When compared to **DoRA**, QPA initially yields slightly higher perplexity at small parameter counts for both GPT-2 and Gemma-2. However, as the number of trainable parameters increases, QPA surpasses DoRA, eventually achieving competitive or superior results across the board. The strongest reductions in trainable parameters are summarized in Table 3.2, where the ratios are calculated relative to the number of parameters in the target layer. For Gemma-2, the performance gap between QPA and DoRA diminishes at larger parameter budgets, with QPA maintaining strong competitiveness at all scales.

While the absolute performance differences may appear modest, the critical insight is that QPA can drastically reduce the number of trainable parameters in PEFT methods while sustaining, or even improving, task performance.

QPA for Prefix-Tuning and Feed-Forward Adapters. Beyond low-rank methods, we also apply QPA to PT and FFA. In PT, QPA generates the tunable prefix vectors prepended to the inputs of the target linear layer. Unlike conventional PT, where prefix length is tied to input size, QPA introduces flexibility by supporting alternative parameter

doi:10.6342/NTU202504677

configurations. For GPT-2, we test $n_{mlp} \in \{256, 512, 1024, 2048, 4096, 8192\}$, while for Gemma-2 we use $n_{mlp} \in \{1024, 2048, 4096, 8192, 16258, 32768\}$, with an additional setting of $n_{mlp} = 65536$ for QPA-FFA experiments. This extended configurability highlights QPA's ability to adaptively tune different PEFT architectures within the same unifying framework.

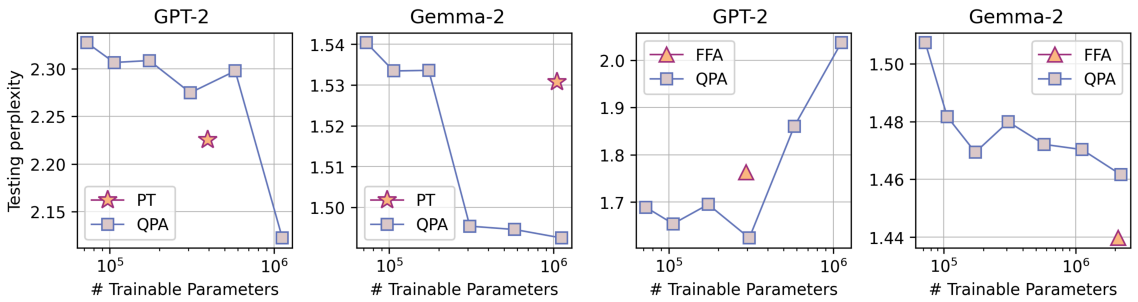


Figure 3.3: Testing perplexity of GPT-2 and Gemma-2 models compared to the number of trainable parameters for prefix-tuning (PT), feed-forward adapter (FFA), and QPA on the WikiText-2 dataset.

As illustrated in Fig. 3.3 and Table. 3.2, the performance of QPA relative to PT and FFA shows distinct trends across different parameter scales.

For **GPT-2**, QPA does not surpass PT in the low-parameter regime but demonstrates clear potential as the parameter space grows. In the strongest compression setting, QPA achieves 72,552 trainable parameters with a perplexity of 2.327, compared to PT with 393,216 parameters and a perplexity of 2.225. This corresponds to a reduction to 18.45% of the parameters, at the expense of a 4.38% increase in perplexity.

For **Gemma-2**, QPA initially matches PT performance and eventually exceeds it as the number of parameters increases, delivering superior perplexity at larger scales. This adaptability highlights QPA's ability to flexibly explore and optimize within a broader parameter space than conventional PT.

In the case of **FFA**, QPA generates the small feed-forward layers inserted before

doi:10.6342/NTU202504677

the target linear layer. On GPT-2, QPA outperforms FFA at lower parameter counts but gradually falls behind as the number of parameters increases. For Gemma-2, QPA does not surpass classical FFA across the tested range, but it maintains competitive perplexity even when scaled to larger configurations.

Although QPA does not consistently outperform PT or FFA in raw perplexity, its ability to achieve drastic reductions in trainable parameters—e.g., from $0.20\% \rightarrow 0.01\%$ for PT and $0.40\% \rightarrow 0.01\%$ for FFA on Gemma-2 (Table 3.2)—while incurring only modest performance loss underscores its promise as a highly efficient adaptation strategy. The trade-off between parameter count and accuracy demonstrates that QPA offers meaningful efficiency gains, particularly in scenarios where parameter budgets are highly constrained.

3.6 Effects of Hyperparameter Settings

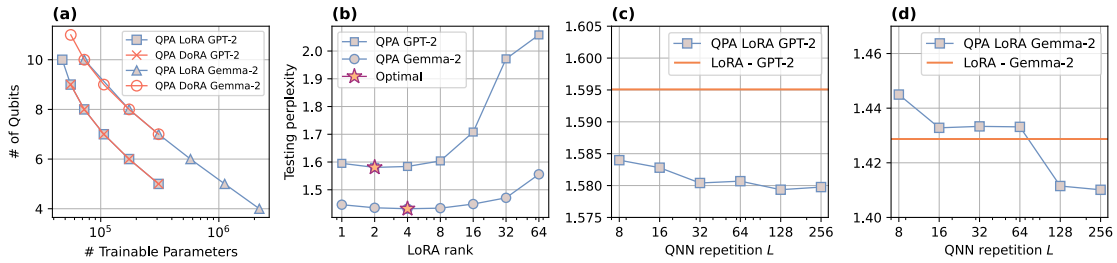


Figure 3.4: (a) Qubit usage versus the number of trainable parameters for QPA applied to LoRA and DoRA on GPT-2 and Gemma-2 models. (b) The relationship between testing perplexity and LoRA rank for QPA applied to GPT-2 and Gemma-2. (c) and (d) Testing perplexity depending on the QNN repetition L for QPA applied to LoRA on GPT-2 and Gemma-2.

Trade-off between qubit count and parameter size. As discussed in Sec. 3.4, the qubit usage in QPA follows Eq. 3.8. Increasing the chunk size n_{mlp} in batched parameter generation expands the output size of the mapping model, thereby increasing the total number of trainable parameters. At the same time, Eq. 3.8 shows that larger n_{mlp} reduces

doi:10.6342/NTU202504677

the number of required qubits. Fig. 3.4(a) illustrates the qubit usage corresponding to the results in Fig. 3.2, where the practical range is between 4 and 11 qubits. Such requirements are well within reach of both classical simulators and near-term fault-tolerant quantum devices[2]. While higher qubit usage generally reduces the number of trainable PEFT parameters, it may also lead to higher perplexity. Thus, careful tuning of n_{mlp} is necessary to balance parameter reduction against performance.

Optimal LoRA rank. Prior studies on LoRA and DoRA have shown that increasing the rank of low-rank matrices does not monotonically improve performance; instead, there exists an optimal rank. A similar trend emerges when applying QPA to low-rank adaptation. As shown in Fig. 3.4(b), the optimal LoRA ranks (marked by stars) are 2 for GPT-2 and 4 for Gemma-2, with Gemma-2 achieving consistently lower perplexity across ranks. For GPT-2, testing perplexity increases sharply as rank grows, whereas Gemma-2 remains relatively stable. In these experiments, the chunk size is fixed at $n_{mlp} = 2048$ for GPT-2 and $n_{mlp} = 8192$ for Gemma-2. These findings suggest that QPA can identify rank settings that strike a favorable balance between model compactness and accuracy.

Effect of deeper QNNs. The expressiveness of the QNN ansatz (Eq. 3.1), composed of R_Y rotations and CNOT gates, strongly depends on the number of repetitions L. A larger L produces a deeper QNN, enhancing its representational power. Figures 3.4(c) and (d) show the effect of increasing L on testing perplexity for QPA applied to LoRA on GPT-2 and Gemma-2. The orange lines mark the baseline LoRA results without QPA (LoRA rank = 4 for GPT-2 and rank = 2 for Gemma-2). For GPT-2, QPA-LoRA achieves lower perplexity overall, with slight improvements as L increases at $n_{mlp} = 2048$. For Gemma-2, QPA-LoRA initially matches the baseline but begins to outperform it once

[2]For instance, IBM recently demonstrated quantum error-correcting codes that preserve 12 logical qubits using 288 physical qubits [9].

doi:10.6342/NTU202504677

$L > 64$, achieving larger perplexity reductions at higher depths with $n_{mlp} = 8192$.

Theoretically, with a universal gate set such as $\{R_X, R_Y, R_Z, P(\varphi), \text{CNOT}\}$, the Solovay–Kitaev theorem [15] guarantees that QNNs can approximate any unitary to arbitrary precision, given sufficient depth. Thus, as L increases, QNNs gain the capacity to represent increasingly complex transformations [13, 18], enabling them to better approximate the optimal PEFT parameters. In practice, however, good performance can still be achieved with more restricted gate sets and moderate depths. In our main experiments, we use R_Y and CNOT gates with $L = 8$, and extend up to $L = O(N^2)$ in this section to demonstrate scalability.

Finally, gradient variance analysis (Sec. 3.10) shows that QPA does not exhibit exponential gradient suppression as qubit count increases. Although a mild downward trend appears with larger L, the absence of barren plateau behavior indicates that QPA remains trainable even with deeper QNNs, reinforcing its practical viability.

3.7 Training Hyperparameter Configuration

In this section, we provide the training hyperparameter configuration used for the results presented in the main text. Notably, α represents the scaling factor in the low-rank adaptation methods. All experiments were conducted on NVIDIA V100S and NVIDIA H100 GPUs.

doi:10.6342/NTU202504677

Table 3.3: Hyperparameter configurations of LoRA and QPA LoRA for fine-tuning GPT-2 and Gemma-2 with WikiText-2 dataset.

Hyperparameters	LoRA		QPA LoRA	
	GPT-2	Gemma-2	GPT-2	Gemma-2
α	2r		2r	
Dropout	0.05	0.0	0.05	0.0
Optimizer	AdamW		AdamW	
LR	1e-5		1e-5	
LR Scheduler	Linear		Linear	
Batch size	1		1	
Warmup Steps	0		0	
Epochs	3	5	3	5

Table 3.4: Hyperparameter configurations of DoRA and QPA DoRA for fine-tuning GPT-2 and Gemma-2 with WikiText-2 dataset.

Hyperparameters	DoRA		QPA DoRA	
	GPT-2	Gemma-2	GPT-2	Gemma-2
α	2r		2r	
Dropout	0.0		0.0	
Optimizer	AdamW		AdamW	
LR	2e-6		2e-6	
LR Scheduler	Linear		Linear	
Batch size	1		1	
Warmup Steps	100		100	
Epochs	5		5	

Table 3.5: Hyperparameter configurations of PT and QPA PT for fine-tuning GPT-2 and Gemma-2 with WikiText-2 dataset.

Hyperparameters	PT		QPA PT	
	GPT-2	Gemma-2	GPT-2	Gemma-2
Dropout	0.0		0.0	
Optimizer	AdamW		AdamW	
LR	5e-6		1e-6	
LR Scheduler	Linear		Linear	
Batch size	1		1	
Warmup Steps	0		0	
Epochs	5		5	

doi:10.6342/NTU202504677

Table 3.6: Hyperparameter configurations of FFA and QPA FFA for fine-tuning GPT-2 and Gemma-2 with WikiText-2 dataset.

Hyperparameters	FFA		QPA FFA	
	GPT-2	Gemma-2	GPT-2	Gemma-2
Dropout	0.0		0.0	
Optimizer	AdamW		AdamW	
LR	5e-6		5e-6	
LR Scheduler	Linear		Linear	
Batch size	1		1	
Warmup Steps	0		0	
Epochs	5	10	5	10

3.8 Effects of Different Circuit Ansatz

In the main experiments, we adopt a circuit ansatz constructed from R_Y rotations and CNOT gates. The motivation stems from the fact that the rotation

$$R_Y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

produces quantum states with real-valued amplitudes. This property aligns naturally with the goal of generating parameters for PEFT methods, which are typically real numbers. By contrast, R_X and R_Z rotations,

$$R_X(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \quad R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix},$$

introduce complex amplitudes, which are not directly required for this task. For this reason, the R_Y ansatz was selected as the default configuration.

Nevertheless, alternative circuit constructions can be explored by replacing or aug-

doi:10.6342/NTU202504677

menting the R_Y rotations. For instance, an ansatz can be defined with R_X gates:

$$|\psi(\boldsymbol{\theta})\rangle = \left(\prod_{i=1}^{N-1} \mathrm{CNOT}^{i,i+1} \prod_{j=1}^{N} R_X^j(\theta_j^{(L)})\right)^L |0\rangle^{\otimes N} \quad (R_X+\text{ CNOT circuit}). \tag{3.11}$$

Hybrid constructions combining both R_Z and rotation gates are also possible:

$$|\psi(\boldsymbol{\theta})\rangle = \left(\prod_{i=1}^{N-1} \mathrm{CNOT}^{i,i+1} \prod_{j=1}^{N} R_Z^j(\theta_j^{(L,Z)}) \prod_{i=1}^{N-1} \mathrm{CNOT}^{i,i+1} \prod_{j=1}^{N} R_X^j(\theta_j^{(L,X)})\right)^L |0\rangle^{\otimes N}$$
$$(R_X R_Z+\text{ CNOT circuit}), \tag{3.12}$$

$$|\psi(\boldsymbol{\theta})\rangle = \left(\prod_{i=1}^{N-1} \mathrm{CNOT}^{i,i+1} \prod_{j=1}^{N} R_Z^j(\theta_j^{(L,Z)}) \prod_{i=1}^{N-1} \mathrm{CNOT}^{i,i+1} \prod_{j=1}^{N} R_Y^j(\theta_j^{(L,Y)})\right)^L |0\rangle^{\otimes N}$$
$$(R_Y R_Z+\text{ CNOT circuit}). \tag{3.13}$$

In Fig. 3.5, we extend the LoRA results from Fig. 3.2 by including these alternative circuit ansatz. Across all cases, the observed performance differences are minor under noiseless simulation. This suggests that, at least in the ideal setting, the choice of circuit construction has only a limited impact on the effectiveness of QPA. The role of circuit choice under realistic conditions (finite shots and hardware noise) will be further examined in the following section.

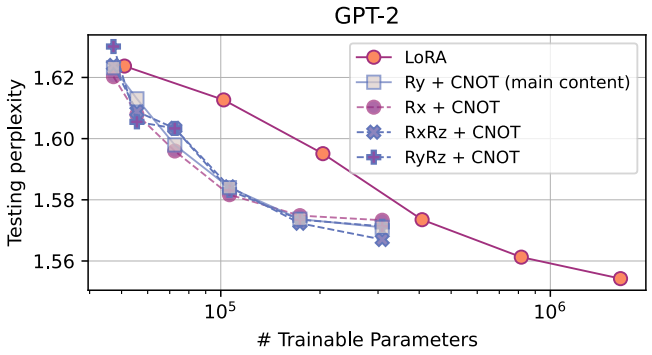


Figure 3.5: Testing perplexity of GPT-2 versus the number of trainable parameters on different circuit ansatz. The comparison includes LoRA and QPA applied at LoRA rank $(r = 4)$.

doi:10.6342/NTU202504677

3.9 Finite Measurement Shots and Noise

In the main experiments, the amplitudes and measurement probabilities of the basis states are computed exactly under noiseless simulation. While implementation on physical quantum hardware is left for future work, it is possible to approximate this setting by simulating the effects of finite measurement shots and incorporating noise models derived from real quantum devices.

On actual hardware, measurement probabilities are estimated from a finite number of shots. A larger number of shots yields more accurate probability estimates, but at the cost of increased runtime. To understand the scaling behavior of shot requirements, we draw parallels to tasks that approximate one quantum state with another. Specifically, quantum fidelity tomography results indicate that, to achieve fidelity $F(|\gamma\rangle, |\varphi\rangle) \geq 1 - \epsilon$ between an output state $|\varphi\rangle$ and an unknown state $|\gamma\rangle$, the sufficient number of measurements is

$$n_{\text{shot}} = O\left(\frac{2^N}{\epsilon} \log\left(\frac{2^N}{\epsilon}\right)\right), \tag{3.14}$$

where N denotes the number of qubits [24]. This highlights the exponential dependence of sampling cost on system size.

Given this scaling, adopting a linear shot allocation proportional to 2^N emerges as a practical heuristic for balancing efficiency and accuracy in larger systems. To evaluate this effect, we analyze LoRA and QPA results for GPT-2 under finite-shot conditions in Fig. 3.6. We compare measurement counts $n_{\text{shot}} \in \{10\times 2^N, 20\times 2^N, 40\times 2^N\}$, where N is the number of qubits. As expected, increasing the number of shots improves performance, with results progressively approaching those obtained from exact probability calculations.

doi:10.6342/NTU202504677

Notably, when $n_{\text{shot}} = 40 \times 2^N$, the outcomes are nearly indistinguishable from the exact results in several cases.

The number of qubits N corresponding to different QPA configurations in Fig. 3.6 ranges from 10 down to 5, matching the qubit usage progression shown in Fig. 3.4(a). These findings underscore the importance of carefully balancing shot allocation with qubit count: while higher shot budgets enhance fidelity, even moderate shot allocations can yield practical accuracy levels for QPA.

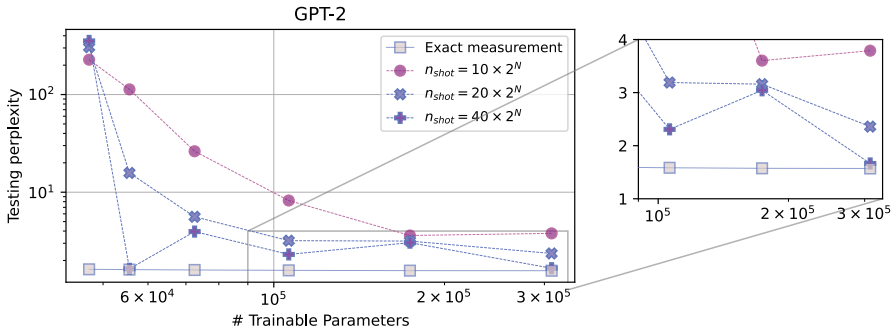


Figure 3.6: Testing perplexity of GPT-2 versus the number of trainable parameters on different number of measurement shots with R_Y+ CNOT ansatz. The comparison includes LoRA and QPA applied at LoRA rank $(r = 4)$.

Consequently, the ability to efficiently gather measurement data is critical for the practical deployment of the QPA method. Generative models offer a promising alternative by simulating quantum measurement outcomes directly [1]. Such models can alleviate the exponential shot requirements implied by fidelity tomography, reducing computational overhead while retaining statistical accuracy. By leveraging generative approaches, it becomes feasible to scale QPA to larger systems, enhancing its applicability in real-world scenarios.

Building on this perspective, we further simulate realistic conditions by incorporating hardware noise models alongside finite-shot sampling. Specifically, we employ noise

doi:10.6342/NTU202504677

profiles derived from the IBM quantum backends ibm_torino and ibm_fez [3]. These simulations allow us to approximate the behavior of QPA when executed on physical devices, providing insights into its robustness against both shot noise and hardware-induced errors. Results of these noise-aware experiments are discussed in the following analysis.

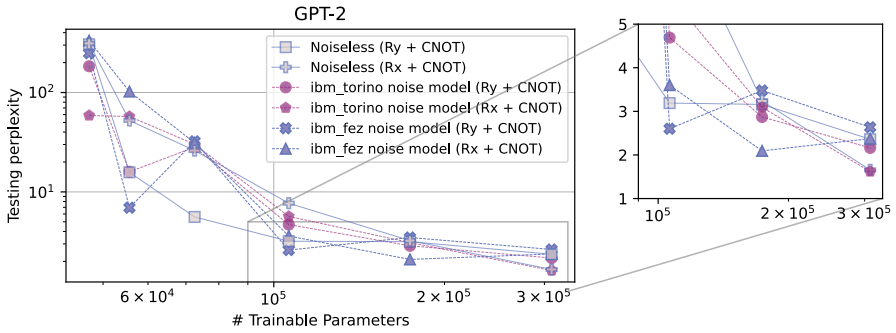


Figure 3.7: Testing perplexity of GPT-2 versus the number of trainable parameters on different noise setting with R_Y+ CNOT and R_X+ CNOT ansatz. The comparison includes LoRA and QPA applied at LoRA rank $(r = 4)$, where n_{shot} is fixed at $n_{\text{shot}} = 20 \times 2^N$.

In Fig. 3.7, we present QPA performance under simulated quantum noise for two circuit ansatzes: $R_X + \text{CNOT}$ and $R_Y + \text{CNOT}$. In the noiseless setting, the R_Y circuit consistently outperforms the R_X circuit across most parameter regimes. Interestingly, when noise models from ibm_torino and ibm_fez are introduced, performance does not uniformly degrade. While accuracy decreases slightly in some configurations, in most cases the noisy simulations yield improved performance compared to the noiseless baseline. This counterintuitive effect is consistent with recent observations in quantum computing, where device noise has been shown to enhance performance in certain computational paradigms [17].

A similar phenomenon has been documented in classical machine learning, where the injection of small amounts of noise can benefit the fine-tuning of large language models (LLMs) [76]. In the QPA setting, the quantum circuit generates measurement probabilities

[3]IBM Quantum provides noise models based on the properties of real hardware backends: https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.providers.aer.noise.NoiseModel.

doi:10.6342/NTU202504677

that are then passed through a classical MLP mapping model to produce PEFT parameters. The empirical observation that quantum noise can improve performance suggests an analogy to this "small noise" effect in classical fine-tuning: noise may encourage exploration of a richer parameter space, thereby aiding adaptation in QPA.

3.10 Gradient Variance of Quantum Circuit Parameters

Barren plateaus are a well-documented challenge in the training of QNNs [55, 80]. They typically arise in settings where the learning objective is defined as the expectation value of a Hermitian operator H, i.e.,

$$E(\theta) = \langle 0|U(\theta)^\dagger H U(\theta)|0\rangle. \tag{3.15}$$

In such cases, the variance of the gradient $\partial_\theta E(\theta)$ can decay exponentially with system size, rendering optimization intractable.

In contrast, QPA departs from this expectation-value framework. Its objective is the loss function of a target classical model (e.g., an LLM), where the parameters of the PEFT method (LoRA, DoRA, PT, etc.) are generated by a mapping model that consumes measurement outcomes of the QNN. Thus, the QNN' s output is the measurement probability distribution over computational basis states, rather than the expectation of a Hermitian operator. This structural difference raises the question of whether the exponential vanishing gradient characteristic of barren plateaus manifests in QPA.

To address this, we empirically analyzed the variance of QNN gradients $\partial\theta$ in QPA-LoRA applied to GPT-2, as shown in Fig. 3.8. Across increasing qubit counts, no significant downward trend in gradient variance is observed. We hypothesize that this robustness

doi:10.6342/NTU202504677

arises because the QNN gradients are propagated through the mapping model into the loss function of the target classical model, preventing the direct cancellation effects seen in expectation-based VQAs.

However, when varying the QNN repetition depth L, we observe a slight downward trend in gradient variance. While far milder than the exponential suppression characteristic of barren plateaus, this behavior resembles gradient attenuation in both classical deep feedforward neural networks [21] and in deep variational quantum circuits [55]. This suggests that although QPA does not exhibit barren plateaus in the conventional sense, its gradient dynamics share certain features of both classical and quantum learning systems.

Overall, these results indicate that QPA maintains informative gradients across scaling regimes, mitigating the barren plateau problem while still reflecting depth-related challenges familiar from both domains.

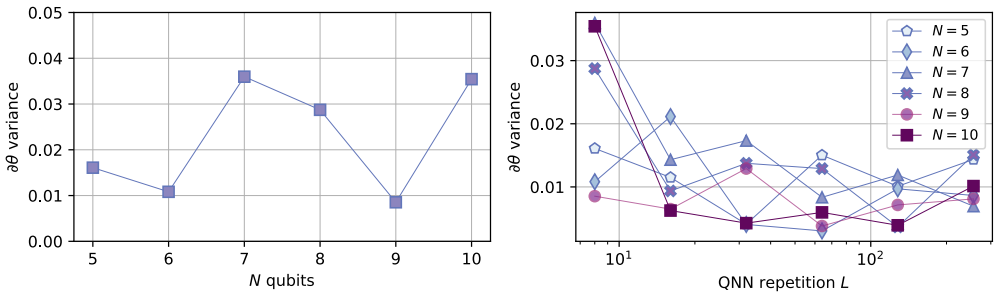


Figure 3.8: Variance of the gradient $\partial\theta$ with respect to the number of qubits (N) and the QNN repetition depth (L).

3.11 On Computational Time

The total execution times corresponding to the LoRA and QPA results in Fig. 3.2 are reported in Table 3.7 and Table 3.8. For GPT-2, the average batch execution times are 0.0154s for classical LoRA and 0.0460s for QPA. For Gemma-2, the corresponding

doi:10.6342/NTU202504677

averages are 0.0207s (LoRA) and 0.0576s (QPA). Thus, due to the additional cost of simulating quantum circuits and the associated quantum-classical mappings, QPA requires approximately three times the execution time of LoRA on average.

This observation suggests a fair comparison framework: under the same total computational budget, LoRA can train for roughly three epochs in the time required for a single epoch of QPA. Fig. 3.9 therefore contrasts the performance of QPA and LoRA when adjusted for computational cost. For GPT-2, since both LoRA and QPA originally train for three epochs, the one-epoch QPA result provides a natural benchmark against LoRA trained for three epochs under equivalent wall-clock time.

A similar rationale applies to Gemma-2, where five training epochs are used as the baseline. To approximate parity in execution time, we report QPA results for one and two epochs, compared against five epochs of LoRA. While the performance advantage of QPA is less pronounced under equalized time budgets, QPA continues to achieve competitive, and in some cases superior, results in the low-parameter regime.

Overall, these findings highlight a trade-off: QPA incurs additional per-epoch computational overhead but retains the ability to reduce parameter counts substantially while preserving model quality. This makes QPA a promising approach for scenarios where parameter efficiency is prioritized over raw training speed, especially when scaling to large models where memory and parameter counts dominate resource constraints.

doi:10.6342/NTU202504677

Table 3.7: Training parameters and total execution time for GPT-2 using LoRA and QPA methods. Corresponding to the results provided in Fig. 3.2

GPT-2 LoRA		GPT-2 QPA	
# Params	**Total execution time (s)**	**# Params**	**Total execution time (s)**
51025	1696	47248	5965
102050	1677	55656	5520
204100	1695	72512	5223
408200	1712	106264	4910
816400	1700	173808	4570
1632800	1729	308936	4245

Table 3.8: Training parameters and total execution time for Gemma-2 using LoRA and QPA methods. Corresponding to the results provided in Fig. 3.2

Gemma-2 LoRA		Gemma-2 QPA	
# Params	**Total execution time (s)**	**# Params**	**Total execution time (s)**
258048	3796	72592	13861
516096	3794	173888	11602
1032192	3798	309016	8316
2064384	3690	575154	9252
4128768	4002	1119944	11768
8257536	3816	2201248	8702

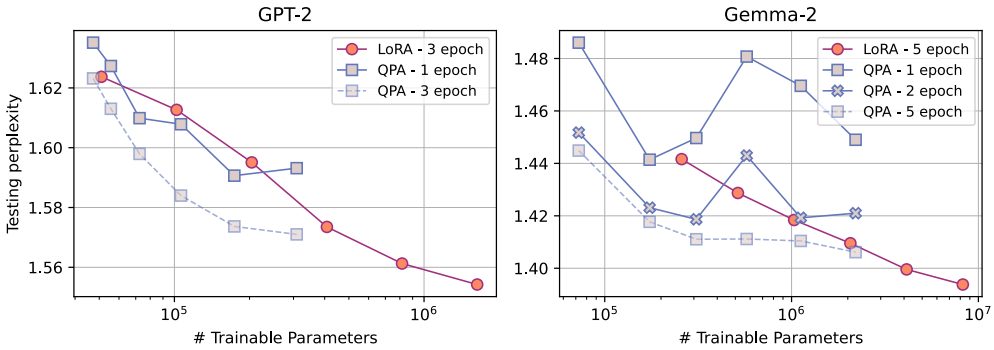


Figure 3.9: Comparison of testing perplexity against the number of trainable parameters for GPT-2 (left) and Gemma-2 (right) models using LoRA and QPA approaches across different epochs.

doi:10.6342/NTU202504677

Chapter 4 Applications of QT and QPA

Chapter Overview

While the previous chapters have focused on the theoretical development and methodological extensions of Quantum-Train (QT) and Quantum Parameter Adaptation (QPA), it is equally important to demonstrate their utility in practical domains. This chapter presents three case studies that highlight how QT and QPA can be applied across diverse real-world tasks:

- **Flood prediction (Section 4.1):** Using QT to parameterize Long Short-Term Memory (LSTM) models for hydrological time-series forecasting.
- **Reinforcement learning (Section 4.2):** Applying QT to compress policy networks in policy gradient methods, yielding efficient yet high-performing agents.
- **Typhoon trajectory forecasting (Section 4.3):** Leveraging QPA to drastically reduce trainable parameters in state-of-the-art spatio-temporal forecasting models, while maintaining predictive accuracy.

These case studies collectively illustrate the practical usage of quantum-enhanced

doi:10.6342/NTU202504677

parameter-efficient learning. Across environmental forecasting and decision-making tasks, QT and QPA demonstrate their ability to achieve significant compression of trainable parameters without sacrificing performance, while ensuring that inference remains fully classical and deployable on existing infrastructure. Each section provides a concise overview of the methodology, experimental results, and implications, together with selected figures for illustration.

4.1 Quantum-Train on Flood Prediction Problem

Flood forecasting is a critical task in climate adaptation, with direct implications for human safety, infrastructure protection, and environmental sustainability. In this section, we demonstrate the application of the Quantum-Train (QT) framework to time-series forecasting of river flood events. The work presented here is based on our study in [41], which was also the winning project of the Deloitte Quantum Climate Challenge 2024. We adopt a Long Short-Term Memory (LSTM) network as the base model for flood prediction. The QT framework is integrated into this setting by using a quantum neural network (QNN) to generate the LSTM parameters, thereby reducing the trainable parameter count from $O(M)$ to $O(\text{polylog}(M))$. Importantly, after training, inference is fully classical, making this approach both resource-efficient and practically deployable in real-world disaster-preparedness systems. We evaluated the QT-LSTM on hydrological and meteorological data from the Wupper river region, including water levels, discharge rates, and reservoir fill levels. Experimental results show that:

- The QT-LSTM reduced trainable parameters by more than 50% compared to the classical LSTM baseline.

doi:10.6342/NTU202504677

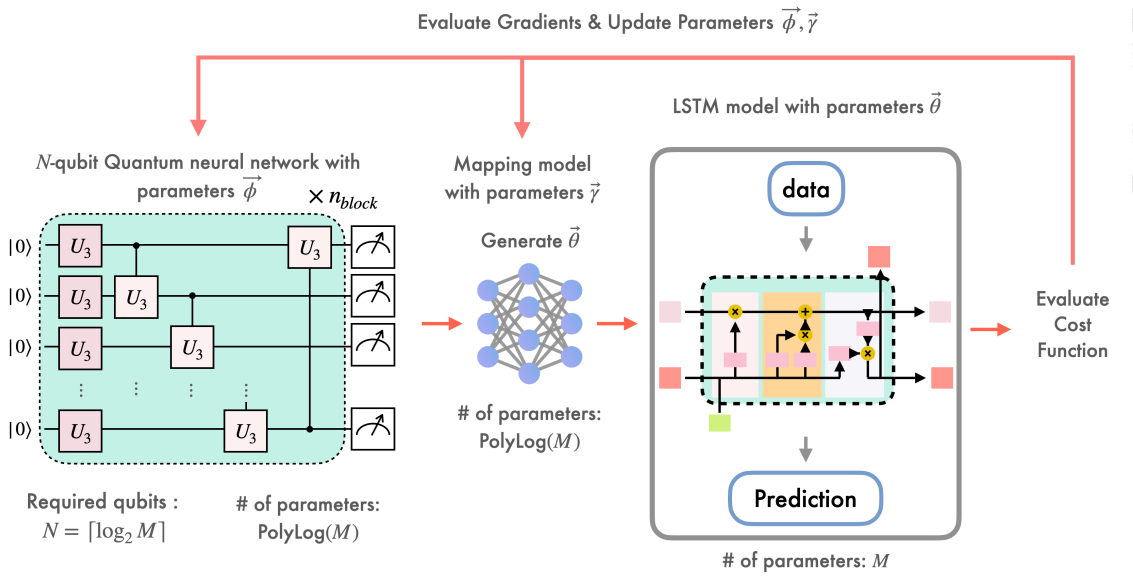


Figure 4.1: Illustration of the Quantum-Train LSTM (QT-LSTM) framework. A QNN maps measurement probabilities into the parameter space of a classical LSTM for flood prediction.

- While the QT-LSTM exhibited slightly higher mean squared error (MSE) than the classical LSTM in raw water-level forecasting, the performance gap narrowed when the task was reformulated as flood-event classification.

- In classification mode (predicting the occurrence of flood warnings), QT-LSTM achieved accuracy comparable to the classical model, despite its significantly smaller parameter budget.

The QT-LSTM results highlight both the promise and the trade-offs of parameter-efficient quantum-assisted training. Although raw prediction error remains higher than the classical LSTM, the comparable performance in flood-event classification suggests that QT can be a viable solution for risk prediction tasks, particularly where computational resources are constrained. Furthermore, QT retains practical advantages: independence from quantum hardware during inference and scalability to large temporal datasets.

This case study demonstrates the practical applicability of QT in environmental fore-

doi:10.6342/NTU202504677

(a)

	Classical	QT (simu.)
MSE	38.02	475.91
MAE	4.19	17.44
# of training parameter	40451	18830
Training time	1.5 min	11 hr

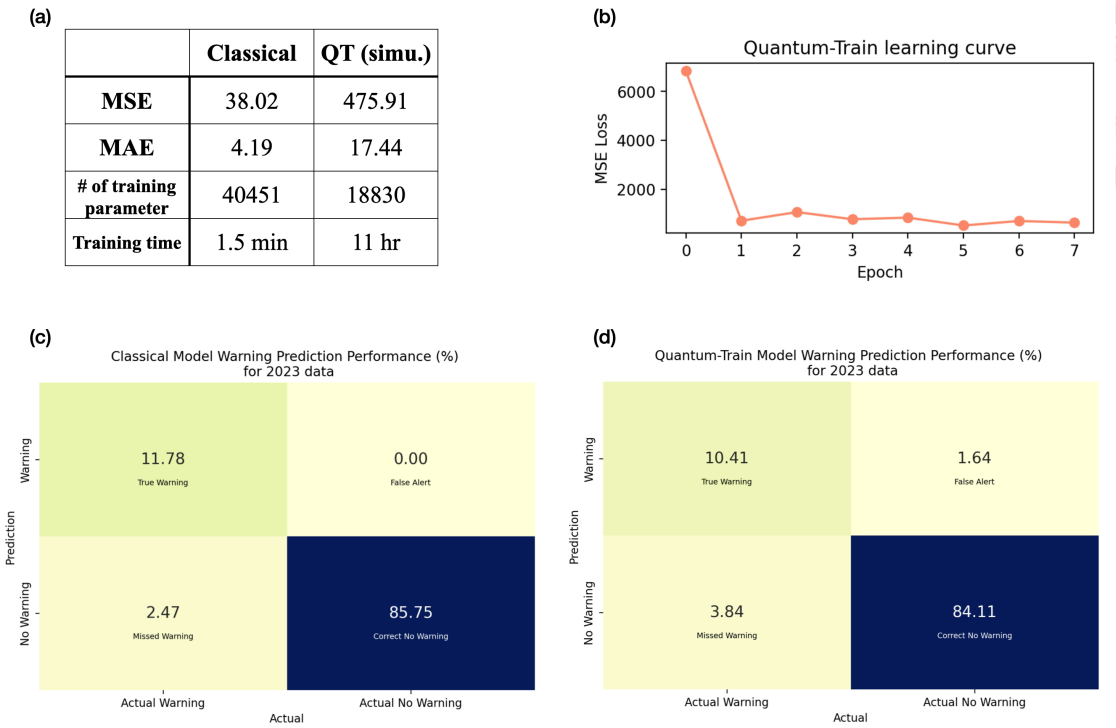


Figure 4.2: Comparison of flood forecasting results between classical LSTM and QT-LSTM. (a) Water level prediction, (b) learning curve, and (c–d) flood warning classification performance.

casting. By integrating QT into LSTM architectures, we establish a proof-of-concept for using quantum-generated parameters to enhance efficiency in real-world time-series prediction. This provides a foundation for subsequent applications of QT and QPA in other domains, as discussed in the following sections.

4.2 Quantum-Train Reinforcement Learning

Reinforcement learning (RL) is a powerful framework for sequential decision-making, but classical policy networks often require a large number of parameters and extensive training resources. Quantum reinforcement learning (QRL) has been proposed as a potential solution, but existing approaches face significant challenges, particularly in efficient data encoding and reliance on quantum hardware for both training and inference. To ad-

doi:10.6342/NTU202504677

dress these issues, we developed the Quantum-Train Reinforcement Learning (QTRL) framework, presented in [50], which adapts the Quantum-Train idea to the policy gradient setting. In QTRL, a parameterized quantum circuit (QNN) combined with a mapping model is used to generate the parameters of a classical policy network. This approach offers two advantages:

- **Parameter efficiency:** A classical policy network with k parameters can be generated by only $O(\text{polylog}(k))$ quantum and mapping model parameters, providing significant compression.
- **Practicality:** The final trained policy is entirely classical, meaning inference requires no quantum hardware. This eliminates latency issues that would otherwise hinder RL tasks requiring fast decision-making (e.g., autonomous driving).

We tested QTRL in two benchmark environments:

1. **CartPole-v1:** The classical policy model contained 898 parameters. QTRL reduced the number of trainable parameters to as few as 361 while maintaining strong performance. With deeper circuits ($L = 5$), QTRL even surpassed the classical baseline in average reward.
2. **MiniGrid-Empty-5x5-v0:** The classical policy model required 4835 parameters. QTRL compressed this to 2529 parameters (with $L = 13$ layers) while achieving nearly identical average reward (0.900 vs. 0.916 for the classical model).

The results indicate that QTRL achieves a favorable balance between parameter efficiency and performance. Shallow quantum circuits reduce parameter counts dramatically,

doi:10.6342/NTU202504677

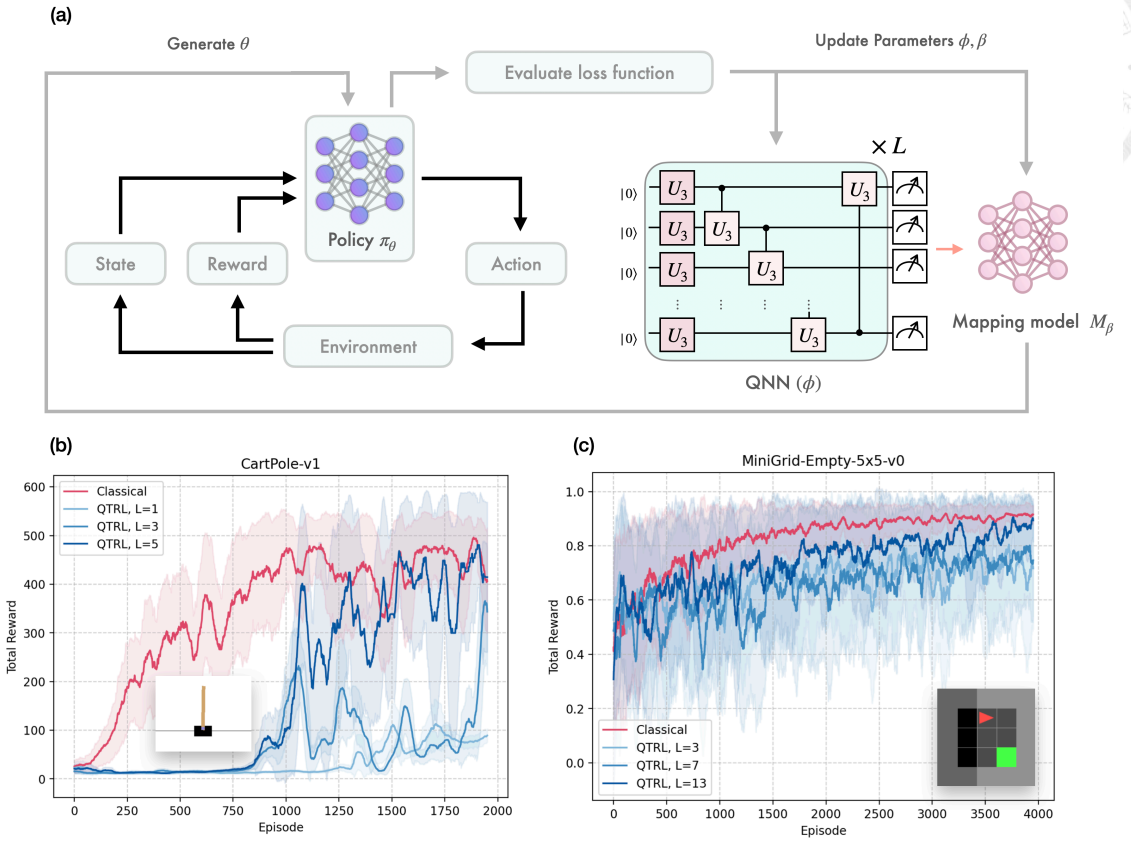


Figure 4.3: (a) Illustration of QTRL framework. (b) Total reward as a function of episode number for the CartPole-v1 environment. The classical method is shown in red, while QTRL with varying depths ($L = 1, 3, 5$) are shown in different shades of blue. (c) Total reward as a function of episode number for the MiniGrid-Empty-5x5-v0 environment. The classical method is shown in red, while QTRL with varying depths ($L = 3, 7, 13$) are shown in different shades of blue. Insets show the visual representation of each environment.

Model	**Topology**	**Last 10 episode average reward**	**Para. size**
Classical Policy Model	(4-128, 128-2)	436.3	898
QTRL-1	QNN: (U_3 block)*1 + Mapping Model: (11-10, 10-10, 10-1)	94.6	361
QTRL-3	QNN: (U_3 block)*3 + Mapping Model: (11-10, 10-10, 10-1)	218.9	531
QTRL-5	QNN: (U_3 block)*5 + Mapping Model: (11-10, 10-10, 10-1)	493.8	651

Table 4.1: Comparative performance of Classical Policy and QT models with varying numbers of QNN blocks for CartPole-v1 environment.

Model	**Topology**	**Last 10 episode average reward**	**Para. size**
Classical Policy Model	(147-32, 32-3)	0.916	4835
QTRL-3	QNN: (U_3 block)*3 + Mapping Model: (11-10, 10-10, 10-1)	0.694	1749
QTRL-7	QNN: (U_3 block)*7 + Mapping Model: (11-10, 10-10, 10-1)	0.827	2061
QTRL-13	QNN: (U_3 block)*13 + Mapping Model: (11-10, 10-10, 10-1)	0.900	2529

Table 4.2: Comparative performance of Classical Policy and QT models with varying numbers of QNN blocks for MiniGrid-Empty-5x5-v0 environment.

doi:10.6342/NTU202504677

while deeper circuits recover or surpass the performance of classical policy networks. Importantly, because the final trained model is purely classical, inference is efficient and deployable on edge devices. This feature makes QTRL particularly suited for real-world RL applications such as robotics and autonomous driving, where low-latency inference is critical.

QTRL demonstrates how Quantum-Train principles can be applied to reinforcement learning, yielding compressed yet effective policy models. By removing data-encoding burdens and eliminating dependence on quantum hardware at inference time, QTRL provides a practical path forward for quantum-enhanced RL.

4.3 QPA for Typhoon Trajectory Forecasting

Typhoon trajectory forecasting is a critical task for disaster preparedness, yet it remains computationally demanding due to the chaotic nature of atmospheric dynamics and the resource-intensive nature of deep learning models. In this section, we present the application of Quantum Parameter Adaptation (QPA) to large-scale typhoon forecasting, as introduced in [44]. This work represents the first use of quantum-inspired parameter-efficient learning for geoscientific climate forecasting. The classical baseline is the Attention-based Multi-ConvGRU (AM-ConvGRU) model [78], consisting of over 8 million trainable parameters, which has shown strong performance in spatio-temporal forecasting of typhoon paths. We integrate QPA with LoRA-style parameter-efficient fine-tuning, applying QPA to generate the adaptation parameters of the last two linear layers while keeping the rest of the model frozen. By combining quantum-generated LoRA parameters with the batched mapping model strategy, we reduce the effective trainable pa-

doi:10.6342/NTU202504677

rameters by over 96% (from 8.39M to approximately 0.3M) while retaining forecasting accuracy.

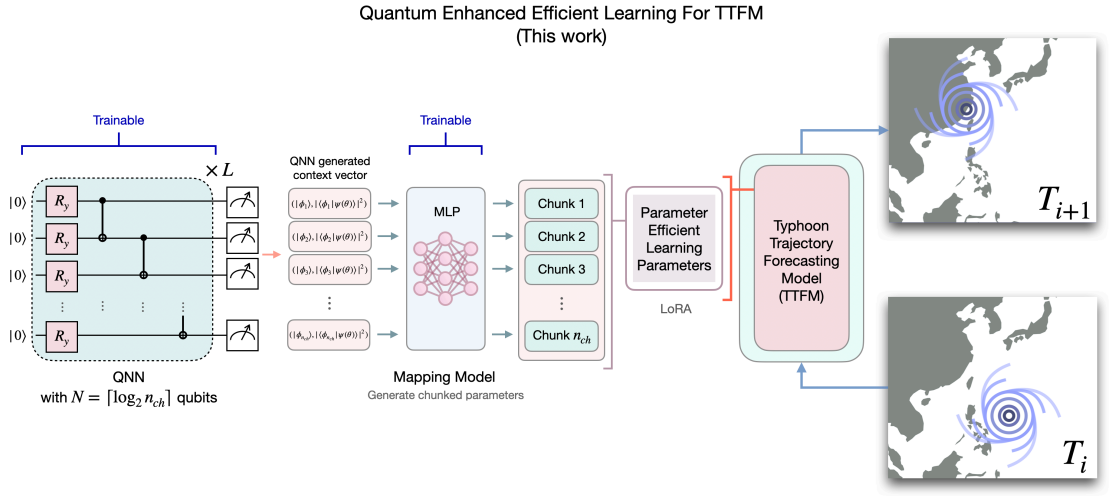


Figure 4.4: QPA-enhanced AM-ConvGRU framework for typhoon trajectory forecasting. Quantum-generated parameters compress the adaptation space while maintaining predictive accuracy.

The model was trained and tested on real-world datasets from the China Meteorological Administration (CMA) and ERA-Interim reanalysis. These datasets provide 2D typhoon path data (latitude, longitude, wind speed, pressure) and 3D atmospheric context across multiple isobaric levels. Preprocessing steps included CLIPER feature extraction, spatial gridding, and isobaric plane selection. The model was trained on typhoon data from 2000–2014 and evaluated on storms from 2015–2018. Key findings include:

- QPA reduced trainable parameters from 8.39M to ∼0.3M, a compression ratio of over 96%.
- Despite extreme compression, QPA matched or outperformed pruning and weight-sharing baselines in forecasting accuracy.
- QPA-based models captured typhoon trajectories nearly as well as the full classical model, with deviations in forecast error remaining minimal for most test cases.

doi:10.6342/NTU202504677

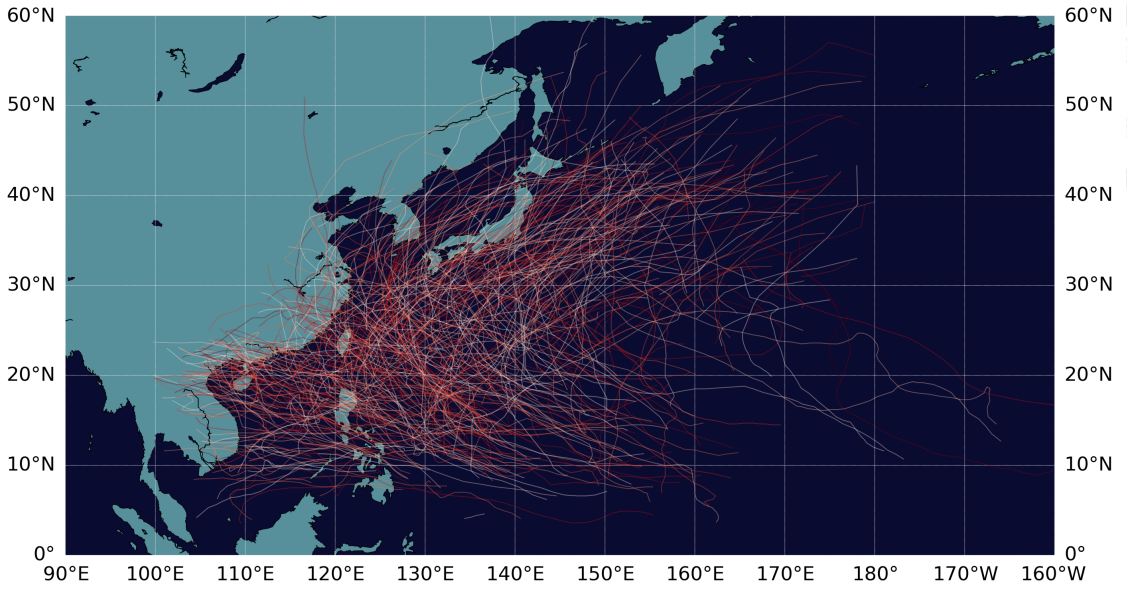


Figure 4.5: Visualization of historical typhoon trajectories used in training and testing.

- Visualization of predicted paths (e.g., for Typhoon Usagi and Typhoon Chan-hom) showed strong alignment with actual trajectories, even under significant compression.

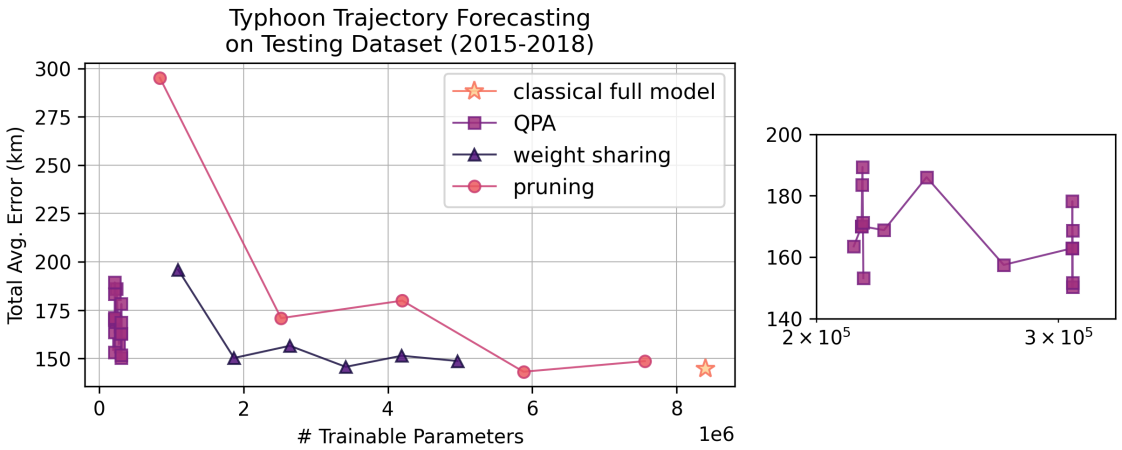


Figure 4.6: Benchmark comparison of classical full model, pruning, weight sharing, and QPA. QPA achieves comparable accuracy with 30× fewer parameters.

These results highlight QPA's potential to scale hybrid quantum–classical learning into large geoscientific forecasting problems. Unlike pruning, which plateaus in performance at low parameter counts, QPA maintains robust accuracy even under extreme compression. This suggests that quantum-inspired parameterization can encode critical atmo-

doi:10.6342/NTU202504677

spheric patterns more efficiently than purely classical compression techniques. Moreover, inference remains fully classical, ensuring compatibility with existing high-performance computing (HPC) weather systems.

QPA for typhoon trajectory forecasting demonstrates the promise of hybrid quantum-classical learning in climate science. By achieving nearly 30× compression with minimal accuracy loss, QPA offers a scalable and energy-efficient approach to forecasting, with implications for both sustainability and accessibility of advanced climate models.

doi:10.6342/NTU202504677

Chapter 5 Conclusion

5.1 Summary of Contributions

This thesis has introduced and developed two complementary frameworks, **Quantum-Train (QT)** and **Quantum Parameter Adaptation (QPA)**, that explore the intersection of quantum computing and parameter-efficient machine learning. Both frameworks address a common challenge in modern machine learning: the growing size of models and the corresponding computational burden of training and fine-tuning. By leveraging quantum circuits as parameter generators, this work demonstrates that quantum resources can be integrated into the training pipeline in a way that yields parameter compression, efficiency gains, and new design possibilities, while still allowing inference to remain purely classical. The main contributions of this thesis can be summarized as follows:

- **Quantum-Train (QT):** We proposed QT as a hybrid quantum-classical framework where variational quantum circuits generate classical neural network parameters via a classical mapping model. A theoretical approximation error bound was derived, showing how model performance depends on circuit depth, mapping expressivity, and shot count. Extensive experiments on MNIST, FashionMNIST, and CIFAR-10 demonstrated that QT achieves competitive accuracy with far fewer parameters

doi:10.6342/NTU202504677

compared to classical baselines, and outperforms established compression methods such as weight sharing and pruning in several regimes.

- **Architectural Extensions:** Beyond multilayer perceptrons, we introduced tensor-network mappings (specifically matrix product states) within QT. These structured mappings showed enhanced robustness under noise and finite-shot conditions and offered improved parameter efficiency. We also explored integration with low-rank adaptation (LoRA), demonstrating that QT can serve as a generator of LoRA matrices and achieve superior performance in the low-parameter regime.

- **Robustness Analysis:** We examined the effects of hardware noise and finite measurement shots using IBM noise models. Empirical studies revealed that QT with structured mappings remains stable even under noisy and shot-limited settings. Moreover, gradient variance analyses showed that QT avoids barren plateau behavior, due to its reliance on probability-based gradients rather than expectation-value objectives.

- **Quantum Parameter Adaptation (QPA):** Building on QT, we introduced QPA for parameter-efficient fine-tuning of large language models (LLMs). QPA generates the parameters of PEFT methods (LoRA, DoRA, Prefix-Tuning, and Feed-Forward Adapters) via quantum-classical mappings, reducing the number of trainable parameters by nearly an order of magnitude. Experiments on GPT-2 and Gemma-2 demonstrated that QPA consistently matches or outperforms classical PEFT baselines in perplexity while dramatically lowering parameter counts.

- **Scalability Enhancements:** To handle billion-parameter models, we developed a batched parameter generation technique that reduces qubit requirements by chunk-

doi:10.6342/NTU202504677

ing model parameters, enabling large-scale tasks with feasible memory and runtime costs. We further analyzed trade-offs between qubit count, chunk size, LoRA rank, and QNN depth, showing how deeper and more expressive QNNs enhance performance.

- **Practical Considerations:** Finally, we evaluated computational costs, showing that while QPA incurs additional overhead due to quantum circuit simulation, it remains competitive when normalized by total training time and demonstrates clear advantages in parameter efficiency. Together with the noise and shot analysis, this highlights the practical viability of QT and QPA on near-term quantum devices and simulations.

5.2 Key Insights

The investigations in this thesis reveal several broader insights into the role of quantum computing in machine learning:

- Quantum circuits can serve not only as direct classifiers or generative models, but also as *parameter generators*, offering a novel use case that sidesteps many of the limitations of data encoding in quantum machine learning.

- The Hilbert space structure provides an exponentially large representation space, which, when coupled with classical mappings, allows for efficient exploration of parameter spaces in both classical neural networks and LLM fine-tuning methods.

- Probability-based training objectives mitigate the barren plateau problem by avoiding expectation-value cancellations, leading to more stable gradient behavior across

doi:10.6342/NTU202504677

depth, qubit count, and noise settings.

- Hybrid strategies, such as QT-LoRA and QPA, demonstrate that quantum methods can be complementary to state-of-the-art classical compression and adaptation techniques, offering parameter reductions and robustness advantages.

5.3 Future Directions

While this work establishes the theoretical foundation and empirical validation of QT and QPA, several avenues remain open for future exploration:

- **Hardware Implementation:** Extending QPA experiments from classical simulations to real quantum devices will provide insights into the impact of hardware constraints, decoherence, and noise mitigation strategies.
- **Generative Sampling for Shots:** Incorporating generative models to simulate measurement outcomes can further reduce the exponential sampling costs of large systems, increasing the practicality of QPA at scale.
- **Integration with Foundation Models:** Applying QPA to larger LLMs beyond GPT-2 and Gemma-2, and exploring multimodal models (vision-language, speech-language) will help assess its utility for industrial-scale deployments.
- **Theoretical Analysis:** A rigorous convergence theory for QPA, particularly in the context of gradient variance and generalization error, remains an important open problem that could further clarify its advantages over classical PEFT methods.

doi:10.6342/NTU202504677

5.3.1 Model Compression in the Inference Stage

A recurring theme throughout this thesis is the importance of model compression for efficient deployment. In the current QT and QPA settings, while training requires significantly fewer tunable parameters, the resulting models at the inference stage still retain the full parameter count of the original architecture. This limits the extent to which compression benefits can be realized in practice, especially in large-scale applications where inference costs often dominate. Future research may address this limitation by developing methods that compress not only the training phase but also the inference model itself, yielding a genuinely smaller network for deployment. A promising direction toward this goal is provided by Quantum Relational Knowledge Distillation (QRKD) [45]. In QRKD, a large teacher model is used to train a compact student model through quantum-enhanced knowledge transfer. Once trained, the student model replaces the teacher during both training and inference, thereby reducing the parameter footprint across the entire pipeline.

The demonstration of QRKD represents an early but significant step toward achieving parameter efficiency in both stages of learning. It opens a new research avenue where quantum resources can play a central role in distilling knowledge more effectively, paving the way for scalable and deployable quantum-enhanced parameter-efficient learning.

5.4 Concluding Remarks

This thesis introduced Quantum-Train and Quantum Parameter Adaptation as two synergistic frameworks that leverage quantum computing to address the pressing chal-

doi:10.6342/NTU202504677

lenge of parameter efficiency in modern machine learning. Together, they represent a step toward practical, quantum-assisted learning pipelines that are both scalable and hardware-conscious. By demonstrating how quantum circuits can compress and adapt parameters for large classical models while retaining competitive performance, this work opens new possibilities for bridging quantum and classical computation in the era of increasingly large and complex machine learning models.

doi:10.6342/NTU202504677

References

[1] S. Ahmed, C. Sánchez Muñoz, F. Nori, and A. F. Kockum. Quantum state tomography with conditional generative adversarial networks. Physical review letters, 127(14):140502, 2021.

[2] H. B. Barlow. Unsupervised learning. Neural computation, 1(3):295–311, 1989.

[3] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information theory, 39(3):930–945, 1993.

[4] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini. Parameterized quantum circuits as machine learning models. Quantum Science and Technology, 4(4):043001, 2019.

[5] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968, 2018.

[6] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning? Proceedings of machine learning and systems, 2:129–146, 2020.

[7] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi,

doi:10.6342/NTU202504677

M. Cain, M. Kalinowski, D. Hangleiter, et al. Logical quantum processor based on reconfigurable atom arrays. Nature, 626(7997):58–65, 2024.

[8] A. Bouland, B. Fefferman, C. Nirkhe, and U. Vazirani. On the complexity and verification of quantum random circuit sampling. Nature Physics, 15(2):159–163, 2019.

[9] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. Nature, 627(8005):778–782, 2024.

[10] T. B. Brown. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

[11] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles. Generalization in quantum machine learning from few training data. Nature communications, 13(1):4919, 2022.

[12] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. Nature Reviews Physics, 3(9):625–644, 2021.

[13] A. M. Childs. Lecture notes on quantum algorithms. Lecture notes at University of Maryland, 5, 2017.

[14] P. Cunningham, M. Cord, and S. J. Delany. Supervised learning. In Machine learning techniques for multimedia: case studies on organization and retrieval, pages 21–49. Springer, 2008.

[15] C. M. Dawson and M. A. Nielsen. The solovay-kitaev algorithm. arXiv preprint quant-ph/0505030, 2005.

doi:10.6342/NTU202504677

[16] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6):141–142, 2012.

[17] L. Domingo, G. Carlo, and F. Borondo. Taking advantage of noise in quantum reservoir computing. Scientific Reports, 13(1):8790, 2023.

[18] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao. Expressive power of parametrized quantum circuits. Physical Review Research, 2(3):033125, 2020.

[19] Y. Du, M.-H. Hsieh, T. Liu, S. You, and D. Tao. Learnability of quantum neural networks. PRX quantum, 2(4):040337, 2021.

[20] Y. Du, Y. Yang, D. Tao, and M.-H. Hsieh. Problem-dependent power of quantum neural networks on multiclass classification. Physical Review Letters, 131(14):140601, 2023.

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[22] Q. A. Google. Choosing hardware for your qsim simulation. https://quantumai.google/qsim/choose_hw, 2024.

[23] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. International Journal of Computer Vision, 129(6):1789–1819, 2021.

[24] J. Haah, A. W. Harrow, Z. Ji, X. Wu, and N. Yu. Sample-optimal tomography of quantum states. IEEE Transactions on Information Theory, page 1–1, 2017.

doi:10.6342/NTU202504677

[25] D. Hangleiter, M. Kalinowski, D. Bluvstein, M. Cain, N. Maskara, X. Gao, A. Kubica, M. D. Lukin, and M. J. Gullans. Fault-tolerant compiling of classically hard instantaneous quantum polynomial circuits on hypercubes. PRX Quantum, 6(2):020338, 2025.

[26] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. Nature, 567(7747):209–212, 2019.

[27] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. IEEE Intelligent Systems and their applications, 13(4):18–28, 1998.

[28] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. 2008.

[29] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In International conference on machine learning, pages 2790–2799. PMLR, 2019.

[30] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[31] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean. Power of data in quantum machine learning. Nature communications, 12(1):2631, 2021.

[32] M. Javaheripi, S. Bubeck, M. Abdin, J. Aneja, S. Bubeck, C. C. T. Mendes, W. Chen, A. Del Giorno, R. Eldan, S. Gopi, et al. Phi-2: The surprising power of small language models. Microsoft Research Blog, 2023.

doi:10.6342/NTU202504677

[33] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

[34] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. Journal of artificial intelligence research, 4:237–285, 1996.

[35] D. P. Kingma. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[36] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. SIAM review, 51(3):455–500, 2009.

[37] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). URL http://www. cs. toronto. edu/kriz/cifar. html, 5(4):1, 2010.

[38] J. Landman, S. Thabet, C. Dalyac, H. Mhiri, and E. Kashefi. Classically approximating variational quantum machine learning with random fourier features. arXiv preprint arXiv:2210.13200, 2022.

[39] R. LaRose and B. Coyle. Robust data encodings for quantum classifiers. Physical Review A, 102(3):032420, 2020.

[40] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190, 2021.

[41] C.-H. A. Lin, C.-Y. Liu, and K.-C. Chen. Quantum-train long short-term memory: Application on flood prediction problem. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 2, pages 268–273. IEEE, 2024.

doi:10.6342/NTU202504677

[42] D. Lin, S. Talathi, and S. Annapureddy. Fixed point quantization of deep convolutional networks. International conference on machine learning, pages 2849–2858, 2016.

[43] Z. Lin, A. Madotto, and P. Fung. Exploring versatile generative language model via parameter-efficient transfer learning. arXiv preprint arXiv:2004.03829, 2020.

[44] C.-Y. Liu, K.-C. Chen, Y.-C. Chen, S. Y.-C. Chen, W.-H. Huang, W.-J. Huang, and Y.-J. Chang. Quantum-enhanced parameter-efficient learning for typhoon trajectory forecasting. arXiv preprint arXiv:2505.09395, 2025.

[45] C.-Y. Liu, K.-C. Chen, K. Murota, S. Y.-C. Chen, and E. Rinaldi. Quantum relational knowledge distillation. arXiv preprint arXiv:2508.13054, 2025.

[46] C.-Y. Liu and S. Y.-C. Chen. Federated quantum-train with batched parameter generation. In 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), pages 1133–1138. IEEE, 2024.

[47] C.-Y. Liu, E.-J. Kuo, C.-H. Abraham Lin, J. Gemsun Young, Y.-J. Chang, M.-H. Hsieh, and H.-S. Goan. Quantum-train: Rethinking hybrid quantum-classical machine learning in the model compression perspective. Quantum Machine Intelligence, 7(2):80, 2025.

[48] C.-Y. Liu, E.-J. Kuo, C.-H. Abraham Lin, J. Gemsun Young, Y.-J. Chang, M.-H. Hsieh, and H.-S. Goan. Quantum-train: Rethinking hybrid quantum-classical machine learning in the model compression perspective. Quantum Machine Intelligence, 7(2):80, 2025.

[49] C.-Y. Liu, E.-J. Kuo, C.-H. A. Lin, S. Chen, J. G. Young, Y.-J. Chang, and M.-H. Hsieh. Training classical neural networks by quantum machine learning. In 2024

doi:10.6342/NTU202504677

IEEE International Conference on Quantum Computing and Engineering (QCE), volume 2, pages 34–38. IEEE, 2024.

[50] C.-Y. Liu, C.-H. A. Lin, C.-H. H. Yang, K.-C. Chen, and M.-H. Hsieh. Qtrl: Toward practical quantum reinforcement learning via quantum-train. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 2, pages 317–322. IEEE, 2024.

[51] C.-Y. Liu, C.-H. H. Yang, H.-S. Goan, and M.-H. Hsieh. A quantum circuit-based compression perspective for parameter-efficient learning. In The Thirteenth International Conference on Learning Representations.

[52] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. Dora: Weight-decomposed low-rank adaptation. arXiv preprint arXiv:2402.09353, 2024.

[53] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran. Transfer learning in hybrid classical-quantum neural networks. Quantum, 4:340, 2020.

[54] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran. Transfer learning in hybrid classical-quantum neural networks. Quantum, 4:340, 2020.

[55] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. Nature communications, 9(1):4812, 2018.

[56] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. Physical Review A, 98(3):032309, 2018.

doi:10.6342/NTU202504677

[57] M. Mohri, A. Rostamizadeh, and A. Talwalkar. Foundations of machine learning. MIT press, 2018.

[58] R. Movassagh. The hardness of random quantum circuits. Nature Physics, 19(11):1719–1724, 2023.

[59] J. O. Neill. An overview of neural network compression. arXiv preprint arXiv:2006.03669, 2020.

[60] M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information. Cambridge university press, 2010.

[61] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight sharing. pages 373–394. CRC Press, 2018.

[62] C.-Y. Park, M. Kang, and J. Huh. Hardware-efficient ansatz without barren plateaus in any depth. arXiv preprint arXiv:2403.04844, 2024.

[63] J. Qi, C.-H. H. Yang, P.-Y. Chen, and M.-H. Hsieh. Theoretical error performance analysis for variational quantum circuit based functional regression. npj Quantum Information, 9(1):4, 2023.

[64] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

[65] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. Ortiz Marrero, M. Larocca, and M. Cerezo. A lie algebraic theory of barren plateaus for deep parameterized quantum circuits. Nature Communications, 15(1):7172, 2024.

[66] N. Sardana, J. Portes, S. Doubov, and J. Frankle. Beyond chinchilla-optimal:

doi:10.6342/NTU202504677

Accounting for inference in language model scaling laws. arXiv preprint arXiv:2401.00448, 2023.

[67] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. Physical Review A, 99(3):032331, 2019.

[68] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. Physical Review A, 99(3):032331, 2019.

[69] S. Sim, P. D. Johnson, and A. Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies, 2(12):1900070, 2019.

[70] E. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. Advances in neural information processing systems, 29, 2016.

[71] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024.

[72] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

[73] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28), 2022.

doi:10.6342/NTU202504677

[74] H. Wang, Y. Ding, J. Gu, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han. Quantumnas: Noise-adaptive search for robust quantum circuits. 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 692–708, 2022.

[75] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin. General parameter-shift rules for quantum gradients. Quantum, 6:677, 2022.

[76] C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie. Noisytune: A little noise can help you finetune pretrained language models better. arXiv preprint arXiv:2202.12024, 2022.

[77] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

[78] G. Xu, D. Xian, P. Fournier-Viger, X. Li, Y. Ye, and X. Hu. Am-convgru: a spatio-temporal model for typhoon path prediction. Neural Computing and Applications, 34(8):5905–5921, 2022.

[79] C.-H. H. Yang, Y.-Y. Tsai, and P.-Y. Chen. Voice2series: Reprogramming acoustic models for time series classification. In International conference on machine learning, pages 11808–11819. PMLR, 2021.

[80] K. Zhang, L. Liu, M.-H. Hsieh, and D. Tao. Escaping from the barren plateau via gaussian initializations in deep variational quantum circuits. Advances in Neural Information Processing Systems, 35:18612–18627, 2022.

doi:10.6342/NTU202504677

Appendix A — Proof of QT Approximation Bound

In this appendix, we provide the detailed derivation of the QT approximation error bound stated in Theorem 1. The proof follows a step-by-step decomposition of the total approximation error, beginning with the difference between the empirical loss of the QT-generated model and the optimal classical neural network (NN) model. We denote the weight and bias generated by QT in the classical NN model as w_{qt} and b_{qt}, such that

$$y_{\text{qt}}(\mathbf{x}) = \sigma(w_{\text{qt}}^T \mathbf{x} + b_{\text{qt}}), \tag{A.1}$$

where w_{qt} and b_{qt} are calculated from mapping model $G_{\theta_{\text{mm}}}$ with different parts of the basis set (equivalently different part of Ψ^c), denoted as

$$w_{\text{qt}} = G_{\theta_{\text{mm}}}(\Psi_w^c(|\langle\phi|\psi(\theta_{\text{qnn}})\rangle|^2)) \tag{A.2}$$

$$= G_{\theta_{\text{mm}}}(\Psi_w^c(P(\theta_{\text{qnn}}))) \tag{A.3}$$

$$= G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ P(\theta_{\text{qnn}}), \tag{A.4}$$

doi:10.6342/NTU202504677

where the measurement probability is represented as $P(\theta_{\text{qnn}})$ for clarity. Similarly,

$$b_{\text{qt}} = G_{\theta_{\text{mm}}}(\Psi_b^c(|\langle\phi|\psi(\theta_{\text{qnn}})\rangle|^2)) \tag{A.5}$$

$$= G_{\theta_{\text{mm}}}(\Psi_b^c(P(\theta_{\text{qnn}}))) \tag{A.6}$$

$$= G_{\theta_{\text{mm}}} \circ \Psi_b^c \circ P(\theta_{\text{qnn}}). \tag{A.7}$$

The QT approximation error can now be derived as:

$$\begin{aligned}
&\mathcal{L}_{\mathcal{D}}(f_S^{\text{QT}*}) - \mathcal{L}_{\mathcal{D}}(f_S^*) \\
&\leq \|f_S^* - f_S^{\text{QT}*}\|_1 \\
&\leq \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + \|\hat{f}_S^{\text{QT}*} - f_S^{\text{QT}*}\|_1 \quad \text{(Triangle Ineq.)} \\
&\approx \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + \|\sigma'(c)\left((\hat{w}_{qt} - w_{qt})^T\mathbf{x} + \hat{b}_{qt} - b_{qt}\right)\|_1 \quad \text{(Mean Value Theorem)} \\
&\leq \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + \frac{1}{4}\|(\hat{w}_{qt} - w_{qt})^T\mathbf{x} + \hat{b}_{qt} - b_{qt}\|_1 \\
&= \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + \frac{1}{4}\|\left(G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ (\hat{P} - P)\right)^T\mathbf{x} + G_{\theta_{\text{mm}}} \circ \Psi_b^c \circ (\hat{P} - P)\|_1 \\
&\leq \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + \frac{1}{4\sqrt{m}}\sqrt{\frac{1}{2}\ln\frac{2}{\delta}} \cdot \left\|\left(G_{\theta_{\text{mm}}} \circ \Psi_w^c(\mathbf{1})\right)^T\mathbf{x} + G_{\theta_{\text{mm}}} \circ \Psi_b^c(\mathbf{1})\right\|_1 \quad \text{(Hoeffding's Ineq.)} \\
&= \|f_S^* - \hat{f}_S^{\text{QT}*}\|_1 + O(\frac{1}{\sqrt{m}}) \quad \text{(where } \|\mathbf{x}\| \leq \text{const.)} \\
&\leq \frac{1}{4}\|(w - \hat{w}_{qt})^T\mathbf{x} + b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}}) \\
&= \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ \hat{P})^T\mathbf{x} + b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}}) \\
&= \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ |\langle\phi|\psi(\theta_{\text{qnn}})\rangle|_{\text{emp}}^2)^T\mathbf{x} + b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}}) \\
&= \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ |\langle\phi|U(\theta_{\text{qnn}})|0\rangle^{\otimes n_{qt}}|_{\text{emp}}^2)^T\mathbf{x} + b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}})
\end{aligned} \tag{A.8}$$

where $\hat{f}_S^{\text{QT}*}$ is defined as the QT operator with finite measurement shots m, thus the probability $\hat{P}$ is empirical, with the corresponding empirical $\hat{w}_{qt}$ and $\hat{b}_{qt}$. c is a mean value

doi:10.6342/NTU202504677

vector of A and B where we use the property $\sigma(A)-\sigma(B)\approx\sigma'(c)(A-B)$ when $A-B$ is small, recall that $\sigma'(c)\leq\frac{1}{4}$. Continue on the QT approximation error:

$$
\begin{aligned}
&\mathcal{L}_{\mathcal{D}}(f_S^{\mathrm{QT}*})-\mathcal{L}_{\mathcal{D}}(f_S^*)\\
&\leq\frac{1}{4}\|(w-G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}})^T\mathbf{x}+b-\hat{b}_{qt}\|_1+O(\frac{1}{\sqrt{m}})\\
&\leq\frac{1}{4}\|(w-G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}})^T\mathbf{x}\|_1+\frac{1}{4}\|b-\hat{b}_{qt}\|_1+O(\frac{1}{\sqrt{m}})\\
&=\frac{1}{4}\|\left(w-G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}}+G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ(|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}}-|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}})\right)^T\mathbf{x}\|_1\\
&+\frac{1}{4}\|b-\hat{b}_{qt}\|_1+O(\frac{1}{\sqrt{m}})\\
&\leq\frac{1}{4}\|(w-G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}})^T\mathbf{x}\|_1\\
&+\frac{1}{4}\|(G_{\theta_{\text{mm}}}\circ\Psi_w^c\circ(|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}}-|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}}))^T\mathbf{x}\|_1\\
&+\frac{1}{4}\|b-\hat{b}_{qt}\|_1+O(\frac{1}{\sqrt{m}})
\end{aligned}
\tag{A.9}
$$

where the initial state $|\text{in}\rangle:=|0\rangle^{\otimes n_{qt}}$, and S represented as an existing optimal operator that $U(\theta_{\text{qnn}})$ is targeting to approximate with the finite number of gate operations. Look into the term:

$$
|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}}-|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}} \tag{A.10}
$$

while considering S as $U(\theta_{\text{qnn}})$ with a perturbation, $S=U+V$ and $\|V\|\leq\epsilon_v$ with small ϵ_v, the term become:

$$
\begin{aligned}
&|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}}-|\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}}\\
&=|\langle\phi|S-U|\text{in}\rangle|^2_{\text{emp}}+2\Re[\langle\phi|U|\text{in}\rangle\langle\text{in}|(S-U)^\dagger|\phi\rangle]_{\text{emp}}
\end{aligned}
\tag{A.11}
$$

With Solovay‐Kitaev theorem $\|S-U\|\leq\epsilon$ [15], the length D of the gate sequence needed to achieve this precision between the target unitary S and the approximated unitary

doi:10.6342/NTU202504677

U (constructed by gates from universal gate set $\mathcal{G}$), is given by $D = O\left(\log^{c_{sk}}(1/\epsilon)\right)$, we obtain

$$
\begin{aligned}
&|\langle\phi|S - U|\text{in}\rangle|^2_{\text{emp}} + 2\Re[\langle\phi|U|\text{in}\rangle\langle\text{in}|(S-U)^\dagger|\phi\rangle]_{\text{emp}} \\
&\leq \epsilon^2 + 2\epsilon \quad (\text{where} \|U\| = 1) \\
&\leq \epsilon(2+\epsilon)
\end{aligned} \tag{A.12}
$$

thus

$$
\begin{aligned}
&\mathcal{L}_{\mathcal{D}}(f_S^{\text{QT}*}) - \mathcal{L}_{\mathcal{D}}(f_S^*) \\
&\leq \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ |\langle\phi|S|\text{in}\rangle|^2_{\text{emp}})^T \mathbf{x}\|_1 \\
&+\frac{1}{4}\|(G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ (|\langle\phi|S|\text{in}\rangle|^2_{\text{emp}} - |\langle\phi|U(\theta_{\text{qnn}})|\text{in}\rangle|^2_{\text{emp}}))^T \mathbf{x}\|_1 \\
&+\frac{1}{4}\|b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}}) \\
&\leq \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ |\langle\phi|S|\text{in}\rangle|^2_{\text{emp}})^T \mathbf{x}\|_1 \\
&+\frac{\epsilon(2+\epsilon)}{4}\|(G_{\theta_{\text{mm}}} \circ \Psi_w^c(\mathbf{1}))^T \mathbf{x}\|_1 + \frac{1}{4}\|b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{m}}) \\
&\leq \frac{1}{4}\|(w - G_{\theta_{\text{mm}}} \circ \Psi_w^c \circ |\langle\phi|S|\text{in}\rangle|^2_{\text{emp}})^T \mathbf{x}\|_1 + \frac{1}{4}\|b - \hat{b}_{qt}\|_1 \\
&+O(\frac{1}{\exp\left(D^{1/c_{sk}}\right)}) + O(\frac{1}{\sqrt{m}}) \quad \text{(Solovay–Kitaev Theorem)}
\end{aligned} \tag{A.13}
$$

where D is the gate sequence length, proportional to the depth of the QNN parameterized by θ_{qnn}, and constant $c_{sk} > 1$ is determined by the efficiency of the recursive refinement algorithm used to approximate a target unitary S with a sequence of gates U from a universal set, where $c_{sk} \approx 3.97$ in the standard Solovay–Kitaev algorithm. In our context, a smaller c_{sk} means that fewer gates are required to approximate the given target unitary. Thus, it is natural to consider that c_{sk} is determined by the QNN architecture and can be reduced through QNN architecture search. Here, we use the behavior of this approximation

doi:10.6342/NTU202504677

to present the precision ϵ decreases exponentially with the sequence length $D^{1/c_{sk}}$.

Next, recall that the mapping model $G_{\theta_{\text{mm}}}$ is a classical NN with a sigmoid activation function. By the universal approximation theorem, such a network, with sufficient neurons, can approximate any continuous function on a compact domain, making it a universal approximator in principle [3]. With $\prod_{k=1}^{K} h_k = h_g$ the neuron number dependency of $G_{\theta_{\text{mm}}}$, the QT approximation error reads:

$$
\begin{aligned}
&\mathcal{L}_{\mathcal{D}}(f_S^{\text{QT}*}) - \mathcal{L}_{\mathcal{D}}(f_S^*) \\
&\leq \frac{1}{4\sqrt{h_g}}\|\mathbf{1x}\|_1 + \frac{1}{4}\|b - \hat{b}_{qt}\|_1 + O(\frac{1}{\exp\left(L^{1/c_{sk}}\right)}) + O(\frac{1}{\sqrt{m}}) \quad \text{(Universal Approx.)} \\
&\leq \frac{1}{4}\|b - \hat{b}_{qt}\|_1 + O(\frac{1}{\sqrt{h_g}}) + O(\frac{1}{\exp\left(L^{1/c_{sk}}\right)}) + O(\frac{1}{\sqrt{m}}) \\
&\leq O(\frac{1}{\sqrt{h_g}}) + O(\frac{1}{\exp\left(D^{1/c_{sk}}\right)}) + O(\frac{1}{\sqrt{m}}) \qquad \text{(A.14)}
\end{aligned}
$$

where we apply the same procedure as $\hat{w}_{qt}$ to $\hat{b}_{qt}$ in the last step, which concludes the proof of Theorem 1.

doi:10.6342/NTU202504677